Raymond Chiong (Ed.)

Nature-Inspired Algorithms for Optimisation

Studies in Computational Intelligence, Volume 193

Editor-in-Chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Raymond Chiong (Ed.)

Nature-Inspired Algorithms for Optimisation

Springer

Raymond Chiong
Swinburne University of Technology
Sarawak Campus, Jalan Simpang Tiga
93350 Kuching
Sarawak, Malaysia
E-mail: rchiong@swinburne.edu.my

and

Swinburne University of Technology
John Street, Hawthorn
Victoria 3122
Australia
E-mail: rchiong@swin.edu.au

ISBN 978-3-642-00266-3 e-ISBN 978-3-642-00267-0

DOI 10.1007/978-3-642-00267-0

Studies in Computational Intelligence ISSN 1860949X

Library of Congress Control Number: 2009920517

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed in acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Foreword

Research on stochastic optimisation methods emerged around half a century ago. One of these methods, evolutionary algorithms (EAs) first came into sight in the 1960s. At that time EAs were merely an academic curiosity without much practical significance. It was not until the 1980s that the research on EAs became less theoretical and more applicable. With the dramatic increase in computational power today, many practical uses of EAs can now be found in various disciplines, including scientific and engineering fields.

EAs, together with other nature-inspired approaches such as artificial neural networks, swarm intelligence, or artificial immune systems, subsequently formed the field of natural computation. While EAs use natural evolution as a paradigm for solving search and optimisation problems, other methods draw on the inspiration from the human brain, collective behaviour of natural systems, biological immune systems, etc. The main motivation behind nature-inspired algorithms is the success of nature in solving its own myriad problems. Indeed, many researchers have found these nature-inspired methods appealing in solving practical problems where a high degree of intricacy is involved and a bagful of constraints need to be dealt with on a regular basis. Numerous algorithms aimed at disentangling such problems have been proposed in the past, and new algorithms are being proposed nowadays.

This book assembles some of the most innovative and intriguing nature-inspired algorithms for solving various optimisation problems. It also presents a range of new studies which are important and timely. All the chapters are written by active researchers in the field of natural computation, and are carefully presented with challenging and rewarding technical content. I am sure the book will serve as a good reference for all researchers and practitioners, who can build on the many ideas introduced here and make more valuable contributions in the future. Enjoy!

November 2008

Professor Zbigniew Michalewicz
School of Computer Science
University of Adelaide, Australia
`http://www.cs.adelaide.edu.au/~zbyszek/`

Preface

Nature has always been a source of inspiration. In recent years, new concepts, techniques and computational applications stimulated by nature are being continually proposed and exploited to solve a wide range of optimisation problems in diverse fields. Various kinds of nature-inspired algorithms have been designed and applied, and many of them are producing high quality solutions to a variety of real-world optimisation tasks. The success of these algorithms has led to competitive advantages and cost savings not only to the scientific community but also the society at large.

The use of nature-inspired algorithms stands out to be promising due to the fact that many real-world problems have become increasingly complex. The size and complexity of the optimisation problems nowadays require the development of methods and solutions whose efficiency is measured by their ability to find acceptable results within a reasonable amount of time. Despite there is no guarantee of finding the optimal solution, approaches based on the influence of biology and life sciences such as evolutionary algorithms, neural networks, swarm intelligence algorithms, artificial immune systems, and many others have been shown to be highly practical and have provided state-of-the-art solutions to various optimisation problems.

This book provides a central source of reference by collecting and disseminating the progressive body of knowledge on the novel implementations and important studies of nature-inspired algorithms for optimisation purposes. Addressing the various issues of optimisation problems using some new and intriguing intelligent algorithms is the novelty of this edited volume. It comprises 18 chapters, which can be categorised into the following 5 sections:

- Section I Introduction
- Section II Evolutionary Intelligence
- Section III Collective Intelligence
- Section IV Social-Natural Intelligence
- Section V Multi-Objective Optimisation

The first section contains two introductory chapters. In the first chapter, *Weise et al.* explain why optimisation problems are difficult to solve by addressing some of the fundamental issues that are often encountered in optimisation tasks such as premature convergence, ruggedness, causality, deceptiveness, neutrality, epistasis, robustness, overfitting, oversimplification, multi-objectivity, dynamic fitness, the No Free Lunch Theorem, etc. They also present some possible countermeasures, focusing on the stochastic based nature-inspired solutions, for dealing with these problematic features. This is probably the very first time in the literature that all these features have been discussed within a single document. Their discussion also leads to the conclusion of why so many different types of algorithms are needed.

While parallels can certainly be drawn between these algorithms and various natural processes, the extent of the natural inspiration is not always clear. *Steer et al.* thus attempt to clarify what it means to say an algorithm is nature-inspired and examine the rationale behind the use of nature as a source of inspiration for such algorithm in the second chapter. In addition, they also discuss the features of nature which make it a valuable resource in the design of successful new algorithms. Finally, the history of some well-known algorithms are discussed, with particular focus on the role nature has played in their development.

The second section of this book deals with evolutionary intelligence. It contains six chapters, presenting several novel algorithms based on simulated learning and evolution – a process of adaptation that occurs in nature. The first chapter in this section by *Salomon and Arnold* describes a hybrid evolutionary algorithm, called the Evolutionary-Gradient-Search (EGS) procedure. This procedure initially uses random variations to estimate the gradient direction, and then deterministically searches along that direction in order to advance to the optimum. The idea behind it is to utilise all individuals in the search space to gain as much information as possible, rather than selecting only the best offspring. Through both theoretical analysis and empirical studies, the authors show that the EGS procedure works well on most optimisation problems where evolution strategies also work well, in particular those with unimodal functions. Besides that, this chapter also discusses the EGS procedure's behaviour in the presence of noise. Due to some performance degradations, the authors introduce the concept of inverse mutation, a new idea that proves very useful in the presence of noise, which is omnipresent in almost any real-world application.

In an attempt to address some limitations of the standard genetic algorithm, *Lenaerts et al.* in the second chapter of this section present an algorithm that mimics evolutionary transitions from biology called the Evolutionary Transition Algorithm (ETA). They use the Binary Constraint Satisfaction Problem (BINCSP) as an illustration to show how ETA is able to evolve increasingly complex solutions from the interactions of simpler evolving solutions. Their experimental results on BINCSP confirm that the ETA is a promising approach that requires more extensive investigation from both theoretical and practical optimisation perspectives.

Following which, *Tenne* proposes a new model-assisted Memetic Algorithm for expensive optimisation problems. The proposed algorithm uses a radial basis function neural network as a global model and performs a global search on this model. It then uses a local search with a trust-region framework to converge to a true optimum. The local search uses Kriging models and adapts them during the search to improve convergence. The author benchmarks the proposed algorithm against four model-assisted evolutionary algorithms using eight well-known mathematical test functions, and shows that this new model-assisted Memetic Algorithm is able to outperform the four reference algorithms. Finally, the proposed algorithm is applied to a real-world application of airfoil shape optimisation, where better performance than the four reference algorithms is also obtained.

In the next chapter, *Wang and Li* propose a new self-adaptive estimation of distribution algorithm (EDA) for large scale global optimisation (LSGO) called the Mixed model Uni-variate EDA (MUEDA). They begin with an analysis on the behaviour and performances of uni-variate EDAs with different kernel probability densities via fitness landscape analysis. Based on the analysis, the self-adaptive MUEDA is devised. To assess the effectiveness and efficiency of MUEDA, the authors test it on typical function optimisation tasks with dimensionality scaling from 30 to 1500. Compared to other recently published LSGO algorithms, the MUEDA shows excellent convergence speed, final solution quality and dimensional scalability.

Subsequently, *Tirronen and Neri* propose a Differential Evolution (DE) with integrated fitness diversity self-adaptation. In their algorithm, the authors introduce a modified probabilistic criterion which is based on a novel measurement of the fitness diversity. In addition, the algorithm contains an adaptive population size which is determined by variations in the fitness diversity. Extensive experimental studies have been carried out, where the proposed DE is being compared to a standard DE and four modern DE based algorithms. Numerical results show that the proposed DE is able to produce promising solutions and is competitive with the modern DEs. Its convergence speed is also comparable to those state-of-the-art DE based algorithms.

In the final chapter of this section, *Patel* uses genetic algorithms to optimise a class of biological neural networks, called Central Pattern Generators (CPGs), with a view to providing autonomous, reactive and self-modulatory control for practical engineering solutions. This work is precursory to producing controllers for marine energy devices with similar locomotive properties. Neural circuits are evolved using evolutionary techniques. The lamprey CPG, responsible for swimming movements, forms the basis of evolution, and is optimised to operate with a wider range of frequencies and speeds. The author demonstrates via experimental results that simpler versions of the CPG network can be generated, whilst outperforming the swimming capabilities of the original CPG network.

The third section deals with collective intelligence, a term applied to any situation in which indirect influences cause the emergence of collaborative effort. Four chapters are presented, each addressing one novel algorithm. The first chapter of the section by *Bastos Filho et al.* gives an overview of a new algorithm for searching in high-dimensional spaces, called the Fish School Search (FSS). Based on the behaviours of fish schools, the FSS works through three main operators: feeding, swimming and breeding. Via empirical studies, the authors demonstrate that the FSS is quite promising for dealing with high-dimensional problems with multimodal functions. In particular, it has shown great capability in finding balance between exploration and exploitation, self-adapting swiftly out of local minima, and self-regulating the search granularity.

The next chapter by *Tan and Zhang* presents another new swarm intelligence algorithm called the Magnifier Particle Swarm Optimisation (MPSO). Based on the idea of magnification transformation, the MPSO enlarges the range around each generation's best individual, while the velocity of particles remains unchanged. This enables a much faster convergence speed and better optimisation solving capability. The authors compare the performance of MPSO to the Standard Particle Swarm Optimisation (SPSO) using the thirteen benchmark test functions from CEC 2005. The experimental results show that the proposed MPSO is indeed able to tremendously speed up the convergence and maintain high accuracy in searching for the global optimum. Finally, the authors also apply the MPSO to spam detection, and demonstrate that the proposed MPSO achieves promising results in spam email classification.

Mezura-Montes and Flores-Mendoza then present a study about the behaviour of Particle Swarm Optimisation (PSO) in constrained search spaces. Four well-known PSO variants are used to solve a set of test problems for comparison purposes. Based on the comparative study, the authors identify the most competitive PSO variant and improve it with two simple modifications related to the dynamic control of some parameters and a variation in the constraint-handling technique, resulting in a new Improved PSO (IPSO). Extensive experimental results show that the IPSO is able to improve the results obtained by the original PSO variants significantly. The convergence behaviour of the IPSO suggests that it has better exploration capability for avoiding local optima in most of the test problems. Finally, the authors compare the IPSO to four state-of-the-art PSO-based approaches, and confirm that it can achieve competitive or even better results than these approaches, with a moderate computational cost.

The last chapter of this section by *Rabanal et al.* describes an intriguing algorithm called the River Formation Dynamics (RFD). This algorithm is inspired by how water forms rivers by eroding the ground and depositing sediments. After drops transform the landscape by increasing or decreasing the altitude of different areas, solutions are given in the form of paths of decreasing altitudes. Decreasing gradients are constructed, and these gradients are followed by subsequent drops to compose new gradients and reinforce the best ones. The authors apply the RFD to solve three NP-complete problems, and compare its performance to Ant Colony Optimisation (ACO). While the RFD normally takes longer than ACO to find good solutions, it is usually able to outperform ACO in terms of solution quality after some additional time passes.

The fourth section contains two survey chapters. The first survey chapter by *Neme and Hernández* discusses optimisation algorithms inspired by social phenomena in human societies. This study is highly important as majority of the natural algorithms in the optimisation domain are inspired by either biological phenomena or social behaviours of mainly animals and insects. As social phenomena often arise as a result of interaction among individuals, the main idea behind algorithms inspired by social phenomena is that the computational power of the inspired algorithms is correlated to the richness and complexity of the corresponding social behaviour. Apart from presenting social phenomena that have motivated several optimisation algorithms, the authors also refer to some social processes whose metaphor may lead to new algorithms. Their hypothesis is that some of these phenomena - the ones with high complexity, have more computational power than other, less complex phenomena.

The second survey chapter by *Bernardino and Barbosa* focuses on the applications of Artificial Immune Systems (AISs) in solving optimisation problems. AISs are computational methods inspired by the natural immune system. The main types of optimisation problems that have been considered include the unconstrained optimisation problems, the constrained optimisation problems, the multimodal optimisation problems, as well as the multi-objective optimisation problems. While several immune mechanisms are discussed, the authors have paid special attention to two of the most popular immune methodologies: clonal selection and immune networks. They remark that even though AISs are good for solving various optimisation problems, useful features from other techniques are often combined with a "pure" AIS in order to generate hybridised AIS methods with improved performance.

The fifth section deals with multi-objective optimisation. There are four chapters in this section. It starts with a chapter by *Jaimes et al.* who present a comparative study of different ranking methods on many-objective problems. The authors consider an optimisation problem to be a many-objective optimisation problem (instead of multi-objective) when it has more than 4 objectives. Their aim is to investigate the effectiveness of different approaches in order to find out the advantages and disadvantages of each of the ranking methods studied and, in general, their performance. The results presented can be an important guide for selecting a suitable ranking method for a particular problem at hand, developing new ranking schemes or extending the Pareto optimality relation.

Next, *Nebro and Durillo* present an interesting chapter that studies the effect of applying a steady-state selection scheme to Non-dominated Sorting Genetic Algorithm II (NSGA-II), a fast and elitist Multi-Objective Evolutionary Algorithm (MOEA). This work is definitely a timely and important one, since not many non-generational MOEAs exist. The authors use a benchmark composed of 21 bi-objective problems for comparing the performance of both the original and the steady-state versions of NSGA-II in terms of the quality of the obtained solutions and their convergence speed towards the optimal Pareto front. Comparative studies between the two versions as well as four state-of-the-art multi-objective optimisers not only demonstrate the significant improvement obtained by the steady-state scheme over the generational one in most of the problems, but also its competitiveness with the state-of-the-art algorithms regarding the quality of the obtained approximation sets and the convergence speed.

The following chapter by *Tan and Teo* proposes two new co-evolutionary algorithms for multi-objective optimisation based on the Strength Pareto Evolutionary Algorithm 2 (SPEA2), another state-of-the-art MOEA. The two new algorithms introduce the concept of competitive co-evolution and cooperative co-evolution respectively to SPEA2. The authors are able to exhibit, through experimental studies, the superiority of these augmented algorithms over the original one in terms of the non-dominated solutions to the true Pareto front, the diversity of the obtained solutions as well as the coverage level. Moreover, the authors observe an increased performance improvement over the original SPEA2 with an increase in the number of dimensions to be optimised. Overall, this chapter shows that the introduction of co-evolution, especially cooperative co-evolution, is able to furnish significant enhancements to the solution of multi-objective optimisation problems.

The final chapter by *Duran et al.* focuses on portfolio optimisation using multi-objective optimisation techniques. Based on the Venezuelan market mutual funds from year 1994 to 2002, the authors conduct a comparative study of three different evolutionary multi-objective approaches – NSGA-II, SPEA2, and Indicator-Based Evolutionary Algorithm (IBEA) – as well as the optimisation portfolios generated by these approaches. Using Sharpe's index as a measure of risk premium for the final solution selection, the authors observe that NSGA-II is able to provide results similar to SPEA2 for mixed and fixed mutual funds, and superior solutions than SPEA2 for variable funds. This observation, the authors argue, is indication that NSGA-II provides better coverage of the regions containing interesting solutions for Sharpe's index. The experimental results presented also demonstrate that IBEA is superior to both NSGA-II and SPEA2 regarding the index value attained, and the portfolios IBEA generates are more profitable than those indexed by the Caracas Stock Exchange.

In closing, I would like to thank all the authors for their excellent contributions to this book. I also wish to acknowledge the help of the editorial advisory board and all reviewers involved in the review process, without whose support this book project could not have been satisfactorily completed. Special thanks go to all those who provided constructive and comprehensive review comments, as well as those who willingly helped in last-minute urgent reviews. A further special note of thanks goes to Dr Thomas Ditzinger (Engineering Senior Editor, Springer-Verlag) and Ms Heather King (Engineering Editorial, Springer-Verlag) for their editorial assistance and professional support. Finally, I hope that readers would enjoy reading this book as much as I have enjoyed putting it together.

December 2008 Raymond Chiong

Organization

Editorial Advisory Board

List of Reviewers

Efrén Mezura-Montes	Laboratorio Nacional de Informática Avanzada, México
Enrique Alba	University of Malaga, Spain
Fernando Buarque Lima Neto	University of Pernambuco, Brazil
Fernando Rubio Diez	Universidad Complutense de Madrid, Spain
Ferrante Neri	University of Jyväskylä, Finland
Francisco Chicano	University of Malaga, Spain
Gary G Yen	Oklahoma State University, USA
Guillermo Leguizamón	Universidad Nacional de San Luis, Argentina
Helio J C Barbosa	Laboratório Nacional de Computação Cientí-fica (LNCC), Brazil
James Montgomery	Swinburne University of Technology, Australia
Jon Timmis	University of York, UK
Jörn Grahl	Johannes Gutenberg University Mainz, Germany
Jose Barahona da Fonseca	New University of Lisbon, Portugal
Kesheng Wang	Norwegian University of Science and Technology, Norway
Laurent Deroussi	IUT de Montluçon, France
Leena N Patel	University of Edinburgh, UK
Maurice Clerc	Independent Consultant, France
Michael Zapf	University of Kassel, Germany
Nguyen Xuan Hoai	Seoul National University, Korea
Thomas Weise	University of Kassel, Germany
Tim Hendtlass	Swinburne University of Technology, Australia
Tom Lenaerts	Université Libre de Bruxelles, Belgium
Uday K Chakraborty	University of Missouri, USA
Walter D Potter	University of Georgia, USA
Ying Tan	Peking University, China
Yoel Tenne	University of Sydney, Australia

Contents

Why Is Optimization Difficult?

Thomas Weise, Michael Zapf, Raymond Chiong, and Antonio J. Nebro

Abstract. This chapter aims to address some of the fundamental issues that are often encountered in optimization problems, making them difficult to solve. These issues include premature convergence, ruggedness, causality, deceptiveness, neutrality, epistasis, robustness, overfitting, oversimplification, multi-objectivity, dynamic fitness, the No Free Lunch Theorem, etc. We explain why these issues make optimization problems hard to solve and present some possible countermeasures for dealing with them. By doing this, we hope to help both practitioners and fellow researchers to create more efficient optimization applications and novel algorithms.

1 Introduction

Optimization, in general, is concerned with finding the best solutions for a given problem. Its applicability in many different disciplines makes it hard to give an exact definition. Mathematicians, for instance, are interested in finding the maxima or minima of a real function from within an allowable set of variables. In computing and engineering, the goal is to maximize the performance of a system or application with minimal runtime and resources.

Thomas Weise · Michael Zapf
Distributed Systems Group, University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany
e-mail: weise@vs.uni-kassel.de and zapf@vs.uni-kassel.de

Raymond Chiong
School of Computing & Design, Swinburne University of Technology (Sarawak Campus), 93350 Kuching, Sarawak, Malaysia
e-mail: rchiong@swinburne.edu.my

Antonio J. Nebro
Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, University of Málaga, Campus de Teatinos, 29071 Málaga, Spain
e-mail: antonio@lcc.uma.es

In the business industry, people aim to optimize the efficiency of a production process or the quality and desirability of their current products.

All these examples show that optimization is indeed part of our everyday life. We often try to maximize our gain by minimizing the cost we need to bear. However, are we really able to achieve an “optimal” condition? Frankly, whatever problems we are dealing with, it is rare that the optimization process will produce a solution that is truly optimal. It may be optimal for one audience or for a particular application, but definitely not in all cases.

As such, various techniques have emerged for tackling different kinds of optimization problems. In the broadest sense, these techniques can be classified into exact and stochastic algorithms. Exact algorithms, such as branch and bound, $A^\star$ search, or dynamic programming can be highly effective for small-size problems. When the problems are large and complex, especially if they are either NP-complete or NP-hard, i.e., have no known polynomial-time solutions, the use of stochastic algorithms becomes mandatory. These stochastic algorithms do not guarantee an optimal solution, but they are able to find quasi-optimal solutions within a reasonable amount of time.

In recent years, metaheuristics, a family of stochastic techniques, has become an active research area. They can be defined as higher level frameworks aimed at efficiently and effectively exploring a search space [25]. The initial work in this area was started about half a century ago (see [175, 78, 24], and [37]). Subsequently, a lot of diverse methods have been proposed, and today, this family comprises many well-known techniques such as Evolutionary Algorithms, Tabu Search, Simulated Annealing, Ant Colony Optimization, Particle Swarm Optimization, etc.

There are different ways of classifying and describing metaheuristic algorithms. The widely accepted classification would be the view of nature-inspired vs. non nature-inspired, i.e., whether or not the algorithm somehow emulates a process found in nature. Evolutionary Algorithms, the most widely used metaheuristics, belong to the nature-inspired class. Other techniques with increasing popularity in this class include Ant Colony Optimization, Particle Swarm Optimization, Artificial Immune Systems, and so on. Scatter search, Tabu Search, and Iterated Local Search are examples of non nature-inspired metaheuristics. Unified models of metaheuristic optimization procedures have been proposed by Vaessens et al [220, 221], Rayward-Smith [169], Osman [158], and Taillard et al [210].

In this chapter, our main objective is to address some fundamental issues that make optimization problems difficult based on the nature-inspired class of metaheuristics. Apart from the reasons of being large, complex, and dynamic, we present a list of problem features that are often encountered and explain why some optimization problems are hard to solve. Some of the issues that will be discussed, such as multi-modality and overfitting, concern global optimization in general. We will also elaborate on other issues which are often linked to Evolutionary Algorithms, e.g., epistasis and neutrality, but can occur in virtually all metaheuristic optimization processes.

These concepts are important, as neglecting any one of them during the design of the search space and operations or the configuration of the optimization algorithms can render the entire invested effort worthless, even if highly efficient optimization methods are applied. To the best of our knowledge, to date there is not a single document in the literature comprising all such problematic features. By giving clear definitions and comprehensive introductions on them, we hope to create awareness among fellow scientists as well as practitioners in the industry so that they could perform optimization tasks more efficiently.

The rest of this chapter is organized as follows: In the next section, premature convergence to local minima is introduced as one of the major symptoms of failed optimization processes. Ruggedness (Section 3), deceptiveness (Section 4), too much neutrality (Section 5), and epistasis (Section 6), some of which have been illustrated in Fig. 1[1], are the main causes which may lead to this situation. Robustness, correctness, and generality instead are features which we expect from valid solutions. They are challenged by different types of noise discussed in Section 7 and the affinity of overfitting or overgeneralization (see Section 8). Some optimization tasks become further complicated because they involve multiple, conflicting objectives (Section 9) or dynamically changing ones (Section 10). In Section 11, we give a short introduction about the No Free Lunch Theorem, from which we can follow that no panacea, no magic bullet can exist against all of these problematic features. We will conclude our outline of the hardships of optimization with a summary in Section 12.

1.1 Basic Terminology

In the following text, we will utilize a terminology commonly used in the Evolutionary Algorithms community and sketched in Fig. 2 based on the example of a simple Genetic Algorithm. The possible solutions x of an optimization problem are elements of the problem space $\mathbb{X}$. Their utility as solutions is evaluated by a set $\mathbf{f}$ of objective functions f which, without loss of generality, are assumed to be subject to minimization. The set of search operations utilized by the optimizers to explore this space does not directly work on them. Instead, they are applied to the elements (the genotypes) of the search space $\mathbb{G}$ (the genome). They are mapped to the solution candidates by a genotype-phenotype mapping $\mathrm{gpm} : \mathbb{G} \mapsto \mathbb{X}$. The term *individual* is used for both, solution candidates and genotypes.

[1] We include in Fig. 1 different examples of fitness landscapes, which relate solution candidates (or genotypes) to their objective values. The small bubbles in Fig. 1 represent solution candidates under investigation. An arrow from one bubble to another means that the second individual is found by applying one search operation to the first one. The objective values here are subject to minimization.

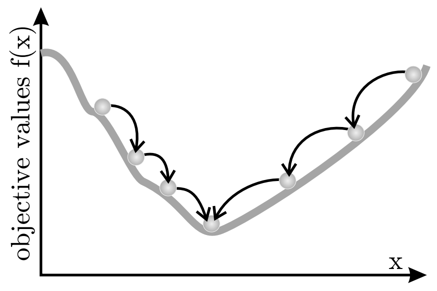


Fig. 1.a: Best Case

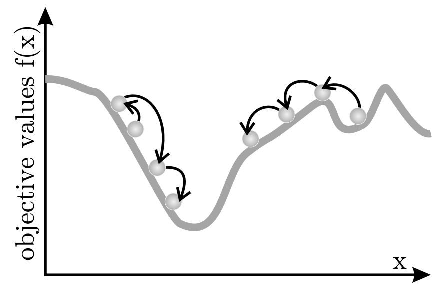


Fig. 1.b: Low Total Variation

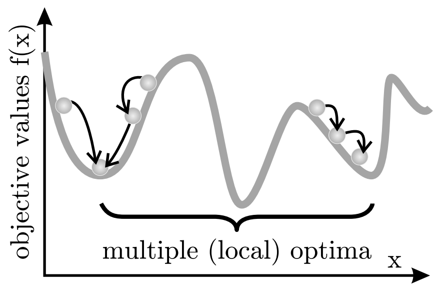


Fig. 1.c: Multimodal

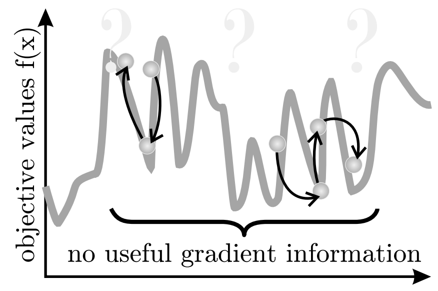


Fig. 1.d: Rugged

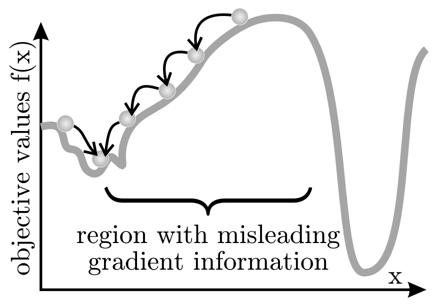


Fig. 1.e: Deceptive

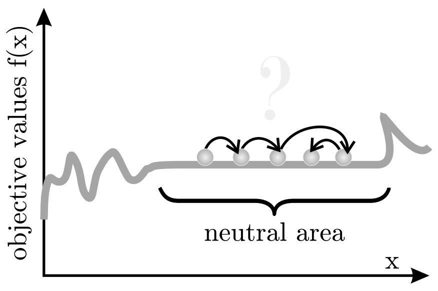


Fig. 1.f: Neutral

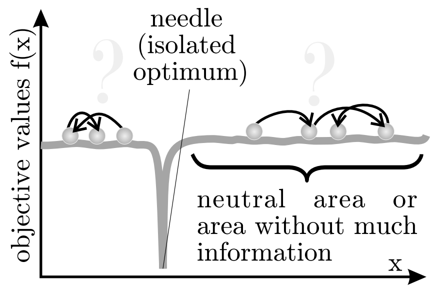

objective values f(x)
needle (isolated optimum)
neutral area or area without much information
x

Fig. 1.g: Needle-In-A-Haystack

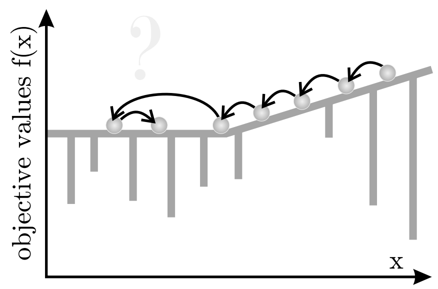


Fig. 1.h: Nightmare

Fig. 1 Different possible properties of fitness landscapes (minimization)

1.2 The Term "Difficult"

Before we go more into detail about what makes these landscapes *difficult*, we should establish the term in the context of optimization. The degree of difficulty of solving a certain problem with a dedicated algorithm is closely related to its *computational complexity*, i.e., the amount of resources such as time and memory required to do so. The computational complexity depends on the number of input elements needed for applying the algorithm. This dependency is often expressed in the form of approximate boundaries with the Big-O-family notations introduced by Bachmann [10] and made popular by Landau [122]. Problems can be further divided into *complexity classes*. One of the most difficult complexity classes owning to its resource requirements is NP, the set of all decision problems which are solvable in polynomial time by non-deterministic Turing machines [79]. Although many attempts have been

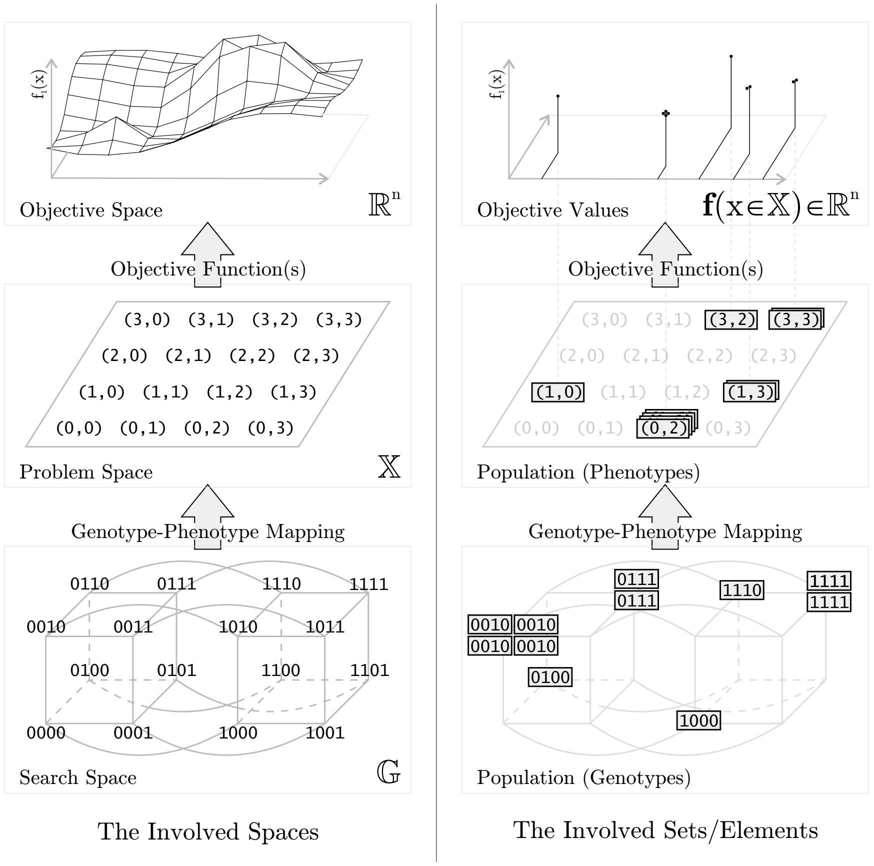


Fig. 2 The involved spaces and sets in optimization

made, no algorithm has been found which is able to solve an NP-complete [79] problem in polynomial time on a deterministic computer. One approach to obtaining near-optimal solutions for problems in NP in reasonable time is to apply metaheuristic, randomized optimization procedures.

As already stated, optimization algorithms are guided by objective functions. A function is *difficult* from a mathematical perspective in this context if it is not continuous, not differentiable, or if it has multiple maxima and minima. This understanding of difficulty comes very close to the intuitive sketches in Fig. 1.

In many real world applications of metaheuristic optimization, the characteristics of the objective functions are not known in advance. The problems are usually NP or have unknown complexity. It is therefore only rarely possible to derive boundaries for the performance or the runtime of optimizers in advance, let alone exact estimates with mathematical precision.

Most often, experience, rules of thumb, and empirical results based on the models obtained from related research areas such as biology are the only guides available. In this chapter we discuss many such models and rules, providing a better understanding of when the application of a metaheuristic is feasible and when not, as well as with indicators on how to avoid defining problems in a way that makes them *difficult*.

2 Premature Convergence

2.1 Introduction

An optimization algorithm has *converged* if it cannot reach new solution candidates anymore or if it keeps on producing solution candidates from a "small"[2] subset of the problem space. Global optimization algorithms will usually converge at some point in time. One of the problems in global optimization is that it is often not possible to determine whether the best solution currently known is situated on a local or a global optimum and thus, if convergence is acceptable. In other words, it is usually not clear whether the optimization process can be stopped, whether it should concentrate on refining the current optimum, or whether it should examine other parts of the search space instead. This can, of course, only become cumbersome if there are multiple (local) optima, i.e., the problem is multimodal as depicted in Fig. 1.c.

A mathematical function is multimodal if it has multiple maxima or minima [195, 246]. A set of objective functions (or a vector function) **f** is multimodal if it has multiple (local or global) optima – depending on the definition of "optimum" in the context of the corresponding optimization problem.

[2] According to a suitable metric like numbers of modifications or mutations which need to be applied to a given solution in order to leave this subset.

2.2 The Problem

An optimization process has *prematurely converged* to a local optimum if it is no longer able to explore other parts of the search space than the area currently being examined *and* there exists another region that contains a superior solution [192, 219]. Fig. 3 illustrates examples of premature convergence.

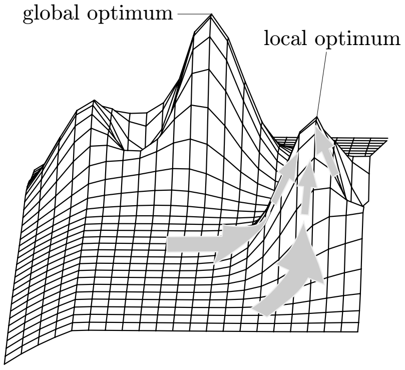


Fig. 3.a: Example 1: Maximization

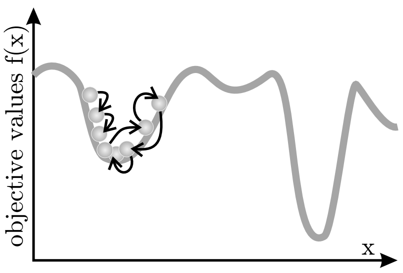


Fig. 3.b: Example 2: Minimization

Fig. 3 Premature convergence in the objective space

The existence of multiple global optima itself is not problematic and the discovery of only a subset of them can still be considered as successful in many cases (see Section 9). The occurrence of numerous local optima, however, is more complicated.

The phenomenon of *domino convergence* has been brought to attention by Rudnick [184] who studied it in the context of his BinInt problem [184, 213]. In principle, domino convergence occurs when the solution candidates have features which contribute significantly to different degrees of the total fitness. If these features are encoded in separate genes (or building blocks) in the genotypes, they are likely to be treated with different priorities, at least in randomized or heuristic optimization methods.

Building blocks with a very strong positive influence on the objective values, for instance, will quickly be adopted by the optimization process (i.e., "converge"). During this time, the alleles of genes with a smaller contribution are ignored. They do not come into play until the optimal alleles of the more "important" blocks have been accumulated. Rudnick [184] called this sequential convergence phenomenon *domino convergence* due to its resemblance to a row of falling domino stones [213].

In the worst case, the contributions of the less salient genes may almost look like noise and they are not optimized at all. Such a situation is also an instance of premature convergence, since the global optimum which would involve optimal configurations of all blocks will not be discovered. In this situation, restarting the optimization process will not help because it will always turn out the same way. Example problems which are often likely to exhibit domino convergence are the Royal Road [139] and the aforementioned BinInt problem [184].

2.3 One Cause: Loss of Diversity

In biology, diversity is *the variety and abundance of organisms at a given place and time* [159, 133]. Much of the beauty and efficiency of natural ecosystems is based on a dazzling array of species interacting in manifold ways. Diversification is also a good investment strategy utilized by investors in the economy in order to increase their profit.

In population-based global optimization algorithms as well, maintaining a set of diverse solution candidates is very important. Losing diversity means approaching a state where all the solution candidates under investigation are similar to each other. Another term for this state is convergence. Discussions about how diversity can be measured have been provided by Routledge [183], Cousins [49], Magurran [133], Morrison and De Jong [148], and Paenke et al [159].

Preserving diversity is directly linked with maintaining a good balance between exploitation and exploration [159] and has been studied by researchers from many domains, such as

- Genetic Algorithms [156, 176, 177],
- Evolutionary Algorithms [28, 29, 123, 149, 200, 206],
- Genetic Programming [30, 38, 39, 40, 53, 93, 94],
- Tabu Search [81, 82], and
- Particle Swarm Optimization [238].

The operations which create new solutions from existing ones have a very large impact on the speed of convergence and the diversity of the populations [69, 203]. The step size in Evolution Strategy is a good example of this issue: setting it properly is very important and leads to the "exploration versus exploitation" problem [102] which can be observed in other areas of global optimization as well.[3]

In the context of optimization, *exploration* means finding new points in areas of the search space which have not been investigated before. Since computers have only limited memory, already evaluated solution candidates usually have to be discarded. Exploration is a metaphor for the procedure which allows search operations to find novel and maybe better solution structures. Such operators (like mutation in Evolutionary Algorithms) have a high chance of creating inferior solutions by destroying good building blocks but

[3] More or less synonymously to exploitation and exploration, the terms *intensifications* and *diversification* have been introduced by Glover [81, 82] in the context of Tabu Search.

also a small chance of finding totally new, superior traits (which, however, is not guaranteed at all).

Exploitation, on the other hand, is the process of improving and combining the traits of the currently known solution(s), as done by the crossover operator in Evolutionary Algorithms, for instance. Exploitation operations often incorporate small changes into already tested individuals leading to new, very similar solution candidates or try to merge building blocks of different, promising individuals. They usually have the disadvantage that other, possibly better, solutions located in distant areas of the problem space will not be discovered.

Almost all components of optimization strategies can either be used for increasing exploitation or in favor of exploration. Unary search operations that improve an existing solution in small steps can be built, hence being exploitation operators (as is done in Memetic Algorithms, for instance). They can also be implemented in a way that introduces much randomness into the individuals, effectively making them exploration operators. Selection operations in Evolutionary Computation choose a set of the most promising solution candidates which will be investigated in the next iteration of the optimizers. They can either return a small group of best individuals (exploitation) or a wide range of existing solution candidates (exploration).

Optimization algorithms that favor exploitation over exploration have higher convergence speed but run the risk of not finding the optimal solution and may get stuck at a local optimum. Then again, algorithms which perform excessive exploration may never improve their solution candidates well enough to find the global optimum or it may take them very long to discover it "by accident". A good example for this dilemma is the Simulated Annealing algorithm [117]. It is often modified to a form called *simulated quenching* which focuses on exploitation but loses the guaranteed convergence to the optimum [110]. Generally, optimization algorithms should employ at least one search operation of explorative character and at least one which is able to exploit good solutions further. There exists a vast body of research on the trade-off between exploration and exploitation that optimization algorithms have to face [7, 57, 66, 70, 103, 152].

2.4 Countermeasures

As we have seen, global optimization algorithms are optimization methods for finding the best possible solution(s) of an optimization problem instead of prematurely converging to a local optimum. Still, there is no general approach to ensure their success. The probability that an optimization process prematurely converges depends on the characteristics of the problem to be solved and the parameter settings and features of the optimization algorithms applied [215].

A very crude and yet, sometimes effective measure is restarting the optimization process at randomly chosen points in time. One example for this method is *GRASP*s, *Greedy Randomized Adaptive Search Procedures* [71, 72], which continuously restart the process of creating an initial solution and refining it with local search. Still, such approaches are likely to fail in domino convergence situations.

In order to extend the duration of the evolution in Evolutionary Algorithms, many methods have been devised for steering the search away from areas which have already been frequently sampled. This can be achieved by integrating density metrics into the fitness assignment process. The most popular of such approaches are sharing and niching based on the Euclidean distance of the solution candidates in objective space [55, 85, 104, 138]. Using low selection pressure furthermore decreases the chance of premature convergence but also decreases the speed with which good solutions are exploited.

Another approach against premature convergence is to introduce the capability of self-adaptation, allowing the optimization algorithm to change its strategies or to modify its parameters depending on its current state. Such behaviors, however, are often implemented not in order to prevent premature convergence but to speed up the optimization process (which may lead to premature convergence to local optima) [185, 186, 187].

3 Ruggedness and Weak Causality

3.1 The Problem: Ruggedness

Optimization algorithms generally depend on some form of gradient in the objective or fitness space. The objective functions should be continuous and exhibit low total variation[4], so the optimizer can descend the gradient easily. If the objective functions are unsteady or fluctuating, i.e., going up and down, it becomes more complicated for the optimization process to find the right directions to proceed to. The more rugged a function gets, the harder it becomes to optimize it. From a simplified point of view, ruggedness is multimodality plus steep ascends and descends in the fitness landscape. Examples of rugged landscapes are Kauffman's NK fitness landscape [113, 115], the p-Spin model [6], Bergman and Feldman's jagged fitness landscape [19], and the sketch in Fig. 1.d.

3.2 One Cause: Weak Causality

During an optimization process, new points in the search space are created by the search operations. Generally we can assume that the genotypes which are the input of the search operations correspond to phenotypes which have previously been selected. Usually, the better or the more promising an individual is, the higher are its chances of being selected for further investigation. Reversing this statement suggests that individuals which are passed to the

[4] `http://en.wikipedia.org/wiki/Total_variation` [accessed 2008-04-23]

search operations are likely to have a good fitness. Since the fitness of a solution candidate depends on its properties, it can be assumed that the features of these individuals are not so bad either. It should thus be possible for the optimizer to introduce slight changes to their properties in order to find out whether they can be improved any further[5]. Normally, such modifications should also lead to small changes in the objective values and, hence, in the fitness of the solution candidate.

Definition 1 (Strong Causality). *Strong causality* (locality) means that small changes in the properties of an object also lead to small changes in its behavior [170, 171, 180].

This principle (proposed by Rechenberg [170, 171]) should not only hold for the search spaces and operations designed for optimization, but applies to natural genomes as well. The offspring resulting from sexual reproduction of two fish, for instance, has a different genotype than its parents. Yet, it is far more probable that these variations manifest in a unique color pattern of the scales, for example, instead of leading to a totally different creature.

Apart from this straightforward, informal explanation here, causality has been investigated thoroughly in different fields of optimization, such as Evolution Strategy [170, 65], structure evolution [129, 130], Genetic Programming [65, 107, 179, 180], genotype-phenotype mappings [193], search operators [65], and Evolutionary Algorithms in general [65, 182, 207].

In fitness landscapes with weak (low) causality, small changes in the solution candidates often lead to large changes in the objective values, i.e., ruggedness. It then becomes harder to decide which region of the problem space to explore and the optimizer cannot find reliable gradient information to follow. A small modification of a very bad solution candidate may then lead to a new local optimum and the best solution candidate currently known may be surrounded by points that are inferior to all other tested individuals.

The lower the causality of an optimization problem, the more rugged its fitness landscape is, which leads to a degradation of the performance of the optimizer [120]. This does not necessarily mean that it is impossible to find good solutions, but it may take very long to do so.

3.3 Countermeasures

To our knowledge, no viable method which can directly mitigate the effects of rugged fitness landscapes exists. In population-based approaches, using large population sizes and applying methods to increase the diversity can decrease the influence of ruggedness, but only up to a certain degree. Utilizing the Baldwin effect [13, 100, 101, 233] or Lamarckian evolution [54, 233], i.e., incorporating a local search into the optimization process, may further help to smoothen out the fitness landscape [89].

[5] We have already mentioned this under the subject of exploitation.

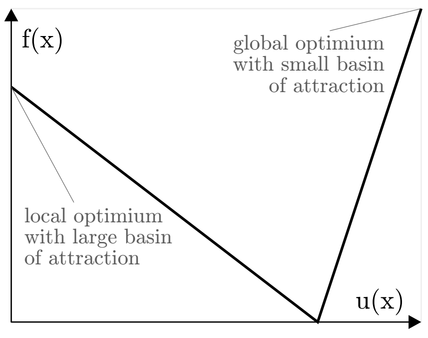
Weak causality is often a home-made problem: it results from the choice of the solution representation and search operations. Thus, in order to apply Evolutionary Algorithms in an efficient manner, it is necessary to find representations which allow for iterative modifications with bounded influence on the objective values.

4 Deceptiveness

4.1 Introduction

Especially annoying fitness landscapes show *deceptiveness* (or deceptivity). The gradient of deceptive objective functions leads the optimizer away from the optima, as illustrated in Fig. 1.e.

The term deceptiveness is mainly used in the Genetic Algorithm community in the context of the Schema Theorem. Schemas describe certain areas (hyperplanes) in the search space. If an optimization algorithm has discovered an area with a better average fitness compared to other regions, it will focus on exploring this region based on the assumption that highly fit areas are likely to contain the true optimum. Objective functions where this is not the case are called deceptive [20, 84, 127]. Examples for deceptiveness are the ND fitness landscapes [17], trap functions [1, 59, 112] like the one illustrated in Fig. 4, and the fully deceptive problems given by Goldberg et al [86, 60].



$$f(x) = \begin{cases} (8n/z)\,(z - u(x)) & \text{if } u(x) \leq z \\ (10n/\,(n - z))\,(u(x) - z) & \text{otherwise} \end{cases}$$

where $u(x)$ is the number of ones in the bit string x of length n and $z = \lfloor 3n/4 \rfloor$. $f(x)$ is subject to maximization.

Fig. 4 Ackley's “Trap” function [1, 112]

4.2 The Problem

If the information accumulated by an optimizer actually guides it away from the optimum, search algorithms will perform worse than a random walk or an exhaustive enumeration method. This issue has been known for a long time [228, 140, 141, 212] and has been subsumed under the No Free Lunch Theorem which we will discuss in Section 11.

4.3 Countermeasures

Solving deceptive optimization tasks perfectly involves sampling many individuals with very bad features and low fitness. This contradicts the basic ideas of metaheuristics and thus, there are no efficient countermeasures against deceptivity. Using large population sizes, maintaining a very high diversity, and utilizing linkage learning (see Section 6.3) are, maybe, the only approaches which can provide at least a small chance of finding good solutions.

5 Neutrality and Redundancy

5.1 The Problem: Neutrality

We consider the outcome of the application of a search operation to an element of the search space as neutral if it yields no change in the objective values [15, 172]. It is challenging for optimization algorithms if the best solution candidate currently known is situated on a plane of the fitness landscape, i.e., all adjacent solution candidates have the same objective values. As illustrated in Fig. 1.f, an optimizer then cannot find any gradient information and thus, no direction in which to proceed in a systematic manner. From its point of view, each search operation will yield identical individuals. Furthermore, optimization algorithms usually maintain a list of the best individuals found, which will then overflow eventually or require pruning.

The degree of neutrality ν is defined as the fraction of neutral results among all possible products of the search operations Op applied to a specific genotype [15]. We can generalize this measure to areas G in the search space $\mathbb{G}$ by averaging over all their elements. Regions where ν is close to one are considered as *neutral*.

$$\forall g_1 \in \mathbb{G} \Rightarrow \nu(g_1) = \frac{|\{g_2 | P(g_2 = Op(g_1)) > 0 \wedge \mathbf{f}(\text{gpm}(g_2)) = \mathbf{f}(\text{gpm}(g_1))\}|}{|\{g_2 | P(g_2 = Op(g_1)) > 0\}|} \quad (1)$$

$$\forall G \subseteq \mathbb{G} \Rightarrow \nu(G) = \frac{1}{|G|} \sum_{g \in G} \nu(g) \quad (2)$$

5.2 Evolvability

Another metaphor in global optimization borrowed from biological systems is evolvability [52]. Wagner [225, 226] points out that this word has two uses in biology: According to Kirschner and Gerhart [118], a biological system is evolvable if it is able to generate heritable, selectable phenotypic variations. Such properties can then be evolved and changed by natural selection. In its second sense, a system is evolvable if it can acquire new characteristics via genetic change that help the organism(s) to survive and to reproduce. Theories about how the ability of generating adaptive variants has evolved have been proposed by Riedl [174], Altenberg [3], Wagner and Altenberg [227], and Bonner [26], amongst others. The idea of evolvability can be adopted for global optimization as follows:

Definition 2 (Evolvability). The evolvability of an optimization process in its current state defines how likely the search operations will lead to solution candidates with new (and eventually, better) objectives values.

The direct *probability of success* [170, 22], i.e., the chance that search operators produce offspring fitter than their parents, is also sometimes referred to as *evolvability* in the context of Evolutionary Algorithms [2, 5].

5.3 Neutrality: Problematic and Beneficial

The link between evolvability and neutrality has been discussed by many researchers. The evolvability of neutral parts of a fitness landscape depends on the optimization algorithm used. It is especially low for Hill Climbing and similar approaches, since the search operations cannot directly provide improvements or even changes. The optimization process then degenerates to a random walk, as illustrated in Fig. 1.f. The work of Beaudoin et al [17] on the ND fitness landscapes shows that neutrality may "destroy" useful information such as correlation.

Researchers in molecular evolution, on the other hand, found indications that the majority of mutations have no selective influence [77, 106] and that the transformation from genotypes to phenotypes is a many-to-one mapping. Wagner [226] states that neutrality in natural genomes is beneficial if it concerns only a subset of the properties peculiar to the offspring of a solution candidate while allowing meaningful modifications of the others. Toussaint and Igel [214] even go as far as declaring it a necessity for self-adaptation.

The theory of *punctuated equilibria* in biology introduced by Eldredge and Gould [67, 68] states that species experience long periods of evolutionary inactivity which are interrupted by sudden, localized, and rapid phenotypic evolutions [47, 134, 12]. It is assumed that the populations explore neutral layers during the time of stasis until, suddenly, a relevant change in a genotype leads to a better adapted phenotype [224] which then reproduces quickly.

The key to differentiating between "good" and "bad" neutrality is its degree ν in relation to the number of possible solutions maintained by the optimization algorithms. Smith et al [204] have used illustrative examples similar to Fig. 5 showing that a certain amount of neutral reproductions can foster the progress of optimization. In Fig. 5.a, basically the same scenario of premature convergence as in Fig. 3.a is depicted. The optimizer is drawn to a local optimum from which it cannot escape anymore. Fig. 5.b shows that a little shot of neutrality could form a bridge to the global optimum. The optimizer now has a chance to escape the smaller peak if it is able to find and follow that bridge, i.e., the evolvability of the system has increased. If this bridge gets wider, as sketched in Fig. 5.c, the chance of finding the global optimum increases as well. Of course, if the bridge gets too wide, the optimization process may end up in a scenario like in Fig. 1.f where it cannot find any direction. Furthermore, in this scenario we expect the neutral bridge to lead to somewhere useful, which is not necessarily the case in reality.

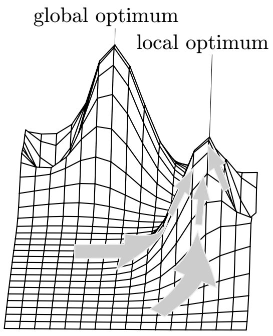


Fig. 5.a: Premature Convergence

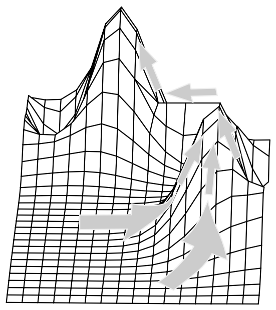


Fig. 5.b: Small Neutral Bridge

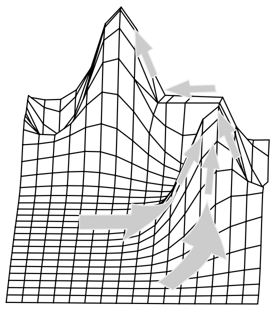


Fig. 5.c: Wide Neutral Bridge

Fig. 5 Possible positive influence of neutrality

Examples for neutrality in fitness landscapes are the ND family [17], the NKp [15] and NKq [155] models, and the Royal Road [139]. Another common instance of neutrality is *bloat* in Genetic Programming [131].

5.4 Redundancy: Problematic and Beneficial

Redundancy in the context of global optimization is a feature of the genotype-phenotype mapping and means that multiple genotypes map to the same phenotype, i.e., the genotype-phenotype mapping is not injective. The role of redundancy in the genome is as controversial as that of neutrality [230]. There exist many accounts of its positive influence on the optimization process. Shackleton et al [194, 197], for instance, tried to mimic desirable evolutionary properties of RNA folding [106]. They developed redundant genotype-phenotype mappings using voting (both, via uniform redundancy and via a non-trivial approach), Turing machine-like binary instructions, Cellular automata, and random Boolean networks [114]. Except for the trivial voting mechanism based on uniform redundancy, the mappings induced neutral networks which proved beneficial for exploring the problem space. Especially the last approach provided particularly good results [194, 197]. Possibly converse effects like epistasis (see Section 6) arising from the new genotype-phenotype mappings have not been considered in this study.

Redundancy can have a strong impact on the explorability of the problem space. When utilizing a one-to-one mapping, the translation of a slightly modified genotype will always result in a different phenotype. If there exists a many-to-one mapping between genotypes and phenotypes, the search operations can create offspring genotypes different from the parent which still translate to the same phenotype. The optimizer may now walk along a path through this neutral network. If many genotypes along this path can be modified to different offspring, many new solution candidates can be reached [197]. The experiments of Shipman et al [198, 196] additionally indicate that neutrality in the genotype-phenotype mapping can have positive effects.

Yet, Rothlauf [182] and Shackleton et al [194] show that simple uniform redundancy is not necessarily beneficial for the optimization process and may even slow it down. There is no use in introducing encodings which, for instance, represent each phenotypic bit with two bits in the genotype where `00` and `01` map to `0` and `10` and `11` map to `1`.

5.5 Summary

Different from ruggedness which is always bad for optimization algorithms, neutrality has aspects that may further as well as hinder the process of finding good solutions. Generally we can state that degrees of neutrality ν very close to 1 degenerate optimization processes to random walks. Some forms of neutral networks [14, 15, 27, 105, 208, 222, 223, 237] accompanied by low (nonzero) values of ν can improve the evolvability and hence, increase the chance of finding good solutions.

Adverse forms of neutrality are often caused by bad design of the search space or genotype-phenotype mapping. Uniform redundancy in the genome should be avoided where possible and the amount of neutrality in the search space should generally be limited.

6 Epistasis

6.1 Introduction

In biology, *epistasis* is defined as a form of interaction between different genes [163]. The term was coined by Bateson [16] and originally meant that one gene suppresses the phenotypical expression of another gene. In the context of statistical genetics, epistasis was initially called "epistacy" by Fisher [74]. According to Lush [132], the interaction between genes is epistatic if the effect on the fitness of altering one gene depends on the allelic state of other genes. This understanding of epistasis comes very close to another biological expression: *Pleiotropy*, which means that a single gene influences multiple phenotypic traits [239]. In global optimization, such fine-grained distinctions are usually not made and the two terms are often used more or less synonymously.

Definition 3 (Epistasis). In optimization, Epistasis is the dependency of the contribution of one gene to the value of the objective functions on the allelic state of other genes [4, 51, 153].

We speak of minimal epistasis when every gene is independent of every other gene. Then, the optimization process equals finding the best value for each gene and can most efficiently be carried out by a simple greedy search [51]. A problem is maximally epistatic when no proper subset of genes is independent of any other gene [205, 153]. Examples of problems with a high degree of epistasis are Kauffman's NK fitness landscape [113, 115], the p-Spin model [6], and the tunable model of Weise et al [232].

6.2 The Problem

As sketched in Fig. 6, epistasis has a strong influence on many of the previously discussed problematic features. If one gene can "turn off" or affect the expression of many other genes, a modification of this gene will lead to a large change in the features of the phenotype. Hence, the *causality* will be weakened and *ruggedness* ensues in the fitness landscape. On the other hand, subsequent changes to the "deactivated" genes may have no influence on the phenotype at all, which would then increase the degree of *neutrality* in the search space. Epistasis is mainly an aspect of the way in which we define the genome $\mathbb{G}$ and the genotype-phenotype mapping gpm. It should be avoided where possible.

Generally, epistasis and conflicting objectives in multi-objective optimization should be distinguished from each other. Epistasis as well as pleiotropy

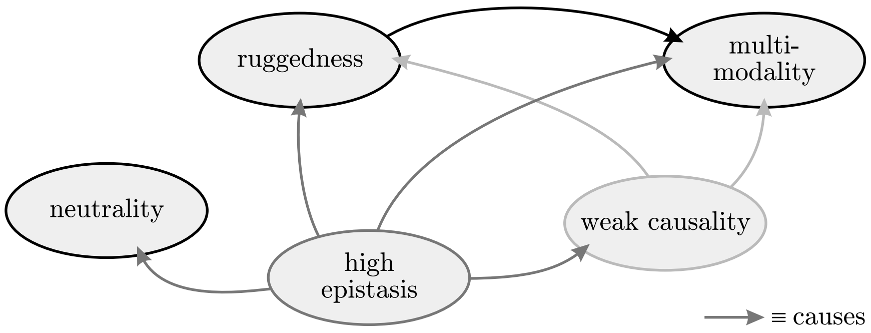


Fig. 6 The influence of epistasis on the fitness landscape

is a property of the influence of the elements (the genes) of the genotypes on the phenotypes. Objective functions can conflict *without* the involvement of any of these phenomena. We can, for example, define two objective functions $f_1(x) = x$ and $f_2(x) = -x$ which are clearly contradicting regardless of whether they are subject to maximization or minimization. Nevertheless, if the solution candidates x as well as the genotypes are simple real numbers and the genotype-phenotype mapping is simply an identity mapping, neither epistatic nor pleiotropic effects can occur.

Naudts and Verschoren [154] have shown for the special case of length-two binary string genomes that deceptiveness does not occur in situations with low epistasis and also that objective functions with high epistasis are not necessarily deceptive. Another discussion about different shapes of fitness landscapes under the influence of epistasis is given by Beerenwinkel et al [18].

6.3 Countermeasures

6.3.1 General

We have shown that epistasis is a root cause for multiple problematic features of optimization tasks. General countermeasures against epistasis can be divided into two groups. The symptoms of epistasis can be mitigated with the same methods which increase the chance of finding good solutions in the presence of ruggedness or neutrality – using larger populations and favoring explorative search operations. Epistasis itself is a feature which results from the choice of the search space structure, the search operations, and the genotype-phenotype mapping. Avoiding epistatic effects should be a major concern during their design. This can lead to a great improvement in the quality of the solutions produced by the optimization process [231]. General advice for good search space design is given in [84, 166, 178] and [229].

6.3.2 Linkage Learning

According to Winter et al [240], *linkage* is "the tendency for alleles of different genes to be passed together from one generation to the next" in genetics. This usually indicates that these genes are closely located in the same chromosome. In the context of Evolutionary Algorithms, this notation is not useful since identifying spatially close elements inside the genotypes is trivial. Instead, we are interested in alleles of different genes which have a joint effect on the fitness [150, 151].

Identifying these linked genes, i.e., learning their epistatic interaction, is very helpful for the optimization process. Such knowledge can be used to protect building blocks from being destroyed by the search operations. Finding approaches for *linkage learning* has become an especially popular discipline in the area of Evolutionary Algorithms with binary [99, 150, 46] and real [63] genomes. Two important methods from this area are the *messy Genetic*

Algorithm (mGA) by Goldberg et al [86] and the *Bayesian Optimization Algorithm* (BOA) [162, 41]. Module acquisition [8] may be considered as a similar effort in the area of Genetic Programming.

Let us take the mGA as an illustrative example for this family of approaches. By explicitly allowing the search operations to rearrange the genes in the genotypes, epistatically linked genes may get located closer to each other by time. As sketched in Fig. 7, the tighter the building blocks are packed, the less likely are they to be destroyed by crossover operations which usually split parent genotypes at randomly chosen points. Hence, the optimization process can strengthen the causality in the search space.

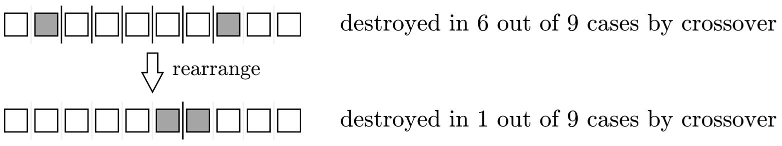

destroyed in 6 out of 9 cases by crossover
rearrange
destroyed in 1 out of 9 cases by crossover

Fig. 7 Two linked genes and their destruction probability under single-point crossover

7 Noise and Robustness

7.1 Introduction – Noise

In the context of optimization, three types of noise can be distinguished. The first form is noise in the training data used as basis for learning *(i)*. In many applications of machine learning or optimization where a model m for a given system is to be learned, data samples including the input of the system and its measured response are used for training. Some typical examples of situations where training data is the basis for the objective function evaluation are

- the usage of global optimization for building classifiers (for example for predicting buying behavior using data gathered in a customer survey for training),
- the usage of simulations for determining the objective values in Genetic Programming (here, the simulated scenarios correspond to training cases), and
- the fitting of mathematical functions to (x, y)-data samples (with artificial neural networks or symbolic regression, for instance).

Since no measurement device is 100% accurate and there are always random errors, noise is present in such optimization problems.

Besides inexactnesses and fluctuations in the input data of the optimization process, perturbations are also likely to occur during the application of its results. This category subsumes the other two types of noise: perturbations that may arise from inaccuracies in *(ii)* the process of realizing the solutions and *(iii)* environmentally induced perturbations during the applications of the products.

This issue can be illustrated using the process of developing the perfect tire for a car as an example. As input for the optimizer, all sorts of material coefficients and geometric constants measured from all known types of wheels and rubber could be available. Since these constants have been measured or calculated from measurements, they include a certain degree of noise and imprecision *(i)*.

The result of the optimization process will be the best tire construction plan discovered during its course and it will likely incorporate different materials and structures. We would hope that the tires created according to the plan will not fall apart if, accidently, an extra 0.0001% of a specific rubber component is used *(ii)*. During the optimization process, the behavior of many construction plans will be simulated in order to find out about their utility. When actually manufactured, the tires should not behave unexpectedly when used in scenarios different from those simulated *(iii)* and should instead be applicable in all driving scenarios likely to occur.

The effects of noise in optimization have been studied by various researchers; Miller and Goldberg [136, 137], Lee and Wong [125], and Gurin and Rastrigin [92] are some of them. Many global optimization algorithms and theoretical results have been proposed which can deal with noise. Some of them are, for instance, specialized

- Genetic Algorithms [75, 119, 188, 189, 217, 218],
- Evolution Strategies [11, 21, 96], and
- Particle Swarm Optimization [97, 161] approaches.

7.2 The Problem: Need for Robustness

The goal of global optimization is to find the global optima of the objective functions. While this is fully true from a theoretical point of view, it may not suffice in practice. Optimization problems are normally used to find good parameters or designs for components or plans to be put into action by human beings or machines. As we have already pointed out, there will always be noise and perturbations in practical realizations of the results of optimization.

Definition 4 (Robustness). A system in engineering or biology is *robust* if it is able to function properly in the face of genetic or environmental perturbations [225].

Therefore, a local optimum (or even a non-optimal element) for which slight deviations only lead to gentle performance degenerations is usually favored over a global optimum located in a highly rugged area of the fitness landscape [31]. In other words, local optima in regions of the fitness landscape with strong causality are sometimes better than global optima with weak causality. Of course, the level of this acceptability is application-dependent. Fig. 8 illustrates the issue of local optima which are robust vs. global optima which are not. More examples from the real world are:

- When optimizing the control parameters of an airplane or a nuclear power plant, the global optimum is certainly not used if a slight perturbation can have hazardous effects on the system [218].
- Wiesmann et al [234, 235] bring up the topic of manufacturing tolerances in multilayer optical coatings. It is no use to find optimal configurations if they only perform optimal when manufactured to a precision which is either impossible or too hard to achieve on a constant basis.
- The optimization of the decision process on which roads should be precautionary salted for areas with marginal winter climate is an example of the need for dynamic robustness. The global optimum of this problem is likely to depend on the daily (or even current) weather forecast and may therefore be constantly changing. Handa et al [98] point out that it is practically infeasible to let road workers follow a constantly changing plan and circumvent this problem by incorporating multiple road temperature settings in the objective function evaluation.
- Tsutsui et al [218, 217] found a nice analogy in nature: The phenotypic characteristics of an individual are described by its genetic code. During the interpretation of this code, perturbations like abnormal temperature, nutritional imbalances, injuries, illnesses and so on may occur. If the phenotypic features emerging under these influences have low fitness, the organism cannot survive and procreate. Thus, even a species with good genetic material will die out if its phenotypic features become too sensitive to perturbations. Species robust against them, on the other hand, will survive and evolve.

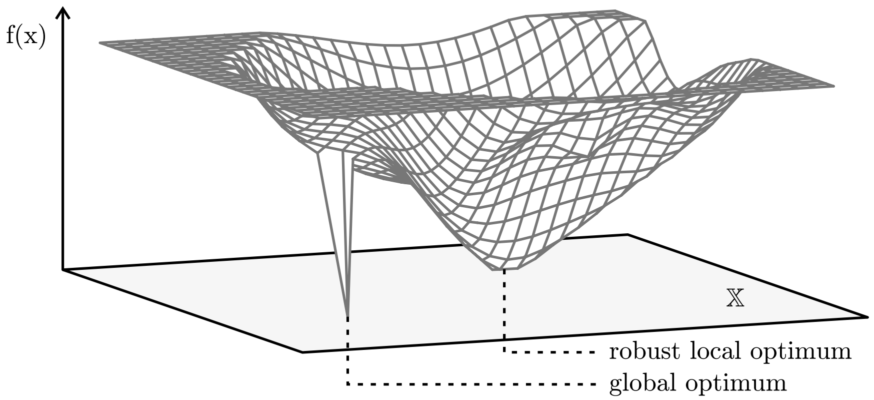


Fig. 8 A robust local optimum vs. a "unstable" global optimum

7.3 Countermeasures

For the special case where the problem space corresponds to the real vectors ($\mathbb{X} \subseteq \mathbb{R}^n$), several approaches for dealing with the problem of robustness have been developed. Inspired by Taguchi methods[6] [209], possible disturbances are represented by a vector $\boldsymbol{\delta} = (\delta_1, \delta_2, .., \delta_n)^T, \delta_i \in \mathbb{R}$ in the method of Greiner [87, 88]. If the distribution and influence of the δ_i are known, the objective function $f(\mathbf{x}) : \mathbf{x} \in \mathbb{X}$ can be rewritten as $\tilde{f}(\mathbf{x}, \boldsymbol{\delta})$ [235]. In the special case where $\boldsymbol{\delta}$ is normally distributed, this can be simplified to $\tilde{f}\Big((x_1 + \delta_1, x_2 + \delta_2, .., x_n + \delta_n)^T\Big)$. It would then make sense to sample the probability distribution of $\boldsymbol{\delta}$ a number of t times and to use the mean values of $\tilde{f}(\mathbf{x}, \boldsymbol{\delta})$ for each objective function evaluation during the optimization process. In cases where the optimal value y of the objective function f is known, Equation 3 can be minimized. This approach is also used in the work of Wiesmann et al [234, 235] and basically turns the optimization algorithm into something like a maximum likelihood estimator.

$$f'(\mathbf{x}) = \frac{1}{t} \sum_{i=1}^{t} \Big(y - \tilde{f}(\mathbf{x}, \boldsymbol{\delta}_i)\Big)^2 \tag{3}$$

This method corresponds to using multiple, different training scenarios during the objective function evaluation in situations where $\mathbb{X} \nsubseteq \mathbb{R}^n$. By adding random noise and artificial perturbations to the training cases, the chance of obtaining robust solutions which are stable when applied or realized under noisy conditions can be increased.

8 Overfitting and Oversimplification

In all scenarios where optimizers evaluate some of the objective values of the solution candidates by using training data, two additional phenomena with negative influence can be observed: overfitting and oversimplification.

8.1 Overfitting

8.1.1 The Problem

Definition 5 (Overfitting). Overfitting is the emergence of an overly complicated model (solution candidate) in an optimization process resulting from the effort to provide the best results for as much of the available training data as possible [64, 80, 190, 202].

A model (solution candidate) $m \in \mathbb{X}$ created with a finite set of training data is considered to be overfitted if a less complicated, alternative model

[6] `http://en.wikipedia.org/wiki/Taguchi_methods` [accessed 2008-07-19]

$m' \in \mathbb{X}$ exists which has a smaller error for the set of all possible (maybe even infinitely many), available, or (theoretically) producible data samples. This model m' may, however, have a larger error in the training data.

The phenomenon of overfitting is best known and can often be encountered in the field of artificial neural networks or in curve fitting [124, 128, 181, 191, 211]. The latter means that we have a set A of n training data samples (x_i, y_i) and want to find a function f that represents these samples as well as possible, i.e., $f(x_i) = y_i \;\forall\, (x_i, y_i) \in A$.

There exists exactly one polynomial of the degree $n-1$ that fits to each such training data and goes through all its points. Hence, when only polynomial regression is performed, there is exactly one perfectly fitting function of minimal degree. Nevertheless, there will also be an infinite number of polynomials with a higher degree than $n-1$ that also match the sample data perfectly. Such results would be considered as overfitted.

In Fig. 9, we have sketched this problem. The function $f_1(x) = x$ shown in Fig. 9.b has been sampled three times, as sketched in Fig. 9.a. There exists no other polynomial of a degree of two or less that fits to these samples than f_1. Optimizers, however, could also find overfitted polynomials of a higher degree such as f_2 which also match the data, as shown in Fig. 9.c. Here, f_2 plays the role of the overly complicated model m which will perform as good as the simpler model m' when tested with the training sets only, but will fail to deliver good results for all other input data.

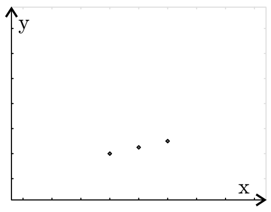


Fig. 9.a: Three sample points of f_1

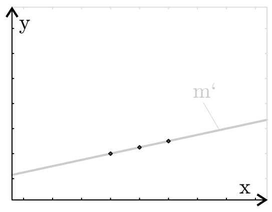


Fig. 9.b: $m' \equiv f_1(x) = x$

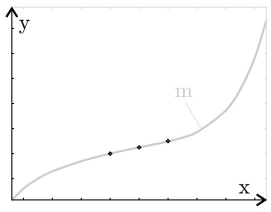


Fig. 9.c: $m \equiv f_2(x)$

Fig. 9 Overfitting due to complexity

A very common cause for overfitting is noise in the sample data. As we have already pointed out, there exists no measurement device for physical processes which delivers perfect results without error. Surveys that represent the opinions of people on a certain topic or randomized simulations will exhibit variations from the true interdependencies of the observed entities, too. Hence, data samples based on measurements will always contain some noise.

In Fig. 10 we have sketched how such noise may lead to overfitted results. Fig. 10.a illustrates a simple physical process obeying some quadratic equation. This process has been measured using some technical equipment

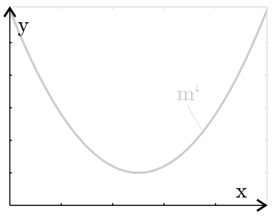


Fig. 10.a: The original physical process

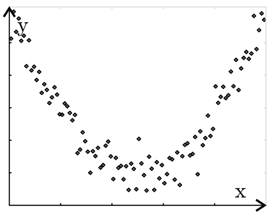


Fig. 10.b: The measurement/training data

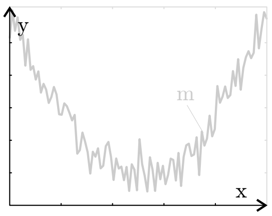


Fig. 10.c: The overfitted result

Fig. 10 Fitting noise

and the 100 noisy samples depicted in Fig. 10.b has been obtained. Fig. 10.c shows a function resulting from an optimization that fits the data perfectly. It could, for instance, be a polynomial of degree 99 that goes right through all the points and thus, has an error of zero. Although being a perfect match to the measurements, this complicated model does not accurately represent the physical law that produced the sample data and will not deliver precise results for new, different inputs.

From the examples we can see that the major problem that results from overfitted solutions is the loss of generality.

Definition 6 (Generality). A solution of an optimization process is general if it is not only valid for the sample inputs $a_1, a_2, \dots, a_n$ which were used for training during the optimization process, but also for different inputs $a \neq a_i \; \forall i : 0 < i \leq n$ if such inputs a exist.

8.1.2 Countermeasures

There exist multiple techniques that can be utilized in order to prevent overfitting to a certain degree. It is most efficient to apply multiple such techniques together in order to achieve best results.

A very simple approach is to restrict the problem space $\mathbb{X}$ in a way that only solutions up to a given maximum complexity can be found. In terms of function fitting, this could mean limiting the maximum degree of the polynomials to be tested. Furthermore, the functional objective functions which solely concentrate on the error of the solution candidates should be augmented by penalty terms and non-functional objective functions putting pressure in the direction of small and simple models [64, 116].

Large sets of sample data, although slowing down the optimization process, may improve the generalization capabilities of the derived solutions. If arbitrarily many training datasets or training scenarios can be generated, there are two approaches which work against overfitting:

1. The first method is to use a new set of (randomized) scenarios for each evaluation of a solution candidate. The resulting objective values may differ

largely even if the same individual is evaluated twice in a row, introducing incoherence and ruggedness into the fitness landscape.

2. At the beginning of each iteration of the optimizer, a new set of (randomized) scenarios is generated which is used for all individual evaluations during that iteration. This method leads to objective values which can be compared without bias.

In both cases it is helpful to use more than one training sample or scenario per evaluation and to set the resulting objective value to the average (or better median) of the outcomes. Otherwise, the fluctuations of the objective values between the iterations will be very large, making it hard for the optimizers to follow a stable gradient for multiple steps.

Another simple method to prevent overfitting is to limit the runtime of the optimizers [190]. It is commonly assumed that learning processes normally first find relatively general solutions which subsequently begin to overfit because the noise "is learned", too.

For the same reason, some algorithms allow to decrease the rate at which the solution candidates are modified by time. Such a decay of the learning rate makes overfitting less likely.

If only one finite set of data samples is available for training/optimization, it is common practice to separate it into a set of training data A_t and a set of test cases A_c. During the optimization process, only the training data is used. The resulting solutions are tested with the test cases afterwards. If their behavior is significantly worse when applied to A_c than when applied to A_t, they are probably overfitted.

The same approach can be used to detect when the optimization process should be stopped. The best known solution candidates can be checked with the test cases in each iteration without influencing their objective values which solely depend on the training data. If their performance on the test cases begins to decrease, there are no benefits in letting the optimization process continue any further.

8.2 Oversimplification

8.2.1 The Problem

Oversimplification (also called overgeneralization) is the opposite of overfitting. Whereas overfitting denotes the emergence of overly-complicated solution candidates, oversimplified solutions are not complicated enough. Although they represent the training samples used during the optimization process seemingly well, they are rough overgeneralizations which fail to provide good results for cases not part of the training.

A common cause for oversimplification is sketched in Fig. 11: The training sets only represent a fraction of the set of possible inputs. As this is normally the case, one should always be aware that such an incomplete coverage may fail to represent some of the dependencies and characteristics of the data,

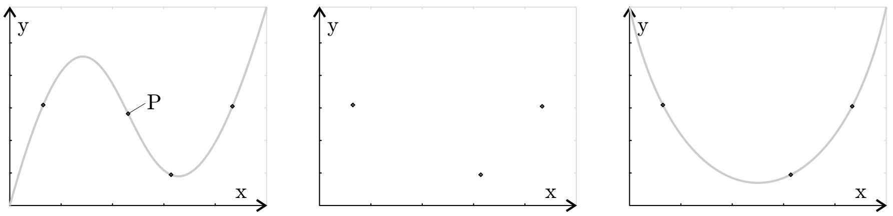


Fig. 11.a: The "real system" and the points describing it

Fig. 11.b: The sampled training data

Fig. 11.c: The oversimplified result

Fig. 11 Oversimplification

which then may lead to oversimplified solutions. Another possible reason is that ruggedness, deceptiveness, too much neutrality, or high epistasis in the fitness landscape may lead to premature convergence and prevent the optimizer from surpassing a certain quality of the solution candidates. It then cannot completely adapt them even if the training data perfectly represents the sampled process. A third cause is that a problem space which does not include the correct solution was chosen.

Fig. 11.a shows a cubic function. Since it is a polynomial of degree three, four sample points are needed for its unique identification. Maybe not knowing this, only three samples have been provided in Fig. 11.b. By doing so, some vital characteristics of the function are lost. Fig. 11.c depicts a square function – the polynomial of the lowest degree that fits exactly to these samples. Although it is a perfect match, this function does not touch any other point on the original cubic curve and behaves totally differently at the lower parameter area.

However, even if we had included point P in our training data, it would still be possible that the optimization process would yield Fig. 11.c as a result. Having training data that correctly represents the sampled system does not mean that the optimizer is able to find a correct solution with perfect fitness – the other, previously discussed problematic phenomena can prevent it from doing so. Furthermore, if it was not known that the system which was to be modeled by the optimization process can best be represented by a polynomial of the third degree, one could have limited the problem space $\mathbb{X}$ to polynomials of degree two and less. Then, the result would likely again be something like Fig. 11.c, regardless of how many training samples are used.

8.2.2 Countermeasures

In order to counter oversimplification, its causes have to be mitigated. Generally, it is not possible to have training scenarios which cover the complete input space of the evolved programs. By using multiple scenarios for each individual evaluation, the chance of missing important aspects is decreased. These scenarios can be replaced with new, randomly created ones in each generation, which will decrease this chance even more. The problem space, i.e., the representation of the solution candidates, should further be chosen in a way which allows constructing a correct solution to the problem defined. Then again, releasing too many constraints on the solution structure increases the risk of overfitting and thus, careful proceeding is recommended.

9 Multi-objective Optimization

9.1 Introduction

Many optimization problems in the real world have k possibly contradictory objectives f_i which must be optimized simultaneously. Furthermore, the solutions must satisfy m inequality constraints g and p equality constraints h. A solution candidate x is *feasible*, if and only if $g_i(x) \geq 0 \; \forall i = 1, 2, .., m$ and $h_i(x) = 0 \; \forall i = 1, 2, .., p$ holds. A *multi-objective optimization problem* (MOP) can then be formally defined as follows:

Definition 7 (MOP). Find a solution candidate $x^\star$ in $\mathbb{X}$ which minimizes (or maximizes) the vector function $\mathbf{f}(x^\star) = (f_i(x^\star), f_2(x^\star), .., f_k(x^\star))^T$ and is feasible, (i.e., satisfies the m inequality constraints $g_i(x^\star) \geq 0 \; \forall i = 1, 2, .., m$, the p equality constraints $h_i(x^\star) = 0 \; \forall i = 1, 2, .., p$).

As in single-objective optimization, nature-inspired algorithms are popular techniques to solve such problems. The fact that there are two or more objective functions implies additional difficulties. Due to the contradictory feature of the functions in a MOP and the fact that there exists no total order in $\mathbb{R}^n$ for $n > 1$, the notions of "better than" and "optimum" have to be redefined. When comparing any two solutions x_1 and x_2, solution x_1 can have a better value in objective f_i, i.e., $f_i(x_1) < f_i(x_2)$, while solution x_2 can have a better value in objective f_j. The concepts commonly used here are *Pareto dominance* and *Pareto optimality.*

Definition 8 (Pareto Dominance). In the context of multi-objective global optimization, a solution candidate x_1 is said to *dominate* another solution candidate x_2 (denoted by $x_1 \preccurlyeq x_2$) if and only if $\mathbf{f}(x_1)$ is partially less than $\mathbf{f}(x_2)$, i.e., $\forall i \in \{1, .., k\} \;\; f_i(x_1) \leq f_i(x_2) \; \wedge \; \exists j \in \{1, .., k\} : \; f_j(x_1) < f_j(x_2)$.

The dominance notion allows us to assume that if solution x_1 dominates solution x_2, then x_1 is preferable to x_2. If both solution are non-dominated (such as candidate ① and ② in Fig. 12), some additional criteria have to be used to choose one of them.

Definition 9 (Pareto Optimality). A feasible point $x^\star \in \mathbb{X}$ is *Pareto-optimal* if and only if there is no feasible $x_b \in \mathbb{X}$ with $x_b \preccurlyeq x^\star$.

This definition states that $x^\star$ is Pareto-optimal if there is no other feasible solution x_b which would improve some criterion without causing a simultaneous worsening in at least one other criterion. The solution to a MOP,

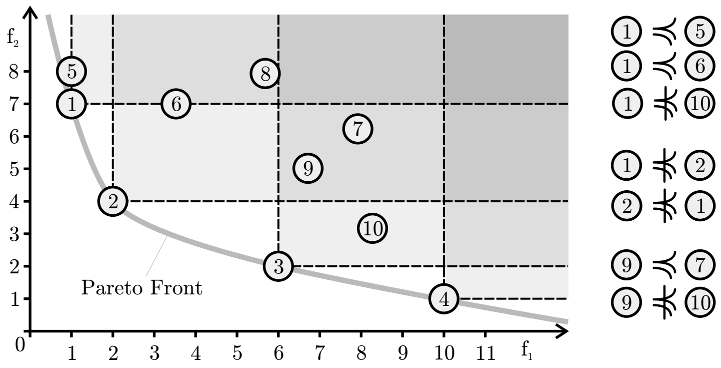


Fig. 12 Some examples for the dominance relation

considering Pareto optimality, is the set of feasible, non-dominated solutions which is known as *Pareto-optimal set*:

Definition 10 (Pareto-Optimal Set). For a given MOP $\mathbf{f}(x)$, the Pareto optimal set is defined as $\mathcal{P}^\star = \{x^\star \in \mathbb{X} | \neg\exists x \in \mathbb{X} : x \preccurlyeq x^\star\}$.

When the solutions in the Pareto-optimal set are plotted in the objective space (as sketched in Fig. 12), they are collectively known as the *Pareto front*:

Definition 11 (Pareto Front). For a given MOP $\mathbf{f}(x)$ and its Pareto-optimal set $\mathcal{P}^\star$, the Pareto front is defined as $\mathcal{PF}^\star = \{\mathbf{f}(x) \,| x \in \mathcal{P}^\star\}$.

Obtaining the Pareto front of a MOP is the main goal of multi-objective optimization. In a real scenario, the solutions in the Pareto front are sent to an expert in the MOP, the decision maker, who will be responsible for choosing the best tradeoff solution among all of them. Fig. 13 depicts the Pareto front of a bi-objective MOP. In a real problem example, f_1 could

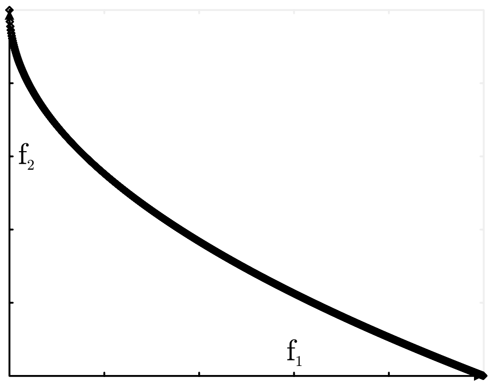


Fig. 13 Example of Pareto front of a bi-objective MOP

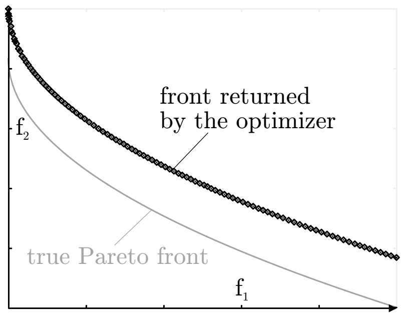


Fig. 14.a: Bad Convergence and Good Spread

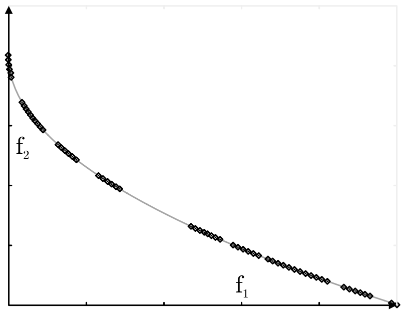


Fig. 14.b: Good Convergence and Bad Spread

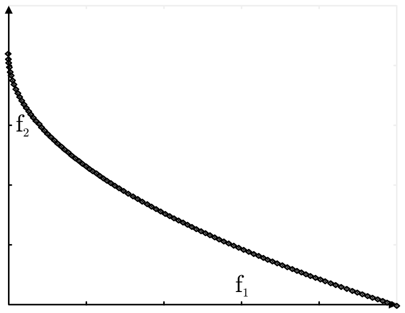


Fig. 14.c: Good Convergence and Spread

Fig. 14 Pareto front approximation sets

represent the time required by a car to cover a given distance, while f_2 could be the fuel consumption.

The Pareto front of a MOP can contain a large (possibly infinite) number of points. Usually, the goal of optimization is to obtain a fixed-size set of solutions called *Pareto front approximation set.* Population-based algorithms, such as Genetic Algorithms, are very popular to solve MOPs because they can provide an approximation set in a single run.

Given that the goal is to find a Pareto front approximation set, two issues arise. First, the optimization process should *converge* to the true Pareto front and return solutions as close to it as possible. Second, they should be uniformly spread along this front.

Let us examine the three fronts included in Fig. 14. The first picture (Fig. 14.a) shows an approximation set having a very good spread[7] of

[7] In MO optimization, this property is usually called *diversity.* In order to avoid confusion with the (related) diversity property from Section 2.3, we here use the term *spread* instead.

solutions, but the points are far away from the true Pareto front. Such results are not attractive because they do not provide Pareto-optimal solutions. The second example (Fig. 14.b) contains a set of solutions which are very close to the true Pareto front but cover it only partially, so the decision maker could lose important trade-off solutions. Finally, the front depicted in Fig. 14.c has the two desirable properties of good convergence and spread.

9.2 The Problem

Features such as multi-modality, deceptiveness, or epistasis found in single-objective optimization also affect MOPs, making them more difficult to solve. However, there are some characteristics that are particular to MOPs. Here we comment on two of them: geometry and dimensionality.

The Pareto front in Fig. 13 has a convex geometry, but there are other different shapes as well. In Fig. 15 we show some examples, including non-convex (concave), disconnected, linear, and non-uniformly distributed Pareto

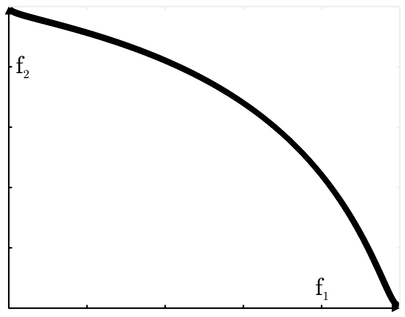


Fig. 15.a: Non-Convex (Concave)

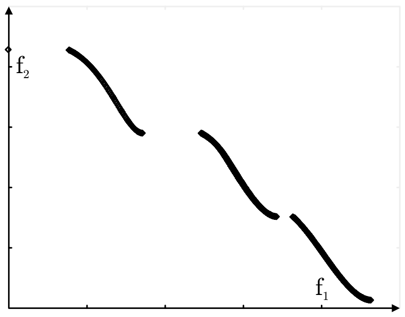


Fig. 15.b: Disconnected

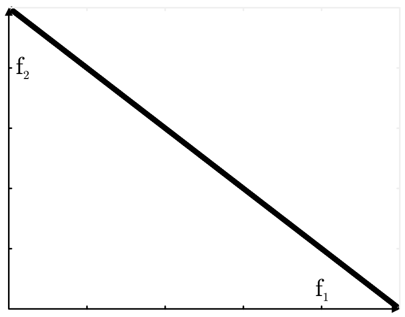


Fig. 15.c: linear

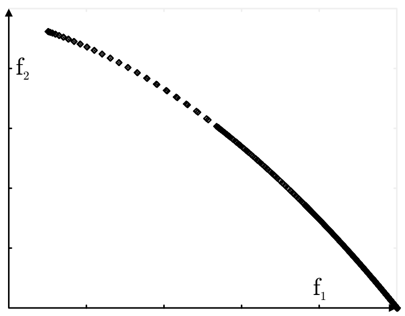


Fig. 15.d: Non-Uniformly Distributed

Fig. 15 Examples of Pareto fronts

fronts. Besides Pareto optimization, there is a wide variety of other concepts for defining what optima are in the presence of multiple objective functions [45]. The simplest approach is maybe to use a weighted sum of all objective values and set $v(x) = \sum_{i=1}^{k} f_i(x)$. Then the optima would be the element(s) $x^\star$ with $\neg\exists x \in \mathbb{X} : v(x) < v(x^\star)$. However, an optimization process driven by such a linear aggregating function will not find portions of Pareto fronts with non-convex geometry as shown by Das and Dennis [50].

Many studies in the literature consider mainly bi-objective MOPs. As a consequence, many algorithms are designed to deal with that kind of problems. However, MOPs having a higher number of objective functions are common in practice, leading to the so-called *many-objective optimization* [165], which is currently a hot research topic. Most of the optimization algorithms applied today utilize the Pareto dominance relation. When the dimension of the MOPs increases, the majority of solution candidates are non-dominated. As a consequence, traditional nature-inspired algorithms have to be redesigned.

9.3 Countermeasures

In order to obtain an accurate approximation to the true Pareto front, many nature-inspired multi-objective algorithms apply a fitness assignment scheme based on the concept of Pareto dominance, as commented before. For example, NSGA-II [61, 62], the most well-known multi-objective technique, assigns to each solution a rank depending on the number of solutions dominating it. Thus, solutions with rank 1 are non-dominated, solutions with rank 2 are dominated by one solution, and so on. Other algorithms, such as SPEA2 [247, 248] introduce the concept of strength, which is similar to the ranking but also considers the number of dominated solutions.

While the use of Pareto-based ranking methods allows the techniques to search in the direction of finding approximations with good convergence, additional strategies are needed to promote spread. The most commonly adopted approach is to include a kind of density estimator in order to select those solutions which are in the less crowded regions of the objective space. Thus, NSGA-II employs the crowding distance [61] and SPEA2 the distance to the k-nearest neighbor [62].

9.4 Constraint Handling

How the constraints mentioned in Definition 7 are handled is a whole research area in itself with roots in single-objective optimization. Maybe one of the most popular approach for dealing with constraints goes back to Courant [48] who introduced the idea of *penalty functions* [73, 44, 201] in 1943: Consider, for instance, the term $f'(x) = f(x) + v\,[h(x)]^2$ where f is the original objective function, h is an equality constraint, and $v > 0$. If f' is minimized, an infeasible individual will always have a worse fitness than a feasible one with the same objective values.

Besides such static penalty functions, *dynamic* terms incorporating the generation counter [111, 157] or *adaptive* approaches utilizing additional population statistics [95, 199] have been proposed. Rigorous discussions on penalty functions have been contributed by Fiacco and McCormick [73] and Smith and Coit [201].

During the last fifteen years, many approaches have been developed which incorporate constraint handling and multi-objectivity. Instead of using penalty terms, Pareto ranking can also be extended by additionally comparing individuals according to their feasibility, for instance. Examples for this approach are the Method of Inequalities (MOI) of Zakian [245] as used by Pohlheim [164] and the Goal Attainment method defined in [76]. Deb [56, 58] even suggested to simply turn constraints into objective functions in his MOEA version of Goal Programming.

10 Dynamically Changing Fitness Landscape

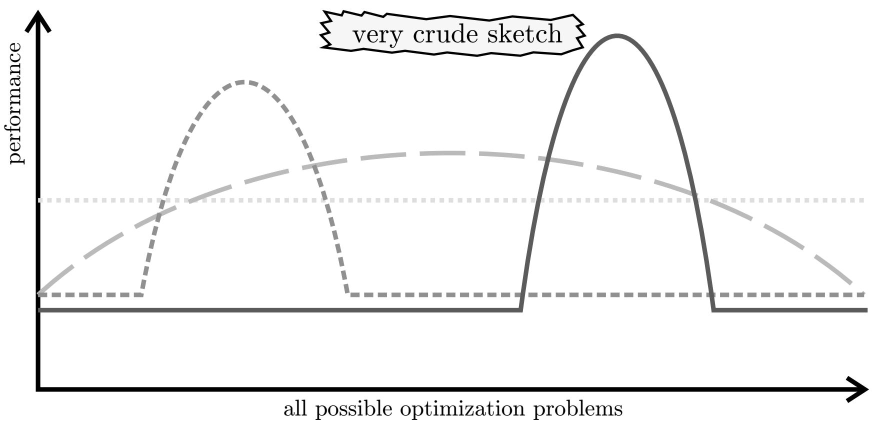
It should also be mentioned that there exist problems with dynamically changing fitness landscapes [33, 32, 36, 147, 173]. The task of an optimization algorithm is, then, to provide solution candidates with momentarily optimal objective values for each point in time. Here we have the problem that an optimum in iteration t will possibly not be an optimum in iteration $t + 1$ anymore.

The moving peaks benchmarks by Branke [33, 32] and Morrison and De Jong [147] are good examples for dynamically changing fitness landscapes. Such problems with dynamic characteristics can, for example, be tackled with special forms [244] of

- Evolutionary Algorithms [9, 34, 35, 145, 146, 216, 236],
- Genetic Algorithms [83, 119, 142, 143, 144],
- Particle Swarm Optimization [23, 42, 43, 126, 160],
- Differential Evolution [135, 243], and
- Ant Colony Optimization [90, 91]

11 The No Free Lunch Theorem

By now, we know the most important problems that can be encountered when applying an optimization algorithm to a given problem. Furthermore, we have seen that it is arguable what actually an optimum is if multiple criteria are optimized at once. The fact that there is most likely no optimization method that can outperform all others on all problems can, thus, easily be accepted. Instead, there exist a variety of optimization methods specialized



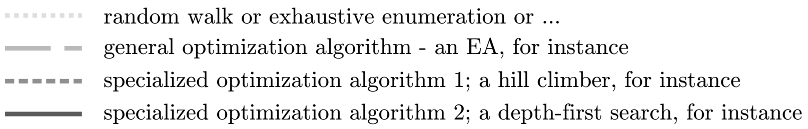


Fig. 16 A visualization of the No Free Lunch Theorem

in solving different types of problems. There are also algorithms which deliver good results for many different problem classes, but may be outperformed by highly specialized methods in each of them.

These facts have been formalized by Wolpert and Macready [241, 242] in their *No Free Lunch Theorems* (NFL) for search and optimization algorithms. Wolpert and Macready [242] focus on single-objective optimization and prove that the sum of the values of any performance measure (such as the objective value of the best solution candidate discovered until a time step m) over all possible objective functions f is always identical for all optimization algorithms.

From this theorem, we can immediately follow that, in order to outperform the optimization method a_1 in one optimization problem, the algorithm a_2 will necessarily perform worse in another. Fig. 16 visualizes this issue. The higher the value of the performance measure illustrated there, the faster will the corresponding problem be solved. The figure shows that general optimization approaches (like Evolutionary Algorithms) can solve a variety of problem classes with reasonable performance. Hill Climbing approaches, for instance, will be much faster than Evolutionary Algorithms if the objective functions are steady and monotonous, that is, in a smaller set of optimization tasks. Greedy search methods will perform fast on all problems with matroid structure. Evolutionary Algorithms will most often still be able to solve these problems, it just takes them longer to do so. The performance of Hill Climbing and greedy approaches degenerates in other classes of optimization tasks as a trade-off for their high utility in their “area of expertise”.

One interpretation of the No Free Lunch Theorem is that it is impossible for any optimization algorithm to outperform random walks or exhaustive enumerations on all possible problems. For every problem where a given method leads to good results, we can construct a problem where the same method has exactly the opposite effect (see Section 4). As a matter of fact, doing so is even a common practice to find weaknesses of optimization algorithms and to compare them with each other.

Another interpretation is that every useful optimization algorithm utilizes some form of problem-specific knowledge. Radcliffe [167] states that without such knowledge, search algorithms cannot exceed the performance of simple enumerations. Incorporating knowledge starts with relying on simple assumptions like "if x is a good solution candidate, than we can expect other good solution candidates in its vicinity", i.e., strong causality. The more (correct) problem specific knowledge is integrated (correctly) into the algorithm structure, the better will the algorithm perform. On the other hand, knowledge correct for one class of problems is, quite possibly, misleading for another class. In reality, we use optimizers to solve a given set of problems and are not interested in their performance when (wrongly) applied to other classes.

Today, there exists a wide range of work on No Free Lunch Theorems for many different aspects of machine learning. The website `http://www.no-free-lunch.org/`[8] gives a good overview about them. Further summaries and extensions have been provided by Köppen et al [121] and Igel and Toussaint [108, 109]. Radcliffe and Surry [168] discuss the NFL in the context of Evolutionary Algorithms and the representations used as search spaces. The No Free Lunch Theorem is furthermore closely related to the Ugly Duckling Theorem proposed by Watanabe [228] for classification and pattern recognition.

12 Concluding Remarks

The subject of this introductory chapter was the question about what makes optimization problems hard, especially for metaheuristic approaches. We have discussed numerous different phenomena which can affect the optimization process and lead to disappointing results. If an optimization process has converged prematurely, it has been trapped in a non-optimal region of the search space from which it cannot "escape" anymore (Section 2). Ruggedness (Section 3) and deceptiveness (Section 4) in the fitness landscape, often caused by epistatic effects (Section 6), can misguide the search into such a region. Neutrality and redundancy (Section 5) can either slow down optimization because the application of the search operations does not lead to a gain in information or may also contribute positively by creating neutral networks from which the search space can be explored and local optima can be escaped

[8] Accessed: 2008-03-28

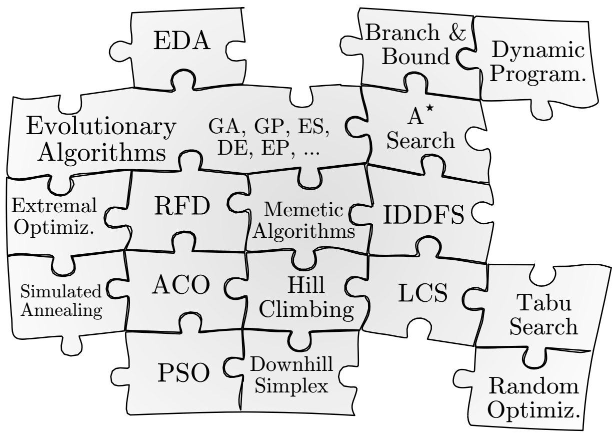


Fig. 17 The puzzle of optimization algorithms

from. The solutions that are derived, even in the presence of noise, should be robust (Section 7). Also, they should neither be too general (oversimplification, Section 8.2) nor too specifically aligned only to the training data (overfitting, Section 8.1). Furthermore, many practical problems are multi-objective, i.e., involve the optimization of more than one criterion at once (Section 9), or concern objectives which may change over time (Section 10).

In the previous section, we discussed the No Free Lunch Theorem and argued that it is not possible to develop the *one* optimization algorithm, the problem-solving machine which can provide us with near-optimal solutions in short time for every possible optimization task. This must sound very depressing for everybody new to this subject.

Actually, quite the opposite is the case, at least from the point of view of a researcher. The No Free Lunch Theorem means that there will always be new ideas, new approaches which will lead to better optimization algorithms to solve a given problem. Instead of being doomed to obsolescence, it is far more likely that most of the currently known optimization methods have at least one niche, one area where they are excellent. It also means that it is very likely that the "puzzle of optimization algorithms" will never be completed. There will always be a chance that an inspiring moment, an observation in nature, for instance, may lead to the invention of a new optimization algorithm which performs better in some problem areas than all currently known ones.

Acknowledgements. We gratefully acknowledge comments on early drafts of this chapter by Peter J Bentley and Patrick Siarry. The last author acknowledges support from the "Consejería de Innovación, Ciencia y Empresa", Junta de Andalucía under contract `P07-TIC-03044 DIRICOM` project, `http://diricom.lcc.uma.es`.

References

1. Ackley, D.H.: A connectionist machine for genetic hillclimbing. The Springer International Series in Engineering and Computer Science, vol. 28. Kluwer Academic Publishers, Dordrecht (1987)
2. Altenberg, L.: The schema theorem and price's theorem. Foundations of Genetic Algorithms 3, 23–49 (1994)
3. Altenberg, L.: Genome growth and the evolution of the genotype-phenotype map. In: Evolution and Biocomputation – Computational Models of Evolution, pp. 205–259. Springer, Heidelberg (1995)
4. Altenberg, L.: Nk fitness landscapes. In: Handbook of Evolutionary Computation, ch.. B2.7.2. Oxford University Press, Oxford (1996)
5. Altenberg, L.: Fitness distance correlation analysis: An instructive counterexample. In: Proceedings of the International Conference on Genetic Algorithms, ICGA, pp. 57–64 (1997)
6. Amitrano, C., Peliti, L., Saber, M.: Population dynamics in a spin-glass model of chemical evolution. Journal of Molecular Evolution 29(6), 513–525 (1989)
7. Amor, H.B., Rettinger, A.: Intelligent exploration for genetic algorithms: Using self-organizing maps in evolutionary computation. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 1531–1538 (2005) doi:10.1145/1068009.1068250
8. Angeline, P.J., Pollack, J.: Evolutionary module acquisition. In: The Second Annual Conference on Evolutionary Programming, Evolutionary Programming Society, pp. 154–163 (1993)
9. Aragón, V.S., Esquivel, S.C.: An evolutionary algorithm to track changes of optimum value locations in dynamic environments. Journal of Computer Science & Technology (JCS&T) 4(3), 127–133 (2004); invited paper
10. Bachmann, P.G.H.: Die Analytische Zahlentheorie / Dargestellt von Paul Bachmann, Zahlentheorie: Versuch einer Gesamtdarstellung dieser Wissenschaft in ihren Haupttheilen, vol. Zweiter Theil. B. G. Teubner, Leipzig, Germany (1894)
11. Bäck, T., Hammel, U.: Evolution strategies applied to perturbed objective functions. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 1, pp. 40–45 (1994) doi:10.1109/ICEC.1994.350045
12. Bak, P., Sneppen, K.: Punctuated equilibrium and criticality in a simple model of evolution. Physical Review Letters 71, 4083–4086 (1993)
13. Baldwin, J.M.: A new factor in evolution. The American Naturalist 30, 441–451 (1896)
14. Barnett, L.: Tangled webs: Evolutionary dynamics on fitness landscapes with neutrality. Master's thesis, School of Cognitive Science, University of East Sussex, Brighton, UK (1997)
15. Barnett, L.: Ruggedness and neutrality – the nkp family of fitness landscapes. In: Artificial Life VI: Proceedings of the sixth international conference on Artificial life, pp. 18–27 (1998)
16. Bateson, W.: Mendel's Principles of Heredity. Cambridge University Press, Cambridge (1909)
17. Beaudoin, W., Verel, S., Collard, P., Escazut, C.: Deceptiveness and neutrality the nd family of fitness landscapes. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 507–514 (2006) doi:10.1145/1143997.1144091

18. Beerenwinkel, N., Pachter, L., Sturmfels, B.: Epistasis and shapes of fitness landscapes. Eprint arXiv:q-bio/0603034 (Quantitative Biology, Populations and Evolution) (accessed 2007-08-05) (2006), `http://arxiv.org/abs/q-bio.PE/0603034`
19. Bergman, A., Feldman, M.W.: Recombination dynamics and the fitness landscape. Physica D: Nonlinear Phenomena 56, 57–67 (1992)
20. Bethke, A.D.: Genetic algorithms as function optimizers. PhD thesis, University of Michigan, Ann Arbor, MI, USA (1980)
21. Beyer, H.-G.: Toward a theory of evolution strategies: Some asymptotical results from the $(1, +\lambda)$-theory. Evolutionary Computation 1(2), 165–188 (1993)
22. Beyer, H.-G.: Toward a theory of evolution strategies: The (μ, λ)-theory. Evolutionary Computation 2(4), 381–407 (1994)
23. Blackwell, T.: Particle swarm optimization in dynamic environments. In: Evolutionary Computation in Dynamic and Uncertain Environments, ch. 2, pp. 29–52. Springer, Heidelberg (2007)
24. Bledsoe, W.W., Browning, I.: Pattern recognition and reading by machine. In: Proceedings of the Eastern Joint Computer Conference (EJCC) – Papers and Discussions Presented at the Joint IRE - AIEE - ACM Computer Conference, pp. 225–232 (1959)
25. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys 35(3), 268–308 (2003)
26. Bonner, J.T.: On Development: The Biology of Form, new ed edn. Commonwealth Fund Publications, Harvard University Press (1974)
27. Bornberg-Bauer, E., Chan, H.S.: Modeling evolutionary landscapes: Mutational stability, topology, and superfunnels in sequence space. Proceedings of the National Academy of Science of the United States of Americs (PNAS) – Biophysics 96(19), 10689–10694 (1999)
28. Bosman, P.A.N., Thierens, D.: Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. International Journal Approximate Reasoning 31(3), 259–289 (2002)
29. Bosman, P.A.N., Thierens, D.: A thorough documentation of obtained results on real-valued continuous and combinatorial multi-objective optimization problems using diversity preserving mixture-based iterated density estimation evolutionary algorithms. Tech. Rep. UU-CS-2002-052, Institute of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands (2002)
30. Brameier, M.F., Banzhaf, W.: Explicit control of diversity and effective variation distance in linear genetic programming. In: Foster, J.A., Lutton, E., Miller, J., Ryan, C., Tettamanzi, A.G.B. (eds.) EuroGP 2002. LNCS, vol. 2278, pp. 37–49. Springer, Heidelberg (2002)
31. Branke, J.: Creating robust solutions by means of evolutionary algorithms. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 119–128. Springer, Heidelberg (1998)
32. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 3, pp. 1875–1882 (1999) doi:10.1109/CEC.1999.785502
33. Branke, J.: The moving peaks benchmark. Tech. rep., Institute AIFB, University of Karlsruhe, Germany (accessed 2007-08-19) (1999), `http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/` Presented in [32]

34. Branke, J.: Evolutionary optimization in dynamic environments. PhD thesis, Universität Karlsruhe (TH), Fakultät für Wirtschaftswissenschaften (2000)
35. Branke, J.: Evolutionary Optimization in Dynamic Environments. Genetic Algorithms and Evolutionary Computation, vol. 3. Kluwer Academic Publishers, Dordrecht (2001)
36. Branke, J., Salihoğlu, E., Uyar, Ş.: Towards an analysis of dynamic environments. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 1433–1440 (2005)
37. Bremermann, H.J.: Optimization through evolution and recombination. Self-Organizing systems pp. 93–100 (1962)
38. Burke, E.K., Gustafson, S.M., Kendall, G.: Survey and analysis of diversity measures in genetic programming. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 716–723 (2002)
39. Burke, E.K., Gustafson, S.M., Kendall, G., Krasnogor, N.: Is increasing diversity in genetic programming beneficial? an analysis of the effects on fitness. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 1398–1405 (2003)
40. Burke, E.K., Gustafson, S.M., Kendall, G.: Diversity in genetic programming: An analysis of measures and correlation with fitness. IEEE Transactions on Evolutionary Computation 8(1), 47–62 (2004)
41. Cantú-Paz, E., Pelikan, M., Goldberg, D.E.: Linkage problem, distribution estimation, and bayesian networks. Evolutionary Computation 8(3), 311–340 (2000)
42. Carlisle, A.J.: Applying the particle swarm optimizer to non-stationary environments. PhD thesis, Graduate Faculty of Auburn University (2002)
43. Carlisle, A.J., Dozier, G.V.: Tracking changing extrema with adaptive particle swarm optimizer. In: Proceedings of the 5th Biannual World Automation Congress, WAC 2002, Orlando, Florida, USA, vol. 13, pp. 265–270 (2002) doi:10.1109/WAC.2002.1049555
44. Carroll, C.W.: An operations research approach to the economic optimization of a kraft pulping process. PhD thesis, Institute of Paper Chemistry, Appleton, Wisconsin, USA (1959)
45. Ceollo Coello, C.A., Lamont, G.B., van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, Genetic and Evolutionary Computation. Genetic and Evolutionary Computation (1st edn., 2002, 2nd edn., 2007), vol. 5. Kluwer Academic Publishers, Springer (2007)
46. Chen, Y.p.: Extending the Scalability of Linkage Learning Genetic Algorithms – Theory & Practice. Studies in Fuzziness and Soft Computing, vol. 190. Springer, Heidelberg (2006)
47. Cohoon, J.P., Hegde, S.U., Martin, W.N., Richards, D.: Punctuated equilibria: a parallel genetic algorithm. In: Proceedings of the Second International Conference on Genetic algorithms and their Application, pp. 148–154 (1987)
48. Courant, R.: Variational methods for the solution of problems of equilibrium and vibrations. Bulletin of the American Mathematical Society 49(1), 1–23 (1943)
49. Cousins, S.H.: Species diversity measurement: Choosing the right index. Trends in Ecology and Evolution (TREE) 6(6), 190–192 (1991)

50. Das, I., Dennis, J.E.: A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. Structural optimization 14(1), 63–69 (1997)
51. Davidor, Y.: Epistasis variance: A viewpoint on GA-hardness. In: Proceedings of the First Workshop on Foundations of Genetic Algorithms, pp. 23–35 (1990)
52. Dawkins, R.: The evolution of evolvability. In: ALIFE – Artificial Life: Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems, pp. 201–220 (1987)
53. de Jong, E.D., Watson, R.A., Pollack, J.B.: Reducing bloat and promoting diversity using multi-objective methods. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 11–18 (2001)
54. de Lamarck, J.B.P.A.d.C.: Philosophie zoologique – ou Exposition des considérations relatives à l'histoire naturelle des Animaux. Dentu / G. Baillière, Paris, France/Harvard University (1809)
55. Deb, K.: Genetic algorithms in multimodal function optimization. Master's thesis, The Clearinghouse for Genetic algorithms, University of Alabama, Tuscaloosa, tCGA Report No. 89002 (1989)
56. Deb, K.: Solving goal programming problems using multi-objective genetic algorithms. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 77–84 (1999) doi:10.1109/CEC.1999.781910
57. Deb, K.: Genetic algorithms for optimization. KanGAL Report 2001002, Kanpur Genetic Algorithms Laboratory (KanGAL), Kanpur, PIN 208 016, India (2001)
58. Deb, K.: Nonlinear goal programming using multi-objective genetic algorithms. Journal of the Operational Research Society 52(3), 291–302 (2001)
59. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In: Foundations of Genetic Algorithms 2, pp. 93–108 (1993)
60. Deb, K., Goldberg, D.E.: Sufficient conditions for deceptive and easy binary functions. Annals of Mathematics and Artificial Intelligence 10(4), 385–408 (1994)
61. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Proceedings of the International Conference on Parallel Problem Solving from Nature, PPSN, pp. 849–858 (2000); KanGAL Report No. 200001
62. Deb, K., Pratab, A., Agrawal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
63. Deb, K., Sinha, A., Kukkonen, S.: Multi-objective test problems, linkages, and evolutionary methodologies. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 1141–1148. ACM, New York (2006)
64. Dietterich, T.: Overfitting and undercomputing in machine learning. ACM Computing Surveys (CSUR) 27(3), 326–327 (1995)
65. Droste, S., Wiesmann, D.: On representation and genetic operators in evolutionary algorithms. Tech. Rep. CI–41/98, Fachbereich Informatik, Universität Dortmund (1998)
66. Eiben, Á.E., Schippers, C.A.: On evolutionary exploration and exploitation. Fundamenta Informaticae 35(1-4), 35–50 (1998)

67. Eldredge, N., Gould, S.J.: Punctuated equilibria: an alternative to phyletic gradualism. In: Schopf, T.J.M. (ed.) Models in Paleobiology, ch. 5, pp. 82–115. W.H. Freeman, New York (1972)
68. Eldredge, N., Gould, S.J.: Punctuated equilibria: The tempo and mode of evolution reconsidered. Paleobiology 3(2), 115–151 (1977)
69. Eshelman, L.J., Schaffer, J.D.: Preventing premature convergence in genetic algorithms by preventing incest. In: Proceedings of the International Conference on Genetic Algorithms, ICGA, pp. 115–122 (1991)
70. Eshelman, L.J., Caruana, R.A., Schaffer, J.D.: Biases in the crossover landscape. In: Proceedings of the third international conference on Genetic algorithms, pp. 10–19 (1989)
71. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6(2), 109–133 (1995)
72. Festa, P., Resende, M.G.: An annotated bibliography of grasp. AT&T Labs Research Technical Report TD-5WYSEW, AT&T Labs (2004)
73. Fiacco, A.V., McCormick, G.P.: Nonlinear Programming: Sequential Unconstrained Minimization Techniques. John Wiley & Sons Inc., Chichester (1968)
74. Fisher, S.R.A.: The correlations between relatives on the supposition of mendelian inheritance. Philosophical Transactions of the Royal Society of Edinburgh 52, 399–433 (1918)
75. Fitzpatrick, J.M., Grefenstette, J.J.: Genetic algorithms in noisy environments. Machine Learning 3(2–3), 101–120 (1988)
76. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Proceedings of the 5th International Conference on Genetic Algorithms, pp. 416–423 (1993)
77. Forst, C.V., Reidys, C., Weber, J.: Evolutionary dynamics and optimization: Neutral networks as model-landscapes for RNA secondary-structure folding-landscapes. In: European Conference on Artificial Life, pp. 128–147 (1995)
78. Friedberg, R.M.: A learning machine: Part i. IBM Journal of Research and Development 2, 2–13 (1958)
79. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Series of Books in the Mathematical Sciences. W. H. Freeman & Co., New York (1979)
80. Geman, S., Bienenstock, E., Doursat, R.: Neural networks and the bias/variance dilemma. Neural Computation 4(1), 1–58 (1992)
81. Glover, F.: Tabu search – part ii. Operations Research Society of America (ORSA) Journal on Computing 2(1), 190–206 (1990)
82. Glover, F., Taillard, É.D., de Werra, D.: A user's guide to tabu search. Annals of Operations Research 41(1), 3–28 (1993)
83. Gobb, H.G., Grefenstette, J.J.: Genetic algorithms for tracking changing environments. In: Proceedings of the International Conference on Genetic Algorithms, ICGA, pp. 523–529 (1993)
84. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Longman Publishing Co., Amsterdam (1989)
85. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the Second International Conference on Genetic algorithms and their Application, pp. 41–49 (1987)
86. Goldberg, D.E., Deb, K., Korb, B.: Messy genetic algorithms: motivation, analysis, and first results. Complex Systems 3, 493–530 (1989)

87. Greiner, H.: Robust filter design by stochastic optimization. Proceedings of SPIE (The International Society for Optical Engineering) 2253, 150–161 (1994)
88. Greiner, H.: Robust optical coating design with evolutionary strategies. Applied Optics 35, 5477–5483 (1996)
89. Gruau, F., Whitley, L.D.: Adding learning to the cellular development of neural networks: Evolution and the baldwin effect. Evolutionary Computation 1(3), 213–233 (1993)
90. Guntsch, M., Middendorf, M.: Pheromone modification strategies for ant algorithms applied to dynamic TSP. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 213–222. Springer, Heidelberg (2001)
91. Guntsch, M., Middendorf, M., Schmeck, H.: An ant colony optimization approach to dynamic TSP. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 860–867 (2001)
92. Gurin, L.S., Rastrigin, L.A.: Convergence of the random search method in the presence of noise. Automation and Remote Control 26, 1505–1511 (1965)
93. Gustafson, S.M.: An analysis of diversity in genetic programming. PhD thesis, University of Nottingham, School of Computer Science & IT (2004)
94. Gustafson, S.M., Ekárt, A., Burke, E.K., Kendall, G.: Problem difficulty and code growth in genetic programming. Genetic Programming and Evolvable Machines 5(3), 271–290 (2004)
95. Hadj-Alouane, A.B., Bean, J.C.: A genetic algorithm for the multiple-choice integer program. Tech. Rep. 92-50, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbour, MI 48109-2117, USA (1992)
96. Hammel, U., Bäck, T.: Evolution strategies on noisy functions: How to improve convergence properties. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 159–168. Springer, Heidelberg (1994)
97. Han, L., He, X.: A novel opposition-based particle swarm optimization for noisy problems. In: ICNC 2007: Proceedings of the Third International Conference on Natural Computation, vol. 3, pp. 624–629 (2007) doi:10.1109/ICNC.2007.119
98. Handa, H., Lin, D., Chapman, L., Yao, X.: Robust solution of salting route optimisation using evolutionary algorithms. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 3098–3105 (2006) doi:10.1109/CEC.2006.1688701
99. Harik, G.R.: Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. PhD thesis, University of Michigan, Ann Arbor (1997)
100. Hinton, G.E., Nowlan, S.J.: How learning can guide evolution. Complex Systems 1, 495–502 (1987)
101. Hinton, G.E., Nowlan, S.J.: How learning can guide evolution. In: Adaptive individuals in evolving populations: models and algorithms, pp. 447–454. Addison-Wesley Longman Publishing Co., Inc., Amsterdam (1996)
102. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. The University of Michigan Press, Ann Arbor (1975); reprinted by MIT Press, NetLibrary, Inc. (April 1992)

103. Holland, J.H.: Genetic algorithms. Scientific American 267(1), 44–50 (1992)
104. Horn, J., Nafpliotis, N., Goldberg, D.E.: A niched pareto genetic algorithm for multiobjective optimization. In: Proceedings of the First IEEE Conference on Evolutionary Computation, vol. 1, pp. 82–87 (1994) doi:10.1109/ICEC.1994.350037
105. Huynen, M.A.: Exploring phenotype space through neutral evolution. Journal of Molecular Evolution 43(3), 165–169 (1996)
106. Huynen, M.A., Stadler, P.F., Fontana, W.: Smoothness within ruggedness: The role of neutrality in adaptation. Proceedings of the National Academy of Science, USA 93, 397–401 (1996)
107. Igel, C.: Causality of hierarchical variable length representations. In: Proceedings of the 1998 IEEE World Congress on Computational Intelligence, pp. 324–329 (1998)
108. Igel, C., Toussaint, M.: On classes of functions for which no free lunch results hold. Information Processing Letters 86(6), 317–321 (2003)
109. Igel, C., Toussaint, M.: Recent results on no-free-lunch theorems for optimization. ArXiv EPrint arXiv:cs/0303032 (Computer Science, Neural and Evolutionary Computing) (accessed 2008-03-28) (2003), `http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0303032`
110. Ingber, L.: Adaptive simulated annealing (asa): Lessons learned. Control and Cybernetics 25(1), 33–54 (1996)
111. Joines, J.A., Houck, C.R.: On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. In: Proceedings of the First IEEE Conference on Evolutionary Computation, pp. 579–584 (1994) doi:10.1109/ICEC.1994.349995
112. Jones, T.: Evolutionary algorithms, fitness landscapes and search. PhD thesis, The University of New Mexico (1995)
113. Kauffman, S.A.: Adaptation on rugged fitness landscapes. In: Stein, D.L. (ed.) Lectures in the Sciences of Complexity: The Proceedings of the 1988 Complex Systems Summer School. Santa Fe Institute Studies in the Sciences of Complexity, vol. Lecture I, pp. 527–618. Addison-Wesley, Reading (1988)
114. Kauffman, S.A.: The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, Oxford (1993)
115. Kauffman, S.A., Levin, S.A.: Towards a general theory of adaptive walks on rugged landscapes. Journal of Theoretical Biology 128(1), 11–45 (1987)
116. Kearns, M.J., Mansour, Y., Ng, A.Y., Ron, D.: An experimental and theoretical comparison of model selection methods. In: COLT 1995: Proceedings of the eighth annual conference on Computational learning theory, pp. 21–30. ACM Press, New York (1995)
117. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
118. Kirschner, M., Gerhart, J.: Evolvability. Proceedings of the National Academy of Science of the USA (PNAS) 95(15), 8420–8427 (1998)
119. Kita, H., Sano, Y.: Genetic algorithms for optimization of noisy fitness functions and adaptation to changing environments. In: 2003 Joint Workshop of Hayashibara Foundation and 2003 Workshop on Statistical Mechanical Approach to Probabilistic Information Processing (SMAPIP) (2003)

120. Kolarov, K.: Landscape ruggedness in evolutionary algorithms. In: Proceedings of the IEEE Conference on Evolutionary Computation, pp. 19–24 (1997)
121. Köppen, M., Wolpert, D.H., Macready, W.G.: Remarks on a recent paper on the "no free lunch" theorems. IEEE Transactions on Evolutionary Computation 5(3), 295–296 (2001)
122. Landau, E.: Handbuch der Lehre von der Verteilung der Primzahlen. B. G. Teubner, Leipzig (1909); reprinted by Chelsea, New York (1953)
123. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: On the convergence and diversity-preservation properties of multi-objective evolutionary algorithms. Tech. Rep. 108, Computer Engineering and Networks Laboratory (TIK), Department of Electrical Engineering, Swiss Federal Institute of Technology (ETH) Zurich and Kanpur Genetic Algorithms Laboratory (KanGAL), Department of Mechanical Engineering, Indian Institute of Technology Kanpur (2001)
124. Lawrence, S., Giles, C.L.: Overfitting and neural networks: Conjugate gradient and backpropagation. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000), vol. 1, pp. 1114–1119. IEEE Computer Society, Los Alamitos (2000)
125. Lee, J.Y.B., Wong, P.C.: The effect of function noise on gp efficiency. In: Progress in Evolutionary Computation, pp. 1–16 (1995)
126. Li, X., Branke, J., Blackwell, T.: Particle swarm with speciation and adaptation in a dynamic environment. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 51–58 (2006) doi:10.1145/1143997.1144005
127. Liepins, G.E., Vose, M.D.: Deceptiveness and genetic algorithm dynamics. In: Proceedings of the First Workshop on Foundations of Genetic Algorithms (FOGA), pp. 36–50 (1991)
128. Ling, C.X.: Overfitting and generalization in learning discrete patterns. Neurocomputing 8(3), 341–347 (1995)
129. Lohmann, R.: Structure evolution and neural systems. In: Dynamic, Genetic, and Chaotic Programming: The Sixth-Generation, pp. 395–411. Wiley Interscience, Hoboken (1992)
130. Lohmann, R.: Structure evolution and incomplete induction. Biological Cybernetics 69(4), 319–326 (1993)
131. Luke, S., Panait, L.: A comparison of bloat control methods for genetic programming. Evolutionary Computation 14(3), 309–344 (2006)
132. Lush, J.L.: Progeny test and individual performance as indicators of an animal's breeding value. Journal of Dairy Science 18(1), 1–19 (1935)
133. Magurran, A.E.: Biological diversity. Current Biology Magazine 15, R116–R118 (2005)
134. Martin, W.N., Lienig, J., Cohoon, J.P.: Island (migration) models: Evolutionary algorithms based on punctuated equilibria. In: Handbook of Evolutionary Computation, ch. 6.3. Oxford University Press, Oxford (1997)
135. Mendes, R., Mohais, A.S.: Dynde: a differential evolution for dynamic optimization problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 3, pp. 2808–2815 (2005)
136. Miller, B.L., Goldberg, D.E.: Genetic algorithms, tournament selection, and the effects of noise. IlliGAL Report 95006, Illinois Genetic Algorithms Laboratory, Department of General Engineering, University of Illinois (1995)

137. Miller, B.L., Goldberg, D.E.: Genetic algorithms, selection schemes, and the varying effects of noise. Evolutionary Computation 4(2), 113–131 (1996)
138. Miller, B.L., Shaw, M.J.: Genetic algorithms with dynamic niche sharing for multimodal function optimization. IlliGAL Report 95010, Department of General Engineering, University of Illinois at Urbana-Champaign (1995)
139. Mitchell, M., Forrest, S., Holland, J.H.: The royal road for genetic algorithms: Fitness landscapes and GA performance. In: Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, pp. 245–254 (1991)
140. Mitchell, T.M.: Generalization as search. In: Webber, B.L., Nilsson, N.J. (eds.) Readings in Artificial Intelligence, 2nd edn., pp. 517–542. Tioga Pub. Co. Press, Morgan Kaufmann Publishers, Elsevier Science & Technology Books (1981)
141. Mitchell, T.M.: Generalization as search. Artificial Intelligence 18(2), 203–226 (1982)
142. Mori, N., Kita, H., Nishikawa, Y.: Adaptation to a changing environment by means of the thermodynamical genetic algorithm. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 513–522. Springer, Heidelberg (1996)
143. Mori, N., Imanishi, S., Kita, H., Nishikawa, Y.: Adaptation to changing environments by means of the memory based thermodynamical genetic algorithm. In: Proceedings of the International Conference on Genetic Algorithms, ICGA, pp. 299–306 (1997)
144. Mori, N., Kita, H., Nishikawa, Y.: Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 149–158. Springer, Heidelberg (1998)
145. Morrison, R.W.: Designing evolutionary algorithms for dynamic environments. PhD thesis, George Mason University, USA (2002)
146. Morrison, R.W.: Designing Evolutionary Algorithms for Dynamic Environments. Natural Computing 24(1), 143–144 (2004)
147. Morrison, R.W., De Jong, K.A.: A test problem generator for non-stationary environments. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 3, pp. 2047–2053 (1999) doi:10.1109/CEC.1999.785526
148. Morrison, R.W., De Jong, K.A.: Measurement of population diversity. In: Collet, P., Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M. (eds.) EA 2001. LNCS, vol. 2310, pp. 1047–1074. Springer, Heidelberg (2002)
149. Mostaghim, S.: Multi-objective evolutionary algorithms: Data structures, convergence and, diversity. PhD thesis, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Deutschland, Germany (2004)
150. Munetomo, M., Goldberg, D.E.: Linkage identification by non-monotonicity detection for overlapping functions. Evolutionary Computation 7(4), 377–398 (1999)
151. Munetomo, M., Goldberg, D.E.: Linkage identification by non-monotonicity detection for overlapping functions. IlliGAL Report 99005, Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois at Urbana-Champaign (1999)
152. Muttil, N., Liong, S.-Y.: Superior exploration–exploitation balance in shuffled complex evolution. Journal of Hydraulic Engineering 130(12), 1202–1205 (2004)

153. Naudts, B., Verschoren, A.: Epistasis on finite and infinite spaces. In: Proceedings of the 8th International Conference on Systems Research, Informatics and Cybernetics, pp. 19–23 (1996)
154. Naudts, B., Verschoren, A.: Epistasis and deceptivity. Bulletin of the Belgian Mathematical Society 6(1), 147–154 (1999)
155. Newman, M.E.J., Engelhardt, R.: Effect of neutral selection on the evolution of molecular species. Proceedings of the Royal Society of London B (Biological Sciences) 256(1403), 1333–1338 (1998)
156. Oei, C.K., Goldberg, D.E., Chang, S.J.: Tournament selection, niching, and the preservation of diversity. IlliGAl Report 91011, Illinois Genetic Algorithms Laboratory (IlliGAL), Department of Computer Science, Department of General Engineering, University of Illinois at Urbana-Champaign (1991)
157. Olsen, A.L.: Penalty functions and the knapsack problem. In: Proceedings of the First IEEE Conference on Evolutionary Computation, vol. 2, pp. 554–558 (1994)
158. Osman, I.H.: An introduction to metaheuristics. In: Lawrence, M., Wilsdon, C. (eds.) Operational Research Tutorial Papers, pp. 92–122. Stockton Press, Hampshire (1995); publication of the Operational Research Society, Birmingham, UK
159. Paenke, I., Branke, J., Jin, Y.: On the influence of phenotype plasticity on genotype diversity. In: First IEEE Symposium on Foundations of Computational Intelligence (FOCI 2007), pp. 33–40 (2007)
160. Pan, G., Dou, Q., Liu, X.: Performance of two improved particle swarm optimization in dynamic optimization environments. In: ISDA 2006: Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA 2006), vol. 2, pp. 1024–1028. IEEE Computer Society Press, Los Alamitos (2006)
161. Pan, H., Wang, L., Liu, B.: Particle swarm optimization for function optimization in noisy environment. Applied Mathematics and Computation 181(2), 908–919 (2006)
162. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: Boa: The bayesian optimization algorithm. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 525–532 (1999)
163. Phillips, P.C.: The language of gene interaction. Genetics 149(3), 1167–1171 (1998)
164. Pohlheim, H.: Geatbx introduction – evolutionary algorithms: Overview, methods and operators. Tech. rep., documentation for GEATbx version 3.7 (2005) (accessed, 2007-07-03), `http://www.GEATbx.com`
165. Purshouse, R.C.: On the evolutionary optimisation of many objectives. PhD thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield (2003)
166. Radcliffe, N.J.: Non-linear genetic representations. In: Proceedings of the International Conference on Parallel Problem Solving from Nature, PPSN, pp. 259–268. Elsevier, Amsterdam (1992)
167. Radcliffe, N.J.: The algebra of genetic algorithms. Annals of Mathematics and Artificial Intelligence 10(4) (1994) doi:10.1007/BF01531276
168. Radcliffe, N.J., Surry, P.D.: Fundamental limitations on search algorithms: Evolutionary computing in perspective. In: van Leeuwen, J. (ed.) Computer Science Today. LNCS, vol. 1000, pp. 275–291. Springer, Heidelberg (1995)

169. Rayward-Smith, V.J.: A unified approach to tabu search, simulated annealing and genetic algorithms. In: Rayward-Smith, V.J. (ed.) Applications of Modern Heuristic Methods – Proceedings of the UNICOM Seminar on Adaptive Computing and Information Processing, Brunel University Conference Centre, London, UK, vol. I, pp. 55–78. Alfred Waller Ltd / Nelson Thornes Ltd / Unicom Seminars Ltd (1994)
170. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog Verlag, Stuttgart (1973)
171. Rechenberg, I.: Evolutionsstrategie 1994. Werkstatt Bionik und Evolutionstechnik, vol. 1. Frommann Holzboog (1994)
172. Reidys, C.M., Stadler, P.F.: Neutrality in fitness landscapes. Applied Mathematics and Computation 117(2–3), 321–350 (2001)
173. Richter, H.: Behavior of evolutionary algorithms in chaotically changing fitness landscapes. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 111–120. Springer, Heidelberg (2004)
174. Riedl, R.J.: A systems-analytical approach to macroevolutionary phenomena. Quarterly Review of Biology, 351–370 (1977)
175. Robbins, H., Monro, S.: A stochastic approximation method. Annals of Mathematical Statistics 22(3), 400–407 (1951)
176. Ronald, S.: Preventing diversity loss in a routing genetic algorithm with hash tagging. Complexity International 2 (1995) (accessed 2008-12-07), `http://www.complexity.org.au/ci/vol02/sr_hash/`
177. Ronald, S.: Genetic algorithms and permutation-encoded problems. diversity preservation and a study of multimodality. PhD thesis, University Of South Australia. Department of Computer and Information Science (1996)
178. Ronald, S.: Robust encodings in genetic algorithms: A survey of encoding issues. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 43–48 (1997) doi:10.1109/ICEC.1997.592265
179. Rosca, J.P.: An analysis of hierarchical genetic programming. Tech. Rep. TR566, The University of Rochester, Computer Science Department (1995)
180. Rosca, J.P., Ballard, D.H.: Causality in genetic programming. In: Proceedings of the International Conference on Genetic Algorithms, ICGA, pp. 256–263 (1995)
181. Rosin, P.L., Fierens, F.: Improving neural network generalisation. In: Proceedings of the International Geoscience and Remote Sensing Symposium, Quantitative Remote Sensing for Science and Applications, IGARSS 1995, vol. 2, pp. 1255–1257. IEEE, Los Alamitos (1995)
182. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms, 2nd edn. Physica-Verlag (2006) (1st edn., 2002)
183. Routledge, R.D.: Diversity indices: Which ones are admissible? Journal of Theoretical Biology 76, 503–515 (1979)
184. Rudnick, W.M.: Genetic algorithms and fitness variance with an application to the automated design of artificial neural networks. PhD thesis, Oregon Graduate Institute of Science & Technology (1992)
185. Rudolph, G.: Self-adaptation and global convergence: A counter-example. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 1, pp. 646–651 (1999)

186. Rudolph, G.: Self-adaptive mutations may lead to premature convergence. IEEE Transactions on Evolutionary Computation 5(4), 410–414 (2001)
187. Rudolph, G.: Self-adaptive mutations may lead to premature convergence. Tech. Rep. CI–73/99, Fachbereich Informatik, Universität Dortmund (2001)
188. Sano, Y., Kita, H.: Optimization of noisy fitness functions by means of genetic algorithms using history of search. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 571–580. Springer, Heidelberg (2000)
189. Sano, Y., Kita, H.: Optimization of noisy fitness functions by means of genetic algorithms using history of search with test of estimation. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 360–365 (2002)
190. Sarle, W.: What is overfitting and how can i avoid it? Usenet FAQs: compaineural-nets FAQ 3: Generalization(3) (2007)
191. Sarle, W.S.: Stopped training and other remedies for overfitting. In: Proceedings of the 27th Symposium on the Interface: Computing Science and Statistics, pp. 352–360 (1995)
192. Schaffer, J.D., Eshelman, L.J., Offutt, D.: Spurious correlations and premature convergence in genetic algorithms. In: Proceedings of the First Workshop on Foundations of Genetic Algorithms (FOGA), pp. 102–112 (1990)
193. Sendhoff, B., Kreutz, M., von Seelen, W.: A condition for the genotype-phenotype mapping: Causality. In: Proceedings of the International Conference on Genetic Algorithms, ICGA, pp. 73–80 (1997)
194. Shackleton, M., Shipman, R., Ebner, M.: An investigation of redundant genotype-phenotype mappings and their role in evolutionary search. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 493–500 (2000)
195. Shekel, J.: Test functions for multimodal search techniques. In: Proceedings of the Fifth Annual Princeton Conference on Information Science and Systems, pp. 354–359. Princeton University Press, Princeton (1971)
196. Shipman, R.: Genetic redundancy: Desirable or problematic for evolutionary adaptation? In: Proceedings of the 4^{th} International Conference on Artificial Neural Nets and Genetic Algorithms, pp. 1–11 (1999)
197. Shipman, R., Shackleton, M., Ebner, M., Watson, R.: Neutral search spaces for artificial evolution: a lesson from life. In: Bedau, M., McCaskill, J.S., Packard, N.H., Rasmussen, S., McCaskill, J., Packard, N. (eds.) Artificial Life VII: Proceedings of the Seventh International Conference on Artificial Life. The MIT Press, Bradford Books, Complex Adaptive Systems (2000)
198. Shipman, R., Shackleton, M., Harvey, I.: The use of neutral genotype-phenotype mappings for improved evolutionary search. BT Technology Journal 18(4), 103–111 (2000)
199. Siedlecki, W.W., Sklansky, J.: Constrained genetic optimization via dynamic reward-penalty balancing and its use in pattern recognition. In: Proceedings of the third international conference on Genetic algorithms, pp. 141–150 (1989)
200. Singh, G., Deb, K.: Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 1305–1312 (2006)
201. Smith, A.E., Coit, D.W.: Penalty functions. In: Handbook of Evolutionary Computation, ch. 5.2. Oxford University Press, Oxford (1997)

202. Smith, M.: Neural Networks for Statistical Modeling. John Wiley & Sons, Inc. International Thomson Computer Press (1993/1996)
203. Smith, S.S.F.: Using multiple genetic operators to reduce premature convergence in genetic assembly planning. Computers in Industry 54(1), 35–49 (2004)
204. Smith, T., Husbands, P., Layzell, P., O'Shea, M.: Fitness landscapes and evolvability. Evolutionary Computation 10(1), 1–34 (2002)
205. Spatz, B.M., Rawlins, G.J.E. (eds.): Proceedings of the First Workshop on Foundations of Genetic Algorithms. Morgan Kaufmann Publishers, Inc., San Francisco (1990)
206. Spieth, C., Streichert, F., Speer, N., Zell, A.: Utilizing an island model for ea to preserve solution diversity for inferring gene regulatory networks. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 1, pp. 146–151 (2004)
207. Stagge, P., Igel, C.: Structure optimization and isomorphisms. In: Theoretical Aspects of Evolutionary Computing, pp. 409–422. Springer, Heidelberg (2000)
208. Stewart, T.: Extrema selection: accelerated evolution on neutral networks. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 1 (2001)
209. Taguchi, G.: Introduction to Quality Engineering: Designing Quality into Products and Processes. Asian Productivity Organization / American Supplier Institute Inc. / Quality Resources / Productivity Press Inc., translation of Sekkeisha no tame no hinshitsu kanri (1986)
210. Taillard, É.D., Gambardella, L.M., Gendrau, M., Potvin, J.-Y.: Adaptive memory programming: A unified view of metaheuristics. European Journal of Operational Research 135(1), 1–16 (2001)
211. Tetko, I.V., Livingstone, D.J., Luik, A.I.: Neural network studies, 1. comparison of overfitting and overtraining. Journal of Chemical Information and Computer Sciences 35(5), 826–833 (1995)
212. Thierens, D.: On the scalability of simple genetic algorithms. Tech. Rep. UU-CS-1999-48, Department of Information and Computing Sciences, Utrecht University (1999)
213. Thierens, D., Goldberg, D.E., Pereira, Â.G.: Domino convergence, drift, and the temporal-salience structure of problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 535–540 (1998), doi:10.1109/ICEC.1998.700085
214. Toussaint, M., Igel, C.: Neutrality: A necessity for self-adaptation. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 1354–1359 (2002)
215. Trelea, I.C.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85(6), 317–325 (2003)
216. Trojanowski, K.: Evolutionary algorithms with redundant genetic material for non-stationary environments. PhD thesis, Instytut Podstaw Informatyki PAN, Institute of Computer Science, Warsaw, University of Technology, Poland (1994)
217. Tsutsui, S., Ghosh, A.: Genetic algorithms with a robust solution searching scheme. IEEE Transactions on Evolutionary Computation 1, 201–208 (1997)
218. Tsutsui, S., Ghosh, A., Fujimoto, Y.: A robust solution searching scheme in genetic search. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 543–552. Springer, Heidelberg (1996)

219. Ursem, R.K.: Models for evolutionary algorithms and their applications in system identification and control optimization. PhD thesis, Department of Computer Science, University of Aarhus, Denmark (2003)
220. Vaessens, R.J.M., Aarts, E.H.L., Lenstra, J.K.: A local search template. In: Proceedings of the International Conference on Parallel Problem Solving from Nature, PPSN, pp. 67–76 (1992)
221. Vaessens, R.J.M., Aarts, E.H.L., Lenstra, J.K.: A local search template. Computers and Operations Research 25(11), 969–979 (1998)
222. van Nimwegen, E., Crutchfield, J.P.: Optimizing epochal evolutionary search: Population-size dependent theory. Machine Learning 45(1), 77–114 (2001)
223. van Nimwegen, E., Crutchfield, J.P., Huynen, M.: Neutral evolution of mutational robustness. Proceedings of the National Academy of Science of the United States of Americs (PNAS) – Evolution 96(17), 9716–9720 (1999)
224. van Nimwegen, E., Crutchfield, J.P., Mitchell, M.: Statistical dynamics of the royal road genetic algorithm. Theoretical Computer Science 229(1–2), 41–102 (1999)
225. Wagner, A.: Robustness and Evolvability in Living Systems. Princeton Studies in Complexity. Princeton University Press, Princeton (2005)
226. Wagner, A.: Robustness, evolvability, and neutrality. FEBS Lett 579(8), 1772–1778 (2005)
227. Wagner, G.P., Altenberg, L.: Complex adaptations and the evolution of evolvability. Evolution 50(3), 967–976 (1996)
228. Watanabe, S.: Knowing and Guessing: A Quantitative Study of Inference and Information. John Wiley & Sons, Chichester (1969)
229. Weicker, K.: Evolutionäre Algorithmen. Leitfäden der Informatik, B. G. Teubner GmbH (2002)
230. Weicker, K., Weicker, N.: Burden and benefits of redundancy. In: Sixth Workshop on Foundations of Genetic Algorithms (FOGA), pp. 313–333. Morgan Kaufmann, San Francisco (2000)
231. Weise, T., Zapf, M., Geihs, K.: Rule-based Genetic Programming. In: Proceedings of BIONETICS 2007, 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems (2007)
232. Weise, T., Niemczyk, S., Skubch, H., Reichle, R., Geihs, K.: A tunable model for multi-objective, epistatic, rugged, and neutral fitness landscapes. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 795–802 (2008)
233. Whitley, L.D., Gordon, V.S., Mathias, K.E.: Lamarckian evolution, the baldwin effect and function optimization. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 6–15. Springer, Heidelberg (1994)
234. Wiesmann, D., Hammel, U., Bäck, T.: Robust design of multilayer optical coatings by means of evolutionary algorithms. IEEE Transactions on Evolutionary Computation 2, 162–167 (1998)
235. Wiesmann, D., Hammel, U., Bäck, T.: Robust design of multilayer optical coatings by means of evolutionary strategies. Sonderforschungsbereich (sfb) 531, Universität Dortmund (1998)
236. Wilke, C.O.: Evolutionary dynamics in time-dependent environments. PhD thesis, Fakultät für Physik und Astronomie, Ruhr-Universität Bochum (1999)
237. Wilke, C.O.: Adaptive evolution on neutral networks. Bulletin of Mathematical Biology 63(4), 715–730 (2001)

238. Wilke, D.N., Kok, S., Groenwold, A.A.: Comparison of linear and classical velocity update rules in particle swarm optimization: notes on diversity. International Journal for Numerical Methods in Engineering 70(8), 962–984 (2007)
239. Williams, G.C.: Pleiotropy, natural selection, and the evolution of senescence. Evolution 11(4), 398–411 (1957)
240. Winter, P.C., Hickey, G.I., Fletcher, H.L.: Instant Notes in Genetics, 3rd edn. Springer, New York (2006) (1st edn. 1998, 2nd edn. 2002)
241. Wolpert, D.H., Macready, W.G.: No free lunch theorems for search. Tech. Rep. SFI-TR-95-02-010, The Santa Fe Institute (1995)
242. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)
243. Wu, N.: Differential evolution for optimisation in dynamic environments. Tech. rep., School of Computer Science and Information Technology, RMIT University (2006)
244. Yang, S., Ong, Y.S., Jin, Y.: Evolutionary Computation in Dynamic and Uncertain Environments. Studies in Computational Intelligence, vol. 51(XXIII). Springer, Heidelberg (2007)
245. Zakian, V.: New formulation for the method of inequalities. Proceedings of the Institution of Electrical Engineers 126(6), 579–584 (1979)
246. Žilinskas, A.: Algorithm as 133: Optimization of one-dimensional multimodal functions. Applied Statistics 27(3), 367–375 (1978)
247. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Tech. Rep. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich (2001)
248. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems. Proceedings of the EUROGEN 2001 Conference, pp. 95–100 (2001)

The Rationale Behind Seeking Inspiration from Nature

Kent C.B. Steer, Andrew Wirth, and Saman K. Halgamuge

Abstract. There are currently numerous heuristic algorithms for combinatorial optimisation problems which are commonly described as nature-inspired. Parallels can certainly be drawn between these algorithms and various natural processes, but the extent of the natural inspiration is not always clear. This chapter attempts to clarify what it means to say an algorithm is nature-inspired. Additionally, we will discuss the features of nature which make it a valuable resource in the design of successful new algorithms. Not only does nature provide processes which can be used for optimisation, but it is also a popular source of useful metaphors, which assist the designer. Finally, the history of some well-known algorithms will be discussed, with particular attention to the role nature has played in their development.

1 Introduction

In this chapter we will examine the rationale behind the use of nature as a source of inspiration for optimisation algorithms. We will consider the features of nature that contribute to its status as a valuable and popular resource.

The field of nature-inspired computing has grown in popularity over the last fifty years. The algorithms that the field has produced can often be traced to the use of computer simulations to investigate nature. Their popularity is largely driven by the success of these algorithms, many of which were discovered by interdisciplinary partnerships.

Nature-inspired algorithms for optimisation generally fall into the category of heuristic methods. That is, the algorithms will tend to improve computation time at the cost of solution quality—a satisfactory trade-off in many real world problems, which are often *NP-hard*. The best known exact methods for *NP-hard* problems

Kent C.B. Steer · Andrew Wirth · Saman K. Halgamuge
Department of Mechanical Engineering, The University of Melbourne,
Victoria 3010, Australia
e-mail: k.steer1@pgrad.unimelb.edu.au

require time exponential in problem size. As such, there are no known feasible exact solution methods for large problems of this kind.

It is hoped that by exploring the strengths and weaknesses of nature as a source of inspiration we can help readers better utilise this valuable resource in finding solutions to real problems. Furthermore, we hope to emphasise the potential benefits of interdisciplinary research in discovery of new methods and improvement of existing ones.

Here we use the term *nature* to refer to any part of the physical universe which is not a product of intentional human design. We would like to distinguish between two forms of inspiration. The first, which we will call 'strong' inspiration, involves the investigation of some existing problem-solving mechanism, the extraction of some qualitative process description, and the application to some alternative purpose. The second, which we will call 'weak' inspiration, is the less formal role of some phenomenon in the creative stage of solution formulation.

2 Nature's Résumé

We shall now examine various aspects of nature relevant to optimisation and algorithm design. This will hopefully give some insight into the popularity and success of nature-inspired methods for both research and practical applications.

Specifically, we will consider the origins of optimising phenomena in nature, and the strengths of emergent behaviours. Additionally, we will discuss the role of metaphor in algorithm research, and how nature achieves and encourages creativity. This should not be seen as an exhaustive account of what nature has to offer, but rather, it focuses on the aspects we consider to be important and interesting.

2.1 Optimisation by Natural Selection

If we momentarily restrict our attention to the biological branch of nature, we can highlight some of the useful characteristics of this plentiful supplier of inspiration. Undoubtedly the most important contribution to modern biology was made by Charles Darwin with his Theory of Evolution by Natural Selection. Observing the achievements of animal husbandry, he writes:

> Why, if man can by patience select variations most useful to himself, should nature fail in selecting variations useful, under changing conditions of life, to her living products . . . I can see no limit to this power, in slowly and beautifully adapting each form to the most complex relations of life. [17]

The immense explanatory power of a relatively simple set of rules—*reproduction*, *mutation* and *selection*—has often earned Darwin's theory the title of the most significant scientific discovery of the 19th century. As the evolutionary biologist Theodosius Dobzhansky writes, "nothing in biology makes sense except in the light of evolution." [19]

The period between the birth of an organism and the birth of its offspring can be decades. The optimality of its behaviour during this period will influence the likelihood of its genes being propagated. We should not then be surprised to find evolution producing numerous ‘optimisation sub-processes’ suited to different time-scales. To achieve this, organisms use various mechanisms to interact with their environment, which may be of use to an algorithm designer. Not only is natural selection itself a source of much inspiration, but it is also key to the existence of all the biological problem-solving mechanisms we find in nature. Accordingly, a solid understanding of evolution by natural selection is of use to any researcher interested in nature-inspired techniques.

If we are to extract an optimisation method from nature, it seems appropriate to ask exactly what nature was using it for. What is being optimised by natural selection? Is there some approximation to an objective function? How is the problem constrained? How can we measure success?

Ants make up 10 percent of the biomass of all animals in the Amazon rain forest [53], but that does not necessarily mean they are a superior solution. Should we measure success by the longevity of the gene? the individual? or perhaps the species? Maybe the efficiency of energy use is important? In *The Diversity of Life*, E. O. Wilson writes,

> The hallmark of life is this: a struggle among an immense variety of organisms weighing next to nothing for a vanishingly small amount of energy. [53]

Hopfield and Tank [35] describe the challenges when modelling a natural process as an optimisation problem:

> While a cost function may be specified, real world data used to evaluate it is generally not precise. Also, complex cost functions usually involve somewhat arbitrary weightings and forms of the various contributions. From an engineering viewpoint, these complications imply that little meaning can be attached to “best”. Often, what is truly desired is a very *good* solution, . . . computed on a time scale short enough so that the solution can be used in the choice of appropriate action . . . This is especially true in . . . perception and pattern recognition, because these problems typically have an immense number of variables and the task of searching for the mathematical optimum of the criterion can often be of considerable combinatorial difficulty, and hence time consuming [emphasis in the original].

In nature, managing the trade-off between ‘solution quality’ and ‘computation time’ is essential to survival. A similar trade-off is made when using a heuristic to solve the various optimisation problems faced by engineers.

In 1932 Sewall Wright introduced a vivid metaphor to help visualise natural evolution. The *fitness landscape*, according to T. Smith *et al.*,

> describes the search space as a multidimensional landscape defined by the genotype-to-fitness mapping through which evolution moves. The classical idea of searching this landscape for good genotypes focuses on the difficulty of climbing up to the globally optimal fitness solution and avoiding locally optimal solutions. [51]

The genotype can be described as the encoded blueprint for producing the organism. Wright states,

> The problem of evolution as I see it is that of a mechanism by which the species may continually find its way from lower to higher peaks in this field. [55]

We can consider a species as occupying some portion of the fitness landscape, or a *niche*. De Castro defines a *niche* as "the region consisting of the set of possible environments in which a species can persist; members of one species occupy the same ecological niche." He then elaborates,

> [A fitness landscape] is a topographic map used to represent the degree of adaption of individuals in a given environment. Individuals that only reproduce with each other are part of the same species, which occupies one or more biological niche. As the fittest individuals of the population have higher chances of surviving and reproducing, the outcome of evolution is a population increasingly more fit to its environment. [13]

The fittest individuals are those with superior problem-solving mechanisms. It is these problem-solving mechanisms which have been the source of 'strong' inspiration for many popular optimisation algorithms.

While it is difficult to say exactly what is being optimised by natural selection, we do observe it to have produced certain useful features. These features will now be discussed in more detail, along with their applicability to optimisation.

2.1.1 Adaptation

The natural world is not a stagnant place; meteorological events, tidal forces, plate tectonics, and all the biological activities. Evolution by natural selection is a dynamic process, where the fitness landscape is always changing. As individuals and populations search for new ways to exploit their environment, the environment changes. For example, if a species becomes too skilled at hunting a certain prey, the food supply may run out. To survive, organisms must be able to cope with changing environmental conditions. This change can occur over millennia, a few generations, an individual's lifetime, or in an instant.

Some organisms have the ability to withstand large variations in the environment. This approach can be thought of as change tolerance, or *robustness*. Other organisms respond to change more dynamically, using a process called *adaptation*.

> In the most general sense, *adaptation* is a feedback process in which external changes in an environment are mirrored by compensatory internal changes in an adaptive system. [23]

Nature has been observed to achieve this adaptive ability in many ways, and biologists will undoubtedly continue to discover new mechanisms in the future. An important feature of any adaptive process is some form of *memory*, either implicit or explicit. Memory allows previous experience to influence future actions.

Closely related to memory is the concept of a *learning* mechanism. Learning mechanisms process experience and store it in memory. This ability is clearly seen in the human brain, although the mechanism is still poorly understood [38]. A less obvious example is the human immune system, which is capable of recognising and combating infectious foreign elements with specialised responses based on previous exposure [14].

Due to the niche-filling tendency of natural selection we find a range of novel adaptive methods. In this way, nature provides us with a library of mechanisms for dealing with environmental changes. These mechanisms are tuned to different scales of change, be it fast, slow, mild or severe. Many real world problems we are faced with have some dynamic component. Jin and Branke [36] describe four classes of uncertainty in dynamic environments,

1. *noise*—the fitness evaluation is subject to noise,
2. *robustness*—the design variables are subject to perturbations or changes after the optimal solution has been determined,
3. *fitness approximation*—the fitness function is too expensive to evaluate exactly, or is unavailable and must be estimated from experimental data, and
4. *time-varying fitness function*.

With some creativity, we can see most of these kinds of uncertainty in nature, and the natural problem-solving mechanisms must handle them. Accordingly, we can hope to learn from the natural world techniques for successfully overcoming uncertainty.

The ability to adapt also helps deal with uncertainty in static environments. If little is known about a problem, it may be desirable to have an algorithm which learns and adapts as it searches for a solution.

2.1.2 Efficiency

In the opening paragraph of his 1922 article *Contribution to the Energetics of Evolution*, Lotka writes:

> ...the fundamental object of contention in the life-struggle, in the evolution of the organic world, is available energy. In accord with this observation is the principle that, in the struggle for existence, the advantage must go to those organisms whose energy-capturing devices are most efficient in directing available energy into channels favourable to the preservation of the species. [39]

Nature is often under pressure to produce efficient solutions. Given the unpredictability of the environment, whenever resources become scarce, the efficient individuals will have an advantage. Perhaps an analogy can be made between the computational efficiency of an optimisation algorithm and the energy efficiency of the natural mechanism upon which it is based.

2.1.3 Generality

Many of the nature-inspired algorithms currently in use are being applied to a wide range of problems. This puts them in the category of *metaheuristics*, where little or no problem specific information is used in the design of the algorithm. But is this kind of generality found in nature, or is it a human innovation?

Generality is related to the concept of adaptability. Some problem-solving mechanisms found in nature can be viewed as hierarchic algorithms. A successful high level algorithm will often use various adaptive subroutines. For example, ants build nests in many different environments, using the most suitable available materials. As generations pass they may adjust to better collect local materials, but the general rules of assembly are retained. This can be tied back to the use of *diversity* as a means of preservation. A species which survives only in a very small niche is far more likely to suffer extinction when the environment changes. On the other hand some degree of specialisation will be advantageous, especially during periods of stability. As such, natural selection must find a balance between generality and specialisation.

Natural selection itself is certainly a widespread process in nature, capable of finding novel and elaborate solutions to a huge number of problems. Accordingly, it is not surprising that the evolutionary algorithms have been so broadly and successfully applied [27].

It is interesting to consider natural algorithms in terms of the No Free Lunch Theorems [54]. Since all problem-solving techniques found in nature are to some extent specialised to real problems, there is at least an intuitive reason to think they will perform better than random search on the set of problems arising from real world situations.

2.2 Complex Systems and Emergent Behaviour

We now move to a discussion of complex systems and the phenomenon of emergent behaviour. At this point we broaden our scope to include all physical systems, not merely the biological systems previously considered.

In general terms, emergent behaviour is the appearance of some high level function as a result of the interactions of some collection of independent elements. A more formal definition is given by El-Hani and Emmeche [22], who treat the topic from a philosophical perspective.

A property P is said to be an emergent property of an object O if and only if:

1. P supervenes on properties and relations of the parts of O;
2. P is not observed in any of the parts of O; and
3. O has a downward causal influence over its parts, constraining their relations in space-time so that the pattern of constraints realises and, thus, explains P.

Emergence has also been described as *non-linear aggregate behaviour*, where the behaviour of a whole is not simply the sum of the parts. This does not, however, rule out a deterministic relationship between the parts and the whole. De Castro [13] writes:

> there are many systems that can be described adequately as being strictly deterministic but that still remain unpredictable.

and also,

> One of the very important theoretical consequences of chaos theory is the divorce between determinism and predictability.

This is seen in dynamical systems with highly sensitive initial conditions, that is, a slight change in initial conditions can cause a large change in the state of the system at some later time. Given the limitations of our ability to make precise measurements, we find that a system can be both fully deterministic yet entirely unpredictable.

This leads to an important provision of nature—emergent problem-solving mechanisms. A few billion years of trial-and-error have allowed nature to find simple rules for producing useful emergent behaviour. An algorithm designer can look to nature for the desired emergent behaviour, and then—possibly with the help of a biologist—extract the basic rules. In this way we can develop algorithms which are relatively simple to code, yet capable of achieving complex emergent behaviour. These emergent approaches to optimisation are also well suited to parallel computation, and as such, capable of utilising recent developments in the field. Furthermore, being derived from natural processes, these algorithms are likely to be robust, adaptive and efficient, albeit approximate. The human immune system is, again, a good example:

> From the information processing perspective, the immune system can be seen as a parallel and distributed adaptive system. It is capable of learning, it uses memory and is capable of associative retrieval of information in recognition and classification tasks. Particularly, it learns to recognise patterns, it remembers patterns that it has been shown in the past and its global behaviour is an emergent property of many local interactions. All these features of the immune system provide, in consequence, great robustness, fault tolerance, dynamism and adaptability. These are the properties of the immune system that mainly attract researchers to try to emulate it in a computer. [14]

Any heuristic which exploits some emergent behaviour observed in nature is in the 'strong' inspiration category.

2.3 Natural Metaphors

As astute pattern matchers, humans are capable of recognising similarities between our own particular problems and others found in nature. This has two immediate advantages, first we can see how nature solves the problem, and second the metaphor helps us think abstractly about new solution methods.

For example, it has been suggested by some that the fitness landscape proposed by Wright [55] be inverted, and thus evolution viewed as a minimisation problem. From a computational perspective this is a trivial modification, but as a metaphor to assist human comprehension some have found it useful.

> Such a viewpoint is intuitively appealing. Searching for peaks depicts evolution as a slowly advancing, tedious, uncertain process. Moreover, there appears to be a certain fragility to an evolving phyletic line; an optimized population might be expected to quickly fall off the peak under slight perturbations. The inverted topography leaves an altogether different impression. [25]

The way we think about problems has a significant impact on the kinds of solutions we are able to produce. Thinking metaphorically about a problem gives the mind much greater flexibility in exploring solutions. Artificially imposed constraints can be removed when a problem is considered from an alternative perspective.

Physicist David Bohm and scientist David Peat, in their book *Science, order and creativity* [9], describe the role of metaphor as a facilitator of the 'free play' of thought. The use of metaphor in science is compared to its literary use:

> For, in perceiving a new idea in science, the mind is involved in a similar form of creative perception as when it engages a poetic metaphor. However, in science it is essential to unfold the meaning of the metaphor in even greater and more "literal" detail, while in poetry the metaphor may remain relatively implicit. [9]

Nature can inspire a new optimisation algorithm without providing a mechanism. Heuristics produced in this manner can be attributed to 'weak' inspiration.

2.4 Creativity

To create an entirely novel heuristic which performs well for some optimisation problem—or set of problems—is a difficult task. This task itself can be conceived of as a search procedure, with various local optima and vast plateaux. The number of possibilities is often large, and the islands of success can be sparsely located. In such a situation, discoveries are largely due to luck and perseverance.

Dean Simonton gives examples of various scientists describing their discoveries as the result of a mechanistic *combinatorial* search process. However, he is cautious of drawing any conclusions from these anecdotal accounts:

> Admittedly, these introspective reports cannot be considered empirical proof that scientific creativity operates according to a chance combinatorial mechanism. [48]

For some problems it may be possible for heuristics to be designed in a logical procedural fashion. However, as the complexity of the problem increases, this approach becomes unrealistic, and we usually resort to some semi-blind exploration. In some cases, a little knowledge can be inhibitive to the discovery—or even consideration—of alternatives.

Nature has been conducting this blind search continuously for billions of years. In this way, nature provides a shortcut in the creative process. Assuming an analogous problem can be found, nature will do the hard work. In *Archimedes' Bathtub*, by David Perkins, we read:

> Mother nature may repurpose, but do we see the full pattern of breakthrough thinking in nature—the long search, little apparent progress, the precipitating event, some non-mental equivalent of the cognitive snap, and transformation? Arguably, yes! [41]

3 Selected Examples

We will now examine some examples of nature-inspired algorithms for optimisation, with reference to the topics discussed in the previous section. It is hoped that an awareness of the history of nature-inspired methods will be useful for future heuristic development. In examining these examples, the following questions have also been considered:

- To what extent does the natural phenomenon optimise?
- How well is (or was) the natural phenomenon understood?
- How similar are the inspired algorithm and the natural phenomenon?
- What is the relationship between the *investigation* of nature and the *exploitation* of nature?

While at some points it will be helpful to discuss the details of the natural mechanism and the algorithmic abstraction, these are peripheral considerations. The central issue is the role of nature in the inspiration process.

3.1 Evolutionary Algorithms

There are many members in the family of evolutionary algorithms, each with a large number of variants and degrees of specialisation. Many hybrid methods have been developed in an attempt to combine particular characteristics. All these algorithms share certain features which were initially inspired by ideas traceable to the Darwinian Theory of Evolution by Natural Selection. In broad terms, these algorithms contain a *population* of *individuals* (solutions) capable of *reproduction* (recombination), undergoing *mutation* and *selection* [27].

The work of evolutionary biologists in the 1950s and 1960s, a key period in the history of evolutionary computation, lead to many of the algorithms currently in use [13]. As the availability and processing power of computers grew, researchers began using computer simulations to test hypothesis about natural evolution, with the emphasis on understanding nature [16, 30].

The history of evolutionary algorithms, as with almost any history, is not a linear sequence of events, but a tapestry of innovations made by various different researchers. The transition from pure investigation of natural evolution to the combined investigation and exploitation occurred in numerous locations. D. B. Fogel has compiled many important papers from this period in [26]. Among the most widely recognised are German researchers I. Rechenberg and H.-P. Schwefel [43], L. J. Fogel [29] of the U.S.A., and J. H. Holland [32], also from the U.S.A., whose respective work gave rise to *evolution strategies*, *evolutionary programming* and *genetic algorithms*.

The view of natural evolution as a process of optimisation is widely recognised, as was discussed in Section 2.1. Furthermore, the problems faced in nature share many features with the hard optimisation problems faced by engineers and computer scientists. D. B. Fogel writes:

> Evolved biota demonstrate optimized complex behavior at every level: the cell, the organ, the individual, and the population. The problems that biological species have solved are typified by chaos, chance, temporality, and nonlinear interactivity. These are also characteristics of problems that have proved to be especially intractable to classic methods of optimization. [25]

He also notes the need for researchers in the field of evolutionary computation to pay attention to the developments being made in the natural world:

> the ultimate advancement of the field will, as always, rely on the careful observation and abstraction of the natural process of evolution. [25]

The basic principles of natural evolution are quite clearly fundamental to the general success of evolutionary algorithms, in all their variants. However, the finer details can be quite different from those found in the natural mechanism [5]. For example, the evaluation of a fitness function and the selection of individuals to reproduce are two areas in which researchers have explored numerous possibilities not thought to occur in nature. Similarly, evolutionary algorithms are typically run iteratively, with all members of a population reproducing and dying simultaneously, whereas natural life-cycles proceed in a more asynchronous fashion.

Discussing the key differences between the various evolutionary algorithms, D. B. Fogel writes:

> The differences between the procedures are characterized by the typical data representations, the types of variations that are imposed on solutions to create offspring, and the methods employed for selecting new parents. Over time, however, these differences have become increasingly blurred, and will likely become of only historical interest. [26]

Some of the differences between the various algorithms require a distinction be made between 'genotypic' and 'phenotypic' spaces, concepts borrowed from biology. The representation of an individual in genotypic space is typically encoded using some finite set of symbols, such as a bit string, or a DNA strand. The behaviour of these individuals is determined by decoding the genotype, thereby mapping it into phenotypic space. The evaluation of an individual's fitness usually occurs in the phenotypic space.

3.1.1 Evolutionary Operations

In 1957, George E. P. Box proposed a "method for increasing industrial productivity", which he called 'Evolutionary Operation' [11]. He noted the similarities between the evolution of living things and the advances in industrial processes. The presence of evolutionary processes in non-biological systems has often been observed by researchers,

> Artifacts, cultures, and technologies change and evolve. There are no molecular "genes," but there is change and evolution. Research and development efforts are deliberate attempts to develop methods and technologies which bring about a speedy evolution of products [12].

More recently, such observations have lead to the development of *cultural algorithms* and *memetic algorithms* [49].

Box developed a strategy for continual improvement which made the parallels to natural evolution explicit, and thereby sought to improve the effectiveness of this process.

His model iterates through two phases, the first of which is a tight local search similar to basic hill-climbing, which operates on a subset of the variable parameters. The second phase, which Box likens to mutations, involves a team of human experts meeting at regular intervals to suggest more significant modifications to the process. Box emphasises these experts should be from different backgrounds, to promote diversity.

A termination criterion—a feature of many modern evolutionary algorithms without a natural counterpart—was also considered, again requiring human interaction:

> Only if it seemed that more would be lost than gained from the evolutionary procedure would the reintroduction of static operation be justified. [11]

Box does not explicitly consider the limitations of local optima in his parameter optimisation. However, the modifications—or mutations—suggested by the team of experts could allow the process to escape local optima.

Since the process Box proposed was to run continuously, without disrupting standard production, he restricted the local search to small variations in a regular pattern. This increased the likelihood of becoming trapped in local optima. A further precaution also stifled the evolutionary process:

> The plant manager is himself a part of the 'closed loop', thus ensuring that sensible action will be taken even in unforeseen circumstances [11].

By restricting operation to what is 'sensible', the potentially superior unknown regions remain unexplored. In contrast, natural evolution has no notion of the sensible, no manager with veto rights. Understandably, the occasional catastrophic failure may not be acceptable in a chemical plant. Such limitations serve to illuminate the importance of computers and computer simulations in the development of evolutionary algorithms.

Importantly, Box recognised natural evolution as a process which could form the basis for an iterative improvement strategy: "What we have to do is to imitate this process"[11].

3.1.2 Genetic Algorithms

Genetic algorithms are an optimisation technique which has readily borrowed terminology, processes, theory and even researchers from the study of natural evolution, specifically at the level of genes.

> The evolving entity within [genetic algorithms] ... is the genome, typically represented by a binary string. The main source of variation is the crossover of two parental strings .... Exploitation is performed ... by means of fitness proportional mating selection alone. [47]

Yet for reasons of necessity or simplicity, there remain some important differences, for example, "although genetic algorithms mimic the effects of natural selection, until now they have operated on a much smaller scale than does biological evolution" [33]. Nonetheless, they have become one of the most widely used tools in heuristic optimisation.

In the 1950s and 1960s, A. S. Fraser was using one of the early computers—the SILLIAC—to simulate evolution in genetic systems [28]. His models used bit-string representations for individuals in a population, which underwent iterative recombination and strategic selection. Each individual was assigned a phenotypic value based on some function of the bit-string, which would in turn be used for selection. Using these models, Fraser was able to empirically investigate various theories in evolutionary biology, an approach which he saw as complementary to the mathematical investigation of such processes.

> Monte Carlo analyses can never be a substitute for complete mathematical simplifications. They can, however, be regarded as complementary to the present incomplete mathematical solutions, and provide an easily used tool for those geneticist, such as myself, who cannot otherwise add to the mathematical theory of natural selection. [30]

Although Fraser's early efforts were focused on the investigation of nature, due to the broader applicability of his work, he is considered a pioneer in the field of evolutionary computation.

> By 1968, Fraser had placed his work in the context of purposive learning systems .... In retrospect, his computational procedures presaged the mechanisms that would later become common in traditional genetic algorithms. [28]

Hans J. Bremermann was also experimenting with computer simulations of genetic systems, particularly with recombination methods:

> the characteristics of offspring were determined by summing up corresponding genes in two parents. This mating procedure was limited, however, because it could apply only to characteristics that could be added together in a meaningful way. [33]

Bremermann was interested not only in the use of simulated evolution for optimisation, but also as a means of better understanding natural evolution. He mentions the apparent mathematical intractability of natural evolution: "a system of such complexity is beyond the reach of an explicit mathematical analysis" [12]. He claimed that the assumptions necessary to obtain solutions were oversimplifying, and thus the results were of diminished significance. Accordingly, he suggests computer simulation may be the more enlightening research area.

> Biology today would be unthinkable without the theory of evolution. Nevertheless there are vast areas of ignorance ... the manifold interactions of proteins with each other and with the cell environment are not well understood ... the details of the process by which anatomy and behavior of macro-organisms arise from their DNA are largely unknown. [12]

Bremermann also experimented with variants which were not based on any natural mechanism, and in 1966 claimed to have discovered "evolutionary schemes that converge much better, but with no known biological counterpart" [12]. These results prompted him to question the optimality of biological species, and to suggest they may in fact be trapped in ecological niches.

In the 1960s John H. Holland became interested in the use of evolutionary techniques to investigate adaptive processes. He was concerned with producing mathematical descriptions of strategies for the collection and application of information in unknown environments. While the models he developed are responsible for the form of current genetic algorithms, the focus was not initially on optimisation [26]. But rather, they were a tool for the study of adaptation in natural and artificial systems.

In 1992 Holland reiterated the use of simulated evolution in the investigation of nature:

> as researchers probe the natural selection of programs under controlled and well-understood conditions, the practical results they achieve may yield some insight into the details of how life and intelligence evolve in the natural world. ... Eventually artificial adaptation may repay its debt to nature by increasing researchers understanding of natural ecosystems and other complex adaptive systems. [33]

3.1.3 Evolution Strategies

In the mid sixties, Ingo Rechenberg, Hans-Paul Schwefel and Peter Bienert developed a technique for optimising the parameters in various fluid mechanics problems. Their approach was based on ideas from natural evolution, and accordingly was named *evolution strategies*.

Rechenberg, Schwefel and Bienert were graduate students at the Technical University of Berlin. They were searching for an automatic means of solving engineering problems, such as minimising drag over a surface. The conventional methods—single parameter variation, and discrete gradient search—were found to be insufficient because they could not escape local optima. Rechenberg was aware of parallels between the natural world and engineering problems:

> In cybernetics it is axiomatic that common theories can be applied to apparently widely separated fields of science. The increasingly evident points in common between biology as the theory of organisms and technology as the theory of mechanisms provide a good example of this. [43]

Making an analogy to natural evolution, Rechenberg suggested randomly varying all parameters simultaneously, followed by a selection stage. The selection compared the single parent with the single offspring, always choosing the superior. Experiments were initially conducted on physical apparatus, with manually adjustable design variables, and dice were used for random number generation. Their approach successfully found a pipe elbow shape superior to any previously known, which both confirmed the potential of the technique and encouraged further research.

In 1965 Schwefel implemented the procedure on a computer allowing more rigourous tests to be conducted. He found that the previously used discrete binomial distribution for generating random mutations led to premature stagnation, and at times provided solutions that were not even locally optimal [26]. Further experimentation led to the use of continuous variables and normally distributed random mutations. In 2002 Schwefel explained:

> Broadly accepted hereditary evidence has led to saying: *the apple does not fall far off of the tree*. A better model of variations from one generation to the next, at least for

multi-cellular individuals—an early evolutionary achievement with no simple genotype/phenotype mapping—may be a normal or Gaussian probability density distribution for phenotypic changements between generations, its maximum and expectation being centered at the respective ancestors position. [47]

Schwefel is suggesting that the variation in physical traits between parents and their offspring approximates a normal distribution. This can be contrasted with genetic mutations—both in nature and in genetic algorithms—which do not require continuity between the original and the mutated value, although the organism as a whole may be quite similar to its parent(s).

Whilst acknowledging the complexity and importance of the numerous—and possibly unknown—mechanisms between genetic code and functioning organisms, Schwefel suggests much can be learnt from simulations of evolution at higher levels of abstraction.

A descriptive or Keplerian model is sufficient for the investigation, and there is no need for a Newtonian or explicative model. [47]

Such tiered investigations are important in both the exploration and exploitation of nature, and are connected to previously mentioned ideas of complexity and emergent behaviour (Section 2.2).

Various extensions were later introduced, many of which were based on some aspect of natural evolution. Such extensions include, but are not limited to: a population of individuals (solutions), multiple offspring, recombination, variable length encoding (with duplication and deletion operators), various strategies for selecting individuals to retain for the next generation, evolution of strategy parameters (self-adaptivity), and limited life-spans [26].

Often, some limitation or shortcoming of the existing approach would lead researchers to ask: *how does nature do it?* However, since there were—and still are—many unanswered questions in the field of evolutionary biology, improvements were sought through trial-and-error experiments. Fortunately, natural evolution provided a 'shortlist' of candidate extensions which fuelled progress in this area.

Drawing from his experience using evolutionary algorithms as heuristic optimisation techniques, Schwefel has written about the insights that can be gained into the natural world by the study of simulated evolution [47]. Among these insights are the role of death and forgetting, and the importance of diversity in an uncertain world.

3.1.4 Evolutionary Programming

The technique known as Evolutionary Programming has its origins in the work of Lawrence J. Fogel, who, along with his colleagues, was developing a new approach to artificial intelligence. L. J. Fogel treated intelligence as the ability to seek goals by responding appropriately to predictions of relevant future events, as explained by D. B. Fogel:

Intelligence can be viewed as that property which allows a system to adapt its behavior to meet desired goals in a range of environments ... in that light, prediction is a keystone to intelligent behaviour because it is only through predicting the expected outcomes of alternative actions and assessing their projected worth in light of the system's purpose that a system can adapt its behaviour. [24]

At the time, many other researchers in the field of artificial intelligence were giving their attention to the object most commonly associated with intelligence, the human brain. This was pursued from two directions: neural networks and expert systems. Expert systems attempt to explicitly capture the knowledge of human experts, and as such are not generally considered 'intelligent'; in essence they could only repeat what they had been told, albeit in a useful manner.

Alternatively, research into artificial neural networks aimed to copy the hardware which was considered responsible for human intelligence. L. J. Fogel *et al.* pointed to the difficulties in directly replicating nature:

The immense complexity of the central nervous system, coupled with our incomplete knowledge of the neural and molecular mechanisms, limits our ability to replicate the biological entity which provides human intellect. Networks of threshold elements may simulate arrays of neurons, but this is a far cry from providing behavior at higher levels of abstraction. In short, the case for replicating nature in terms of physical correspondence stands on weak ground. [29]

One's ability to physically replicate biological problem-solving mechanisms is limited by the available technology. However, if the observed mechanism is a specific implementation of a universal, 'platform independent' principle, then it may be possible to study that principle, and implement it with technology that is available. Based on the state of research in the 1960s, L. J. Fogel *et al.* suggested that neural networks were unlikely to produce 'intelligent' behaviour, at least not at that point in time. This was partly because the underlying principle was insufficiently understood, and partly because the replication technology was not yet advanced enough.

As an alternative, they sought to mimic the system responsible for producing intelligence:

Man may be recognized to be but a single artifact of the natural experiment called evolution…Might it not be far wiser to model the process of evolution—iterative mutation and selection—in order to discover successively better logic for seeking the given goal under the constraint imposed by the environment? [29]

Based on this insight, L. J. Fogel *et al.* began investigating evolution-inspired approaches to producing a Finite State Machine.

To provide maximum generality, in a series of experiments, a simulated environment was described as a sequence of symbols taken from a finite alphabet. The problem was then defined to evolve an algorithm that would operate on the sequence of symbols thus far observed in such a manner as to produce an output symbol that is likely to maximize the benefit to the algorithm in light of the next symbol to appear in the environment and a well-defined payoff function. Finite-state machines provided a useful representation for the required behaviour. [24]

In L. J. Fogel's initial model, each member in a population of 'parent' finite-state machines produced a single offspring by mutation, thereby doubling the overall population. The fitness of all members was evaluated and the ‘best’ fifty percent were retained.

The link between environmental prediction and optimisation is mentioned by Atmar:

> Evolutionary optimization is not characterized by simple combinatorial optimizations. Rather, it is intrinsically composed of problems that minimize the costs of mispredicting sequences of environmental stimuli. Misprediction of a forthcoming event (surprise) is generally costly, if not occasionally lethal. [4]

While L. J. Fogel *et al.* mentioned the greater applicability of their approach [29], the move from artificial intelligence to a more widely applicable optimisation procedure was largely due to D. B. Fogel in the early 1990s.

> Models used to predict the system’s environment reflect the system’s understanding of its surroundings. These models can then be used for the sake of control by examining the projected outcomes of alternative allocations of available resources and selecting those which are believed to be most favourable. [24]

D. B. Fogel showed how the evolutionary programming technique can be applied to pattern discovery, system identification and automatic control, performing a form of numerical optimisation in each.

The development of evolutionary programming is a case of strong inspiration, yet many important aspects of natural evolution are excluded. For example, the typical formulation has no means by which members can combine or share their ‘knowledge’. Furthermore, there is some ambiguity as to whether the individuals in a population correspond to individual organisms or entire species. Schwefel suggests the lack of recombination in typical implementations is because the model is based on the evolution of species:

> [Evolutionary programming] intends to model the birth and death of species and thus, generally does not include recombination.[47]

But if this is the case, then there are some inconsistencies with other elements of the approach. For example, natural mutation is a phenomenon that occurs at the genetic level, and as such it is inaccurate to talk about the mutation of the individual, of the species more so. Variation at the species level, based on the fossil record, can be characterised by long periods of stability interspersed with occasional periods of rapid change. It may be possible to draw an analogy with Gould and Eldredge’s theory of *punctuated equilibrium*[1], but there are no suggestions that the initial conception was inspired by these ideas.

3.1.5 Future Directions

Advances in our understanding of natural evolutionary processes are often unnoticed by optimisation practitioners. There have, however, been exceptions. Whitacre

[1] An alternative optimisation heuristic, *extremal optimisation* [8], is inspired by the Bak–Sneppen model of evolution, which reproduces various events thought to occur in natural evolution, including punctuated equilibrium.

et al. [52] have attempted to bring the most recent work in complex biological systems into the field of evolutionary computation. Specifically, self-organised locality and the effects of gene interactions. They show that "sustainable coexistence of genetically distinct individuals" can be achieved in an emergent fashion using nature-inspired mechanisms.

> Population diversity was not imposed upon the [evolutionary algorithm] as is traditionally done but instead emerges in the system as a natural consequence of population dynamics. The environmental conditions which enable sustainable diversity are similar to what is observed in complex biological systems. [52]

Another promising area of evolutionary algorithm research—and many other nature inspired heuristics—involves "implementation on parallel machines, for evolution is an inherently parallel process" [25]. The separation of populations across multiple processors with intermittent migrations is a natural extension of the evolutionary analogy.

3.2 *Particle Swarm Optimisation*

Particle swarm optimisation is a population based method which has received a lot of attention since it was published in 1995. Kennedy and Eberhart describe how the "method was discovered through simulation of a simplified social model", and provide a detailed account of the development process. They write,

> [Particle swarm optimisation] can be implemented in a few lines of code. It requires only primitive mathematical operators, and is computationally inexpensive in terms of both memory requirements and speed. Early testing has found the implementation to be effective with several kinds of problems. [37]

Kennedy and Eberhart mention the work of C. W. Reynolds [44], who had made progress in the realistic simulation of 'flocks, herds and schools' for computer animation. They also note the work of Heppner and Grenander, who write in their 1990 paper:

> Certain small birds such as pigeons, starlings, and shorebirds fly in coordinated flocks that display strong synchronization in turning, initiation of flight, and landing. Experimental efforts to find leaders in such flocks have to date failed. We propose that synchronization of movement may be a byproduct of "rules" for movement followed by each bird in the flock. Accordingly, we have developed a computer-simulated bird flock employing stochastic differential equations which demonstrates realistic "flocking" behavior. [31]

The success of these models for movement followed from the assumption that the emergent flocking behaviour was a product of the efforts of individuals to optimise their position relative to neighbours. Kennedy and Eberhart then extended this concept to social interactions:

> It does not seem a too-large leap of logic to suppose that some same rules underlie animal social behavior, including herds, schools, and flocks, and that of humans. [37]

This is an informal argument, based on intuition and a form of pattern matching. They elaborate:

It seems reasonable, in discussing human social behavior, to map the concept of change into the bird/fish analogy of movement. This is consistent with the classic Aristotelian view of qualitative and quantitative change as types of movement. Thus, besides moving through three-dimensional physical space, and avoiding collisions, humans change in abstract multidimensional space, collision-free. [37]

As such, their investigation of one part of nature—social interactions—is somewhat inspired by discoveries in a different, albeit related, part of nature.

From this point Kennedy and Eberhart began experimenting with variations in the low level behaviour of individuals. They consciously "thought of agents as collision-proof birds", with the tentative goal of producing "graceful but unpredictable choreography of a bird flock" [37]. This experimentation involved the introduction of simple rules for individuals to follow at each iteration. While one rule, *neighbour tracking*, was based on a mechanism thought to exist in nature, another, *craziness*, was more arbitrarily invented to produce the desired 'lifelike' behaviour.

Unsatisfied with this solution, Kennedy and Eberhart gave further consideration to the dynamic forces used in Heppner's simulation, and again they drew inspiration from nature:

Heppner's birds knew where their roost was, but in real life birds land on any tree or telephone wire that meets their immediate needs. Even more importantly, bird flocks land where there is food. How do they find food? [37]

They hypothesised that birds make some form of evaluation of their current position, retain the location of the best position they have encountered, and eagerly share this information with the whole flock. To allow the evaluation of their current position they introduced the equation:

$$Eval. = \sqrt{(presentx - 100)^2} + \sqrt{(presenty - 100)^2}, \tag{1}$$

where lower values are considered better. As such, (100,100) was the target.

Without going into the specifics of the algorithm, after some parameter adjustments the individuals exhibited realistic approach behaviour. But perhaps more importantly, Kennedy and Eberhart recognised the parallels to an optimisation task. At this stage they began removing features of the algorithm to find the essential components. Notably, the *flock* of *individuals*, who paid attention to the actions of neighbours, became a *swarm* of *particles*, who paid attention to the global best. The focus shifted to the emergent properties of the swarm, and thus can be considered a single, purposeful entity.

It was found that their flocks could find the optimum in a simple, two-dimensional, linear field. They proceeded to prune parts of the simulation unimportant to optimisation. What is relevant to our discussion here is the extent of the inspiration from nature. While it is true that Kennedy and Eberhart were experimenting with models of nature, it was not necessary for the models to accurately represent the natural process in order to produce the optimising behaviour. In their own words:

> the social metaphor is discussed, though the algorithm stands without metaphorical support. [37]

The benefits of interdisciplinary collaboration are also mentioned:

> The authors of this paper are a social psychologist and an electrical engineer. The particle swarm optimiser serves both of these fields equally well. Why is social behaviour so ubiquitous in the animal kingdom? Because it optimises. What is a good way to solve engineering optimisation problems? Modelling social behaviour. [37]

Pithy as this statement may be, in practice the situation is more complicated. It is difficult to say exactly what is being optimised by social behaviour, and how we define ‘good’ solution methods is equally challenging. Nonetheless, from this example we can see the collaboration of researchers from different backgrounds has the potential for mutual gains.

If we look at how the understanding of the relevant natural phenomena has progressed since the discovery of the particle swarm optimiser, we can further construct a picture of how the inspiration has occurred. Current studies in collective animal behaviour—such as bird flocking—acknowledge that these phenomena are still not well understood. Various models have been proposed and evaluated somewhat subjectively by visual inspection. Recent work by Ballerini *et al.* [6] has attempted to capture detailed three-dimensional position information of a large flock of starlings. They hope that this will enable better testing of proposed models. Similarly, current work in social interaction modelling—an undoubtedly more complicated challenge—shows there is much still to be discovered [7]. Accordingly, we find a nature-inspired algorithm can still be successful even if it is based on an incorrect or incomplete understanding of nature.

Since the Kennedy and Eberhart paper was published, there has been a vast spectrum of extensions and modifications to the original. Each new algorithm published generally reports some superior behaviour over some set of problems. These variations usually retain the metaphorical language, and often add to the vocabulary (e.g. *scouts*, *sentinels*, etc.). It seems reasonable then to attribute some of the popularity of this method to the ease with which the solution process can be perceived.

3.3 Ant Colony Optimisation

The behaviours exhibited by ant colonies when locating and collecting food, or ‘foraging’, were a ‘strong’ inspiration for a class of techniques known as *ant colony optimisation*, or *ant inspired algorithms*. However, the diversity of behaviours observed in different species of ants suggest

> it is more accurate to describe ACO as being inspired by the recruitment strategy of ants which use chemical markers (pheromone trails) to mark the location of a rich food source such as the *Iridomyrmex humilis* (Argentine ant) species. [2]

Differences in foraging strategies arise from differences in the evolutionary environments. The strategy employed by *I. humilis*, which arose in regions abundant in food supplies, relies on optimal allocation of resources [2]. Another species of ant, *cataglyphis*, evolved in a harsh desert environment, employs a strategy which places more importance on the previous success of each individual ant. The *cataglyphis* species does not use intra-colony communication, but rather depends on the memory of individuals. These differences accommodate the expected pay-off for taking risks in the respective environments.

The emergent behaviour of *I. humilis* was demonstrated by Deneubourg *et al.* in their double bridge experiment [18]. Using a simple model for the behaviour of individual ants, a Monte Carlo simulation was able to reproduce the complex collective behaviour of an actual colony of ants as they explore their environment.

> The simplicity of the mechanisms involved, and Occam's principle of scientific parsimony, leads us to speculate that only two conditions are necessary for this phenomenon to appear. The workers outside the nest must continually lay pheromone and must leave the nest together in sufficient numbers. [18]

The platform independence of this process can be observed in "the formation of trails, for example, [made] by mammals in scrub or grassland or even by students on a snowed-under campus!"[18]. The advantages of the emergent, distributed approach to foraging are more apparent when compared with a centralised strategy.

> Indeed if one considers how many different and complicated "instructions" would be necessary explicitly to coordinate a swarm, one may readily appreciate both the genetic economy and the added reliability that come with such simplicity. [18]

This model of ant behaviour was the basis for the stochastic, population based, combinatorial optimisation technique proposed by Dorigo *et al.* [20], which they called *Ant System.*

> One of the problems studied by ethologists was to understand how almost blind animals like ants could manage to establish shortest route paths from their colony to feeding sources and back. ... The algorithms that we are going to define ... are models derived from the study of real ant colonies. [20]

While Deneubourg *et al.* were seeking to understand the mechanisms behind the behaviour of *I. humilis*, Dorigo *et al.* focused on producing an optimisation technique:

> As we are not interested in simulation of ant colonies, but in the use of artificial ant colonies as an optimization tool, our system will have some major differences with a real (natural) one:
>
> - artificial ants will have some memory,
> - they will not be completely blind,
> - they will live in an environment where time is discrete.

Once a mechanism has been extracted from a natural system and formulated into an algorithm, experimentation with the various parameters typically focuses on improving the performance of the algorithm, with little regard for remaining faithful to the natural origins.

> But the ants' success in collectively selecting the shortest path is only statistical: the colony may occasionally get 'stuck' on a longer path if by chance the longer path is the first one to be marked. In using the 'trail laying–trail following' metaphor for optimization purposes, computer scientists have found it essential to improve the convergence properties of their algorithms by artificially increasing the rate of pheromone evaporation beyond biological plausibility. [10]

These points of difference make the technique more suited to the problems to which it is applied, and the system on which it is implemented. Nonetheless, the underlying mechanism is retained, and furthermore, "the ant colony metaphor can be useful to explain [their] model" [20].

Since ant colony algorithms appeared, they have been applied with great success to many problems, achieving state-of-the-art results for the quadratic assignment problem, sequential ordering, vehicle routing, and others [15].

3.4 Artificial Neural Networks

> The most extensive computation known has been conducted over the last billion years on a planet-wide scale: it is the evolution of life. The power of this computation is illustrated by the complexity and beauty of its crowning achievement, the human brain. [46]

Artificial neural networks (ANNs) are a group of computational methods based on the central nervous system found in many organisms. They consist of a large number of interconnected computational elements, each of which is defined by relatively simple input-output relations [34]. They are well suited to clustering, classification, pattern recognition, and function approximation problems [13], and as such are found in a broad variety of applications. Of particular interest to us is their application to combinatorial optimisation problems, an area where they have achieved mixed success [50]. Nonetheless, the history of ANNs—even outside optimisation problems—is worth discussing for what it reveals about the process of natural inspiration.

In 1943 the neuropsychiatrist Warren McCulloch and the mathematician Walter Pitts published a paper—*A Logical Calculus of the Ideas Immanent in Nervous Activity*—which would later be considered "a landmark event in the history of cybernetics, and fundamental to the development of cognitive science and artificial intelligence" [1]. In it they write:

> Many years ago one of us ... was led to conceive of the response of any neuron as factually equivalent to a proposition which proposed its adequate stimulus. [40]

In 1923 McCulloch was working on models of human thought, specifically propositional logic. These investigations played an important role in the formulation of the neuron model twenty years later.

My object, as a psychologist, was to invent a kind of least psychic event, or 'psychon,' that would have the following properties: First, it was to be so simple an event that it either happened or else it did not happen. Second, it was to happen only if its bound cause had happened ... that is, it was to imply its temporal antecedent. Third, it was to propose this to subsequent psychons. Fourth, these were to be compounded to produce the equivalents of propositions concerning their antecedents. (McCulloch, 1965a, p. 8) [cited by Abraham [1]].

Despite these origins as a model of actual neuronal process, in [40] there is no ambiguity about the explanatory nature of the proposed model.

But one point must be made clear: neither of us conceives the formal equivalence to be a factual explanation. [40]

That is, the proposed neuronal model is not considered a sufficient description of the workings of the human brain. McCulloch and Pitts go on to emphasise the strength of their conclusions, regardless of the correspondence to the natural phenomenon:

The importance of the formal equivalence lies in this: that the alterations actually underlying facilitation, extinction and learning in no way affect the conclusions which follow from the formal treatment of the activity of nervous nets, and the relations of the corresponding propositions remain those of the logic of propositions. [40]

Many improvements and variations have been made since this initial proposal, and the McCulloch-Pitts model is now thought of as a specific instance of the current general model used in the field of artificial intelligence [13]. The field of neuroscience has also come a long way; recent developments include the persistence of neuro-plasticity in adults [21], and the role of mirror neurons in learning [45]. However, while occasional parallels and idea exchanges can be seen, they have largely been pursued independently.

In 1975 Michael A. Arbib published *Artificial Intelligence and Brain Theory: Unities and Diversities*, in which he wrote:

While many workers in the field of [artificial intelligence] believe that introspection on the way they solve problems can help them program computers to solve those problems, a majority of such workers feel that the analysis of brain mechanisms involved in intelligent behaviour is irrelevant to them. Rather, they take as their maxim "Airplanes do not flap their wings". [3]

Noting the overwhelming absence of overlap in the respective literature, Arbib argued that the fields of neuroscience and artificial intelligence were suffering from a premature divergence, and that a unified perspective would be beneficial for both.

In 1985 Hopfield and Tank extended ANNs to produce solutions to certain combinatorial optimisation problems. They claimed to identify a link between the problems "in engineering and commerce, and in perceptual problems which must be rapidly solved by the nervous systems of animals."

Recognizing that the nature of perceptual problems is similar to other optimization problems..., we will show here how to organize a computational network of extensively interconnected nonlinear analog neurons so that it will solve a well characterized, but non-biological, optimization problem. [35]

They consider recognition tasks solved by the nervous system to require “truly immense” computational power. The importance of parallel processing in achieving this feat is noted, and emphasis is put on the analog nature of the biological computation mechanism. This analog computation is claimed to improve speed at the cost of accuracy, however, the “computational load is meaninglessly increased by high digital accuracy” [35]. This assertion is justified by pointing out the inaccuracy in the definition of the ‘good’ solution being searched for.

Hopfield and Tank credit other authors as identifying the connection between perceptual problems and optimisation problems. Poggio *et al.* [42] were working on computer vision techniques and posed the image regularisation problem as an optimisation problem, and discussed its plausibility as a biological process.

Despite the initial enthusiasm for the Hopfield and Tank approach, attempts to reproduce their results “raised doubts as to the reliability and validity of the H-T approach to solving [combinatorial optimisation problems]” [50]. It seemed that the majority of solutions produced were infeasible. In the years that followed some researchers abandoned the approach while others sought rectifying modifications. While the success of ANNs for optimisation is inconclusive, their origins are an example of the joint investigation and exploitation of nature. This can be seen in [42]’s description of the goals of computer vision:

> [The aims of the field of computer vision are] to develop image understanding systems, which automatically construct scene descriptions from image input data, and to understand human vision.

Again we observe a mutually beneficial investigation of a natural phenomena; to understand it in its own right, and to utilise the mechanism.

4 Conclusions

We have seen that nature provides inspiration for optimisation algorithms in numerous ways. Some natural systems exhibit behaviour which can be accurately described as optimum-seeking. From these we can extract the essential features and achieve qualitatively similar behaviour on some application of interest.

Other natural systems can be more accurately described as metaphor inspiring. They provide a conceptual framework in which researchers can more easily comprehend problems and more freely produce creative solutions. Additionally, there are natural systems which both inspire creative thought and provide useful mechanisms, such as *swarm intelligence* methods.

It has also been shown that interdisciplinary collaboration can be mutually beneficial. Theories about natural phenomena can inspire heuristic algorithms, and the simulation of natural phenomena can help evaluate theories about nature.

Finally, as the technology to produce massively parallel computing devices is improving, algorithms which fully exploit these systems are required. Many of the complex problems in nature are solved using essentially parallel techniques—swarms, herds, neural networks, immune systems, societies. Accordingly, future heuristics for hard optimisation problems are likely to benefit from close observation of nature.

Acknowledgements. The comments provided by the anonymous referees have helped improve this chapter, and the authors are sincerely grateful.

References

1. Abraham, T.H.: (physio)logical circuits: The intellectual origins of the mcculloch–pitts neural networks. Journal of the History of the Behavioral Sciences 38(1), 3–25 (2002)
2. Angus, D.: Ant colony optimisation: From biological inspiration to an algorithmic framework. Tech. rep., Swinburne University of Technology (2006)
3. Arbib, M.: Artificial intelligence and brain theory: Unities and diversities. Annals of Biomedical Engineering 3(3), 238–274 (1975)
4. Atmar, W.: Notes on the simulation of evolution. Neural Networks, IEEE Transactions on 5(1), 130–147 (1994)
5. Bäck, T., Schwefel, H.P.: An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation 1(1), 1–23 (1993)
6. Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., Orlandi, A., Parisi, G., Procaccini, A., Viale, M., Zdravkovic, V.: Empirical investigation of starling flocks: a benchmark study in collective animal behaviour. Animal behaviour 76, 201–215 (2008)
7. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.U.: Complex networks: Structure and dynamics. Physics Reports-Review Section of Physics Letters 424(4-5), 175–308 (2006)
8. Boettcher, S., Percus, A.: Nature's way of optimizing. Artificial Intelligence 119(1-2), 275–286 (2000)
9. Bohm, D., Peat, D.: Science, Order, and Creativity. Routledge (2000)
10. Bonabeau, E., Dorigo, M., Theraulaz, G.: Inspiration for optimization from social insect behaviour. Nature 406(6791), 39–42 (2000)
11. Box, G.E.P.: Evolutionary operation: A method for increasing industrial productivity. Applied Statistics 6(2), 81–101 (1957)
12. Bremermann, H.J., Rogson, M., Salaff, S.: Global properties of evolution processes. In: Fogel, D.B. (ed.) Evolutionary Computation: The Fossil Record, pp. 314–352. Wiley/ IEEE Press (1998)
13. de Castro, L.N.: Fundamentals of natural computing: an overview. Physics of Life Reviews 4(1), 1–36 (2007)
14. Coello, C.A.C., Cortés, N.C.: Solving multiobjective optimization problems using an artificial immune system. Genetic Programming and Evolvable Machines 6(2), 163–190 (2005)
15. Cordón, O., Herrera, F., Stützle, T.: A review of the ant colony optimization metaheuristic: Basis, models and new trends. Mathware & Soft Computing 9, 141–175 (2002)
16. Crosby, J.L.: Computers in the study of evolution. In: Fogel, D.B. (ed.) Evolutionary Computation: The Fossil Record, pp. 230–254. Wiley/ IEEE Press (1998)
17. Darwin, C.: The Origin of Species. Avenel Books (1979)
18. Deneubourg, J.L., Aron, S., Goss, S., Pasteels, J.M.: The self-organizing exploratory pattern of the argentine ant. Journal of Insect Behavior 3(2), 159–168 (1990)
19. Dobzhansky, T.: Biology, molecular and organismic. American Zoologist 4, 443–452 (1964)

20. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics 26(1), 29–41 (1996)
21. Draganski, B., Gaser, C., Busch, V., Schuierer, G., Bogdahn, U., May, A.: Neuroplasticity: Changes in grey matter induced by training - newly honed juggling skills show up as a transient feature on a brain-imaging scan. Nature 427(6972), 311–312 (2004)
22. El-Hani, C.N., Emmeche, C.: On some theoretical grounds for an organism-centered biology: Property emergence, supervenience, and downward causation. Theory in Biosciences 119(3-4), 234–275 (2000)
23. Flake, G.W.: The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation. MIT Press, Cambridge (2000)
24. Fogel, D.B.: Evolutionary programming–an introduction and some curent directions. Statistics and Computing 4(2), 113–129 (1994)
25. Fogel, D.B.: An introduction to simulated evolutionary optimization. IEEE Transactions on Neural Networks 5(1), 3–14 (1994)
26. Fogel, D.B. (ed.): Evolutionary Computation: The Fossil Record. Wiley-IEEE Press (1998)
27. Fogel, D.B.: What is evolutionary computation? Spectrum, IEEE 37(2), 26, 28–32 (2000)
28. Fogel, D.B.: In memoriam Alex S. Fraser [1923-2002]. IEEE Transactions on Evolutionary Computation 6(5), 429–430 (2002)
29. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial intelligence through a simulation of evolution. In: Fogel, D.B. (ed.) Evolutionary Computation: The Fossil Record, pp. 230–254. Wiley/ IEEE Press (1998)
30. Fraser, A.S.: Monte carlo analyses of genetic models. Nature 181, 208–209 (1958)
31. Heppner, F., Grenander, U.: A stochastic nonlinear model for coordinated bird flocks. In: The Ubiquity of chaos, Washington, D.C, AAAS (1990)
32. Holland, J.H.: Adaptation in natural and artificial systems, 3rd edn. MIT Press, Cambridge (1992)
33. Holland, J.H.: Genetic algorithms. Scientific American 267, 66–72 (1992)
34. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. 79, 2554–2558 (1982)
35. Hopfield, J.J., Tank, D.W.: Neural computation of decisions in optimization problems. Biological Cybernetics 52(3), 141–152 (1985)
36. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments-a survey. IEEE Transactions on Evolutionary Computation 9(3), 303–317 (2005)
37. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
38. Koch, C., Laurent, G.: Complexity and the nervous system. Science 284(5411), 96–98 (1999)
39. Lotka, A.J.: Contribution to the energetics of evolution. Proceedings of the National Academy of Sciences of the United States of America 8(6), 147–151 (1922)
40. McCulloch, W., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biology 5(4), 115–133 (1943)
41. Perkins, D.: Archimedes' Bathtub. W.W. Norton & Company (2000)
42. Poggio, T., Torre, V., Koch, C.: Computational vision and regularization theory. Nature 317(6035), 314–319 (1985)
43. Rechenberg, I.: Cybernetic solution path of an experimental problem. In: Fogel, D.B. (ed.) Evolutionary Computation: The Fossil Record. Wiley/ IEEE Press (1998)

44. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. SIGGRAPH Comput. Graph 21(4), 25–34 (1987)
45. Rizzolatti, G., Craighero, L.: The mirror-neuron system. Annual review of neuroscience 27, 169–192 (2004)
46. Rogers, D.: Weather prediction using a genetic memory. Tech. rep., Research Institute for Advance Computer Science, NASA Ames Research Center (1990)
47. Schwefel, H.P.: Deep insight from simple models of evolution. Biosystems 64(1-3), 189–198 (2002)
48. Simonton, D.K.: Creativity in Science. Cambridge University Press, Cambridge (2004)
49. Smith, J.E.: Coevolving memetic algorithms: A review and progress report. IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics 37(1), 6–17 (2007)
50. Smith, K.A.: Neural networks for combinatorial optimization: a review of more than a decade of research. INFORMS J. on Computing 11(1), 15–34 (1999)
51. Smith, T., Husbands, P., Layzell, P., O'Shea, M.: Fitness landscapes and evolvability. Evolutionary Computation 10(1), 1–34 (2002)
52. Whitacre, J.M., Sarker, R.A., Pham, Q.T.: The self-organization of interaction networks for nature-inspired optimization. IEEE Transactions on Evolutionary Computation 12(2), 220–230 (2008)
53. Wilson, E.O.: The Diversity of Life. Belknap Press of Harvard University Press, Cambridge (1992)
54. Wolpert, D.H., MacReady, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)
55. Wright, S.: Evolution in mendelian populations. Bulletin of Mathematical Biology 52(1), 241–295 (1990)

The Evolutionary-Gradient-Search Procedure in Theory and Practice

Ralf Salomon and Dirk V. Arnold

Abstract. The pertinent literature controversially discusses in which respects evolutionary algorithms differ from classical gradient methods. This chapter presents a hybrid, called the evolutionary-gradient-search procedure, that uses evolutionary variations to estimate the gradient direction in which it then performs an optimization step. Both standard benchmarks and theoretical analyses suggest that this hybrid yields superior performance. In addition, this chapter presents *inverse mutation*, a new concept that proves particularly useful in the presence of noise, which is omnipresent in almost any real-world application.

1 Introduction

In the field of continuous parameter optimization, an optimization algorithm aims at finding a set of n real-valued parameters x_i^o, also called optimizer $\mathbf{x}^o$, such that an objective function $f(x_1^o,\ldots,x_n^o) = f(\mathbf{x}^o)$ assumes an optimal value. Depending on the particular application, the term "optimal value" refers to an overall minimum $\forall\mathbf{x} : f(\mathbf{x}^o) \leq f(\mathbf{x})$ or an overall maximum $\forall\mathbf{x} : f(\mathbf{x}^o) \geq f(\mathbf{x})$. Without loss of generality, it is sufficient to consider only minimization tasks, since maximizing $f()$ is equivalent to minimizing $-f()$. In practice, many problems impose further constraints on the parameters, also called feasible search space, that have to be taken into account.

In case the objective function cannot be explicitly solved for the parameters x_i, the designer must resort to iterative methods where a sequence $\mathbf{x}_{t\geq 1}$ is to be generated. The goal is that in the limit $\lim_{t\to\infty} \|\mathbf{x}_t - \mathbf{x}^o\| \leq \varepsilon$, the search points $\mathbf{x}^t$ arrive

Ralf Salomon
University of Rostock, Germany
e-mail: `ralf.salomon@uni-rostock.de`

Dirk V. Arnold
Dalhousie University, Canada
e-mail: `dirk@cs.dal.ca`

arbitrarily close to the optimizer $\mathbf{x}^o$. Depending on the problem's nature and the chosen requirements, it might also be sufficient to get arbitrarily close to a local optimum.

The very many algorithms provided by the pertinent literature on optimization can be categorized by the way in which they utilize knowledge gathered in previous steps to generate new search points $\mathbf{x}_{t+1}$. The *Monte Carlo search,* for example, generates *all* search points at random, and thus, does not utilize any previous search points at all.

Newton's gradient search method, also known as steepest descent, uses the first derivatives $\partial f/\partial x_i$ along each coordinate axis to determine the n-dimensional gradient $\mathbf{g}$

$$\mathbf{g} = \nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)^T . \tag{1}$$

If the objective function is not given in an explicit analytical form, the first derivatives can be approximated by n differences, which require $n+1$ experiments (i.e., function evaluations):

$$\partial f/\partial x_i \approx \frac{f(x_1^t, \ldots, x_i^t + \Delta x, \ldots, x_n^t) - f(x_1^t, \ldots, x_i^t, \ldots, x_n^t)}{\Delta x} . \tag{2}$$

At any point $\mathbf{x}$, the gradient $\mathbf{g}$ always points in the direction of the maximal increase of $f(\mathbf{x})$. Hence, it is always perpendicular to the (n-1)-dimensional hyper surface $f(\mathbf{x}) = c$ with constant objective function values. Thus, by repeatedly subtracting sufficiently small fractions η of the locally calculated gradients $\mathbf{g}_t$

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \mathbf{g}_t = \mathbf{x}_t - \eta \nabla f(\mathbf{x}_t) , \tag{3}$$

the steepest-descent method converges to the next (local) optimum of any continuously differentiable objective function $f(\mathbf{x})$ from any initial point $\mathbf{x}_0$. The literature [12, 14] provides numerous programming examples and acceleration methods.

Evolutionary algorithms, such as genetic algorithms [10], evolutionary programming [8, 9], and evolution strategies [15, 21] (see [5] for a comparison of these methods), operate in a different manner. They all maintain a population of μ search points $\mathbf{x}_t^{1..\mu}$ from which they generate λ offspring $\mathbf{x}_t^{1..\lambda}$ by applying random variation operators, such as mutation and recombination. After assigning a fitness or error value by using the objective function $f(\cdot)$, specific offspring are selected as parents for the next generation.

At first glance, gradient descent methods and evolutionary algorithms seem to be two frameworks that do not have much in common. Some believe that evolutionary algorithms are not gradient descent methods [6], whereas others believe they are [15] or partly work like them [18]. Regardless of a particular perception, the following property can be noted: gradient descent methods use all n experiments (fitness evaluations) to calculate the direction of progress, i.e. the gradient direction, whereas evolutionary algorithms advance towards better fitness values by *selecting* offspring with above-average fitness values. In other words, since mutation and recombination are normally *unbiased* random variations, selection is *the* mechanism with which evolutionary algorithms proceed towards better solutions.

As has been indicated above, selection is a key element of evolutionary algorithms for advancing to better solutions. It is interesting to note though that selection is inherently linked to discarding potentially valuable information. This raises the question of whether or not *all* offspring can be easily used to gain progress. To provide a partial answer, Section 2 describes a hybrid method, called evolutionary-gradient-search (EGS) procedure, that incorporates elements from both approaches. The EGS procedure is a *hybrid* in that it estimates the local gradient direction $\mathbf{g}$ by applying random variations (i.e., mutations), and then *deterministically* searches along that direction. The results, as presented in Section 3, indicate that indeed all offspring can be beneficially utilized. Using all individuals rather than some selected ones improves the procedure's performance.

Experimental results are important in order to validate a new concept. However, experiments are limited to the investigation of certain aspects. Section 4 thus presents a short theoretical analysis, which supports the experimental findings. Further theoretical analyses, as presented in Section 5, indicate a progressive performance degradation in case of noisy fitness evaluations. Since noise is present in virtually any practical application, Section 5 analyses the main mechanisms responsible for the observable performance degradation. In order to overcome these limits, Section 6 discusses the concept of inverse mutations.

The pertinent literature on traditional and evolutionary optimization provides a large number of possible enhancements, such as the momentum term, individual step sizes, and correlated mutations. Section 7 discusses the incorporation of some of these into the EGS procedure. Finally, Section 8 concludes with a brief discussion.

2 The EGS Procedure

This section starts off with an informal description of the EGS procedure's main concepts. Figure 1 presents a generic situation, which allows for the following observations:

1. The three randomly generated mutation vectors $\mathbf{z}_t^{(1..3)}$ lead to superior individuals $\mathbf{y}_t^{(1..3)}$ with fitness values $f(\mathbf{y}_t^{(1..3)}) < f(\mathbf{x}_t)$.
2. The three other mutations $\mathbf{z}_t^{(4..6)}$ result in inferior individuals $\mathbf{y}_t^{(4..6)}$ with fitness values $f(\mathbf{y}_t^{(4..6)}) > f(\mathbf{x}_t)$.
3. The fitness advantage or disadvantage $f(\mathbf{y}_t^{(i)}) - f(\mathbf{x}_t)$ depends on the mutation vector $\mathbf{z}_t^{(i)}$ and is thus different for every offspring $\mathbf{y}_t^{(i)}$, which is also called a trial point.
4. The fitness difference $f(\mathbf{y}_t^{(i)}) - f(\mathbf{x}_t)$ is negative for superior trial points and positive for inferior ones. Hence, the weighted trial point $(f(\mathbf{y}_t^{(i)}) - f(\mathbf{x}_t))\mathbf{z}_t^{(i)}$ always points away from the optimum.

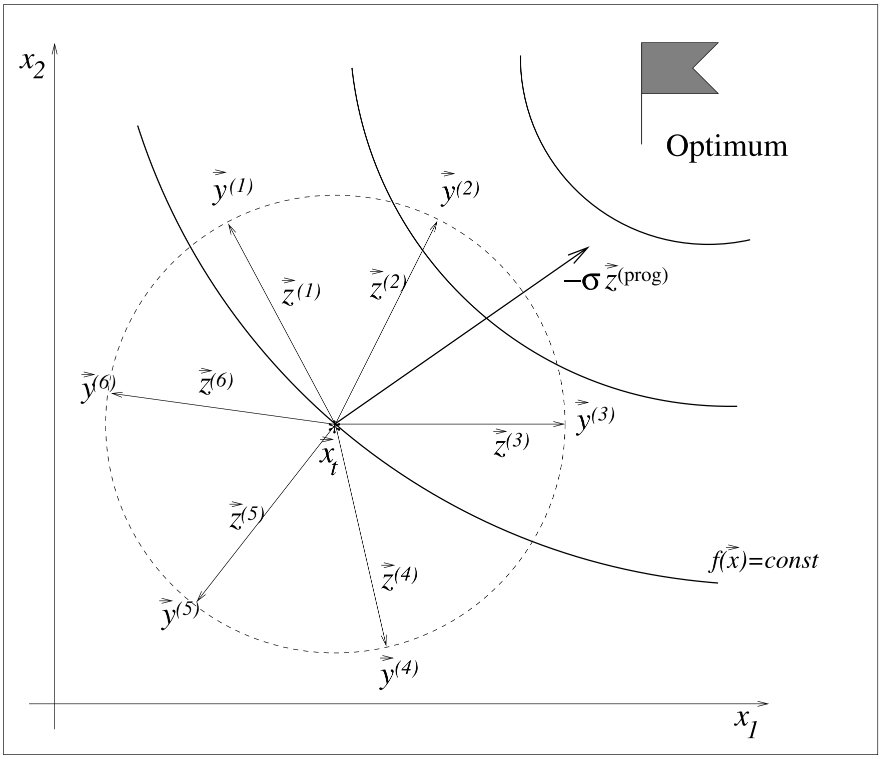


Fig. 1 In the displayed situation, trial points $\mathbf{y}^{(1..3)}$ yield some progress, whereas trial points $\mathbf{y}^{(4..6)}$ yield candidates with inferior fitness than the parents. However, candidates $\mathbf{y}^{(4..6)}$ suggest that some progress might be attainable in the reversal directions $-\mathbf{z}^{(4)}$, $-\mathbf{z}^{(5)}$, and $-\mathbf{z}^{(6)}$

As has already been argued above, most evolution strategies, such as a (3,6)-evolution strategy would probably discard those offspring that "go into the wrong direction". By contrast, however, the EGS procedure *assumes* that inferior offspring, such as $\mathbf{y}_t^{(4..6)}$ in Figure 1, suggest that some progress may be obtained in the reversal direction $-\mathbf{z}_t^{(4)}$, $-\mathbf{z}_t^{(5)}$, and $-\mathbf{z}_t^{(6)}$. Based on this observation, it estimates the true local gradient direction $\mathbf{g}_t$ by using *all* mutation vectors $\mathbf{z}_t^{(i)}$, which are weighted by the fitness difference $f(\mathbf{y}_t^{(i)}) - f(\mathbf{x}_t)$. After estimating the gradient direction $\mathbf{g}_t$, it applies a rather traditional gradient-descent step $\mathbf{x}_{t+1} = \mathbf{x}_t - \sigma_{t+1}\mathbf{g}_t$ at the current search point $\mathbf{x}_t$. In summary, the EGS procedure uses populations of offspring to explicitly estimate the gradient direction in which it tries to advance to the optimum. It then collapses the entire population to only one individual in order to apply a gradient-descent step to a single search point. The remainder of this section focuses on a rather formal description of the procedure.

In each iteration (generation) t, the procedure starts off at one particular search point, denoted as $\mathbf{x}_t$. In its simplest form, it then applies the following operations:

1. Generate $i{=}1,\ldots,\lambda$ offspring $\mathbf{y}_t^{(i)}$ as trial points

$$\mathbf{y}_t^{(i)} = \mathbf{x}_t + \sigma \mathbf{z}_t^{(i)} \tag{4}$$

from the current point $\mathbf{x}_t$ at time step t, with $\sigma{>}0$ denoting a step size, $\mathbf{z}^{(i)}$ denoting a mutation vector consisting of n independent, normally distributed components.
2. Estimate the gradient direction $\tilde{\mathbf{g}}_t$:

$$\tilde{\mathbf{g}}_t = \sum_{i=1}^{\lambda} \left(f(\mathbf{y}_t^{(i)}) - f(\mathbf{x}_t) \right) \mathbf{z}_t^{(i)} \; . \tag{5}$$

3. Perform a step

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \sigma_{t+1} \sqrt{n} \frac{\tilde{\mathbf{g}}_t}{\|\tilde{\mathbf{g}}_t\|} = \mathbf{x}_t - \sigma_{t+1} \mathbf{z}_t^{(\mathrm{prog})} \; , \tag{6}$$

with $\mathbf{z}_t^{(\mathrm{prog})} = \sqrt{n}\tilde{\mathbf{g}}_t / \|\tilde{\mathbf{g}}_t\|$ denoting the progress vector.

The EGS procedure can self-adapt the step size σ_t by performing the following simple test (see, also, [18]):

$$\sigma_{t+1} = \begin{cases} \sigma_t \, \zeta & \text{if} f(\mathbf{x}_t - \sigma_t \, \zeta \, \mathbf{z}_t^{(\mathrm{prog})}) \le f(\mathbf{x}_t - (\sigma_t / \zeta) \, \mathbf{z}_t^{(\mathrm{prog})}) \\ \sigma_t / \zeta & \text{otherwise} \end{cases} \tag{7}$$

with $\zeta \approx 1.8$ denoting a variation factor (see, also, [20]). The adaptation step in Eq. (7) requires two function evaluations. However, since one of the two test steps is equivalent to the actual progress step of Eq. (6), the self-adaptation of the step size σ_t requires only one additional function evaluation, which is small in comparison to the number λ of trial points. That is, the procedure's computational cost is $\lambda + 2$ function evaluations per iteration, which already includes the necessary self-adaptation of the step size σ_t.

3 Basic Behavior

The purpose of this section is to examine the procedure's basic behavior and the possible benefits of using all offspring (for estimating the local gradient direction). Unless otherwise stated, the procedure has used the following parameter settings: $\sigma_0 = 0.1$, $\zeta = 1.8$, $n = 10$ dimensions, $\mathbf{x}_0 = (1000,\ldots,1000)^T$, and $t = 100$ steps averaged over 50 independent trials. In these experiments, the initial step size σ_0 is deliberately ill-set in order to demonstrate the procedure's self-adaptation ability.

The *sphere model* $f_{\mathrm{sphere}}(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$ is the simplest test case. Figure 2 shows the procedure's optimization behavior as a function of the number λ of offspring (trial points). The figure clearly shows that the number λ of offspring significantly influences the procedure's convergence speed, which can be expected from the gradient estimate. According to Eq. (5), an increasing number λ of offspring per iteration increases the accuracy with which the actual gradient is estimated. It is interesting to note, however, that a saturation at $\lambda \geq 200$ can be observed. From that point on, a larger number of offspring does not lead to a more accurate estimate of the gradient. The influence of the number of offspring λ can be summarized by saying that up to a saturation point, which depends on the fitness function $f(\mathbf{x})$ and the n number of dimensions, the accuracy of the gradient estimation can be increased by increasing the number λ of offspring.

Figure 2 also shows the procedure's adaptation behavior of the step size σ_t. Due to the initialization of $\mathbf{x}_0 = (1000, \ldots, 1000)^T$, optimization starts at an initial fitness of $f(\mathbf{x}_0) = 10^7$. During approximately the first 16 iteration steps, no significant fitness improvement can be observed, since the step size σ_t has a value that is way too small. At iteration 16, however, the step size has reached the value $\sigma_{16} = \sigma_0 \zeta^{16} = 0.1 \times 1.8^{16} \approx 1214$, which is in the order of the remaining distance to the optimum. In the subsequent generations, linear convergence can be observed, which goes along with a continuous decrease of the step size. It should be noted that due to its step-size update in (7), the procedure requires merely 4 iterations to update the step size by an order of magnitude.

The behavior of the EGS procedure is further illustrated in Figure 3. This figure shows the evolution of two selected variables, x_1 and x_2, over time for two different values $\lambda = 1$ and $\lambda = 100$ of the number of trial points per iteration. The t-index is neglected in the figure for readability. It can be seen that for $\lambda = 1$ the evolutionary

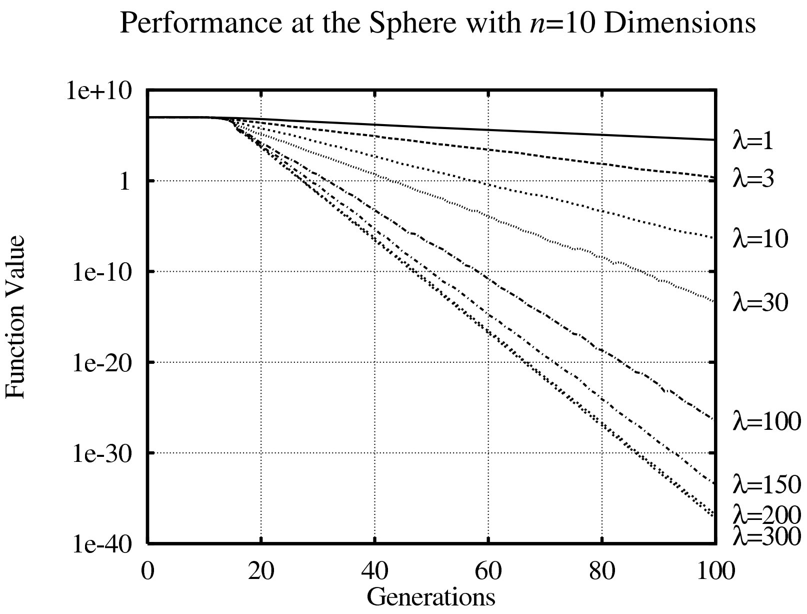


Fig. 2 The convergence speed over 100 steps as a function of the number λ of offspring. It can be observed that up to a saturation point at $\lambda \approx 200$, an increase of λ leads to an increase of the convergence speed.

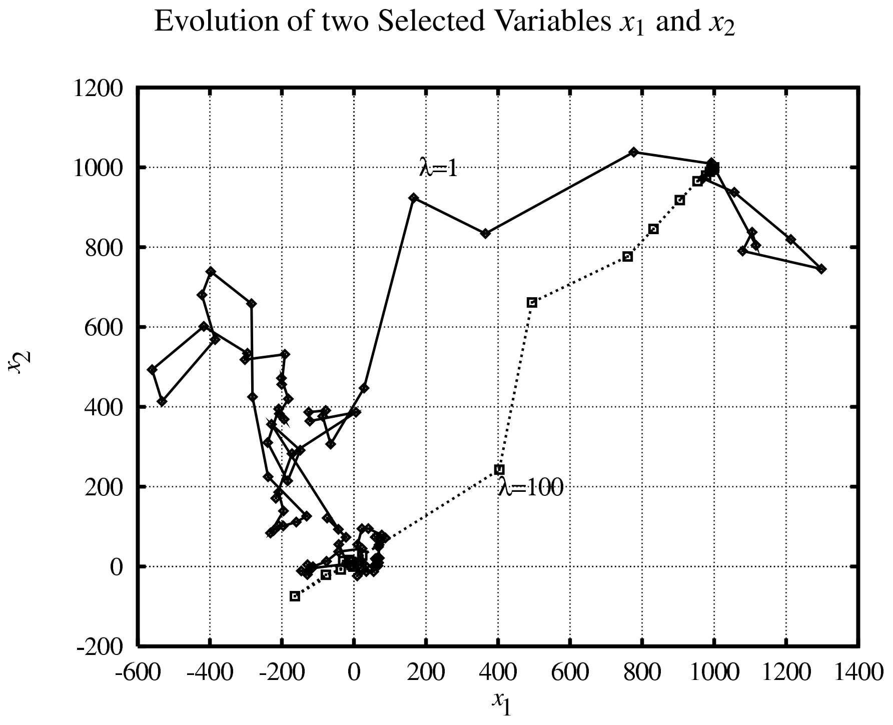


Fig. 3 The evolution over time of two selected variables, x_1 and x_2, as a function of the number λ of trial points. For $\lambda = 1$ the evolutionary path is rather erratic, whereas it is almost along the gradient direction for $\lambda = 100$.

path is rather erratic, whereas the path for $\lambda = 100$ is close to the direction of the gradient. This observation supports the behavior described above.

Figure 4 illustrates the scaling behavior of the EGS procedure when applied to the sphere model f_{sphere} with different dimensions n. The figure shows the average number of generations required to reach a precision of $\|x_i - x_i^o\| \leq \varepsilon = 10^{-5}$ for all parameters x_i when using a constant number $\lambda = 30$ of test candidates. From the figure, it can be seen that the EGS procedure exhibits an almost linear scaling behavior.

Figure 5 shows the performance of the EGS procedure when applied to the ellipsoid $f_{\text{ellipsoid}}(\mathbf{x}) = \sum_{i=1}^{n} i x_i^2$. In order to be independent of the initialization, the figure plots the normalized, logarithmic performance $\log(f^*) = \log(f(\mathbf{x}_0)/f(\mathbf{x}_t))$, which indicates the achieved improvements in orders of magnitude. The general behavior is similar to the case of the sphere model. The progress is smaller, though, due to the different eigenvalues.

Figure 6 shows the performance of the EGS procedure when minimizing the step function $f_{\text{step}}(\mathbf{x}) = \sum_{i=1}^{n} \lfloor |x_i| + 0.5 \rfloor^2$, with $\lfloor \cdot \rfloor$ denoting the floor function that yields the largest integer smaller than its argument. Again, it can be seen that increasing the number λ of trial points $\mathbf{y}_t^{(i)}$ per iteration accelerates the convergence speed.

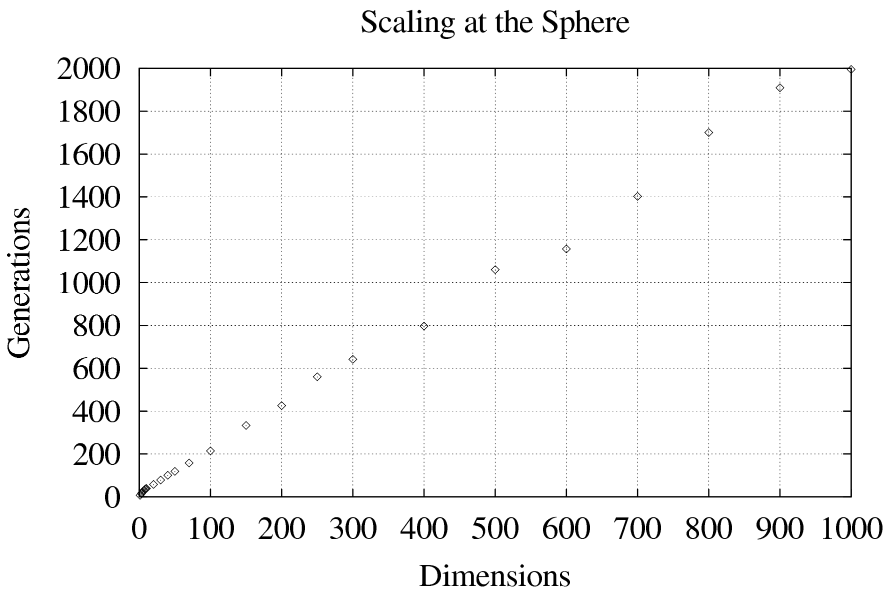


Fig. 4 The average number of generations of the EGS procedure with $\lambda = 30$ trial points to acquire a precision of $\|x_i - x_i^o\| \leq \varepsilon = 10^{-5}$ for all parameters x_i when minimizing the sphere model f_{sphere} with different dimensions n. It can be clearly seen that the EGS procedure exhibits an almost linear scaling behavior.

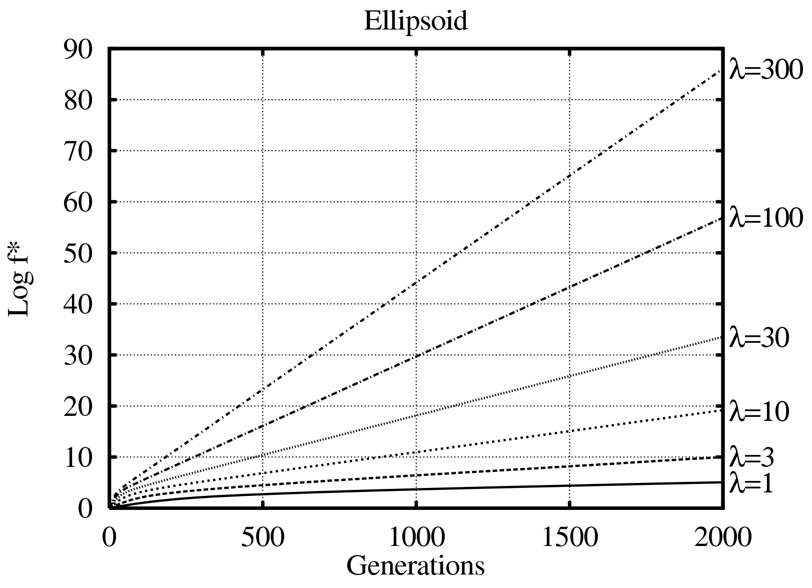


Fig. 5 The normalized, logarithmic progress over 2000 steps as a function of the number λ of trial points when EGS minimizes the ellipsoid $f_{\text{ellipsoid}}$

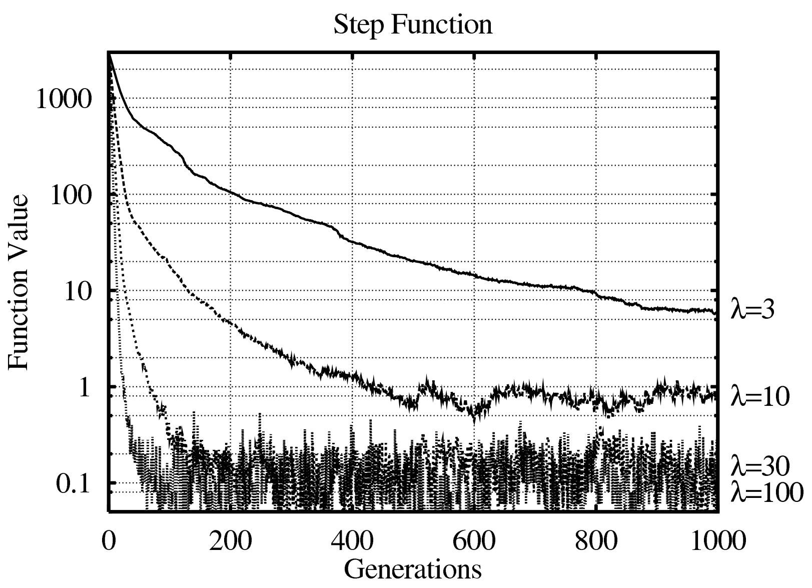


Fig. 6 Performance of the EGS procedure momentum as a function of the number λ of trial points $\mathbf{y}_t^{(i)}$ when applied to the step function $f_{\text{step}}(\mathbf{x}) = \sum_{i=1}^{n} \lfloor |x_i| + 0.5 \rfloor^2$. After reaching the minimum, the procedure oscillates in the neighborhood. In all 50 runs, the EGS procedure found the optimum.

The observable oscillations are due to intermittent increases of the step size σ_t. In all four test cases $\lambda \in \{3, 10, 30, 100\}$ the procedure found the optimum in all 50 runs. On average, the procedure requires 3107 ($\lambda = 3$), 317 ($\lambda = 10$), 85.2 ($\lambda = 30$), 24.3 ($\lambda = 100$), and 15.6 ($\lambda = 300$) generations for finding the optimum with $f_{\text{step}}(\mathbf{0}) = 0$. In all runs, the initial step size was set to $\sigma_0 = 3$ as has also been done for evolution strategies [5]. As has been reported, a $(30, 200)$-evolution strategy with individual step sizes requires approximately 4000 generations, whereas a simple $(30, 200)$-evolution strategy with one global step size fails on this task [5]. Further comparisons with evolutionary programming and genetic algorithms have shown [5] that evolutionary programming requires approximately 7500 generations, whereas a canonical genetic algorithm with bit-coding representations failed on this task.

4 Theoretical Analysis

Most theoretical performance analyses so far have been done on the quadratic function $f(x_1, \ldots, x_n) = \sum_i x_i^2$, which is also known as the sphere model. This choice has two main reasons: First, most other functions are currently too difficult to analyze, and second, the sphere model approximates the optimum's vicinity of many (real-world) applications reasonably well. Thus, the remainder of this chapter also uses this choice.

The theoretical analysis presented in this chapter also considers the $(\mu/\mu,\lambda)$-evolution strategy, since it is mathematically well analyzed and since it yields a very good performance [7, 15]. The $(\mu/\mu,\lambda)$-evolution strategy maintains μ parents, applies global-intermediate recombination on all parents, and applies normally distributed random numbers to generate λ offspring, from which the μ best ones are selected as parents for the next generation. In addition, this strategy also features a step size adaptation mechanism. Further details can be found in the literature [1, 7].

As a first performance measure, the rate of progress is defined as

$$\varphi = f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}) \tag{8}$$

in terms of the best population members' objective function values in two subsequent time steps t and t+1. For standard (μ,λ)-evolution strategies operating on the sphere model, Beyer [7] has derived the following rate of progress φ:

$$\varphi \approx 2Rc_{\mu,\lambda}\sigma - n\sigma^2 , \tag{9}$$

with $R = \|\mathbf{x}_t\|$ denoting the distance of the best population member to the optimum and $c_{\mu,\lambda}$ denoting a constant that subsumes all influences of the population configuration as well as the chosen selection scheme. Typical values are: $c_{1,6}$=1.27, $c_{1,10}$=1.54, $c_{1,100}$=2.51, and $c_{1,1000}$=3.24 [15]. To be independent of the current distance to the optimum, normalized quantities are usually considered (i.e., relative performance):

$$\begin{aligned} \varphi^* &= \sigma^* c_{\mu,\lambda} - 0.5(\sigma^*)^2 \text{ with} \\ \varphi^* &= \varphi\tfrac{n}{2R^2}, \sigma^* = \sigma\tfrac{n}{R} \quad . \end{aligned} \tag{10}$$

Similarly, the literature [1, 3, 7, 15] provides the following rate of progress formulas for EGS and the $(\mu/\mu,\lambda)$-evolution strategies:

$$\varphi^*_{EGS} \approx \sigma^* \sqrt{\frac{\lambda}{1+{\sigma^*}^2/4}} - \frac{{\sigma^*}^2}{2}, \tag{11}$$

$$\varphi^*_{\mu/\mu,\lambda} \approx \sigma^* c_{\mu/\mu,\lambda} - \frac{\sigma^*}{2\mu} . \tag{12}$$

Both formulas assume a large number of dimensions $n \gg 1$.

The rate of progress formula is useful to gain insights about the influences of various parameters on the performance. However, it does not consider the computational costs required for the evaluation of all λ offspring. For the common assumption that all offspring be evaluated sequentially, the literature often uses a second performance measure, called the efficiency $\eta = \varphi^*/\lambda$. In other words, the efficiency η expresses the sequential run time of an algorithm.

Figure 7 shows the efficiency of both algorithms according to Eqs. (11) and (12). It can be seen that for small numbers of offspring (i.e., $\lambda\approx$5), EGS is most efficient in terms of sequential run time and superior to the $(\mu/\mu,\lambda)$-evolution strategy.

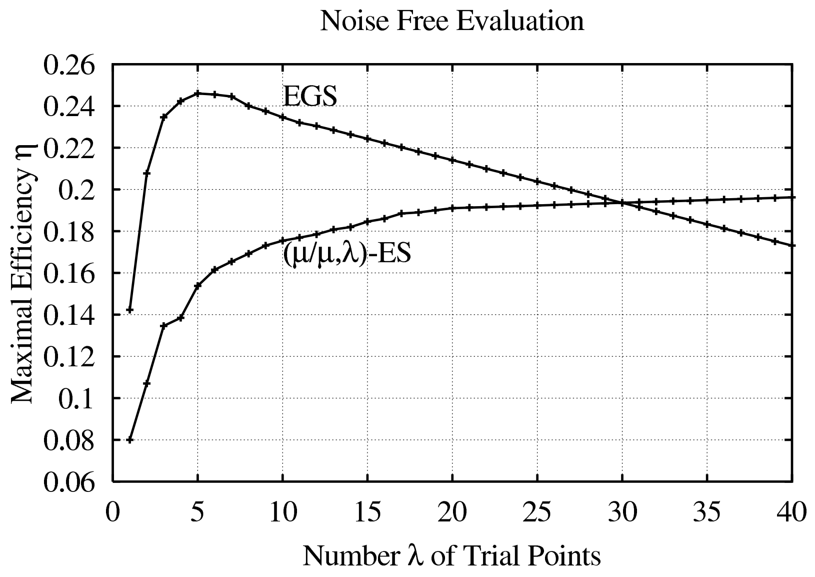


Fig. 7 Rate of progress of EGS and $(\mu/\mu,\lambda)$-evolution strategies according to Eqs. (11) and (12). Further details can be found in [1, 3].

5 The Problem of Noise

Section 4 has analyzed the performance in terms of obtainable progress rates for the undisturbed, noise-free case. The situation changes, however, when considering noise, i.e., noisy fitness evaluations. Noise is present in many if not all real-world applications. For the viability of practially relevant optimization procedures, noise robustness is thus of high importance.

5.1 Performance Analysis of Noisy Fitness Evaluations

Noise is most commonly modeled by additive $N(0,\sigma_\varepsilon)$ normally distributed random numbers with standard deviation σ_ε. For noisy fitness evaluations, Arnold [1] has derived the following rate of progress for the EGS procedure

$$\varphi^*_{EGS} \approx \sigma^* \sqrt{\frac{\lambda}{1+\sigma^{*2}/4+\sigma_\varepsilon^{*2}/\sigma^{*2}}} - \frac{\sigma^{*2}}{2}\,, \tag{13}$$

with $\sigma^*_\varepsilon = \sigma_\varepsilon n/(2R^2)$ denoting the normalized noise strength.

Figure 8 demonstrates how the presence of noise σ^*_ε reduces the rate of progress of the EGS procedure. Eq. (13) reveals that positive progress can be achieved only if $\sigma^*_\varepsilon \leq \sqrt{4\lambda+1}$ holds. In other words, the required number of offspring (trial points) required to estimate the gradient direction grows quadratically with the noise strength σ^*_ε.

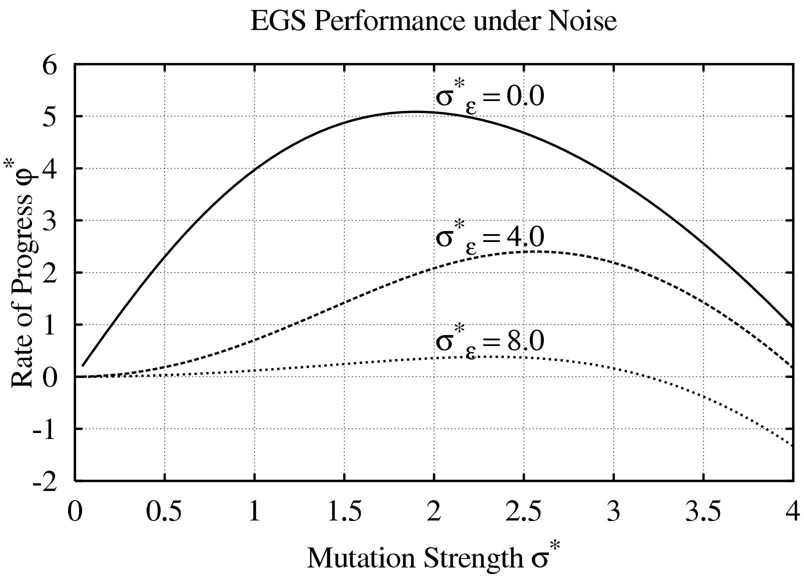


Fig. 8 The rate of progress φ^* of EGS progressively degrades under the presence of noise σ^*_ε. The example has used λ=25 offspring. Further details can be found in [1].

Another point to note is that the condition $\sigma^*_\varepsilon \leq \sqrt{4\lambda+1}$ incorporates the *normalized* noise strength. Thus, if the procedure approaches the optimum, the distance R decreases and the normalized noise strength increases. Consequently, the procedure exhibits an increasing performance loss as it advances towards the optimum.

By contrast, the $(\mu/\mu,\lambda)$-evolution strategy benefits from an effect called genetic repair induced by the global-intermediate recombination, and is thus able to operate with larger mutation step sizes σ^*. For the $(\mu/\mu,\lambda)$-evolution strategy, the literature [3] suggests that only a linear growth in the number of offspring λ is required.

5.2 Causes

Now that Subsection 5.1 has analyzed the performance of the EGS procedure in the presence of noise, this subsection examines the reasons and mechanisms that cause the observable performance degradation. In so doing, this subsection bases its analysis on the description presented in Section 2. Particularly Figure 1 illustrated how the EGS procedure estimates the gradient direction according to Eq. (5): $\tilde{\mathbf{g}}_t = \sum_{i=1}^{\lambda}(f(\mathbf{y}_t^{(i)}) - f(\mathbf{x}_t))\mathbf{z}_t^{(i)}$. The following two points should be mentioned here:

1. The EGS procedure uses the same step size σ_t for generating trial points and performing the step from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$.
2. For small step sizes σ, the probability for an offspring to be better or worse than its parents is half chance. With increasing step sizes, the chances of generating

trial points that improve on those of the previous time step steadily decrease, and they tend to zero for very large σ. For small λ, unequal chances have a negative effect on the accuracy of the gradient approximation.

Both points can be further elaborated as follows: A modified version of the EGS procedure uses two independent step sizes σ_g and σ_p for generating trial points (offspring) $\mathbf{y}_t^{(i)} = \mathbf{x}_t + \sigma_g \mathbf{z}_t^{(i)}$ and performing the actual progress step $\mathbf{x}_{t+1} = \mathbf{x}_t - \sigma_p \sqrt{n}\tilde{\mathbf{g}}_t / \|\tilde{\mathbf{g}}_t\|$ in accordance with Eqs. (4) and (6), respectively. Figure 9 illustrates the effect of these two step sizes for the example of the sphere model with $R = 1$, $n = 10$ dimensions, and $\lambda = 10$ trial points. It can be seen that the rate of progress φ^* significantly degrades for large step sizes σ_g. Since Figure 9 plots the performance for 'all possible' step sizes σ_p, it can be concluded that the accuracy of the gradient estimation significantly degrades for step size σ_g being too large. If the estimation $\tilde{\mathbf{g}} = \mathbf{g}$ was precise, the attainable rate of progress would be $\varphi^* = (f(\mathbf{x}_0) - f(\mathbf{0}))n/(2R^2) = (1-0)10/2 = 5$.

The hypothesis that the accuracy of the gradient estimation significantly degrades when the step size σ_g is too large is supported by Figure 10, which shows the angle $\cos\alpha = \mathbf{g}\tilde{\mathbf{g}}/(\|\mathbf{g}\|\,\|\tilde{\mathbf{g}}\|)$ between the true gradient $\mathbf{g}$ and its estimate $\tilde{\mathbf{g}}$. It can be seen that in addition to being dependent on the number of trial points λ, the gradient estimate's accuracy significantly depends on the step size σ_g. The qualitative behavior is more or less equivalent for all three numbers of trial points $\lambda \in \{5, 10, 25\}$. It is good for small step sizes, but starts degrading at $\sigma_g \approx R$, and quickly approaches zero for large σ_g. Figure 11 shows how the situation changes when noise is present. It can be seen that below the noise level, the accuracy of the gradient estimate degrades.

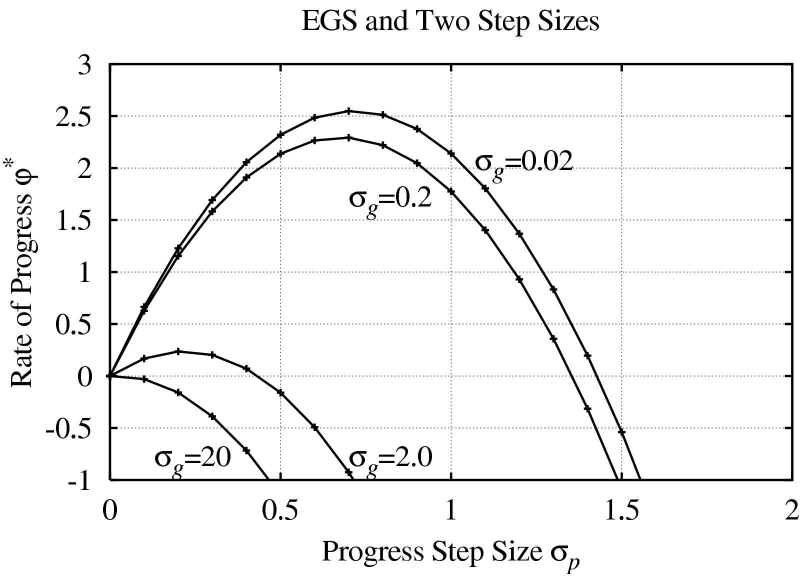


Fig. 9 Normalized rate of progress φ^* when using different step sizes σ_g and σ_p for the sphere example with $n = \lambda = 10$

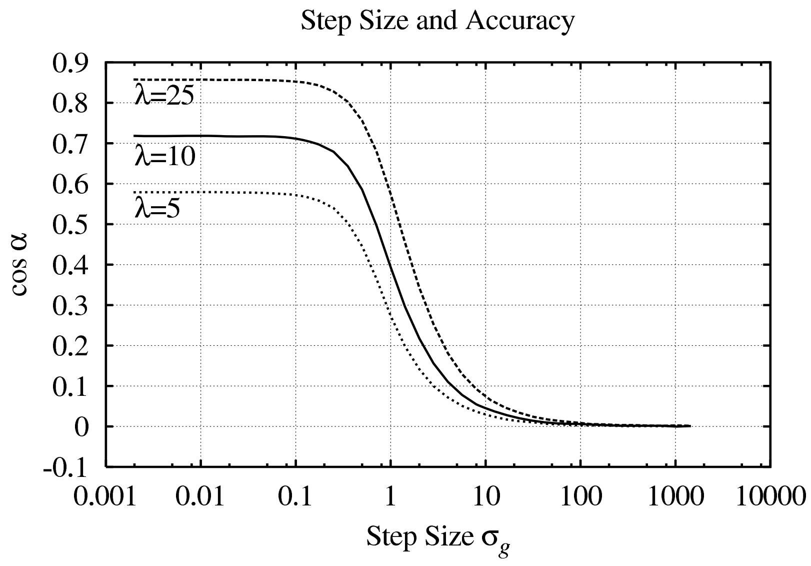


Fig. 10 $\cos\alpha$ between the true gradient $\mathbf{g}$ and its estimate $\tilde{\mathbf{g}}$ as a function of the number of trial points λ and $n = 10$ dimensions

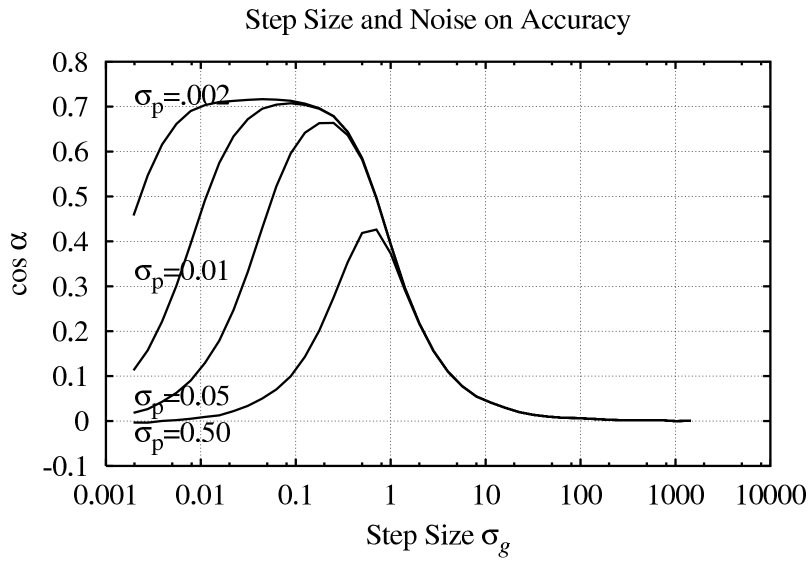


Fig. 11 $\cos\alpha$ between the true gradient $\mathbf{g}$ and its estimate $\tilde{\mathbf{g}}$ as a function of the noise strength $\sigma_e \in \{0.002, 0.01, 0.05, 0.5\}$. In this example, $n{=}\lambda{=}10$ and $R{=}1$ were used.

In summary, this section has shown that it is generally advantageous to employ two step sizes σ_g and σ_p, and that the performance of the EGS procedure degrades when the step size σ_g is either below the noise strength or above the distance to the optimum. The next section shows how to largely mitigate these problems.

6 Inverse Mutations

Section 5.2 has shown that both for small σ_g (where any information gained from evaluating trial points is hidden in the noise) as well as for large σ_g gradient estimates are poor. This problem could be tackled by by increasing the number of different trial points. However, Eq. (13) suggest only a performance gain of only $\sqrt{\lambda}$, which is significantly lower than the linear performance gain the $(\mu/\mu,\lambda)$-evolution strategy yields due to its genetic repair [1]. Therefore, this section considers the concept of *inverse mutations.*

6.1 The Concept of Inverse Mutations

When employing *inverse mutations*, the EGS procedure still generates λ trial points (offspring). However, half of them are mirrored with respect to the parent $\mathbf{x}_t$, that is, they are pointing in the opposite direction. Inverse mutations can be formally defined as:

$$\begin{aligned}\mathbf{y}_t^{(i)} &= \mathbf{x}_t + \sigma_g \mathbf{z}_t^{(i)} \text{ for } i = 1 \ldots \lceil \lambda/2 \rceil \\ \mathbf{y}_t^{(i)} &= \mathbf{x}_t - \sigma_g \mathbf{z}_t^{(i-\lceil \lambda/2 \rceil)} \text{ for } i = \lceil \lambda/2 \rceil + 1, \ldots \lambda\end{aligned} \tag{14}$$

In other words, each mutation vector $\mathbf{z}_t^{(i)}$ is used twice, once in its original form (upper part of Eq. (15)) and once as $-\mathbf{z}_t^{(i)}$ (lower part of Eq. (15)).

Figure 12 illustrates the effect of introducing inverse mutations. The performance gain is clear: The observable accuracy of the gradient estimate is constant over the entire range of σ_g values. This is in sharp contrast to the regular case, which exhibits the performance loss as discussed above. The figure, however, indicates that a slight disadvantage exists in that the number of trial points need to be twice as much in order to gain the same performance as in the regular case. The figure also shows the accuracy for $\lambda = 6$, which is smaller than the number of search space dimensions; the accuracy is $\cos\alpha \approx 0.47$ and thus still reasonably good.

Figure 13 illustrates the performance that inverse mutations yield in the presence of noise. For comparison purposes, the figure also shows the regular case for $\lambda = 10$ and $\sigma_\varepsilon \in \{0.05, 0.5\}$, which are plotted in dashed lines. Again, the performance gain is obvious: the accuracy degrades for small σ_g but is high for large step sizes, which is in contrast to the case where no inverse mutations are used.

Figure 14 shows the rate of progress φ^* of the EGS procedure using two step sizes σ_g and σ_p and inverse mutations for the example $n = 40$ and λ=24 and a normalized noise strength of $\sigma_\varepsilon^* = 8$. When comparing the figure with Figure 8, it can be seen that the rate of progress is almost that of the undisturbed case, i.e., $\sigma_\varepsilon^* = 0$. Furthermore, it can be seen that the performance starts degrading only for too small a step size σ_g. It should be mentioned here that for the case of $\sigma_\varepsilon^* = 0$, the graphs are virtually identical to the case $\sigma_\varepsilon^* = 8$ and $\sigma_g = 32$, and are therefore not shown.

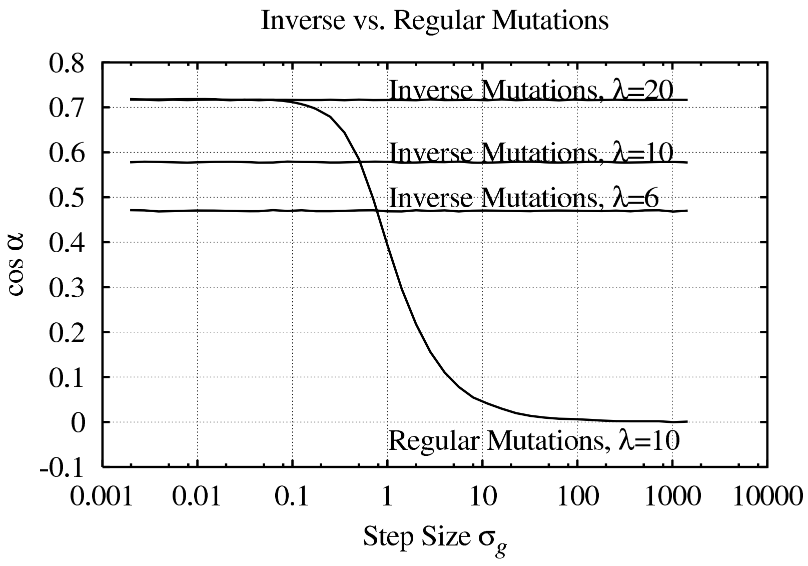


Fig. 12 $\cos\alpha$ between the true gradient $\mathbf{g}$ and its estimate $\tilde{\mathbf{g}}$ for various numbers of trial points λ. For comparison purposes, also the regular case with $\lambda = 10$ is shown. In this example, $n = 10$ and $R = 1$ were used.

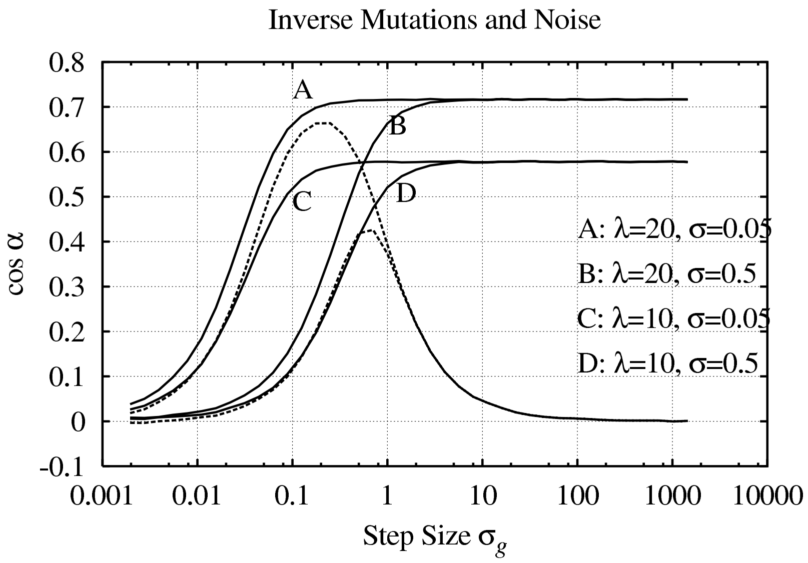


Fig. 13 $\cos\alpha$ between the true gradient $\mathbf{g}$ and its estimate $\tilde{\mathbf{g}}$ for $\lambda \in \{10, 20\}$ and $\sigma_\varepsilon \in \{0.05, 0.5\}$. For comparison purposes, also the regular case with $\lambda = 10$ and $\sigma_\varepsilon \in \{0.05, 0.5\}$ (dashed lines). In this example, $n = 10$ and $R = 1$ were used.

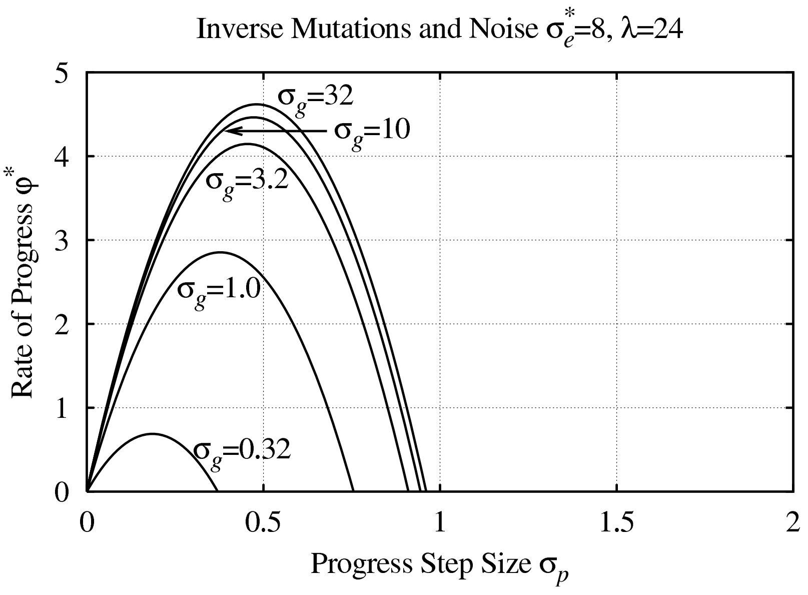


Fig. 14 The rate of progress φ^* of EGS with two step sizes and inverse mutations under the presence of noise $\sigma_\varepsilon^* = 8$. The example has used $n = 40$ and λ=24. In comparison to Figure 8, the performance is not affected by the noise.

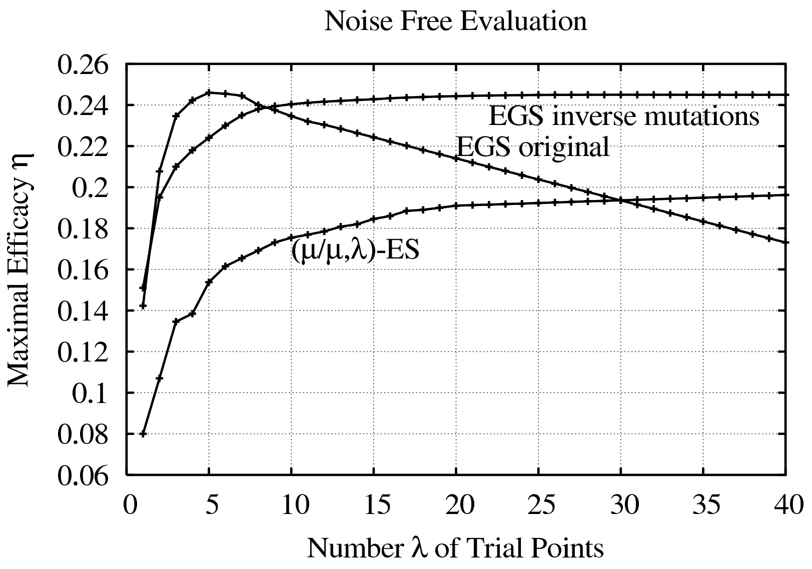


Fig. 15 Rate of progress of the modified EGS procedure in comparison to the original version and the $(\mu/\mu, \lambda)$-evolution strategies according to Eqs. (15), (11) and (12) with $\kappa = 1$. Further details can be found in [4].

6.2 Analysis of Inverse Mutations

Because of its complexity, a thorough analysis of the behavior of inverse mutations is beyond the scope of this chapter. However, it may be worthwhile to present the main results. For the detailed theory, the reader is referred to [4]. The normalized rate of progress of the EGS procedure *with* inverse mutations can be expressed as follows:

$$\varphi^*_{EGS-IV} \approx \frac{1}{\kappa}\left(\sigma^*\sqrt{\lambda} - \frac{{\sigma^*}^2}{2\kappa}\right) , \tag{15}$$

with $\kappa{=}\sigma_g/\sigma_p$. Figure 15 compares the progress of the modified EGS procedure with the progress $\varphi^*_{EGS} \approx \sigma^*\sqrt{\lambda/(1+{\sigma^*}^2/4)} - 0.5{\sigma^*}^2$ according to Eq. (11) of the original algorithm and the progress $\varphi^*_{\mu/\mu,\lambda} \approx \sigma^* c_{\mu/\mu,\lambda} - 0.5\sigma^*/\mu$ of the $(\mu/\mu,\lambda)$-evolution strategy according to Eq. (12). It can be seen that inverse mutations lead to a rate of progress qualitatively similar to that of the $(\mu/\mu,\lambda)$-evolution strategies but with significantly better values.

7 Enhancements

The form of the EGS procedure, as discussed so far, is simple and has its limitations, especially when applied to functions $f(\mathbf{x}) = 0.5\omega_1 x_1^2 + \cdots + 0.5\omega_n x_n^2$, which have very different eigenvalues $\omega_i \ll \omega_{j\neq i}$. From classical optimization techniques, it is well known that in such situations, rotationally invariant methods, such as steepest-descent or simple evolution strategies, exhibit useless oscillations of the gradient and thus of the optimization path: rather than going along a narrow valley, the optimization path might be oscillating between both sides resulting in a small effective progress along the valley's direction. To this end, this section discusses three enhancements, which have been adopted from other research and which are orthogonal in that they are independent of each other and in that they can be freely combined with inverse mutations.

7.1 Momentum

It has been shown [17, p. 330] [20] that a *momentum term* can alleviate the problem of useless oscillations, since it provides a memory by incorporating previous steps. The momentum term can be expressed as follows:

$$\begin{aligned} \Delta\mathbf{x}_{t+1} &= -\sigma_{t+1}\mathbf{e}_t + \alpha\Delta\mathbf{x}_t \\ \mathbf{x}_{t+1} &= \mathbf{x}_t + \Delta\mathbf{x}_{t+1} \end{aligned} \tag{16}$$

with $0 \le \alpha \le 1$ and $\Delta\mathbf{x}_0 = \mathbf{0}$. The parameter α is often simply called momentum. It should be noted that when $\alpha = 0$, the update rule in Eq. (16) is identical to that in Eq. (6).

It is straightforward to self-adapt the momentum α_t with the same adaptation mechanism previously described for the step size σ_t. In this case, the procedure has to adapt two parameters, which can be done by testing all four combinations, $(\sigma_t\,\zeta, \alpha_t\,\zeta)$, $(\sigma_t\,\zeta, \alpha_t/\zeta)$, $(\sigma_t/\zeta, \alpha_t\,\zeta)$, and $(\sigma_t/\zeta, \alpha_t/\zeta)$. In a more general form, the procedure has to test logarithmic-normally-distributed combinations of σ_t and α_t. The procedure can achieve this at no extra cost, if it adapts σ_t and α_t alternately.

As an example, Figure 16 illustrates the effect of the momentum term on the ellipsoid $f_{\text{ellipsoid}}(\mathbf{x}) = \sum_{i=1}^{n} i x_i^2$. A comparison with Figure 5 indicates that with $\lambda = 300$ trial points the momentum term accelerates the progress by about 70 %, i.e., 144 orders of magnitude as compared to 86 without momentum. The label "ES" refers to a (15,100)-evolution strategy *with* recombination and individual step sizes for which the literature [22] reports only 80 orders of magnitude. The concept of individual step sizes assigns one particular σ_i to each parameter x_i. These individual step sizes are used to generate adapted mutations $x_i \leftarrow x_i + \sigma_i z_i$, with z_i denoting $N(0,1)$ normally distributed random numbers. In comparison to the momentum term, individual step sizes are not rotationally invariant. That is, the performance progressively degrades if the functions are rotated in n dimensions.

Schwefel's ridge function $f_{\text{ridge}}(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$ may serve as a second example, which demonstrates the benefits of using the momentum term. A comparison of Figures 17 and 18 shows that for $n = 30$ dimensions, the momentum

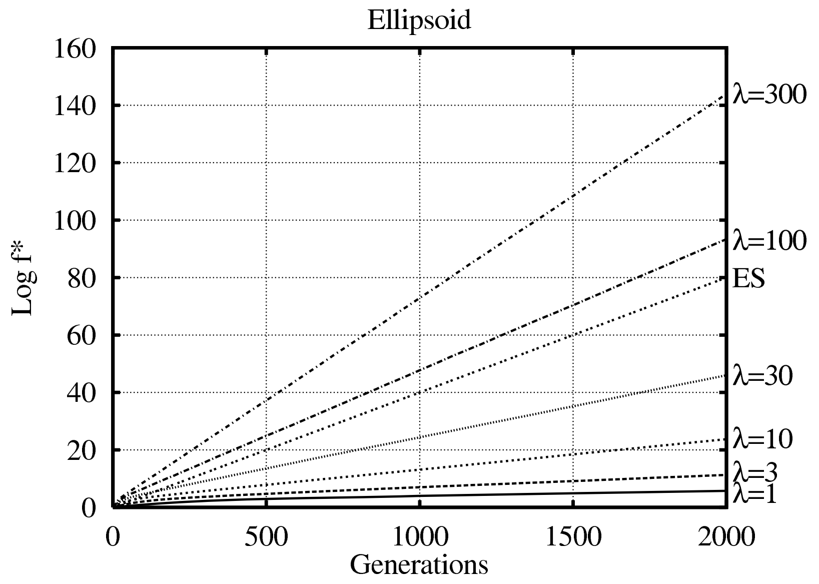


Fig. 16 The normalized, logarithmic progress over 2000 steps as a function of the number λ of test candidates when minimizing the ellipsoid $f_{\text{ellipsoid}}$ with n=30 dimensions by means of the EGS procedure *with* momentum. The label "ES" refers to a (15,100)-evolution strategy with individual step sizes σ_t (result taken from [22]).

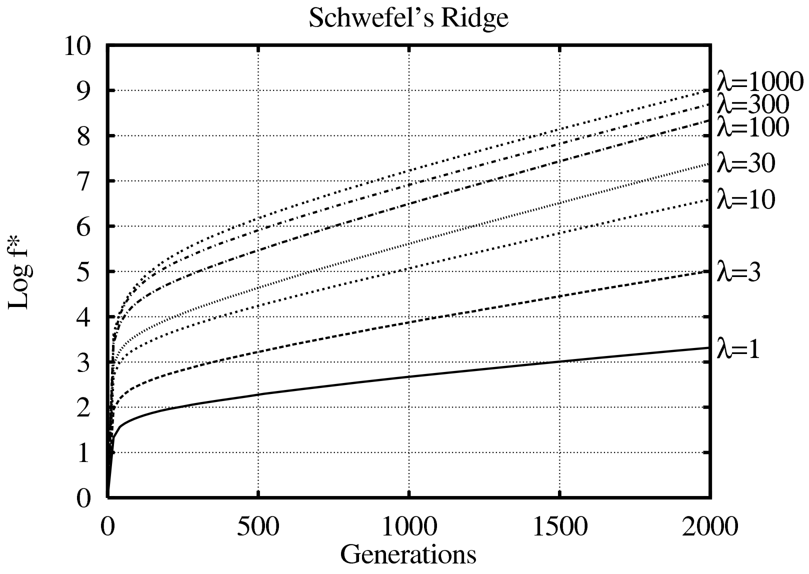


Fig. 17 The normalized, logarithmic progress over 2000 steps as a function of the number λ of test candidates when minimizing Schwefel's ridge f_{ridge} by means of the EGS procedure *without* momentum

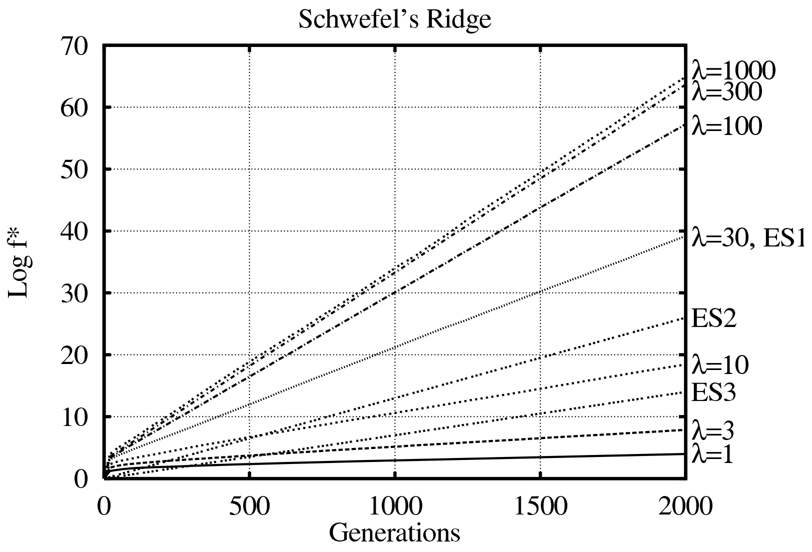


Fig. 18 The normalized, logarithmic progress over 2000 steps as a function of the number λ of test candidates by means of the EGS procedure *with* momentum when minimizing Schwefel's ridge f_{ridge}. The label ES1 refers to a (15,100)-evolution strategy with correlated mutations, ES2 to a (15,100)-evolution strategy with individual step sizes σ_t, and ES3 to a (1,100)-evolution strategy, respectively.

term yields seven *times* more orders of magnitude than that obtained without momentum. Figure 18 also indicates the performance of a simple (15,100)-evolution strategy, a (15,100)-evolution strategy with individual step sizes, and a (15,100)-evolution strategy with *correlated mutations*. Correlated mutations employ a general n-dimensional normal distribution for generating mutation vectors. The variances and covariances of that distribution are adapted using self-adaptation [5, 21]. for details. In practice, the covariance matrix adaptation mechanism described below is a more reliable approach to adapting the parameters of general normal distributions.

7.2 Individual Step Sizes

Previous research [19] has also investigated the utility of individual step sizes. Here, a particular step size σ_i is assigned to its corresponding coordinate axis. Experiments have shown that similar to standard (μ, λ)-evolution strategies, the EGS procedure benefits from this concept when optimizing quadratic functions with very different eigenvalues. However, this concept is not rotationally invariant and its benefit is thus limited to certain functions (the function's natural axes need to be nearly aligned with the coordinate axes x_i). Therefore, this concept is not further considered here and performance figures are omitted due to space limitations. For some results, the interested reader is referred to [19].

7.3 CMA-EGS

The basic EGS strategy described in Section 2 uses an isotropic normal distribution for generating mutation vectors. That distribution is fully described by a single parameter (the common variance of the individual components). The surfaces of equal probability density of the offspring candidate solutions are concentric hyperspheres with the search point $\mathbf{x}_t$ at their center. Individual step sizes as described in Section 7.2 generalize the procedure by using a normal distribution with potentially differing variances of the components, but with zero covariances between them. The distribution is characterized by n independent parameters, and the surfaces of equal probability density are hyperellipsoids with principal axes that are parallel to the axes of the coordinate system. The procedure can be generalized further by generating mutation vectors using general n-dimensional normal distributions. Such distributions are characterized by $n(n+1)/2$ parameters (the variances and covariances that form the covariance matrix), and the surfaces of equal probability density are general hyperellipsoids of arbitrary orientation.

Using a general mutation covariance matrix that is adapted to the problem at hand can speed up convergence by several orders of magnitude. As recognized by Rudolph [16] for evolution strategies, ideally, the covariance matrix is the inverse of the Hessian matrix of the objective function at the current search point. In that case, under certain conditions, the local performance of the strategy is identical to that of a strategy that uses isotropically distributed mutations on the sphere model.

However, the potential performance gain comes at the cost of the need to control $n(n+1)/2$ parameters (as opposed to a single one for the basic strategy, and n for the case of individual step sizes). Hansen and Ostermeier [11] have developed a powerful, derandomized algorithm for adapting the mutation covariance matrix in evolution strategies (CMA-ES) that has been adapted for use in EGS in [2]. That algorithm accumulates consecutive search steps in order to provide information on the basis of which adaptation of the mutation covariance matrix is performed. Realizing that it may be advantageous to adapt the overall step length on a time scale shorter than that used for adapting the shape of the distribution, trial points are generated with covariance matrix $\sigma^2\mathbf{C}$, where step length parameter σ is adapted separately from symmetric, positive definite $n \times n$ matrix $\mathbf{C}$. Adaptation of the former uses the idea of cumulative step length adaptation introduced by Ostermeier et al [13]. Adaptation of the latter is done with the implicit goal of maximizing the probability of replicating successful steps. Somewhat inaccurately, $\mathbf{C}$ is referred to as the mutation covariance matrix. As the CMA-ES, CMA-EGS utilizes two n-dimensional vectors $\mathbf{p}$ and $\mathbf{q}$ referred to as search paths that hold exponentially fading records of the most recently taken steps. An iteration of CMA-EGS updates the search paths along with the search point $\mathbf{x}$, the mutation strength σ, and matrix $\mathbf{C}$ using the following six steps:

1. Compute an eigen decomposition $\mathbf{C}_t = \mathbf{B}_t\mathbf{D}_t(\mathbf{B}_t\mathbf{D}_t)^{\mathrm{T}}$ of the mutation covariance matrix such that the columns of $n \times n$ matrix bfseries B_t are the normalized eigenvectors of $\mathbf{C}_t$ and $\mathbf{D}_t$ is a diagonal $n \times n$ matrix the diagonal elements of which are the square roots of the eigenvalues of $\mathbf{C}_t$.
2. Generate λ trial points

$$\mathbf{y}_t^{(i)} = \begin{cases} \mathbf{x}_t + \sigma_t\mathbf{B}_t\mathbf{D}_t\mathbf{z}_t^{(i)} & \text{for } i = 1,\ldots,\lceil\lambda/2\rceil \\ \mathbf{x}_t - \sigma_t\mathbf{B}_t\mathbf{D}_t\mathbf{z}_t^{(i-\lceil\lambda/2\rceil)} & \text{for } i = \lceil\lambda/2\rceil+1,\ldots,\lambda \end{cases}$$

   where the $\mathbf{z}_t^{(i)}$ consist of n independent, standard normally distributed components.
3. Determine the objective function values $f(\mathbf{y}_t^{(i)})$ of the trial points and compute the weighted sum

$$\tilde{\mathbf{g}}_t = \sum_{i=1}^{\lambda} \left(f(\mathbf{y}_t^{(i)}) - f(\mathbf{x}_t)\right)\mathbf{z}_t^{(i)}$$

   as an estimate of the gradient direction.
4. Perform a step

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \sigma_t\mathbf{B}_t\mathbf{D}_t\mathbf{z}_t^{\mathrm{prog}}$$

   where

$$\mathbf{z}_t^{\mathrm{prog}} = \frac{\sqrt{n}}{\kappa}\frac{\tilde{\mathbf{g}}_t}{\|\tilde{\mathbf{g}}_t\|}$$

   points in direction of the gradient estimate.

5. Update the search paths according to

$$\mathbf{p}_{t+1} = (1-c_{\mathbf{C}})\mathbf{p}_t + \kappa\sqrt{c_{\mathbf{C}}(2-c_{\mathbf{C}})}\mathbf{B}_t\mathbf{D}_t\mathbf{z}_t^{\mathrm{prog}}$$

   and

$$\mathbf{q}_{t+1} = (1-c_\sigma)\mathbf{q}_t + \kappa\sqrt{c_\sigma(2-c_\sigma)}\mathbf{B}_t\mathbf{z}_t^{\mathrm{prog}}$$

   where $c_{\mathbf{C}} = c_\sigma = 4/(n+4)$ as recommended in [11].
6. Update covariance matrix and step length according to

$$\mathbf{C}_{t+1} = (1-c_{\mathrm{cov}})\mathbf{C}_t + c_{\mathrm{cov}}\mathbf{p}_t\ \mathbf{p}_t^{\mathbf{T}}$$

   and

$$\sigma_{t+1} = \sigma_t \exp\left(\frac{\|\mathbf{q}_{t+1}\|^2 - N}{2DN}\right)$$

   where $c_{\mathrm{cov}} = 2/(n+\sqrt{2})^2$ and $D = 1 + 1/c_\sigma$ as recommended in [11].

Notice that steps 2. to 4. closely parallel steps 1. to 3. of the basic algorithm in Section 2. Differences include the use of two separate mutation strengths for generating trial and search steps as proposed in Section 5 (the quotient κ determines the size of the latter relative to the size of the former) and the use of inverse mutations as described in Section 6. The scaling and rotation with matrices **D** and **B**, respectively, of the mutation vectors in step 2. ensures that offspring are generated with covariance matrix $\sigma\mathbf{C}$. Steps 5. and 6. implement cumulative step length and covariance matrix adaptation that replace the simple mutation strength adaptation rule from Section 2. See [11] for a more thorough motivation of the algorithm and the settings of its parameters. Finally, realizing that the eigen decomposition in step 1. is expensive and that its cost may, for large n, outweigh the cost of evaluating the trial points, Hansen and Ostermeier [11] suggest to perform it only every $n/10$ steps, and to use slightly outdated matrices **B** and **D** in between. The loss in performance from that modification is typically negligible.

8 Summary

This chapter has described a hybrid evolutionary algorithm, called the evolutionary-gradient-search (EGS) procedure. Rather than selecting only the best offspring, the procedure utilizes them all to gain as much information as possible. The theoretical analysis as well as the experimental results have shown that the use of all offspring, including the inferior ones, can improve a procedure's performance. Due to its design, the EGS procedure works well on those optimization problems on which evolution strategies also work well. This mainly includes unimodal functions.

This chapter has also discussed the EGS procedure's behavior in the presence of noise. Due to some performance degradations, this chapter has also discussed the concept and performance of inverse mutations, which use a very same mutation vector $\mathbf{z}_t^{(i)}$ twice as $\mathbf{z}_t^{(i)}$ and $-\mathbf{z}_t^{(i)}$. Finally, this chapter has also discussed the incorporation of further enhancements, such as the momentum term, individual step sizes, and correlated mutations, as known from classical optimization procedures and standard evolutionary algorithms. These enhancements are orthogonal to the basic concepts of the EGS procedure and can thus be freely incorporated.

References

1. Arnold, D.: An analysis of evolutionary gradient search. In: Proceedings of the Congress on Evolutionary Computation (CEC 2004), pp. 47–54. IEEE Press, Los Alamitos (2004)
2. Arnold, D., Salomon, R.: Evolutionary gradient search revisited. Techreport CS-2005-09, Faculty of Computer Science, Dalhousie University (2005)
3. Arnold, D., Beyer, H.G.: Local performance of the $(\mu/\mu,\lambda)$-Evolution Strategy in a noisy environment. In: Martin, W.N., Spears, W.M. (eds.) Proceeding of Foundation of Genetic Algorithms 6 (FOGA 2006), pp. 127–141. Morgan Kaufmann, San Francisco (2001)
4. Arnold, D., Salomon, R.: Evolutionary gradient search revisited. IEEE Transactions on Evolutionary Computation 11(4), 480–495 (2007)
5. Bäck, T., Schwefel, H.P.: An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation 1(1), 1–23 (1993)
6. Beyer, H.G.: An alternative explanation for the manner in which genetic algorithms operate. BioSystems 41, 1–15 (1997)
7. Beyer, H.G.: The Theory of Evolution Strategies. Springer, Heidelberg (2001)
8. Fogel, D.B.: Evolutionary Computation: Toward a New Philosophy of Machine Learning Intelligence. IEEE Press, Piscataway (1995)
9. Fogel, L.J.: Autonomous automata. Industrial Research 4, 14–19 (1962)
10. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
11. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
12. Luenberger, D.G.: Linear and Nonlinear Programming. Addison-Wesley, Reading (1984)
13. Ostermeier, A., Gawelczyk, A., Hansen, N.: Step-size adaptation based on non-local use of selection information. In: Davidor, Y., änner, R.M., Schwefel, H.P. (eds.) Proceedings of the 3rd International Conference on Parallel Problem Solving from Nature (PPSN III), pp. 189–198. Springer, Heidelberg (1994)
14. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C. Cambridge University Press, Cambridge (1994)
15. Rechenberg, I.: Evolutionsstrategie 1994. Frommann-Holzboog, Stuttgart (1994)
16. Rudolph, G.: On correlated mutations in evolution strategies. In: änner, R.M., Manderick, B. (eds.) Proceedings of the 2nd International Conference on Parallel Problem Solving from Nature (PPSN II), pp. 105–114. Elsevier, Amsterdam (1992)
17. Rumelhart, et al. (eds.): Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 2. MIT Press, Cambridge (1986)
18. Salomon, R.: Evolutionary algorithms and gradient search: similarities and differences. IEEE Transactions on Evolutionary Computation 2(2), 45–55 (1998)
19. Salomon, R.: Accelerating the Evolutionary-Gradient-Search procedure: Individual step sizes. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.P. (eds.) Proceedings of the 5th International Conference on Parallel Problem Solving from Nature (PPSN V), pp. 408–417. Springer, Heidelberg (1998)

20. Salomon, R., van Hemmen, J.L.: Accelerating backpropagation through dynamic self-adaptation. Neural Networks 9(4), 589–601 (1996)
21. Schwefel, H.P.: Evolution and Optimum Seeking. John Wiley and Sons, Chichester (1995)
22. Schwefel, H.P.: Evolutionary Computation — A Study on Collective Learning. In: Callaos, N., Khoong, C.M., Cohen, E. (eds.) Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Orlando, FL, USA. International Institute of Informatics and Systemics, vol. 2, pp. 198–205 (1997)

The Evolutionary Transition Algorithm: Evolving Complex Solutions Out of Simpler Ones

Tom Lenaerts, Anne Defaweux, and Jano van Hemert

Abstract. Capturing the metaphor of evolutionary transitions in biological complexity, the Evolutionary Transition Algorithm (ETA) evolves solutions of increasing structural and functional complexity from the symbiotic interaction of partial ones. In this chapter we show that the ETA indeed captures this idea and we illustrate this on instances of the Binary Constraint Satisfaction problem. The results make the ETA a promising optimization approach that requires more extensive investigation from both a theoretical and optimization perspective. We analyze here, in depth, some of the design choices that are made for the algorithm. The analysis of these choices provides insight on the plasticity of the algorithm toward alternative choices and other kinds of problems.

1 Introduction

In biology, evolutionary transitions theory [14, 15] provides a generalized explanation of how organisms of increasing complexity may have emerged from the interaction of simpler life forms. The Evolutionary Transition Algorithm (ETA), presented here, captures this metaphor to create structurally and functionally more complex solutions from the combination (interactions) of simpler ones (solutions that solve

Tom Lenaerts
ML-group, Département d'Informatique, Université Libre de Bruxelles,
Boulevard du Triomphe CP212, 1050 Brussels, Belgium
e-mail: `tlenaert@ulb.ac.be`

Anne Defaweux
COMO, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium
e-mail: `adefaweu@gmail.com`

Jano van Hemert
National e-Science Centre, The University of Edinburgh,
15 South College Street Edinburgh EH8 9AA, UK
e-mail: `jano@vanhemert.co.uk`

a smaller part of the problem, for example). As such, the goal of this chapter is to show how a particular biological metaphor was transformed into an algorithm with definite potential. Future investigations will attempt to show its relevance for real-world applications.

Apart from its general contribution to the development of nature-inspired algorithms, this new evolutionary algorithm was introduced to address some limitations of the standard genetic algorithm (GA). One of these limitations follows from the dependencies between the variables in the solution representation. In the standard GA, new genotypes are produced by the recombination of existing ones. To avoid the disruption of good building blocks, one needs to ensure that correlated parts of a solution are close to one another in the genotype . For many problems, determining the internal structure of the genotype is a problem on its own since the interdependencies between the variables of the problem are not known. The compositional mechanism implicit to the ETA provides a way to discover the closely related variables of the problem during the evolutionary process while at the same time looking for the solution to the complete problem. An additional motivation follows from the difficulty of scaling the GA to evolve larger solutions for more extensive problems. By taking the compositional approach the evolutionary process can first focus on solving parts of the problem, which can be combined later just like the modular watch constructed by Simon's watchmaker [22].

We show here, using the Binary Constraint Satisfaction problem (BINCSP) as an illustration, that the ETA evolves increasingly complex solutions from the interactions of simpler evolving solutions. The results for BINCSP confirm that the ETA is promising approach that requires more extensive investigation from both a theoretical and practical optimization perspective. Especially decision problems in general [11] and BINCSPs in particular seem to be very well suited for this new evolutionary algorithm. We provide an in depth analysis of the design choices that are made for the algorithm: choices related to the configuration of the initial population, the introduction of a decomposition operator which breaks down more complex solutions into simpler components and the impact of the transition condition on the performance of the algorithm.

This chapter is partitioned as follows. Before discussing the algorithm, which will be explained using BINCSP, a non-exhaustive overview of related work is provided. Afterwards, a set of BINCSP simulations and their results are shown and explained. Apart from a brief overview of earlier results, we provide new results on particular design choices that can be made for the ETA. Finally, a summary is provided and some conclusions are drawn concerning the usefulness and the future of this algorithm.

2 Related Work

There are two ideas behind the compositional search approach: the first idea concerns the use of symbiotic relations to identify good collaborations, and the second idea is concerned with the aggregation of complex solutions from the interaction of partial ones. The related work listed here falls under either one or both of these categories. It is necessary to note that what we are going to provide is not a full list but a rough picture of the major research work related to the topic. Consequently, certain publications which are very similar to those mentioned here might be missing.

The very first GA-related reference that can be found on the evolution of complete solutions from the combination of partial ones is the work on the messy GA [5, 6, 7]. The messy GA has the structure of a classical GA. When two partial solutions are selected for reproduction in the messy GA, their combined genetic material creates a bigger solution that is defined at a higher level of complexity. Messy GAs therefore have the idea of combining partial solutions. However, this approach differs from the one we propose here: fitness effects caused by the interactions of the partial solutions are not taken into account. Partial solutions are evaluated and selected on their own as in the classical GA. It is during the process of reproduction that combination occurs. Messy GAs therefore lack the idea of a transition that operates on the behavior of good symbionts.

Other approaches focus on the collaboration between partial solutions to construct a full solution for a particular problem. In the Parisian Approach [17, 18] for example, the algorithm intends not to evolve a full solution to a problem but rather a collection of partial solutions that together solve the entire problem. This approach takes an additional step into the direction of compositional evolutionary search. However, the Parisian approach lacks a transition step that merges the set of partial solutions into a full solution. In this respect, the Parisian approach is more related to previous work on multilevel selection [13].

Other approaches introduce an evolutionary "divide and conquer" mechanism like the cooperative co-evolutionary GA [16, 30]. In this algorithm, the problem consists of a collection of sub-problems that can be solved in isolation. Afterwards these partial solutions can be recombined into a full solution for the global problem. The major difference with the ETA resides in the explicit divide and conquer framework. In the cooperative co-evolutionary GA, the process divides the problem into collaborating sub-problems; for each sub-problem, the algorithm evolves solutions that collaborate with one another in order to address the entire problem. The way to divide the problem can become an issue when the problem structure to learn is not trivial or cannot be easily identified.

Finally, a truly compositional approach that uses both the concept of symbiosis and transitions is the Symbiogenetic Model (SEAM) [27, 28, 29]. This model considers a population of partial solutions that interact with one another through the mechanism of symbiosis. In this population, the good symbiotic relations are identified and produce a new partial solution through a transition. This model initially appears very similar to ETA. There is, however, a major difference between SEAM and ETA: SEAM does not generate a succession of populations by reproducing the parent population through the mechanism of fitness proportional selection. It randomly selects pairs of individuals and places them into a symbiotic relation. It then replaces the symbiotic relation by a new individual when it performs better than the parents in isolation. The condition under which this transition occurs also differs from ETA: it uses the concept of pareto optimality with respect to an evolutionary context to define whether a symbiosis performs better or worse than the parents alone. The result of this approach requires the model to include features about the structure of the problem that needs to be solved like for instance the degree of modularity of the problem. Furthermore, in order to work, it requires the problem to be fully hierarchically decomposable, which covers only a subset of the structured problem class.

Finally, the Hierarchical Genetic Algorithm (HGA) [23] is an algorithm very similar to SEAM. The major difference resides in the fact that the HGA introduces, in its implementation, explicit information concerning the hierarchical nature of the problem that needs to be solved (by hard coding the hierarchical search in the process and therefore disclosing the emergent aspect present in SEAM) and optimizes the sampling approach so that only the best samples are present for evaluation at all time, considerably speeding up the search process.

3 BINCSP: The Illustrating Test Case

Before we explain the algorithm we briefly define the problem that is used to illustrate the technique.

Constraint Satisfaction Problems (CSP) [24] are NP-complete problems that are defined by a set of variables X associated with possible domain values D and a set of constraints C defined on this set of variables that specify which combinations of assignments can or cannot occur. The problem consists of finding an assignment to the whole set of variables from the associated domain values so that all constraints are satisfied. This makes the problem a decision problem [11]. If an assignment is impossible then the corresponding problem is said to be unsolvable. A variant of this problem is the BINCSP, where each constraint is defined on at most two variables. This introduces no restriction on the general form of CSP as every CSP can be rewritten into a BINCSP and vice versa [19].

Let us take as an illustration the following BINCSP: consider a set of six variables: $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ all taking values in $D = \{1, 2, 3\}$. We consider the following set of constraints:

$$\begin{aligned} C = \{&(x_1 \neq x_2), (x_2 \neq x_3), (x_3 \neq x_1), \\ &(x_4 \neq x_5), (x_5 \neq x_6), (x_6 \neq x_4), \\ &(x_1 = x_4), (x_2 = x_5), (x_3 = x_6)\} \end{aligned} \tag{1}$$

This constraints setup consists of 9 binary constraints. Each binary constraints defines a relation on 2 variables. For each pair of variables, only one binary constraint may be defined.

The problem consists of finding the right assignment for the variables so that all these constraints are satisfied. We denote the assignment of one variable $x_i \in X$ with value $d \in D$ by $\langle i, d \rangle$ where i is the index of the variable we consider. Using this notation, we represent the simultaneous assignments of variables x_1, x_2 and x_4 with respective values v_1, v_2 and v_4 as

$$(\langle 1, v_1 \rangle, \langle 2, v_2 \rangle, \langle 4, v_4 \rangle)$$

4 General Description of the ETA

As can be seen in Figure 1, the ETA follows a classical *evaluate-select-reproduce* evolutionary loop. As a GA it works with a limited fixed size population which is evaluated and reproduced at each iteration. Before the evaluation phase, all (free) individuals are paired up. This grouping can be performed in different ways. Here we opted for the most simple form, i.e. random pairing. Once each individual has a partner, a fitness score is assigned based on their own properties and the properties they obtain from the interaction with the other individual. The combination of both properties is referred to as the induced phenotype, which will be explained in more detail later. The fitness score will guide the selection process as in the GA and selected individuals can reproduce in three different ways:

1. They can just reproduce their own genetic information, as in the GA.
2. To maintain interesting links, we also provide the possibility that both individuals and their interaction are reproduced.

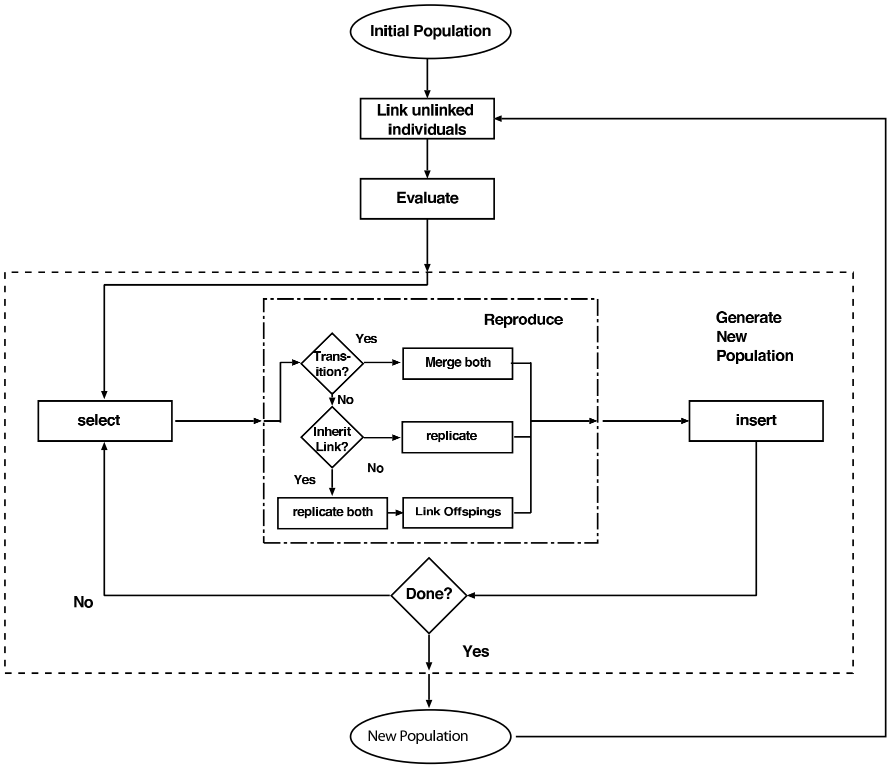


Fig. 1 Schematic overview of the ETA. The algorithm consists of three phases: 1) the pairing up of partial solutions, 2) the evaluations of the interacting solutions and 3) the reproduction of the solutions. Every iteration of these three steps produces a new population.

3. Good symbionts which solve the partial problem completely are reproduced as individuals at a higher level of complexity, meaning that their genetic information is combined into a new individual.

Repeated iterations of this process produce individuals of increasing complexity that solve an increasing proportion of the problem.

The following sections explain each part of the ETA in detail.

4.1 Representation of Basic Elements and Partial Solutions

A partial solution for BINCSP is a compound label which is not defined on all variables of the variable set X. A compound label on, for instance, variable x_1,x_3 and x_7 where each variable is instantiated with values d_1, d_3 and d_7 respectively is denoted by

$$(\langle 1,d_1\rangle,\langle 3,d_3\rangle,\langle 7,d_7\rangle).$$

In the current description of ETA, we make a distinction between a *partial solution* and a *solution*: A partial solution for the BINCSP is a compound label which does not assign all variables of the variable set defined by the BINCSP. When all variables are assigned, we obtain a solution to the problem. The underlying idea behind this distinction is that a solution defining a full genotype is the achievement of the compositional search algorithm. We therefore speak of partial solution as long as the algorithm is still in the process of evolving complexity, meaning it is still trying various combinations of partial solutions with the hope of obtaining a full solution to the problem. The representation of a (partial) solution is also called the *genotype*.

In its most basic form, a partial solution should correspond to the assignment of exactly one variable. However, such solutions are meaningless by themselves as each constraint is defined on exactly two variables. As a consequence, they need to be evaluated during their interaction with other individuals.

The fact that basic elements of length 1 have a zero fitness has consequences when considering a fitness-proportional selection model. Indeed, initializing the population with length 1 partial solutions means that none of them can expect a positive fitness, i.e. be selected, without interacting with others. This lead us to think that the most basic units of selection are not of length 1 but rather of at least length 2. In our simulations, we evaluated both scenarios (see simulations section for more details) and observed that the length of the initial partial solutions has little impact on the evolutionary process itself.

4.2 Interactions and the Induced Phenotype

In the ETA, pairs of (partial) solutions are bound by *symbiotic relations*. When bound the two solutions exchange information. The result of this exchange is captured in the *induced phenotype* of the solution which can alter positively or negatively the fitness of each of the solutions. We explicitly make the distinction between phenotype and *induced phenotype* to stress that the latter is phenotype produced from both the genetic information of an individual and the effect of external factors, like other population members and the environment, on the final phenotype.

Since the induced phenotype of an individual is constructed by combining the information contained in its own genotype with the information contained in the genotype of the other members of the symbiotic relation, conflicts in this information may occur. In case of a conflict (i.e. two different values are assigned by the partial solutions to the same variable), a conflict mediation strategy is applied here that randomly chooses one of the possible values from the set of conflicting values.

As explained earlier, interactions are pairwise and created randomly using the population of solutions, i.e. the solution can be linked to any other solution in the population. Alternative strategies, based on multilevel selection [13] are currently under investigation.

Let us briefly illustrate the process by which the induced phenotype is constructed in the context of the BINCSP example. Suppose we have a partial solution s that interacts with another partial solution sp. Both partial solutions are represented through the following genotypes:

$$\begin{aligned} s &= \langle 1, \alpha_1 \rangle, \langle 2, \alpha_2 \rangle, \langle 9, \alpha_9 \rangle \\ sp &= \langle 2, \beta_2 \rangle, \langle 4, \beta_4 \rangle, \langle 9, \alpha_9 \rangle \end{aligned}$$

In these two solutions, we observe that s is well defined for the variables 1, 2 and 9 and sp for the variables 2, 4 and 9. The value assigned to variable 9 is the same for both partial solutions. However, the values assigned to variable 2 are not the same, meaning that the solutions have a conflict to solve for this variable: we need to choose between value α_2 or β_2. The conflict mediation strategy randomly chooses between one of the conflicting values yielding in our specific example two possible induced phenotypes:

$$\varphi(s, sp) = \langle 1, \alpha_1 \rangle, \left\langle 2, \begin{pmatrix} \alpha_2 \\ \beta_2 \end{pmatrix} \right\rangle, \langle 4, \beta_4 \rangle, \langle 9, \alpha_9 \rangle$$

where α_2 or β_2 is chosen randomly with equal probability.

After conflict resolution, the resulting phenotype assigns exactly one value for each variable that was assigned in s or sp. This phenotype is then used by the fitness function to evaluate the partial solution. The details concerning the fitness function and the way the partial solutions are evaluated in the context of BINCSP is provided in the following section. Note first that the induced phenotypes of the members of the symbiotic relation do not have to be the same for both partners since the process decides randomly which value to use for each member independently. Thus both s and sp can select α_2, β_2 or make different choices (s takes β_2 and sp takes α_2 or vice versa).

4.3 Evaluation Functions

We consider two evaluation functions for BINCSP. The first function evaluates the quality of the solution's phenotype with respect to the entire constraints set. The second function is restricted only to the constraints that are defined on the variables that are present in the partial solution.

Definition 1. The *classical fitness evaluation for the ETA on BINCSP* of a solution s, $f_{et}(s)$ which interacts with a symbiotic partner sp and for which the induced phenotype of the interaction for s with sp is denoted by $\varphi(s,sp)$ is given by following equation:

$$f_{et}(s) = \frac{1}{|C|}\sum_{c\in C} eval(\varphi(s,sp),c) \tag{2}$$

where $eval_c(\varphi,c)$ checks whether the compound label φ satisfies the constraint c or not.

We have slightly adapted the notion of constraint satisfaction as we also need to address partial solutions that have no assignment or incomplete assignment definitions for a given constraint. We therefore suppose that a compound label satisfies a constraint c if:

- it *covers* the constraints, i.e. there are assignments for all the variables comprised in c in the compound label of the partial solution.
- the assignments do not violate the constraints.

This is summarized in the following equation:

$$eval(\varphi,c) = \begin{cases} 1 \text{ if } (\varphi \; covers \; c) \text{ and } (satisfies(\varphi,c)) \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

This fitness value gives an estimation of how the partial solution scores with respect to the entire set of constraints. This means that small partial solutions, even if they are made of good genetic material, cannot receive a high reward value in comparison with larger partial solutions which, by assigning more variables increase their chance of improving their score, even if some of these variables are incorrectly assigned. To obtain an objective value of how a partial solution scores with respect to the sub-problem of constraints that are covered by the partial solution, we therefore also consider a restricted version of this fitness function which limits the evaluation to the covering set of constraints for this partial solution.

Definition 2. The *covering set* of constraints for a given genotype γ on the constraints set C is denoted by $C_{cov}(\gamma)$ and consists of all the constraints of C which are covered by γ:

$$C_{cov}(\gamma) = \{c \in C \mid \gamma \; covers \; c\}.$$

With this covering set, we can now define the covering fitness function.

Definition 3. The *covering fitness function* of a solution s interacting with sp is denoted f_{cov} and is given by the following equation:

$$f_{cov}(s) = \frac{1}{|C_{cov}(\varphi(s,sp))|}\sum_{c\in C_{cov}(\varphi(s,sp))} eval(\varphi(s,sp),c) \tag{4}$$

We typically use the first evaluation function to guide the evolutionary process, that is, for selecting the best solutions for reproduction, while the second measure is used as an observation measure for deciding whether the interacting partial solutions are merged into one new solution.

The motivation for the use of two different fitness measures is the following: the classical fitness function f_{et} will tend to favor cooperation as bigger partial solutions are more likely to receive a greater fitness than smaller partial solutions and we wish the system to evolve a complete solution to the problem and therefore to favor increase in the length of the solution genotype. The covering fitness does not capture information with respect to the entire problem since partial solutions of any size (bigger than 2) can have maximum fitness, which is 1. This function prevents the system from evolving larger solutions as they will not perform better than smaller partial solutions. This is the reason why we choose the classical fitness for the selection process. However, when it comes to evaluating the quality of a symbiotic relation, the covering fitness contains much more information than the classical fitness. By limiting the evaluation to the problem actually covered by the symbiotic relation, we obtain a measure of how well the relation performs with respect to the sub-problem it addresses. The covering fitness therefore allows us to assess whether a symbiotic relation should be preserved for future generations through the mechanism of transitions.

4.4 Replication and Transitions

When selected for reproduction, a partial solution will have three possible options (examples are based in BINCSP problem defined in Section 3:

1. Simple replication: In this case, one solution is replicated, that is, its genotype is copied and passed on to its offspring (with possible mutations: the value of variable 1 changed from 1 to 2 in the example below). It occurs when the solution scored sufficiently high to survive one more generation but did not perform good enough within its symbiotic relation to see this relation survive or preserved through a transition.

$$s = \langle 1,1\rangle, \langle 2,2\rangle, \langle 3,2\rangle \longrightarrow s' = \langle 1,2\rangle, \langle 2,2\rangle, \langle 3,2\rangle$$

2. Inherit symbiotic link: In this situation, the solution not only replicates itself but also trigger the replication of its symbiotic partner. The symbiotic link that binds both parents is also inherited by their offspring. This reproduction mode is a step toward a transition; it is, however, a reversible step since both individuals still exist independently. In the situation of BINCSP, we did not implement any special condition for this mode to occur and it can therefore occur randomly at each generation.

$$\left\{ \begin{array}{l} s = \langle 1,1\rangle, \langle 2,2\rangle, \langle 3,2\rangle \\ sp = \langle 2,1\rangle, \langle 5,1\rangle, \langle 6,3\rangle \end{array} \right\} \longrightarrow \left\{ \begin{array}{l} s' = \langle 1,1\rangle, \langle 2,2\rangle, \langle 3,3\rangle \\ sp' = \langle 2,1\rangle, \langle 5,1\rangle, \langle 6,3\rangle \end{array} \right\}$$

In the example, the covering fitness of the induced phenotype of both individuals (s and sp) is less than 1. As a consequence, no transition occurs. Yet we can replicate both partners and their link to check if we can improve it by choosing, for instance, another value for variable 2. Again, mutation might occur when the individuals are reproduced, as is the case here for variable 3 in individual s.

3. Transitions: This case occurs when the induced phenotype of an individual actually solves the sub-part of the problem completely, i.e. when the covering fitness is equal to 1.0. In this case, the outcome of the symbiotic relation is replicated instead of the original genotype of the solution. This means that the induced phenotype becomes the genotype of the offspring.

$$\left\{\begin{array}{l} s = \langle 1,1\rangle,\langle 2,2\rangle,\langle 3,3\rangle \\ sp = \langle 2,1\rangle,\langle 5,2\rangle,\langle 6,3\rangle \end{array}\right\} \longrightarrow s' = \langle 1,1\rangle,\langle 2,2\rangle,\langle 3,2\rangle,\langle 5,2\rangle,\langle 6,3\rangle$$

In this example, the induced phenotype of s can be for instance:

$$\varphi(s,sp) = \langle 1,1\rangle,\langle 2,2\rangle,\langle 3,2\rangle,\langle 5,2\rangle,\langle 6,3\rangle$$

The covering fitness of $\varphi(s,sp)$ is 1. As a consequence a transition occurs, making the genotype of s' the same as the induced phenotype produced by the interaction of s and sp.

The second reproduction mode might look like an unnecessary step in the entire process as it occurs randomly without any relation to the transition itself. Nevertheless, this mode may be very important for various reasons:

1. Even though this event occurs randomly, it is performed on individuals selected according to some fitness proportionate selection scheme. In this way, only individuals which are potentially better get the chance to reproduce their symbiotic partner. Consequently, it provides the conflict mediation strategy more chances of finding a good combination.
2. It is a way to keep potentially good genetic material in the population as the symbiotic partner might possess the genetic information that actually causes this solution to be selected. In other words, their high fitness is a result of the combination of both partners. Therefore, we need to explore their relationship further.
3. Through a possible error during copying, a nearly good combination may become the right combination. See the examples above.

Further evaluation is required for this part of the algorithm.

The third reproduction mode, the transition, creates solutions of higher complexity: Small genotypes aggregate into larger genotypes that in turn, through interactions with other genotypes, can aggregate into more complex genotypes until a full solution emerges.

5 Evaluation of the ETA on Concrete BINCSP Instances

The objective of the current simulations is to demonstrate that ETA succeeds in building complex solutions that can actually solve BINCSP problems. Since we will evaluate BINCSP instances of different complexities we may derive the subset of problems on which the ETA performs well. Moreover, we examine here how the ETA behaves when we change certain operations or conditions inherent to the algorithm.

5.1 Simulation Setup

Each simulation has an almost similar setup and uses the same set of BINCSP instances. These instances are randomly generated using RandomCSP package provided by [26] and produce instances with common statistical properties as the one used in [25]. This allows us to compare the ETA with other coevolutionary approaches that were tested on this set (see [2]).

The complexity of the instances created by the tool can be tuned by two parameters p_1 and $\bar{p}_2$. We let p_1 and $\bar{p}_2$ vary from 0.1 up to 0.9 with a step size of 0.2. In Table 1 these instances are classified according to their problem complexity into either the easy or difficult class. For each combination of p_1 and $\bar{p}_2$, 25 different random problem instances for a BINCSP of 15 variables, each taking values in a domain of size 15, were generated.

For each of the 25 problem instances, we perform 10 runs (each run using a different random seed). The maximum amount of generations is set to 100 000. This means that for each setup of p_1 and $\bar{p}_2$, 250 runs are performed in total. This amount of runs for each problem instance should be sufficient to evaluate the algorithm.

The initial population is created such that it contains all partial solutions of length 1, i.e. every variable-value combination is present once. We refer to this population as a *sound population*.

During replication, the genotype may mutate with probability 0.001, i.e. the value of a variable may be altered to some random value in the domain of the variable. Transitions are performed when the covering fitness is 1.0.

The speed of the algorithm is determined as follows: Let e be the number of evaluations in the run r where the optimal solution was found and e_{max} the maximum number of evaluations in the simulation run. We wish to evaluate the speed of the algorithm on a scale of $[0,1]$ where higher values mean that the algorithm needed less evaluations to find a solution than lower values and the value 0 corresponds to

Table 1 Identification of easy and difficult BINCSP instances

		Tightness ($\bar{p}_2$)				
		0.1	0.3	0.5	0.7	0.9
Density (p_1)	0.1	**easy**	easy	easy	easy	**difficult**
	0.3	easy	easy	**easy**	**difficult**	unsolvable
	0.5	easy	easy	**difficult**	unsolvable	unsolvable
	0.7	**easy**	**easy**	**difficult**	unsolvable	unsolvable
	0.9	easy	difficult	unsolvable	unsolvable	unsolvable

the situation where no solution was found in the given time. We have therefore the following equation for the speed of the run r:

$$speed(r) = \begin{cases} 0 & \text{if no solution found during } r \\ \frac{e_{max}-e}{e_{max}} & \text{if the optimum is found in } e \text{ evaluations} \end{cases} \quad (5)$$

5.2 Observations

We collected different types of data over all the runs:

- The size of the solutions over time: We monitor the maximum and minimum size of the solutions over time, together with the amount of overlap between the symbiotic partners.
- The fitness of the solutions over time: We trace the best fitness, the average fitness, the worst fitness in the population and the fitness of the largest individual (i.e. the individual that achieved the highest level of complexity). We collect the values of the classical fitness as provided by Equation 2 for these observations.
- Success ratio: This gives the ratio of successful runs over the total amount of runs.
- Average number of generations required to reach a complete solution for all BINCSP instances of a particular complexity: Since the population is evaluated exactly once in each generation, this number is strongly correlated to the speed of the algorithm (see Section 5.1).

5.3 Previous Results

In this section, we summarize the results of the simulations of the ETA on BINCSP discussed in [3, 4], for reasons of comparison.

Table 2 shows the success ratio, average number of generations to find a solution and standard deviation in this number for each BINCSP setup. The plots in Figures 2 and 3 show the evolution of fitness and genotype complexity over time for 4 easy and 4 difficult BINCSP instances (see Table 1).

From these simulations the following conclusions were drawn. The results in Figure 2 show that the transition model succeeds in finding a solution for the easy BINCSP as the fitness converges rapidly to 1.0. They also show that, for difficult BINCSP (after over 2 000 generations), the algorithm converges to a population that contains a combination of partial solutions that solve over 90% of the constraints but are unable to resolve the conflicts between the interacting partial solutions in order to solve the entire set of constraints.

We observe in Figure 3 that for easy BINCSP, the intersection (reflecting the number of variables that are in common between the two solutions) is small while exploring, which corresponds to a high complementarity. Indeed for the easy BINCSP (left column of Figure 3) we see that initially the intersection between the two solutions is rather small. Once the algorithm begins to converge toward a

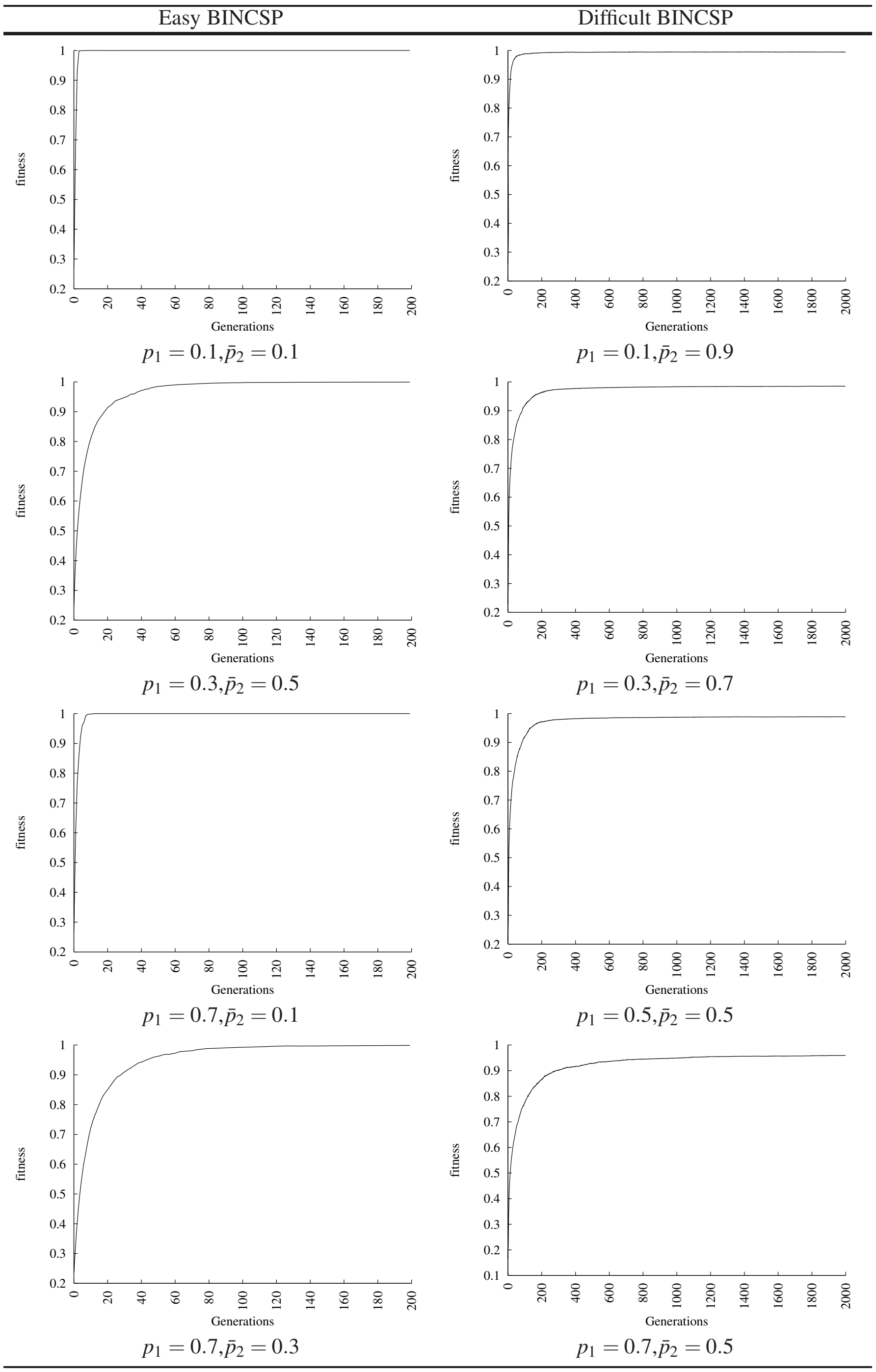


Fig. 2 Evolutionary dynamics of the ETA. All plots show the evolution of the best fitness.

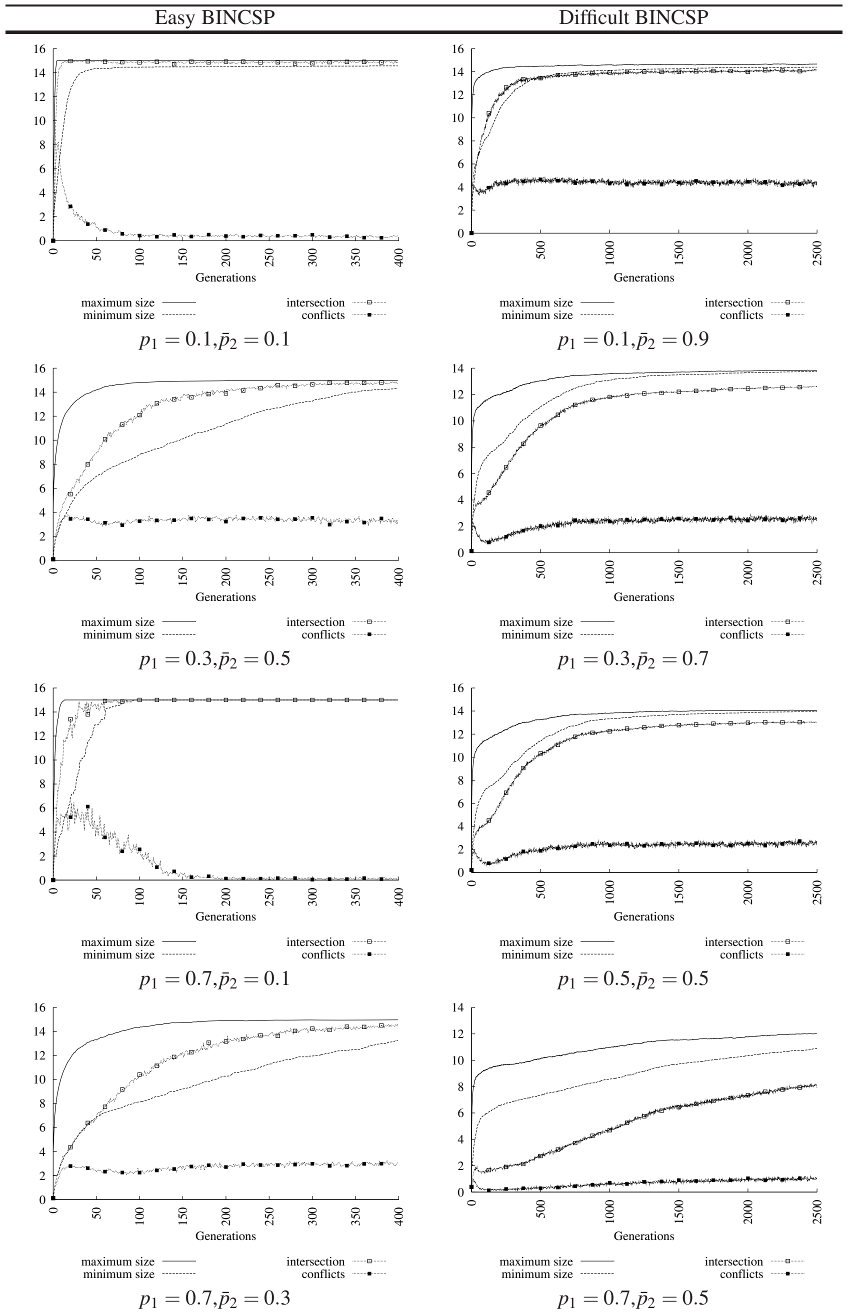


Fig. 3 Evolutionary dynamics of the ETA: All plots show the evolution of the genotype complexity. Each plot contains the maximum and minimum size of the individuals in the population, the average number of variables in common (intersection) and the average number of remaining conflicts (conflicts).

Table 2 Results of the simulations. Each square contains the success ratio, average number of generations, standard deviation and average number of evaluation (between braces).

Density	Tightness				
p_1	0.1	0.3	0.5	0.7	0.9
0.1	1.0	1.0	1.0	1.0	0.992
	(4)	(5)	(7)	(13)	(3627)
	[0]	[0]	[1]	[3]	[1224]
	{900}	{1125}	{1575}	{2925}	{816075}
0.3	1.0	1.0	1.0	0.28	-
	(5)	(12)	(43)	(11758)	-
	[0]	[3]	[25]	[-]	-
	{1125}	{2700}	{9675}	{2645550}	-
0.5	1.0	1.0	0.54	-	-
	(6)	(25)	(11409)	-	-
	[1]	[8]	[-]	-	-
	{1350}	{5625}	{2567025}	-	-
0.7	1.0	1.0	0.29	-	-
	(8)	(61)	(8673)	-	-
	[1]	[29]	[-]	-	-
	{1800}	{13725}	{1951425}	-	-
0.9	1.0	0.972	-	-	-
	(11)	(2335)	-	-	-
	[2]	[4902]	-	-	-
	{2475}	{525375}	-	-	-

solution, this complementarity decreases and the number of conflicts in the intersection stabilizes. The positive number of the conflicts in this intersection for certain easy BINCSP setups reflects a low but existing variation in the genotype population throughout the evolutionary process. One can also see in the left column that when the population converges completely, all conflicts disappear. In contrast, when $p_1 = 0.3$ and $\bar{p}_2 = 0.5$ (and also $p_1 = 0.7$ and $\bar{p}_2 = 0.3$), there are still alternatives present in the population even though an optimal solution is found.

When we look at difficult BINCSP in the same figure, we observe quite a similar evolution (right column of Figure 3). However, in these cases, the individual size and the intersection (almost) never reaches the maximum number of variables. Since the maximum size is not reached, each genotype needs a symbiotic partner to cover the entire variable set, implying some complementarity to achieve the final solution. Yet the problem is that for these difficult problems it is not sufficient to glue together complementary solutions (see three last plots in the right column of Figure 3). In those cases, the number of variables that are common between the partners in the symbiotic relation is even less than the minimum size of the individuals in the population. This shows that partial solutions seem to agree on certain aspects of the solution yet are divided over the other variables and their assignments that are required to reach the complete solution. So here we have an exploration issue that needs to be solved. The low number of conflicts shows that the ETA evolves complementary partial solutions that specialize in solving two different sub-sets of the constraints set too. However, the difficulty of solving the entire problem makes it impossible to resolve certain conflicts among these variables - the results show that the evolutionary process is stuck in some local optimum, which may require other mechanisms to resolve the issue.

The results in Table 2 were compared to other co-evolutionary BINCSP approaches in [4]. The ETA performs relatively well compared to problem-specific EA techniques and outperforms them in one specific case: those instances where the constraint network has some kind of modular structure due to the sparseness of the connection between the constraints. Even when the constraints to learn are difficult (high $\bar{p}2$), the ETA conquers these constraints by solving the subproblems separately and aggregating them once they are solved.

In [3] we analyzed the effect of initial population composition on the results of the ETA. We compared a randomly generated population of partial solutions of size two and a sound population (as used here). The results showed that the initial population has little overall effect on the chance of success, which is interesting as it indicates that our algorithm performs well with a small initial population and hence does not need a large and complete population to achieve good results. We illustrated that these differences are indeed not significant when comparing the speed (using Equation 5) of the algorithm.

5.4 Introducing Decomposition in the ETA

The ETA, as described so far, only creates more and more complex solutions. This might result in a dead-end, possibly explaining why a higher success score on the difficult BINCSP instances was not achieved. It could then be useful to decompose these bad solutions back into smaller individuals so alternative composition paths may be followed. The motivation for not including this from the beginning stems from the description of the biological metaphor. Evolutionary transitions are considered as a one-way process [15]. Once a transition occurs, the components of the new individual are forced to collaborate with one another. Defection leads irremediably to the death of the organism together with all its components. So, biologically, individuals cannot decompose back into lower level units and such a situation, when it happens, leads to the extinction of the individual and all its components. In an optimization context however, we are not limited by this restriction.

To introduce decomposition in the evolutionary process, we slightly adapted the ETA described at the beginning of this chapter so that, at each generation, a certain fraction of the population has the possibility of being decomposed into smaller units. These smaller units of selection can then recombine with other units and evolve a new solution which is defined at a higher level of complexity. The deconstruction (decomposition) therefore introduces a new exploration mechanism that tries other combinations of building blocks that had been identified as potentially good units in former symbiotic relations.

This notion of decomposition is, in a certain sense, related to the backtracking process implicit to depth-first and related search methods [8, 20]. Indeed, these deterministic approaches decompose invalid solutions to reconstruct them afterwards with valid assignments. In the evolutionary approach, the process is no more deterministic but stochastic [11]. However, the idea of decomposing solutions which lead to a dead-end in the optimization process remains.

In this simulation setup, we observe whether the introduction of decomposition, i.e. deconstruction operation, can affect the quality of the results. We first describe the adaptation of the existing algorithm with the introduction of a decomposition mechanism and then we present the simulation setup and results.

5.4.1 Evolutionary Transition Algorithm with Decomposition

ETA with decomposition introduces two new features to the existing framework of the algorithm: the decomposition condition and the decomposition operator. The *decomposition condition* determines when a given solution should decompose into elementary solutions. This condition may trivially be implemented as a random process but may also relate to observations of the evolutionary process (for example the process is stuck in a local optimum, the solution has lasted many generation without resolving its conflicts with the symbiotic partners) or even to problem-specific aspects. The *decomposition operator* may be implemented in different ways to induce different dynamics in the search process. For example, it can decompose one solution all the way back to its elementary units. Such an operator, ensures the presence of elementary solution units during the entire run. Another possibility consists of decomposing one solution back into 2 (or more) partial solutions, reducing the complexity of the given solution by a factor 2 (or more) but without asking the process to rebuild a complex solution with these partial units from scratch.

Given the operator and the condition, the ETA can now be extended. During the replication process, the selected solutions are passed one by one through the reproduction operator. This operator, as before, verifies if the transition condition is met, that is, if the symbiotic relation performs well enough to emerge a new individual at a higher level of complexity. This part of the replication process is unchanged with respect to the classical implementation of ETA. If the transition condition is not met, which means that the selected solution will self-replicate and will possibly replicate its symbiotic relation, the replication process will first check whether the decomposition condition is met before proceeding with self-replication as before. If the decomposition condition is met, then the solution decomposes to a lower level of complexity.

For the simulations, we consider random decomposition. Each selected solution which does not perform a transition may be decomposed with a certain probability. The choice for random decomposition instead of a more problem-or-process related condition is motivated by the idea that we are mainly concerned with the evaluation of the decomposition process. We want to see if decomposition brings something to the system. Random decomposition, not being problem related, offers an objective view of the induced dynamics of the decomposition process.

When the decomposition condition is met, the selected solution will not replicate as a whole but split itself into two or more parts according to the decomposition operator. In our implementation of ETA with decomposition for this set of simulations, we implement a simple decomposition operator where the solution is divided

```
decompose(solution)
for $pos := 1$ to $solution.size$ step 1 do
  $allele := solution[pos]$
  $prob := 0.5$
  if $offspring_1.size > offspring_2.size$
    then
      $prob := prob \times (1 - \frac{offspring_1.size}{solution.size})$
    else
      $prob := prob \times (1 + \frac{offspring_2.size}{solution.size})$
  if $random(prob)$
    then
      $offspring_1.add(allele)$
    else
      $offspring_2.add(allele)$
```

Fig. 4 Pseudo-code for the decomposition operation

into two parts of approximatively the same size. This strategy means that we do not really backtrack to the former situation consisting of the two partial solutions that were merged into this parent solution. We obtain, however, a decomposition in terms of complexity as the solution goes back to a former level of complexity with respect to this parent solution. The way the two offspring solutions are generated in corresponds to a random process where, on average, each allele of the parent solution is passed on to one of the two offspring solutions with the same probability. The pseudo-code of the decomposition of a solution is given in Figure 4. The two offsprings that are created are then added to the offspring population.

5.4.2 Simulation Setup

The problem setup is the same as before. The goal here is to compare the transition model without decomposition (see Table 2) with a transition model using decomposition. Both models will use initial sound populations as described in the previous simulation. They also share exactly the same parameters setup, not considering the additional *decomposition probability*. To explore the impact of this parameter on the behavior of the process, we consider 3 possible values for this parameter:

- Small probability for decomposition: the value of the decomposition probability scales from 0.001 to 0.05 which means that, at each generation, a very small part of the population will be decomposed.
- Medium probability for decomposition: the value of the decomposition probability scales from 0.15 to 0.35. In this situation, at each generation, a larger part of the population will be decomposed. However, decomposition remains a rare event as the solutions have more chance to remain unchanged than to undergo a decomposition operation.
- High probability for decomposition: the value of the decomposition is set to 0.5 and above. In this situation, at each generation, the population will perform at

least as much decomposition operations as self-replication operations. This situation corresponds to a situation where exploration is preferred over exploitation.

For each setup of the BINCSP and the decomposition probability, we perform 10 runs each with a different seed on the 25 problem instances that were randomly generated. We collect the same information as for the previous simulation.

5.4.3 Simulation Results

In Table 3, we gather results concerning the success ratio and average number of generations to reach a solution for the 5 difficult test cases identified in Table 1. For

Table 3 Comparison of success ratio between different decomposition setups. Each entry consists of three values: 1) the success ratio, 2) the average number of generations and 3) the standard deviation. The upper row corresponds to the situation of a generic transition model operating with initial sound population and no decomposition, the other rows gives the results for different decomposition probabilities (DP). Results better than the generic model are shown in bold.

		p_1: 0.9	0.7	0.5	0.3	0.1
		$\bar{p}_2$: 0.3	0.5	0.5	0.7	0.9
	DP:					
No		0.972	0.29	0.54	0.28	0.992
	Generic	(2335)	(8673)	(11409)	(11758)	(3627)
		[4902]	[-]	[-]	[-]	[1224]
Low		0.976	0.276	0.54	0.236	0.992
	0.001	(3106)	(4654)	(7955)	(10534)	(2803)
		[5077]	[1546]	[8128]	[33000]	[2220]
		0.992	0.308	**0.604**	0.236	**1.0**
	0.01	(2476)	(8507)	(12993)	(8230)	(2835)
		[2956]	[9492]	[10729]	[8053]	[4201]
		0.984	**0.36**	0.58	**0.3**	**1.0**
	0.05	(2922)	(13705)	(11879)	(9248)	(1457)
		[3869]	[14214]	[7222]	[5126]	[2407]
Medium		0.736	0.232	0.3	0.172	0.884
	0.15	(7869)	(13168)	(13542)	(19029)	(3003.5)
		[2166]	[11514]	[1246]	[16504]	[4857.5]
		0.648	0.26	0.248	0.156	0.856
	0.25	(9492)	(15437)	(12890)	(10436)	(3471)
		[1191]	[12867]	[3104]	[12701]	[6290]
		0.736	0.1	0.26	0.096	0.892
	0.35	(8774)	(12836)	(17794)	(18952)	(4223)
		[6220]	[9810]	[13272]	[12820]	[7425]
High		0.38	0.08	0.076	0.048	0.604
	0.5	(18237)	(9509)	(25637)	(25561)	(13627.5)
		[13186]	[3147]	[20368]	[23721]	[9136]
		0.004	0.0	0.0	0.0	0.02
	0.65	(46)	(-)	(-)	(-)	(39)
		[0]	[-]	[-]	[-]	[14]

each test case, runs were performed for different decomposition probabilities scaling from low to high. The success ratios that are printed bold in the table represent the best results achieved for the corresponding problem setup.

We observe that decomposition influences the results of the ETA. It is not a surprise to see that high values for the decomposition probability yield the worst results. Indeed, high decomposition probabilities result in an almost systematic decomposition of solutions. As a consequence, partial solutions do not have the necessary time to perform conflict mediation or produce new levels of complexity. Medium values do not perform well either. We had expected that the significant increase in exploration provided by medium levels of decomposition would have helped the process in finding the good solutions. However, even the medium range values for probability did not score well. Actually, only low values for the probability, that scale from 0.01 to 0.05 have a positive impact on the simulation results. In this situation, the ETA with decomposition is able to match and even outperform the results of the generic ETA. This leads to the conclusion that, if the decomposition does influence the search process, it should remain a rare event like the standard mutation operator.

To conclude this comparison of the generic ETA with the ETA with decomposition, we performed the Wilcoxon rank-test between the speeds (defined in Equation 5) of the generic ETA and the best performing ETA with decomposition on each test case. These tests which are summarized in Table 4 compare the speed of each run to find an optimum between a generic ETA and the best performing ETA with decomposition. It evaluates whether the difference in success ratio we observed in Table 3 between both algorithms is significant or not in the global optimization process. The null hypothesis of these tests is that the average speeds of both algorithms are equivalent. The speed is computed the same way as in Equation 5 (see page 114) and captures the average speed of each run in a measure between 0 and 1 where 0 means that the run did not find any optimum value in the given time while 1 means that the solution was found from the very first generation. Higher values for the speed mean that the run was able to find a solution faster than others.

In Table 4 we see that the null hypothesis cannot be rejected under the classical 5% confidence interval except for the test case $p_1 = 0.9, \bar{p}_2 = 0.3$. We can conclude from this that, in general, the differences in performance we observed between the generic ETA and the best scoring ETA with decomposition in Table 3 are not significant. So, introducing a small fraction of decomposition at each generation during the evolutionary process may increase the expected chance of success, although this

Table 4 Wilcoxon-Rank Test between the Generic ETA and the ETA with Decomposition (for the decomposition probability DP that scored best)

BINCSP problem	DP	p-value
$p_1 = 0.9, \bar{p}_2 = 0.3$	0.01	0.01826
$p_1 = 0.7, \bar{p}_2 = 0.5$	0.05	0.2406
$p_1 = 0.5, \bar{p}_2 = 0.5$	0.01	0.08414
$p_1 = 0.3, \bar{p}_2 = 0.7$	0.05	0.7706
$p_1 = 0.1, \bar{p}_2 = 0.9$	0.05	0.955

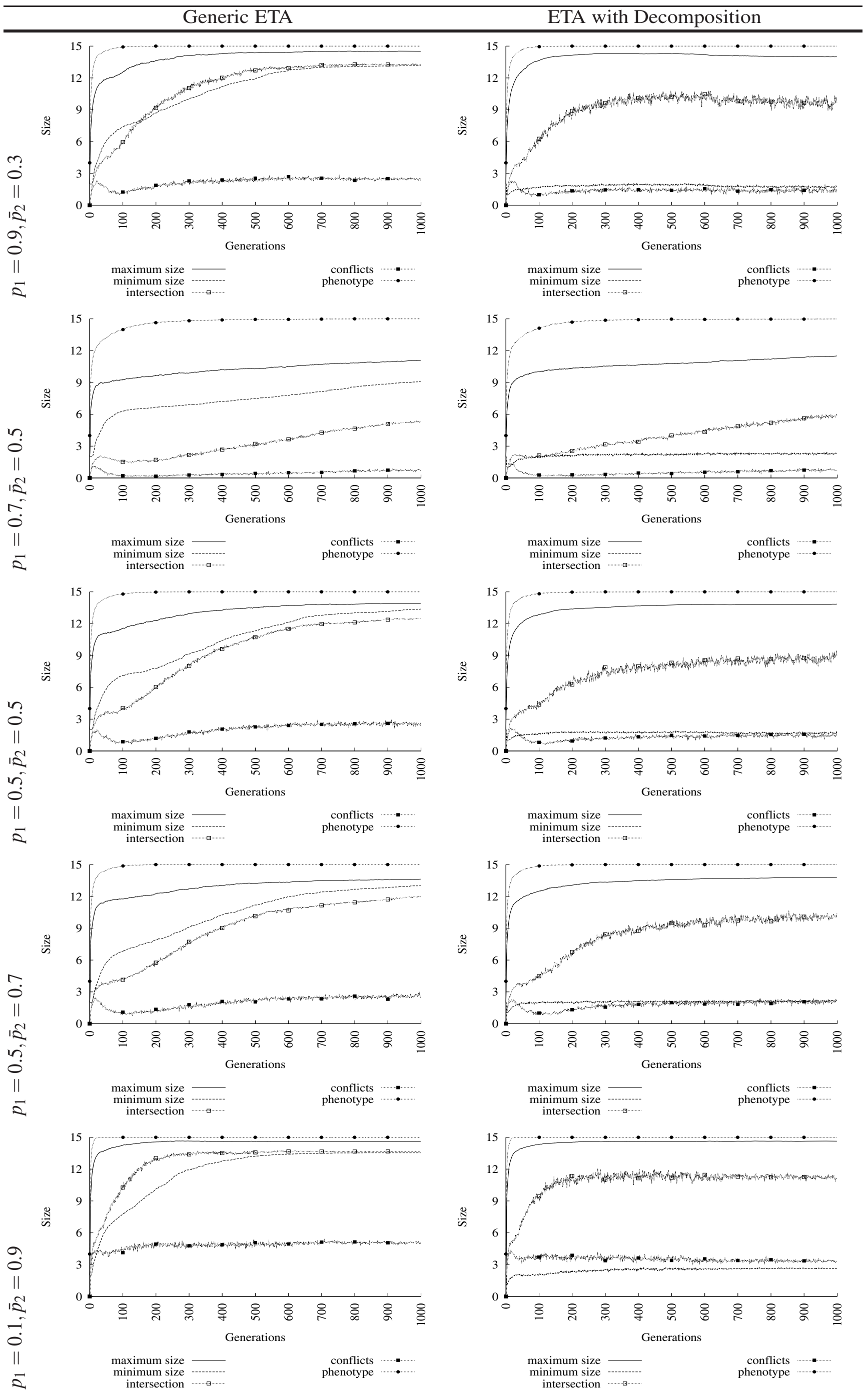


Fig. 5 Comparing size dynamics between generic ETA and ETA with Decomposition

increase is not significant with respect to the dynamics one obtains with a generic ETA without decomposition.

In Figure 5, we illustrate that the ETA with decomposition techniques has a relatively equivalent dynamics for the sizes when compared to a generic ETA. However, decomposition guarantees the survival of smaller building blocks as we can observe that the average minimum size of the ETA with decomposition remains below 3 during the evolutionary process, meaning that the smallest unit of selection is on average of length 2 while for generic ETA the minimum size increases slowly during the process and tends to converge toward the maximum size. This capacity to keep lower level building blocks during the process enforces exploration through combination of these units during the entire process and therefore, through the mechanism of conflict mediation, yields better results as it provides the system with a mechanism to escape local peaks.

5.5 The Relevance of a Good Transition Condition

All the simulation results discussed so far used the covering fitness to determine whether a transition occurred or not. This problem-specific approach will now be compared to a process of random transitions. The motivation for this study is twofold. First of all, problem-specific transition conditions may be difficult to determine. This issue is not new: in the context of learning classifier systems (LCS) [12, 10], the problem of finding a good mechanism to evaluate partial solutions is known as the credit assignment problem [9]. It is clear that given the problems on which one wants to apply the ETA, one will spend some time in determining how to decide on the quality of a partial solution. The difference with the assignment problem in LCS is that we do not need to address how the success of a bunch of classifiers has to be distributed over all classifiers since two partial solutions are merged as soon as they are considered to be adequate, which means they are no longer independent partial solutions.

If random transitions show good performance, a broader class of problems could be addressed with evolutionary transitions as well. Our second motivation is to compare the behavior of both setups and observe how significant the impact of a problem-specific transition condition is on the overall dynamics and success ratio. The random results form a baseline against which we can compare the results of problem-specific transition functions.

5.5.1 Simulations Setup

Our simulation uses the same setup as before. It uses a sound population and performs a total of 250 runs (10 seeds, 25 instances) on different types of BINCSP. In the case of a random transition condition, each selected solution will perform a transition with a certain probability (the Transition Probability TP) whose distribution law is uniform. We consider three possible setups for this:

1. A low transition rate (0.2): this means that each selected solution makes its symbiotic relation permanent with a relatively low probability. At each generation, approximatively 20 % of the population will perform a transition.
2. A medium transition rate (0.5): approximatively 1 solution out of 2 will increase in complexity.
3. A High transition rate (0.8): only 20 % of the population does not perform a transition at each time step.

The random transition condition does not require the symbiotic relation to be successful to see an increase in the genotype complexity. Therefore, all of these rates are more likely to induce a relatively fast emergence of maximum size genotypes in the population. This means that after several generations, the evolution toward the solution only occurs through recombination of existing solutions by means of symbiosis[1]. In these simulations, we wish to observe whether the observed transitions perform significantly better than random based transitions.

5.5.2 Simulation Results

In Table 5, we show the results obtained for a set of difficult BINCSP instances. The results show that the choice of transition condition has quite a significant impact on the performance of the algorithm. It is not surprising that the generic ETA outperforms the random transition ETA. The performance of the generic ETA in comparison to the random transition ETA is, however, particularly dominant. Table 6 illustrates that this difference is significant for all test cases and setups of TP. Random transitions produce a fully defined genotype too quickly. As a consequence, the ETA can only evolve a solution through the recombination operation provided by

Table 5 Comparison of the success ratio between an ETA that uses a problem-specific transition condition and ETAs that perform transition on a randomly (with the Transition Probability TP)

		p_1: 0.9	0.7	0.5	0.3	0.1
		$\bar{p}_2$: 0.3	0.5	0.5	0.7	0.9
	TP:					
		0.972	0.29	0.54	0.28	0.992
	Generic	(2335)	(8673)	(11409)	(11758)	(3627)
		[4902]	[-]	[-]	[-]	[1224]
		0.464	0.064	0.136	0.036	0.804
Low	0.2	(6936)	(10431)	(12624)	(2011)	(3917)
		[7774]	[5725]	[11080]	[2310]	[3484]
		0.612	0.08	0.196	0.072	0.868
Med.	0.5	(7248)	(2546)	(6256)	(20503)	(4244)
		[8446]	[4169]	[5708]	[26279]	[4529]
		0.0	0.0	0.0	0.0	0.0
High	0.8	(-)	(-)	(-)	(-)	(-)
		[-]	[-]	[-]	[-]	[-]

[1] Although similar to uniform cross-over in GAs, this recombination mechanism differs in the sense that recombination occurs prior to selection while GAs perform recombination after selection.

Table 6 Comparisons tests between the different random based transition and the generic ETA. The values in each column correspond to the p-value of the Wilcoxon-Mann-Whitney ranking test.

	0.2	0.5	0.8
$p_1 = 0.9, \bar{p}_2 = 0.3$	< 0.0001	< 0.0001	-
$p_1 = 0.7, \bar{p}_2 = 0.5$	< 0.0001	< 0.0001	-
$p_1 = 0.5, \bar{p}_2 = 0.5$	< 0.0001	< 0.0001	-
$p_1 = 0.3, \bar{p}_2 = 0.7$	< 0.0001	< 0.0001	-
$p_1 = 0.1, \bar{p}_2 = 0.9$	< 0.0001	< 0.0001	-

the symbiotic relation. It ignores one important step of compositional search: First conquer the lower level of complexity before tackling higher levels of complexity.

Figures 6, 7, 8 and 9 show the evolution of the fitness, maximum size, minimum size and conflicts of each algorithm on the difficult test cases.

The same observation can be made for each test case; High values of TP lead to the situation where the worst average fitness is obtained. This means that the high probability of transition prevents selection from performing its task well (i.e. distinguishing the good symbiotic relations from the bad ones). This observation is confirmed by the conflicts' curves in the figures. We observe that the highest value of TP yields the highest number of conflicts at the beginning of the evolutionary process (which results from premature transitions). Furthermore, this number of conflicts soon tends to become zero. This means that the evolutionary process converges

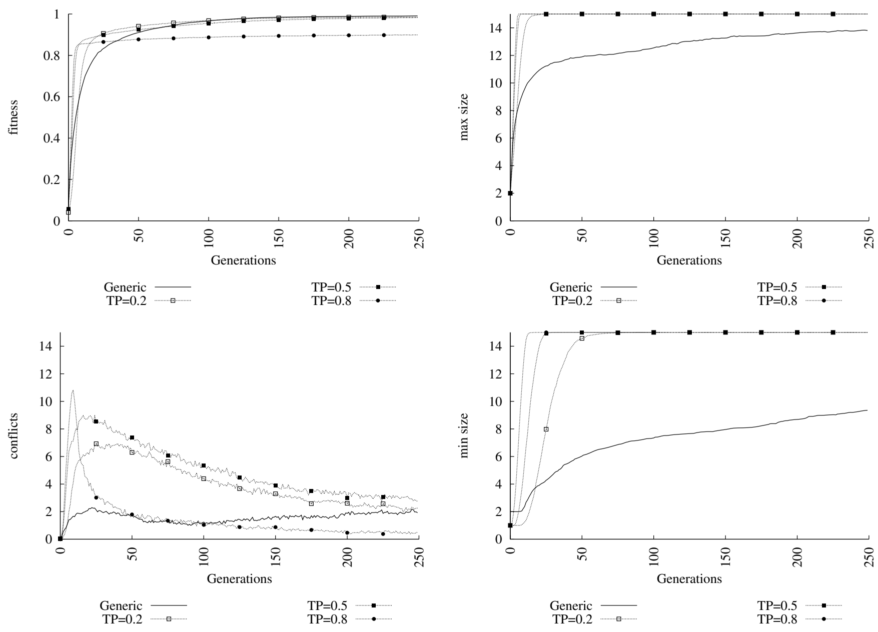


Fig. 6 Comparison between random base transition and observed transition (generic ETA). $p_1 = 0.9, \bar{p}_2 = 0.3$.

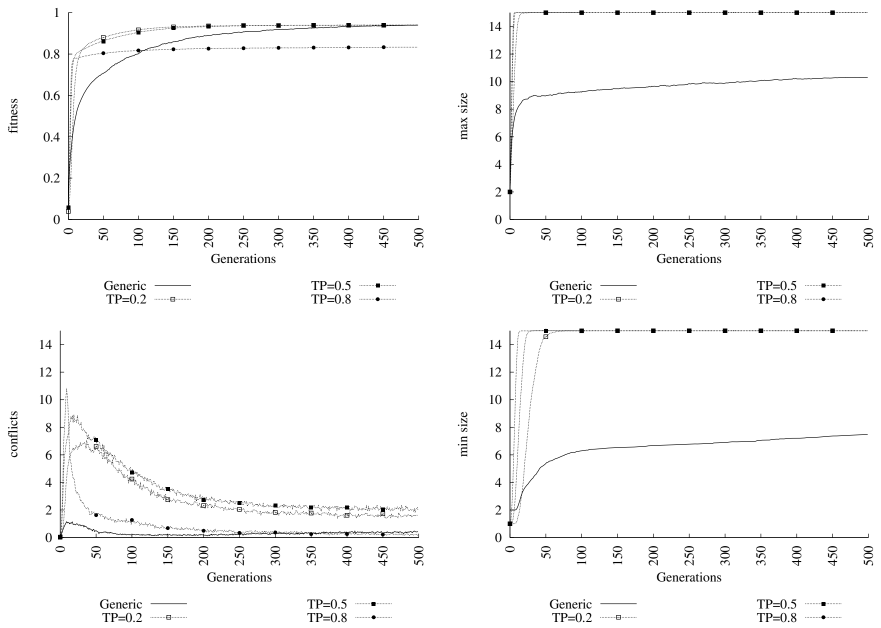


Fig. 7 Comparison between random base transition and observed transition (generic ETA). $p_1 = 0.7, \bar{p}_2 = 0.5$.

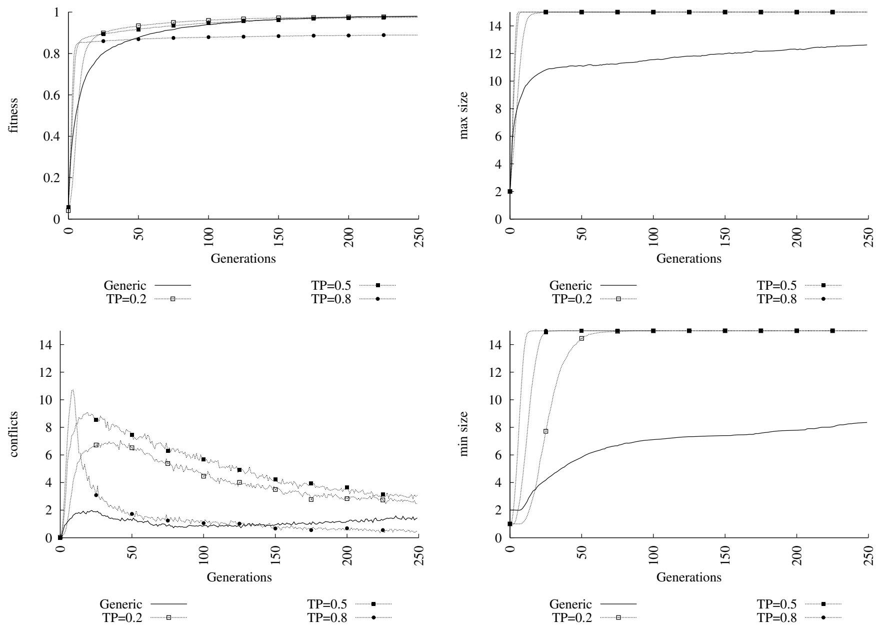


Fig. 8 Comparison between random base transition and observed transition (generic ETA). $p_1 = 0.5, \bar{p}_2 = 0.5$.

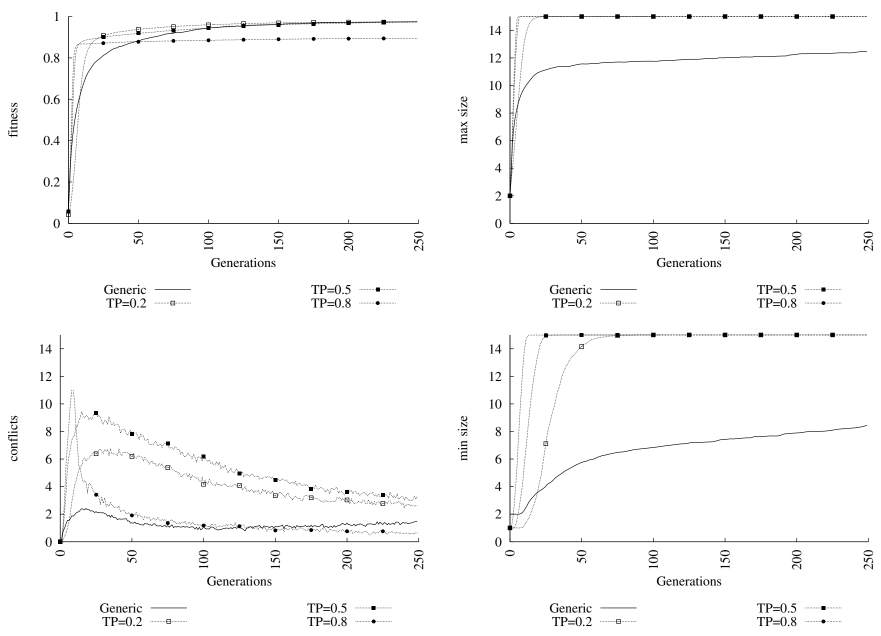


Fig. 9 Comparison between random base transition and observed transition (generic ETA). $p_1 = 0.3, \bar{p}_2 = 0.7$.

toward a sub-optimal solution and is lacking of the necessary exploration to evolve a solution to the problem. The other *TP* values (0.2 and 0.5) perform much better, achieving similar average fitness levels as the generic ETA. The conflicts' curves with scores above the generic case imply that the population is able to maintain sufficient diversity and therefore, some exploration is still active. However, a brief look at the evolution of the maximum and minimum sizes reveals that for all the random based transition instances, full genotypes are produced from the very beginning of the process (in less than 100 generations) and the population soon contains only fully defined genotypes (as the minimum size also converges to a fully defined genotype). Conversely, in the generic ETA, we observe that the maximum and minimum sizes remain at lower complexity levels and that different levels of complexity are present in the population (as the maximum size differs from the minimum size). This means that random-based transitions lose the efficiency inherent in the compositional approach and that these processes have to, relatively early in the evolutionary process, rely only on recombination of fully defined genotypes to evolve a solution.

6 Conclusion

In this chapter, we introduced an algorithm that mimics evolutionary transitions from biology and tackles evolutionary compositional search. We applied the ETA on a test case which has many practical applications [20, 21]: the Binary Constraints Satisfaction Problems. We wished to evaluate how ETA performs on a challenging test case and simulated the ETA on randomly generated problem instances for various setups of the BINCSP. These instances scale from easy to difficult depending on the parameters p_1 and $\bar{p}_2$ [25]. We extensively analyzed the relevance of certain design decisions that were made for the ETA. From the simulation results, the following conclusions can be drawn.

First of all, ETA succeeds in compositionally building complex solutions by aggregating partial solutions through the mechanism of symbiotic relations and transitions. In the case where the system fails to evolve a fully defined genotype for the given problem instances, it is able to evolve two different partial genotypes that together through, their symbiotic relation, yield a full representation for a solution. Yet, this fully defined solution is sub-optimal. A possible reason explaining why ETA failed to evolve complete solutions for the difficult problem instances may reside in the random nature of the instances themselves. These instances are randomly generated, which means that the constraints network is, most likely, totally unstructured and that the resulting epistasis (correlation between the variables) is high. ETA, by its compositional nature, is more likely to work fine on structured instances where modularity in the problem is more apparent. This requirement for structured problem instances is confirmed when we observe the performance of ETA with other co-evolutionary approaches on BINCSP [1]. The fact that the ETA performs well on structured problems makes it promising for real-world applications, since most of them tend to be structured in some way. The study of the impact of the initial population setup on the outcome of ETA demonstrated little differences in the results [3]. This means that modeling the initial population setup is not necessarily an issue and can be kept relatively simple (for example, an initial small population of random assignments).

Concerning the impact of the introduction of decomposition techniques to the ETA, we showed that the outcome of the algorithm is positively influenced when decomposition is a rare event (only a very small fraction of the population should see the solutions being decomposed). Even though it may improve the success ratio, a statistical comparison of the best performing decomposition ETA with the generic ETA showed that, in general, there are no significant differences in terms of speed.

To analyze the quantitative relevance of a problem-specific transition condition, we re-examined all previous BINCSP results using a random (performance-independent) transition condition. We conclude that the choice to perform a transition on a symbiotic relation should be considered thoroughly and should be related to the nature of the problem to be solved as we observed that the ETA with problem-specific transition condition significantly outperformed any setup of the ETA with random condition.

Regarding the evaluation of the performance of ETA on structured problems, the ETA will be evaluated on hierarchically structured problems like the Hierarchical if-and-only-if function [27]. Moreover, we are currently investigating the performance of the ETA on combinatorial optimization problems where one does not only need to find an assignment of values to variables but also find the optimal assignment. These new studies should provide additional understanding on the applicability of the ETA for real-world applications.

References

1. Defaweux, A., Lenaerts, T.: Evolutionary transitions in sequence complexity; a proof of concept. In: Proceedings of the Annual Machine Learning Conference of Belgium and The Nederlands, Brussels, Belgium, pp. 38–45 (2004)
2. Defaweux, A., Lenaerts, T., van Hemert, J.: Evolutionary transitions as a metaphor for evolutionary optimisation. In: Capcarrère, M.S., Freitas, A.A., Bentley, P.J., Johnson, C.G., Timmis, J. (eds.) ECAL 2005. LNCS, vol. 3630, pp. 342–352. Springer, Heidelberg (2005)
3. Defaweux, A., Lenaerts, T., van Hemert, J., Parent, J.: Complexity transitions in evolutionary algorithms: Evaluating the impact of the initial population. In: Proceedings of the IEEE Congress on Evolutionary Computation, Edinburgh, UK, pp. 2174–2181 (2005)
4. Defaweux, A., Lenaerts, T., van Hemert, J., Parent, J.: Transition models as an incremental approach for problem solving in evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, Washington DC, USA, pp. 599–607 (2005)
5. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic algorithms, noise, and the sizing of populations. Complex Systems 6, 333–362 (1992)
6. Goldberg, D.E., Korb, B., Deb, K.: Messy genetic algorithms: motivation, analysis, and first results. Complex Systems 3, 493–530 (1989)
7. Goldberg, D.E., Korb, B., Deb, K.: Messy genetic algorithms revisited: Studies in mixed size and scale. Complex Systems 4, 415–444 (1990)
8. Golomb, S., Baumert, L.: Backtrack programming. Journal of the ACM 12, 516–524 (1965)
9. Greffenstette, J.: Credit assignment in rule discovery systems based on genetic algorithms. Machine Learning 3, 225–245 (1988)
10. Holmes, J., Lanzi, P.L., Stolzmann, W., Wilson, S.W.: Learning classifier systems: New models, successful applications. Information Processing Letters 82, 23–30 (2002)
11. Hoos, H.H., Stützle, T.: Stochastic Local Search: Foundations and Applications. Morgan-Kaufmann/ Elsevier (2005)
12. Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.): IWLCS 1999. LNCS, vol. 1813, p. 243. Springer, Heidelberg (2000)
13. Lenaerts, T.: Different Levels of Selection in Artificial Evolutionary Systems: Analysis and Simulation of Selection Dynamics. PhD thesis, Department of Computer Science, Vrije Universiteit Brussel, Belgium (2003)
14. Maynard Smith, J., Szathmáry, E.: The Major Transitions in Evolution. W.H. Freeman, San Francisco (1995)
15. Michod, R.: Darwinian Dynamics: Evolutionary transitions in Fitness and Individuality. Princeton University Press, Princeton (1999)
16. Potter, M.: The Design and Analysis of a Computational Model of Cooperative Coevolution. PhD thesis, Department of Computer Science, George Mason University, USA (1997)

17. Raynal, F., Collet, P., Lutton, E., Schoenauer, M.: Individual gp: an alternative viewpoint for the resolution of complex problems. In: Banzhaf, W., et al. (eds.) Proceeding of the Genetic and Evolutionary Computation Conference, Orlando, Florida, USA, pp. 974–981 (1999)
18. Raynal, F., Collet, P., Lutton, E., Schoenauer, M.: Polar ifs + parisian genetic programming = efficient ifs inverse problem solving. Genetic Programming and Evolvable Machines Journal 1(4), 339–361 (2000)
19. Rossi, F., Dhar, V.: On the equivalence of constraint satisfaction problems. In: Aiello, L.C. (ed.) Proceedings of the 9th European Conference on Artificial Intelligence, Stockholm, Sweden, pp. 550–556 (1990)
20. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs (1995)
21. Salido, M.A., Garrido, A., Barták, R.: Introduction: special issue on constraint satisfaction techniques for planning and scheduling problems. Engineering Applications of Artificial Intelligence 21(5), 679–682 (2008)
22. Simon, H.A.: The Sciences of the Artificial, 3rd edn. MIT Press, Cambridge (1969)
23. Thierens, D., de Jong, E.D., Watson, R.A.: On the complexity of hierarchical problem solving. In: Proceedings of The Genetic and Evolutionary Computation Conference, Washington DC, USA, pp. 1201–1208 (2005)
24. Tsang, E.P.K.: Foundations of Constraint Satisfaction. Academic Press, London (1993)
25. van Hemert, J.: Application of Evolutionary Computation to Constraints Satisfaction and Data Mining. PhD thesis, Universiteit Leiden, Netherlands (2002)
26. van Hemert, J.: RandomCSP Freely (2002), `http://freshmeat.net/projects/randomcsp/`
27. Watson, R.: Compositional Evolution: Interdisciplinary Investigations in Evolvability, Modularity, and Symbiosis. PhD thesis, Brandeis University, USA (2002)
28. Watson, R.A., Pollack, J.B.: Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 425–434. Springer, Heidelberg (2000)
29. Watson, R.A., Pollack, J.B.: A computational model of symbiotic composition in evolutionary transitions. Special Issue on Evolvability Biosystems 69(2-3), 187–209 (2002)
30. Wiegand, P.: An Analysis of Cooperative Coevolutionary Algorithms. PhD thesis, George Mason University, USA (2004)

A Model-Assisted Memetic Algorithm for Expensive Optimization Problems

Yoel Tenne

Abstract. This chapter proposes a new model-assisted memetic algorithm for expensive optimization problems. The algorithm follows successful optimization approaches such as a combined global–local, modelling and memetic optimization. However, compared to existing studies it offers three novelties: a statistically-sound framework for selecting optimal models during both the global and the local search, an improved trust-region framework and a procedure for improved exploration based on modifying previously found sites. The proposed algorithm uses a radial basis function neural network as a global model and performs a global search on this model. It then uses a local search with a trust-region framework to converge to a true optimum. The local search uses Kriging models and adapts them during the search to improve convergence. A rigorous performance analysis is given where the proposed algorithm is benchmarked against four reference algorithms using eight well-known mathematical test functions. The individual contribution of the components of the algorithm is also studied. Lastly, the proposed algorithm is also applied to a real-world application of airfoil shape optimization where it is also benchmarked against the four reference algorithms. Statistical analysis of all these tests highlights the beneficial combination of the proposed global and local search and shows that the proposed algorithm outperforms the reference algorithms.

1 Introduction

Modern engineering design optimization replaces expensive laboratory experiments with *computer experiments*. These are computationally-intensive simulations which accurately model real-world physics. Using computer experiments reduces the cost and time of the design process and so they are used in diverse areas ranging from the design of integrated circuits [105] to complete aircraft [31].

Yoel Tenne
School of Aerospace, Mechanical and Mechatronic Engineering,
The University of Sydney, Australia
e-mail: `joel.tenne@aeromech.usyd.edu.au`

With computer experiments, the engineering design process is cast as a nonlinear optimization problem having three distinct features:

- The objective function (or cost function) being minimized depends on the simulation outputs. However, such simulations are often legacy computer codes available only as executables. As such there is no analytic expression relating the simulation inputs to its outputs and so the simulation is treated as a *black-box function*. This means a suitable optimization algorithm must rely only on the observed responses and not require the function expression or derivatives.
- Each simulation run is *expensive*, that is it requires anywhere from minutes to hours of computer run time [86, 88]. This means only a small number of simulation runs can be made.
- The underlying real-world physics and possibly the numerical solution itself often give a complicated inputs–outputs response surface, for example which is multimodal and nonsmooth [16]. This means an elaborate optimization algorithm should be used.

Due to these issues common optimization algorithms often perform poorly in such problems. For example, nonlinear programming algorithms [23] either require the function derivatives or approximate them by finite-differences which is too expensive. Heuristics and nature-inspired algorithms [70] use only the observed responses but they often require far too many function evaluations to converge.

These difficulties have motivated new optimization approaches and we review two of these in the following two subsections.

1.1 Hybrid or Memetic Algorithms

One approach to improving the optimization search is by combining several algorithms. For example, the chances of locating a good optimum are improved when the search combines both a global and local search. The global search explores the function landscape to identify promising regions. A local search then exploits local information to identify a better solution. As such, finding a good solution requires an *exploration–exploitation balance*. This approach originated in the global optimization community in the mid 1970s and has been applied in various algorithms [96, 116].

In the late 1980s researchers in the evolutionary algorithm community proposed similar approaches. Goldberg [32, p.202–204] described *hybrid schemes* which combine an evolutionary algorithm (EA) with a local search. Norman and Moscato [77] proposed a similar approach for combinatorial optimization. Moscato [74] termed such combined approaches as *memetic algorithms* following the concept of a ‘meme’ which is a unit of imitation in cultural transmission or an abstract unit of information [21]. In all these cases a population-based algorithm explores the function landscape and efficiently adapts to it [49]. A local search is also used to improve individuals (candidate solutions) by focusing on a small region.

Later studies have focused on hybridization with gradient-based algorithms such as finite-difference quasi-Newton [37, 90, 94, 103]. Others have studied hybridization with heuristics such as the simplex or hill-climbing algorithms [93, 122]. Some have replaced the EA by a simulated annealing algorithm (SA) [43].

Recent studies have focused on improving the search by analyzing the exploration–exploitation interaction [44]. If the cost of the local search is low it may be beneficial to apply it to all individuals [55]. Another approach uses fuzzy logic to balance exploration–exploitation [123]. It is also possible to adaptively select the 'best' type of local search to use [78].

Hybrid and memetic algorithms are an active research field and recent reviews are given in [54, 81]. A book [39] and three special issues [38, 82, 83] have focused on these algorithms which indicates their significance as an emerging research field.

1.2 Reducing the Number of Expensive Evaluations

One main difficulty in expensive optimization is the tight limit on function evaluations. To illustrate the problem we consider a population-based algorithm with a population size s and which is run for a total of g generations or iterations. The total run time of the algorithm (T) depends on the time required by the optimization algorithm alone (t_a) for actions such as sorting strings or performing mutations but excluding function evaluations, and the time required by a function evaluation (t_f). As such the total run time is

$$T = g(s \times t_f + t_a) . \tag{1}$$

In expensive problems each function evaluation is a simulation run which requires minutes to hours of computer time. As such we can assume that the algorithm time is negligible, that is $t_f \gg t_a$ and so

$$T \simeq g(s \times t_f) . \tag{2}$$

To get an estimate of the required time we use the EA settings recommended in [50], that is a population size of $s = 50$ and $g = 1000$ generations. This means the EA requires 50,000 function evaluations for its run. If a single simulation requires an hour (but often much longer) that is $t_f = 1\,\text{hr}$ then a full run of the EA would require about 2083 days or 5.5 years. This is unacceptable in practice and shows that population-based algorithms such as EAs cannot be directly applied to expensive problems. As such several approaches have been studied to solve this difficulty.

In *fitness inheritance* only a fraction of the population is evaluated with the expensive function while the remaining candidate solutions inherit their fitness from their 'parents' [95, 107]. An extension of this approach uses clustering to assign a variable fitness to offspring [52].

A second approach uses *hierarchical* or *variable-fidelity* simulations. Here the algorithm uses several simulations which differ by their accuracy (or fidelity) and so by their computational cost. Promising solutions migrate from low- to high-fidelity simulations and vice versa [27, 102]. The approach was also used with deterministic nonlinear programming algorithms [2].

A third approach is that of *parallelization*. It does not reduce the number of expensive evaluations but only the wall-clock time needed to obtain a solution. Population-based algorithms such as EA and SA operate in a decentralized manner and so they are easily implemented on parallel machines [80, 86].

In this study we take the approach of *Modelling*. Models are computationally-cheaper approximations of the expensive black-box function. They are based on function approximation theory, that is, they interpolate the unknown function based on the observed responses [62, 69, 117]. Since they are cheaper to evaluate, a *model-assisted* algorithm uses the model instead of the expensive function during most of the search [4, 31, 106].

The framework of *model-assisted optimization*, also called *design and analysis with computer experiments* [98], involves three main components [28, 99]:

- Selecting the sites where the expensive function will be evaluated (design of experiment).
- Generating a model based on the sample and
- Assessing the model accuracy.

The early approach of Response Surface Methodology was developed for noisy real-world experiments and used least-squares quadratic models and designs which aim to counter the noise, such as full and fractional factorial designs [8, 76].

However computer experiments are deterministic and so are noiseless (the same inputs repeatedly give the same outputs). Therefore more suitable methods have been studied. Designs tailored for computer experiment are *space-filling*, meaning they spread the sample sites over the search space (instead of resampling at the same location to counter noise). These include Latin hypercube designs [67], orthogonal arrays [84] and maximin designs [47]. Also, the lack of noise motivates using more flexible models which interpolate more accurately than the least-squares quadratics. Such models include neural-networks [6, 40] (also discussed in Sect. 3.3), Kriging [20] (also discussed in Sect. 3.5.2) and radial basis functions (RBFs) [11].

Whatever model is used, it is likely to be inaccurate due to the small sample size. It is then necessary to estimate the degree of inaccuracy since a poor model can drive the optimization search to a *false optimum*, that is an optimum of the model but not of the true function [46]. One approach to estimate the model accuracy is with statistics of goodness-of-fit [76, p.28–36], [120]. Another is with resampling methods which train a model using part of the available sample and test the model using the remaining part [61, Ch.2]. Recent studies have compared various methods for model accuracy assessment [68, 112].

1.3 Model-Assisted Algorithms

The modelling approach has proven to be efficient and effective and as such several classes of model-assisted algorithms have emerged.

One class of algorithms uses Kriging models in a Bayesian statistics framework. Kriging models are a statistical-oriented approach to interpolation and are discussed in length in Sect. 3.5.2. Briefly, a Kriging model treats the black-box function as a combination of a deterministic function and a Gaussian random process. Statistical methods select the parameters of these functions to improve the model accuracy. After generating the Kriging model the algorithm seeks the site which is expected to either improve the best value found so far or to improve the overall model accuracy. These two objectives are combined to give a merit value known as the 'expected improvement' (EI). Sites already evaluated have an EI = 0 while all others have an EI which reflects both the predicted model value and the uncertainty about this prediction. At each iteration the algorithm samples the site having the highest EI value and it updates the model, which then results in new EI values for all sites. As such the EI approach balances between global and local search. The approach originated in 1960s with Kushner's univariate method [57]. Later studies extended it to multivariate functions [34, 110] and incorporated the Bayesian framework [72, 119], including the recent paper by Jones et. al. [48].

Another class of algorithms uses quadratic interpolants as models, motivated by a Taylor series function approximation. Quadratics both account for function curvature which assists the optimization and are simple to optimize. Algorithms in this class combine quadratics with a trust-region framework to ensure convergence to an optimum of the true expensive function [17]. Winfield studied in 1970s an early approach of a model-assisted algorithm using quadratics [121]. Powell [89] and Conn et. al. [14, 15] have later improved the approach based on recent advances in interpolation theory.

A third class is the *surrogate-management framework*. It uses a variant of Torczon's pattern search algorithm [115] as the search algorithm. The pattern search seeks the optimum of the current model (termed 'Search Step'). If it fails to find a new optimum then the model is refined (termed 'Poll Step') [7]. No restriction is made on the model type.

A fourth class is that of *model-assisted memetic algorithm*. The idea is to generate a model and seek its optimum with a memetic algorithm. A number of candidate solutions are then evaluated with the expensive function, the model is updated and the process repeats. One algorithm combines an EA, a neural network and a local search [30, 88]. Other algorithms combine an EA with global and local radial basis function models [79, 80, 114, 126, 127]. An algorithm which combines an EA with quadratic models and a local search was studied in [60, 113].

The latter class has proven to be both efficient and effective and following its success we propose a new model-assisted memetic algorithm for expensive optimization problems. Briefly, the algorithm first trains and selects a global model which is an artificial neural network and seeks the optimum of this model. It then uses a local search to improve this predicted optimum. This sequence is repeated until the number of function evaluations reaches the prescribed limit. Compared to existing studies the proposed algorithm contains three main novelties:

- Model selection and model management using statistical model selection: typically there will be a family of candidate models. Due to lack of domain knowledge the user often chooses a non-optimal model which degrades the optimization search. To address this and to improve the search the proposed algorithm selects all models under a unified and statistically-sound framework of model selection. This yields optimal models which are adapted during the search.

- Improved trust-region framework: to converge to a true optimum the proposed algorithm uses a trust-region approach. We describe several improvements to this approach, such as selecting sites to improve the model and more efficient stopping criteria. Further, we replace the quadratic models (in the classical approach) with Kriging model and adapt the models during the search.
- Improved global exploration: a model can lead the optimization search to converge repeatedly to the same optimum without promoting exploration of the search space, a condition termed 'model stall'. To address this we propose a modification of sampled sites which promotes exploration and discovery of new optima. The method is computationally-efficient and applicable to any type of model.

Rigorous performance analysis shows the proposed algorithm outperforms several reference algorithms.

This chapter is organized as follows: Sect. 2 reviews concepts of model selection theory relevant to the proposed algorithm. Section 3 then describes in detail the proposed algorithm and Sect. 4 provides a rigorous performance analysis. It is followed by Sect. 5 which summarizes this chapter.

2 Model Selection and Complexity Control

A major aspect of the proposed algorithm is the selection of optimal models. This section briefly explains the basics of statistical model selection theory and focuses on the approach used in the proposed algorithm.

In a model selection problem we are given a set of sites and responses generated by an unknown function and we wish to select a model which best describes this function [12, 61]. The model is selected from a family of candidate models. The statistical theory of model selection uses a *discrepancy* ($\varDelta$), which is the mismatch between model predictions and the true responses, to measure the goodness-of-fit of a candidate model to the given data. The discrepancy is calculated for each candidate model and the model chosen is the one having the smallest discrepancy.

Based on information theory we consider the *Kulback-Leibler discrepancy* [56]. It uses the *likelihood* of a candidate model given the data, that is the conditional probability of observing the sample of sites and responses

$$\mathcal{X} = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}, \quad i = 1 \ldots n, \tag{3}$$

under the model $\mathcal{S}(\boldsymbol{x})$, or $\mathcal{L}(\mathcal{S}|\mathcal{X})$ [87]. The discrepancy is then

$$\varDelta = -\log \mathcal{L}(\mathcal{S}|\mathcal{X}). \tag{4}$$

With the Kulback-Leibler discrepancy the optimal model is the one having the *maximum likelihood*, that is

$$\varDelta_{\min} = -\log \mathcal{L}_{\max}(\mathcal{S}|\mathcal{X}). \tag{5}$$

For some models a closed-form expression exists for the likelihood and the discrepancy. Otherwise, an empirical discrepancy [61, Ch.5] is used where

$$\Delta = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} (\mathcal{S}(\boldsymbol{x}) - f(\boldsymbol{x}))^2 , \tag{6}$$

which is the mean of sum of squared error between the true responses and the model predictions over the sample. Since the likelihood is a function of the model (and hence of the model parameters), maximizing the likelihood by solving (4) gives the optimal model parameters [65].

The family of feasible models may contain models of different *complexity*, that is the number of model parameters. More complex models may fit the sample better than simpler ones but their prediction at new sites (or *generalization*) may be poor, a condition termed *over-fitting* [6, Ch.9]. Often a simpler model may be a better approximation of the true function.

This motivates the selection of models based not only on their discrepancy but also on their complexity. As such we consider the *Akaike information criterion* (AIC) for complexity control [1], [61, p.243–245]. The criterion uses the Kulback-Leibler discrepancy but adds a penalty which increases with model complexity where

$$\mathrm{AIC} = -\log \mathcal{L}(\mathcal{S}|\mathcal{X}) + 2m , \tag{7}$$

and m is the number of model parameters. As such a more complex model is preferred over a simpler one only if it is significantly more accurate. The optimal model is the one having the lowest AIC value.

The AIC was derived under asymptotic assumptions of a large sample. However in expensive optimization problems the sample is small and the AIC becomes biased [3, 59]. As such we use the *corrected Akaike information criterion* ($\mathrm{AIC_c}$) which is unbiased for small samples where

$$\mathrm{AIC_c} = \mathrm{AIC} + \frac{2(m+1)(m+2)}{n-m-2} , \tag{8}$$

and n is the sample size [42]. The $\mathrm{AIC_c}$ has performed well against other complexity control criteria [3, 42].

3 The Proposed Algorithm

3.1 Initialization and Main Loop

The proposed algorithm begins by generating a Latin hypercube design (LHD) consisting of $k = 0.2 fe_{\max}$ sites, where $fe_{\max}$ is the prescribed limit of expensive evaluations. As mentioned in Sect. 1.2, this design improves the accuracy of the resultant model by spreading the sites in the search space.

To generate a LHD of k sites the range of each variable is split into k equally sized intervals and one point is sampled at random in each interval. To create a LHD site a sampled point is selected at random (without replacement) for each variable and these samples are combined to give a site (a vector). The process is repeated until k sites have been created. Algorithm 1 gives a pseudocode for generating a LHD sample.

Algorithm 1. Generating a LHD sample

```
inputs
    number of variables ($d$);
    sample size ($k$);
    bounds on variables;

for each variable $i = 1 \dots d$ do
    divide the variable range into $k$ equal intervals;
    sample one point (a scalar) at random in each interval;
for each LHD site $\boldsymbol{x}_j$, $j = 1 \dots k$ do
    for each variable $i = 1 \dots d$ do
        select at random and without replacement a sample point;
        set $i$th component of site $j$ (that is $\boldsymbol{x}_{j,i}$) to selected sample point;
Output: a Latin hypercube design of size $k$
```

The expensive function is then evaluated at the LHD sites. During the search the algorithm caches all sites evaluated with the expensive function and their responses to reuse them later in the search and to reduce new evaluations.

The main loop then begins and the algorithm generates a global model of the objective function, as follows. It first uses a modified copy of the cache where sites found during previous local searchs have been ‘masked’ (as described in Sect. 3.2). It then uses this modified copy to train and select an artificial neural network which serves as the global model (as described in Sect. 3.3). Next, it seeks the optimum of the global model (as described in Sect. 3.4) and improves this predicted optimum with a local search (as described in Sect. 3.5). This sequence is

Algorithm 2. Main loop

```
generate an initial LHD;
evaluate and cache sites;
while $fe \leqslant fe_{\max}$ do
    if local searchs have been made then
        create a copy of the cache and ‘mask’ sites found during local searchs;
    using the (modified) cache train and select a global model;
    select initial site for the local search (the model’s predicted optimum);
    improve predicted optimum with a local search;
Output: best solution and response found
```

repeated until the number of function evaluations reaches the prescribed limit $fe_{\max}$ ($fe_{\max} = 100, 150$ and 200 were used for performance analysis). A pseudocode of the main loop is given in Algorithm 2.

3.2 Modifying the Cache to Assist Exploration

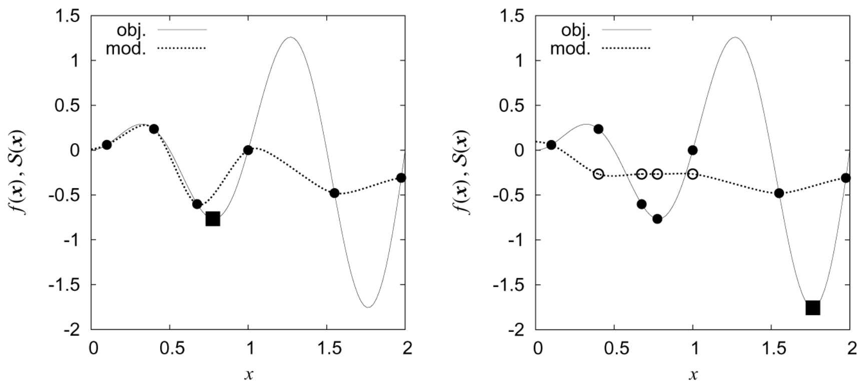
The global model is trained using the cached sites and responses. Elite sites (having a low response), which are typically found during the local searches, can ‘dominate’ the model so other minima will not be represented. To avoid this and to promote exploration such elite sites (and nearby sites) are identified and ‘masked’ by setting their response to the mean response of all cached sites. To identify sites nearby elites the proposed algorithm clusters the cached sites. All sites in clusters with sites found in previous local searchs have their responses set to the mean of responses in the cache ($\bar{f}$). When the global model is trained using this modified cache it will not be dominated by the elite sites and will promote exploration. The algorithm uses the mean response to avoid adding artificial multimodality to the global model.

For clustering the algorithm uses the *k*-harmonic means algorithm [124, 125] which is efficient and has outperformed competing algorithms such as the popular *k*-means [35]. At each iteration the algorithm finds the location of the *k* cluster centres as the harmonic mean of the distance to all sites, so all sites are accounted for. This improves the clustering quality and differs from the popular *k*-means algorithm where only the nearest sites to a centre are accounted for. Algorithm 3 gives a pseudocode of the *k*-harmonic means algorithm.

The *k*-harmonic means requires the number of clusters (*k*) as an input and so to find the optimal *k* the proposed algorithm uses a model selection approach. Following

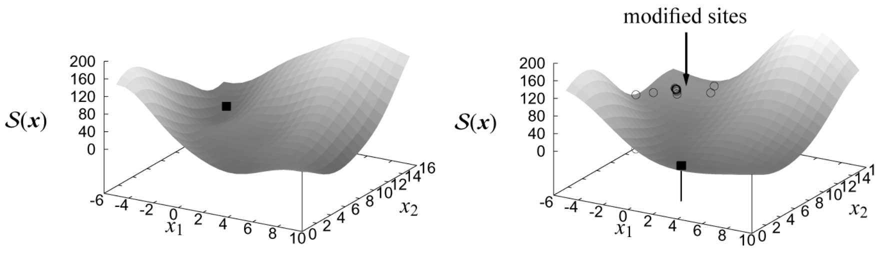
Algorithm 3. *k*-harmonic means clustering

```
Input: sites to cluster $\boldsymbol{x}_j$, $j = 1 \dots n$
set $t = 1$ ;                                    /* iterations counter */
initialize centres $\boldsymbol{c}_i$, $i = 1 \dots k$ at random;
repeat
    for $i = 1 \dots k$ do scan over clusters
        for $j = 1 \dots n$ do scan over sites
            $d_{i,j} = \|\boldsymbol{c}_i - \boldsymbol{x}_j\|_2$ ;          /* distance of centre $i$ to site $j$ */
            $q_{i,j} = {d_{i,j}}^3 \left( \sum_{p=1}^{k} \frac{1}{{d_{p,j}}^2} \right)^2$ ;
        $\boldsymbol{c}_i = \frac{\sum_{j=1}^{n} \frac{1}{q_{i,j}} \boldsymbol{x}_j}{\sum_{j=1}^{n} \frac{1}{q_{i,j}}}$ ;          /* new centre $i$ is harmonic mean */
    t = t + 1;
until change in centres is small or max. iterations ;
```



(a) First local search (b) Second local search

Fig. 1 A 1-D example of the proposed cache modification. The objective function is $f(x) = x\sin(x)$. The sampled sites, objective function and model are shown. (a) shows the first local search finds the local optimum at $x = 0.75$ (■). Sites are then clustered and those in the cluster containing the found optimum have their responses modified (⊙). (b) shows this leads to a model which identifies the second optimum such that the second local search now finds to the optimum at $x = 1.75$



(a) First local search (b) Second local search

Fig. 2 A 2-D example of the proposed cache modification. The objective function is Branin. (a) shows the first local search finds the local optimum at $(-3.1, 12.2)$ (■). Next, cached sites are clustered and those in the cluster containing the found optimum have their responses modified (⊙). (b) shows the second local search used with the model based on the modified sites now converges to a different optimum at $(3.1, 2.2)$ (■)

Sect. 2, the optimal k is found by minimizing the corrected Akaike information criterion (AIC$_c$), where the discrepancy function is the inter-cluster error

$$\Delta = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \|\boldsymbol{c}_i - \boldsymbol{x}_j^{(i)}\|_2 , \tag{9}$$

where n_i is the number of sites in cluster i, $\boldsymbol{c}_i$ is the ith cluster centre and $\boldsymbol{x}_j^{(i)}$ is the jth site in cluster i. This is a univariate minimization problem of minimizing $\mathrm{AIC_c}(k)$. It is solved with Brent's golden-search algorithm [9].

Figures 1 and 2 give a 1D and 2D example, respectively, of the proposed method and Algorithm 4 gives a pseudocode of the proposed method.

Algorithm 4. Modifying the cache to assist exploration

```
Input: cache of sites and responses;
create a copy of cache;
find mean response in cache $\bar{f}$;
cluster cached sites using $k$-harmonic means, find optimal $k$ with $\mathrm{AIC_c}$ criterion;
for clusters containing a site found in previous local searchs set all responses to $\bar{f}$;
Output: modified copy of the cache
```

3.3 Generating the Global Model

Next, the proposed algorithm uses the modified copy of the cache to train a global model of the expensive function. Such a model can be a Lagrangian interpolant so it interpolates exactly at all available sites, that is

$$\mathcal{S}(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)\,,\ i = 1\ldots n\ (= \text{cache size})\,, \tag{10}$$

where $\mathcal{S}(\boldsymbol{x}_i)$ is the model response at the ith site and $f(\boldsymbol{x}_i)$ is the true response there. However, there are two difficulties with such models:

a) they can generalize poorly due to over-fitting to the given data (as mentioned in Sect. 2) so their prediction is likely to be poor at new sites and
b) they become computationally-expensive to handle (since they account for all sites) and numerically unstable (due to ill-conditioning of the interpolation matrix) [22, 51].

To avoid these issues the proposed algorithm uses a *radial basis function network* (RBFN) for the global model which is an artificial neural network with radial basis functions processing units. Artificial neural networks are a form of nonparametric regression [6, 40] and can approximate a continuous function with high accuracy (given sufficient sites) [6, Ch.4]. RBFNs have the advantage of approximating well complicated function landscapes while their structure is simpler compared to other networks and so they are faster to train [6, Ch.5], [10, 73].

Figure 3 shows a typical RBFN. It contains three layers: the input layer, the processing layer containing the processing units (or neurons) and the output layer which is a weighted sum of the units responses. The proposed algorithm uses an RBFN with Gaussian processing units which is equivalent to approximating the objective function by a superposition of Gaussians [69]. The response of this RBFN is

$$\mathcal{S}(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{t}_j\|_2^2}{c_j{}^2}\right), \tag{11}$$

where N is the number of processing units, λ_j is a coefficient, $\boldsymbol{t}_j$ is the centre of the jth Gaussian and c_j is the shape parameter (or hyperparameter) which determines the width of the jth Gaussian.

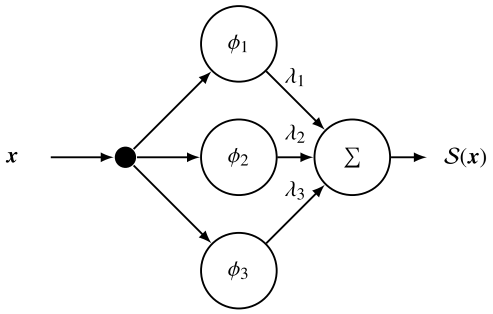


Fig. 3 An RBFN with three neurons (processing units)

An RBFN generalizes well and avoids over-fitting by abstracting the data. This is achieved by using fewer processing units than sample sites ($N < n$) so the centres ($\boldsymbol{t}_j$) typically do not coincide with the sample sites. Training an RBFN requires selecting the number of processing unit (N), the centres ($\boldsymbol{t}_j$), the shape parameters (c_j) and the coefficients (λ_j) . To efficiently select all these the proposed algorithm uses the following two steps.

First, it identifies the optimal number of processing units (N). For a candidate number of processing units the cached sites are clustered using the k-harmonic means algorithm (described in Sect. 3.2) and the resulting centres are taken as the Gaussians' centres. The shape parameters (c_j) are taken as the radii of the corresponding clusters. The coefficients λ_j are obtained from the least-squares solution of the interpolation equations as

$$\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\lambda} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{f}, \tag{12}$$

where $\boldsymbol{\Phi}$ is the interpolation matrix such that

$$\boldsymbol{\Phi} : \Phi_{i,j} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{t}_j\|_2^2}{c_j{}^2}\right), \tag{13}$$

and $\boldsymbol{f}$ is the vector of responses. This linear system is solved by the truncated singular value decomposition method (TSVD) since $\boldsymbol{\Phi}$ may be ill-condtioned [6, p.170–171].

All these parameters define an RBFN with N processing units. Similar to Sect. 3.2, finding the optimal number of processing units is treated as a model selection problem and is solved using the corrected Akaike information criterion (AIC$_\mathrm{c}$). For a network with N units the algorithm finds the empirical discrepancy (6) calculated over all cached sites. Each value of N defines a specific network and a corresponding AIC$_\mathrm{c}$. As such the algorithm finds the optimal N by minimizing AIC$_\mathrm{c}(N)$ using Brent's algorithm [9].

The proposed algorithm then optimizes the shape parameters. Taking the shape parameters found in the previous step ($\boldsymbol{c} = (c_1, \ldots c_N)$) as baseline values it finds the factor l which minimizes the discrepancy (6) of a network with shape parameters $l\boldsymbol{c}$. The number of centres (N) and their location ($\boldsymbol{t}_j$) are fixed to those found in the previous step, but the coefficients λ_j are recalculated for each candidate l value. Similarly to the previous stage, each l value defines a specific network and a corresponding discrepancy and so the algorithm finds the optimal l by minimizing $\varDelta(l)$ using Brent's algorithm. Algorithm 5 gives a pseudocode of the proposed method for generating the RBFN global model.

Algorithm 5. Generating the global model

```
Input: modified copy of cache;
optimize number of units ($N$) by minimizing AIC$_c$($N$):
begin
    for each candidate $N$ cluster sites using $k$-harmonic means;
    set Gaussian centres ($\boldsymbol{t}_j$) to cluster centres;
    set Gaussian widths ($c_j$) to cluster radii;
    find coefficients ($\lambda_j$) by least-squares;
    find discrepancy of candidate network and its AIC$_c$;
end
set $\boldsymbol{c}$ as shape parameters for optimal $N$;
optimize shape parameters by minimizing the discrepancy $\varDelta(l)$:
begin
    for each candidate $l$ set Gaussian widths to $l\boldsymbol{c}$;
    generate network (find coefficients by least-squares);
    find discrepancy of candidate network;
end
Output: an RBFN global model with optimized parameters
```

3.4 Selecting the Starting Site for the Local Search

Next, the proposed algorithm seeks the global optimum of the global model. It uses a real-coded EA [13] followed by a gradient-based finite-differences quasi-Newton BFGS algorithm [23, Ch.5–6]. The EA uses a population size $s_{\text{pop}} = 50$, linear ranking, stochastic universal sampling (SUS), intermediate recombination, elitism with a generation gap $g_{\text{gap}} = 0.9$ and the breeder-genetic-algorithm mutation operator with probability $p_{\text{m}} = 0.05$ [75]. The evolutionary search is stopped when no improvement is observed after $g_{\text{n.i.}} = 10$ generations. The gradient-search then improves the best site found by the EA and this gives the predicted optimum ($\boldsymbol{x}_{\text{p}}$) of the global model.

The predicted optimum is then evaluated with the expensive objective function to give the true response $f(\boldsymbol{x}_{\text{p}})$. If $f(\boldsymbol{x}_{\text{p}})$ is better than the best value found so far then the local search is started from $\boldsymbol{x}_{\text{p}}$. Otherwise, this indicates the model is inaccurate and so the algorithm improves the model by adding a new site to it.

The accuracy of interpolants depends on the spread of the interpolation sites, measured by the *maximin distance* [63, 100]. For a set of sites $\boldsymbol{x}_i$, $i = 1 \ldots k$ the maximin distance is the maximum of all nearest-neighbour distances in the set. The model accuracy improves as the maximin distance increases, that is as the sites are more space-filling. As such, the proposed algorithm improves the global model by seeking the site which maximizes the maximin distance for the cached sites. It finds this site ($\boldsymbol{x}_\mathrm{n}$) by solving the nonlinear optimization problem

$$\boldsymbol{x}_\mathrm{n} : \max_{\boldsymbol{x} \in \mathcal{F}} \min_{\boldsymbol{x}_i \in \mathcal{X}} \{ \, \|\boldsymbol{x}_i - \boldsymbol{x}\|_2 \, \} \tag{14}$$

where $\mathcal{X}$ is the set of cached sites and $\mathcal{F}$ is the search space. This approach generates sites similarly to the *maximin design of experiments* [47].

The new site is evaluated with the expensive function and is cached. The global model is then updated and the process repeats until either a better optimum is found or 10 attempts have been made. In the latter case the best cached site is taken as the starting site for the local search. Algorithm 6 gives a pseudocode for the proposed method for selecting the starting site.

Algorithm 6. Selecting the starting site for the local search

```
Input: cache and modified copy of cache;
set i = 1 ;                                              /* number of attempts */
find best site in cache ($x_b$);
repeat
    generate global model;
    seek optimum of model using an EA followed by a gradient search (SQP);
    evaluate the predicted optimum ($x_p$) with expensive function and cache;
    if optimum is better then current best in cache then
        set $x_0 = x_p$ ;                                    /* set starting site */
    else
        improve model by searching for a site ($x_n$) using maximin distance;
        evaluate $x_n$ with expensive function and cache;
        i = i + 1;
    if $i = 10$ then set $x_0 = x_b$ ;                        /* set starting site */
until $f(x_p) < f(x_b)$ or $i = 10$ ;
Output: initial site for local search ($x_0$)
```

3.5 Improving the Optimum with the Local Search

Next, the proposed algorithm improves the starting site ($\boldsymbol{x}_0$) by using a local search. The proposed local search has three distinct features: a) since it concentrates on a small region, it uses local models to better model the objective function b) it uses an improved trust-region framework to converge to a true optimum of the expensive function but it replaces the classical quadratic models with more flexible ones to improve the search and c) to further assist convergence it continuously adapts the type of model used.

3.5.1 Using a Trust-Region Framework to Converge to a True Optimum

As mentioned in Sect. 1.3, due to model inaccuracy a model-assisted optimization search can converge to a false optimum. To avoid this the proposed algorithm uses a trust-region framework.

The classical trust-region framework generates at each iteration a quadratic model and obtains its constrained optimum (a truncated Newton step) as a quadratic programming problem [17]. The framework guarantees asymptotic convergence to an optimum of the true objective function (a critical site satisfying the first order Karush-Kuhn-Tucker optimality conditions).

For quadratic models the constrained optimum is easily found but such models cannot model a complicated landscape well. As such, the proposed local search replaces them with the more flexible Kriging models which are described in Sect. 3.5.2 which follows. When compared to quadratics, Kriging models can approximate the objective function better over a larger trust-region and so convergence will be faster and require less function evaluations.

The proposed trust-region local search begins with an initial cuboid trust-region centred at $\boldsymbol{x}_{\mathrm{c}} = \boldsymbol{x}_0$ (the starting site) and of size $\Delta = 0.1$, that is

$$\mathcal{T} = \{\boldsymbol{x} : \|\boldsymbol{x}_{\mathrm{c}} - \boldsymbol{x}\|_{\infty} \leqslant \Delta\}\,. \tag{15}$$

To emphasize local function behaviour all cached sites which are in $\mathcal{T}$ are used to generate the local model. If the trust-region contains less than $m = \min\{d+1, 10\}$ sites then the nearest exterior sites are also used. The algorithm selects the optimal type of Kriging model (as described in Sect. 3.5.2) and finds its constrained optimum in the trust-region ($\boldsymbol{x}_{\mathrm{m}}$). However, this is no longer a simple quadratic programming problem (as in the classical framework) and so to find the constrained optimum the algorithm uses an EA followed by a gradient-search, similarly to Sect. 3.4.

Following the classical trust-region approach the objective function is evaluated at the predicted optimum and a merit value is calculated

$$\rho = \frac{f(\boldsymbol{x}_{\mathrm{m}}) - f(\boldsymbol{x}_{\mathrm{c}})}{\mathcal{S}(\boldsymbol{x}_{\mathrm{m}}) - \mathcal{S}(\boldsymbol{x}_{\mathrm{c}})}\,, \tag{16}$$

where $\mathcal{S}(\boldsymbol{x})$ is the current Kriging local model. A value of $\rho \simeq 1$ indicates a good fit of the model to the objective function in the trust-region.

The classical trust-region framework assumes exact derivatives are available so a poor model fit is only due to a trust-region which is too large and so it decreases the trust-region. However, in model-assisted search the model may be inaccurate due to an insufficient number of interpolation sites in the trust-region. This needs to be accounted for to avoid a quick reduction of the trust-region and premature convergence [15].

As discussed in Sect. 3.4, the model's accuracy depends on the maximin distance of the interpolated sites. As such, the proposed algorithm determines if a model is sufficiently accurate (to justify reducing the trust-region) based on the number of space-filling sites in the trust-region. The maximum separation distance for a cuboid trust-region is the diagonal length $\sqrt{d(2\Delta)^2}$ (d is the function dimension). A site is considered space-filling if its nearest-neighbour distance is at least 5% of the diagonal length. The model is considered accurate when the number of space-filling sites in the trust-region (s) is larger than a threshold value ($s^\star = d+1$). This value is based on the number of sites required to model the gradient by well-established methods like quasi-Newton finite-differences [15]. However, as d increases the required number of sites becomes comparable to the total number of function evaluations ($fe_{\max}$). As such, the algorithm uses $s^\star = \min\{d+1\,, 0.1fe_{\max}\}$.

Based on ρ, s and $s^\star$ the algorithm performs as follows:

- if $\rho > 0$: then the local model is accurate since a better solution has been found. Following the classical trust-region framework the algorithm centres the trust-region at the new optimum ($\boldsymbol{x}_{\mathrm{m}}$) and the trust-region is enlarged by a factor δ_+.
- if $\rho \leqslant 0$ and $s < s^\star$: the optimum predicted by the model is a false one, but the poor model accuracy is attributed to an insufficient number of space-filling sites in the trust-region. As such, the algorithm improves the local model by adding a new site $\boldsymbol{x}_{\mathrm{n}}$. Similar to Sect. 3.4, this site ($\boldsymbol{x}_{\mathrm{n}}$) is chosen to give the largest maximin distance with respect to all sites in the trust-region. $\boldsymbol{x}_{\mathrm{n}}$ is evaluated with the expensive function and is cached. If $f(\boldsymbol{x}_{\mathrm{n}}) < f(\boldsymbol{x}_{\mathrm{c}})$ than $\boldsymbol{x}_{\mathrm{n}}$ becomes the new trust-region centre.
- if $\rho \leqslant 0$ and $s \geqslant s^\star$: the local model fails to predict an improvement but its poor accuracy is attributed to the trust-region being too large (the model is considered to be accurate). Following the classical trust-region framework the algorithm decreases the trust-region by a factor δ_-.

As such the local search uses at most two expensive evaluations at each iteration, one for $\boldsymbol{x}_{\mathrm{c}}$ and possibly another for $\boldsymbol{x}_{\mathrm{n}}$. All new sites evaluated with the expensive function are cached.

Next, the local search stops if the trust-region is small enough $\Delta < \Delta_{\min}$ (we use $\Delta_{\min} = \Delta_0 \cdot \delta_-^2$) or if the limit of expensive evaluations has been reached. Otherwise, a new local search iteration begins. Algorithm 7 gives a pseudocode of the proposed trust-region local search.

3.5.2 Selecting Optimal Local Models

To assist the local search the proposed algorithm selects at each iteration an optimal local model. It selects models from a family of Kriging (or spatial-correlation) models as they have performed well compared to other models [45, 58].

Kriging models originated in geostatistics with the work of Krige and Matheron [19, 66]. Such a model has two components: a 'drift' function which models global variations in the objective function and a stochastic function (a stationary Gaussian process) which locally improves the prediction [20].

A common approach is to use a constant drift function (for example set to 1) [53] so the Kriging model is

$$\mathcal{S}(\boldsymbol{x}) = \beta + Z(\boldsymbol{x})\,, \tag{17}$$

Algorithm 7. A trust-region framework for the local search

```
inputs
    $\mathcal{X}$;                                  /* cache of sites and responses */
    $\boldsymbol{x}_0$;                             /* initial site for local search */

$s^\star = \min\{d+1, 0.1fe_{\max}\}$ ;             /* model accuracy threshold */
$\boldsymbol{x}_c = \boldsymbol{x}_0$ ;             /* centre trust-region at $\boldsymbol{x}_0$ */
repeat
    define a cuboid trust-region centred at $\boldsymbol{x}_c$ and of size $\Delta$;
    find the cached sites inside the trust-region (if insufficient use exterior sites);
    select an optimal local model based on these sites;
    find model optimum in trust-region ($\boldsymbol{x}_m$) with EA and gradient-search;
    calculate a trust-region merit value $\rho$;
    update trust-region:
        if $\rho > 0$                      set $\boldsymbol{x}_c = \boldsymbol{x}_m$, increase $\Delta$
        if $\rho \leqslant 0 \bigcap s < s^\star$   improve local model by adding a site $\boldsymbol{x}_n$ , if better set $\boldsymbol{x}_c = \boldsymbol{x}_n$
        if $\rho \leqslant 0 \bigcap s \geqslant s^\star$   decrease $\Delta$
until $\Delta < \Delta_{\min}$ or $fe \geqslant fe_{\max}$ ;
Output: optimum found in local search
```

where β is the drift function coefficient and $Z(\boldsymbol{x})$ is the stochastic function [98]. The latter is taken as a Gaussian process with a zero mean and variance σ . The response of the Kriging model at any site is correlated with that of other sites. The correlation between two sites ($\boldsymbol{x}_1$ and $\boldsymbol{x}_2$) is defined by a covariance function

$$C(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma^2 R(\boldsymbol{x}_1, \boldsymbol{x}_2), \tag{18}$$

where $R(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is a spatial correlation function (SCF). The model is defined by adjusting the free coefficient (β) and the SCF parameters to fit the available data.

Different spatial correlation functions have been studied [53]. Each SCF results in a different model and so the optimal SCF is problem dependant. In practice the SCF is prescribed and fixed throughout the optimization search [71]. To improve this, the proposed algorithm uses the model selection framework (described in Sect. 2) to select the optimal SCF based on maximum likelihood. The likelihood of a Kriging model is given by the closed-form expression [98]

$$\mathcal{L} = -\frac{d}{2}\log(2\pi\sigma^2) - \frac{1}{2}\log(|\boldsymbol{R}|) - \frac{1}{2\sigma^2}(\boldsymbol{f} - \boldsymbol{1}\beta)^{\mathrm{T}}\boldsymbol{R}^{-1}(\boldsymbol{f} - \boldsymbol{1}\beta), \tag{19}$$

where

$$\boldsymbol{R} : R_{i,j} = R(\boldsymbol{x}_i, \boldsymbol{x}_j, \theta) \tag{20}$$

is the correlation matrix of all sites in the sample, θ is a correlation parameter and the spatial correlation function is given by

Table 1 Candidate spatial correlation functions (SCFs)

Name	$\mathcal{R}(\theta, l_k)$
Exponential	$\exp(-\theta\lvert l_k\rvert)$
Gaussian	$\exp(-\theta l_k^2)$
linear	$\max\{0, 1-\theta\lvert l_k\rvert\}$
spherical	$1-1.5\xi_k+0.5\xi_k^3, \quad \xi_k=\min\{1, \theta\lvert l_k\rvert\}$
spline	$\zeta(\xi_k)=\begin{cases}1-15\xi_k^2+30\xi_k^3 & 0 \leqslant \xi_k \leqslant 0.2\\ 1.25(1-\xi_k)^3 & 0.2<\xi_k<1\\ 0 & \xi_k \geqslant 1\end{cases}$ $\xi_k=\theta\lvert l_k\rvert$

$l_k = \boldsymbol{x}_{i,k} - \boldsymbol{x}_{j,k}$ is the difference between the kth component of two sites $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ [109].

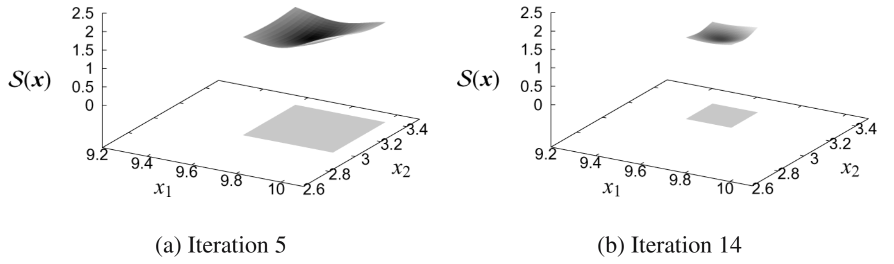


Fig. 4 An example of models used during the local search with the Branin function. (a) shows iteration 2 where the local model used a Gaussian SCF. (b) shows iteration 14 where the local model used a spline SCF. The trust-region is also shown.

Algorithm 8. Selecting optimal local models

```
Input: sites and responses used for the local model
for SCF = exponential, Gaussian, linear, spherical, spline do
    generate Kriging model using candidate SCF;
    find the model's maximum likelihood;
select the Kriging model having the largest maximum likelihood;
Output: optimal Kriging local model
```

$$R(\boldsymbol{x}_i, \boldsymbol{x}_j, \theta) = \prod_{k=1}^{d} \mathcal{R}(\theta, l_k), \quad l_k = \boldsymbol{x}_{i,k} - \boldsymbol{x}_{j,k}, \tag{21}$$

where the functions $\mathcal{R}(\theta, l_k)$ are defined in Table 1. The model parameters (β, σ and θ) are found by maximizing its likelihood (19) [65]. The numerical procedures for generating the Kriging models are given in [109].

As such, at each local search iteration the proposed algorithm builds a Kriging model, one for each of the SCFs in Table 1, and it selects the one giving the largest maximum likelihood. Complexity control is unnecessary since all models have equal complexity, that is the three parameters σ, β and θ. This approach results in a model-adaptive local search. Figure 4 shows an example and Algorithm 8 gives a pseudocode of the proposed method.

3.6 Caching New Sites

During the optimization search new sites and responses are cached for later use. If cached sites are nearly collocated the interpolation matrices used to generate the global and local models will be severely ill-condtioned. To avoid this a new site is added to the cache if it is sufficiently spaced from cached sites (a minimum l_2 distance of $\Delta_{\min}/2$, where $\Delta_{\min}$ is the prescribed minimum trust-region radius, as described in Sect. 3.5.1). Otherwise the new site replaces the cached site nearest to it (in the l_2 norm) if the new response is better than the cached one. Algorithm 9 gives the pseudocode for caching new sites.

Algorithm 9. Caching new sites

```
Input: $\boldsymbol{x}_{\text{new}}, f(\boldsymbol{x}_{\text{new}})$ ;                    /* new site and response */
find the cached site $\boldsymbol{x}_{\text{cac}}$ nearest to $\boldsymbol{x}_{\text{new}}$;
if $\|\boldsymbol{x}_{\text{cac}} - \boldsymbol{x}_{\text{new}}\|_2 \geqslant \Delta_{\min}/2$ then
    add $\boldsymbol{x}_{\text{new}}$ and $f(\boldsymbol{x}_{\text{new}})$ to the cache;
else if $f(\boldsymbol{x}_{\text{new}}) < f(\boldsymbol{x}_{\text{cac}})$ then
    replace $\boldsymbol{x}_{\text{cac}}$ and $f(\boldsymbol{x}_{\text{cac}})$ with $\boldsymbol{x}_{\text{new}}$ and $f(\boldsymbol{x}_{\text{new}})$;
```

4 Performance Analysis

This section gives a detailed performance analysis of the proposed algorithm in three parts. First, we test the proposed algorithm on eight well-known mathematical test functions. In these tests it is also benchmarked against four reference algorithms.

Next, we study the individual contribution of the global search and of the local search components of the proposed algorithm. This is done by comparing the full proposed algorithm to the two cases where its global search is disabled and where its local search is disabled.

Lastly, we apply the proposed algorithm to a real-world application of airfoil shape optimization and we also benchmark it against the four reference algorithms.

4.1 Reference Algorithms and Test Procedure

To obtain a reference of performance we benchmarked the proposed algorithm against four representative model-assisted EAs [91, 92]. These algorithms build a

Algorithm 10. Reference algorithm

```
generate initial sites with LHD and evaluate them;
cache sites and responses;
while $fe \leqslant fe_{\max}$ do
    generate a Kriging model based on cache;
    search for model optimum with an EA for 10 generations;
    evaluate candidate solutions from population with true function;
    cache evaluated candidate solutions and their responses;
Output: best solution and response found
```

Kriging model, seek its optimum and evaluate a certain percentage of the population with the true function. The model is then updated and the process repeats until the limit of function evaluations is reached. Algorithm 10 gives a pseudocode of the reference algorithms.

The four algorithms differ by their Kriging spatial correlation function (SCF) and the percentage of elites and non-elites that they evaluate at each iteration. Table 2 compares the reference algorithms.

Table 3 gives the parameter settings used by the proposed algorithm during the tests. Parameters which define the EA operation were identical in the proposed algorithm and in the four reference algorithms.

To obtain statistically-significant results 30 runs were repeated for each test with the proposed algorithm and the reference algorithms. For each function and each algorithm we provide the statistics mean, standard deviation, median, best and worst result. Also, to determine in a rigorous manner which algorithm performs better we used the Mann–Whitney (or Wilcoxon) significance test [64], which is a nonparametric version of the *t*-test [18, Ch.5], [104, p.513–576]. The Mann–Whitney test is preferable since it is more widely applicable: it is valid for non-normal data while applying the *t*-test on non-normal data can give incorrect inferences [18, Ch.2].

We used the one-tailed Mann–Whitney test which provides a test statistic U. The null and alternative hypothesis are:

$$H_0 : P(r_i \leqslant r_p) \geqslant 0.5 \tag{22a}$$

$$H_1 : P(r_i > r_p) < 0.5\,, \tag{22b}$$

where $P(r_i < r_p)$ is the probability that a result of the proposed algorithm is larger (worse) than a result of the *i*th reference algorithm ($i = 1 \ldots 4$). As such we test if the proposed algorithm is more likely to give a better result than the reference algorithms. The null hypothesis is rejected at the $\alpha = 0.05$ significance level if $U > 1.644$ and at the $\alpha = 0.01$ significance level if $U > 2.326$. For each test function we applied the Mann–Whitney test between results of the proposed algorithm and each of the four reference algorithms and provide the resultant U statistics. As such, for each reference algorithm if $U > 1.644$ or $U > 2.326$ we reject the null hypothesis at the 0.05 and 0.01 significance level, respectively, and accept the proposed algorithm outperformed the reference algorithm.

Table 2 Settings for the reference algorithms

			Evaluated per generation	
Number	Designation	SCF [1]	% elites	% non-elites
1	(G,10,0)	Gaussian	10	0
2	(S,10,0)	spline	10	0
3	(G,5,5)	Gaussian	5	5
4	(S,5,5)	spline	5	5

[1] spatial correlation function.

Table 3 Parameter settings for the proposed algorithm

	General Parameters	
$fe_{\max}$	max. (true) objective function evaluations	100 , 150 , 200[1]
	EA Search	
s_{pop}	population size	50
g_{gap}	generation gap	0.9
$g_{\max}$	maximum generations	20
p_{m}	mutation probability	0.05
$g_{\text{n.i.}}$	no-improvement generations to stop	10
	Trust-region Search	
Δ_0	initial trust-region radius	0.1
δ_+	trust-region size increase factor	2
δ_-	trust-region size decrease factor	0.5
$\Delta_{\min}$	minimum trust-region radius	$\Delta_0 \cdot \delta_-^2$
$\Delta_{\max}$	maximum trust-region radius	$\Delta_0 \cdot \delta_+^2$

[1] 100 and 200 for test functions, 150 for airfoil optimization.

4.2 Mathematical Test Functions

In this section we used eight mathematical test functions which are widely used and well-known: Branin, Hartman 3 and Hartman 6 [24], Greiwank [33], Rastrigin [117, p.185–192], Rosenbrock [97], Schwefel 2.13 [101, p.301–302] and Weierstrass [36]. For the Greiwank, Rastrigin, Rosenbrock, Schwefel and Weierstrass we used the function definitions from [111] along with the supporting files available online. The Branin, Hartman 3 and Hartman 6 have a fixed dimension (2, 3 and 6, respectively) while those from [111] were tested in dimension 10 and 30 to evaluate the 'curse of dimensionality' [5] on the algorithms performance. All functions were tested with a limit on function evaluations ($fe_{\max}$ = 100 for Branin, Hartman 3 and Hartman 6 and $fe_{\max}$ = 200 for all other functions). These are realistic settings for expensive problems and they test the algorithms under a constraint of resources, as

Table 4 Mathematical test functions

Function	Dimension (d)	Definition	Search space	$f(\boldsymbol{x}_{\mathbf{g}})$[1]
Reference: Dixon and Szegö [24]				
Branin	2	$(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$	$[-5,10] \times [0,15]$	0
Hartman 3	3	$\sum_{i=1}^{4} c_i \exp[\sum_{j=1}^{4} a_{i,j}(x_i - p_{i,j})^2]$	$[0,1]^d$	−3.86
Hartman 6	6	$\sum_{i=1}^{4} c_i \exp[\sum_{j=1}^{6} a_{i,j}(x_i - p_{i,j})^2]$	$[0,1]^d$	−3.32
Reference: Suganthan et. al. [111]				
Griewank	10, 30	$\sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600,600]^d$	0
Rastrigin	10, 30	$\sum_{i=1}^{d} \left\{x_i^2 - 10 \cdot \cos(2\pi x_i) + 10\right\}$	$[-5,5]^d$	0
Rosenbrock	10, 30	$\sum_{i=1}^{d} \left\{(2x_{i-1} - x_i^2)^2 + (1 - x_i)^2\right\}$	$[-2,2]^d$	0
Schwefel 2.13	10, 30	$\sum_{i=1}^{d} \left(\sum_{j=1}^{d} a_{i,j}\sin(\alpha_j) + b_{i,j}\cos(\alpha_j) - \sum_{j=1}^{d} a_{i,j}\sin(x_j) + b_{i,j}\cos(x_j)\right)^2$	$[-\pi,\pi]^d$	0
Weierstrass	10, 30	$\sum_{i=1}^{d} \sum_{k=0}^{20} a^k \cos(2\pi b^k(x_i + 0.5))$	$[-0.5,0.5]^d$	$-d$

[1] Value at global optimum.

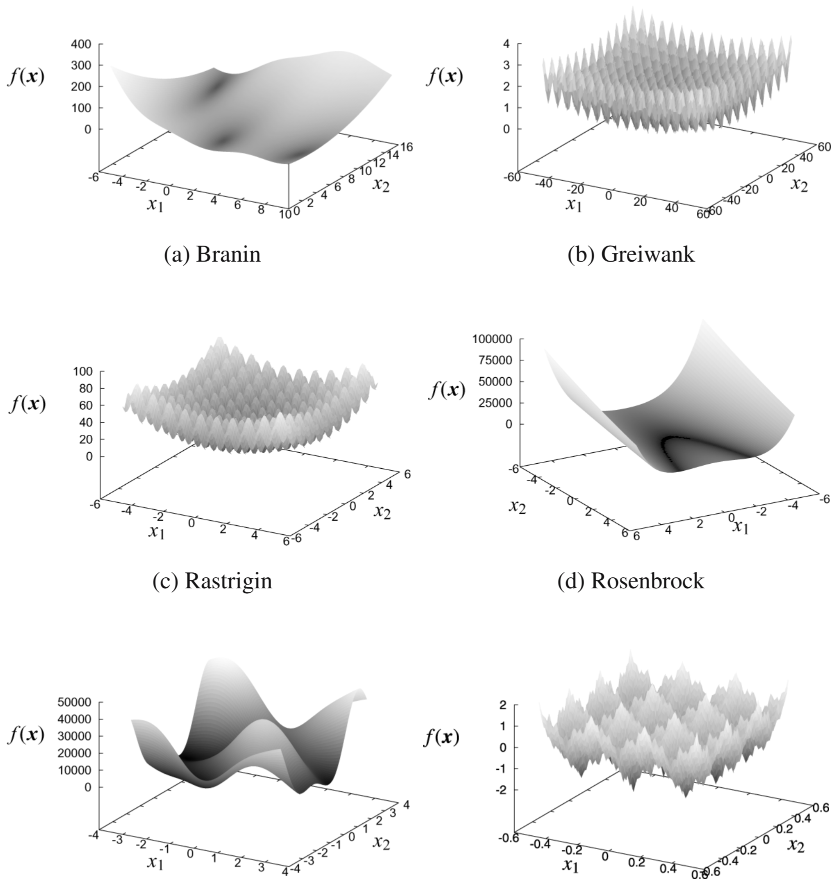
suggested in [118]. Table 4 gives the test functions' details and Fig. 5 shows their bivariate version (excluding Hartman 3 and Hartman 6 which are not bivariate).

Tables 5–7 provide the resultant test statistics for the comparisons with the four reference algorithms over the eight test functions. Results for the Branin, Hartman 3 and Hartman 6 for all algorithms are similar since they all obtained a good approximation of the global optimum. This indicates that these functions were not challenging to all five algorithms and there is no clear winner.

A significant difference in performance between the proposed algorithm and the reference algorithms is seen with the more complicated functions (Greiwank, Rastrigin, Rosenbrock, Schwefel and Weierstrass). The mean and median statistics indicate that the proposed algorithm found a better solution than the reference algorithms. This is attributed to the combined global and local search and the careful selection of models in these searches. Also, the standard deviation of results for the proposed algorithm is typically lower than that of the reference algorithms which indicates its performance is more stable. Overall, based on the Mann–Whitney test in all cases we reject the null hypothesis in (22) at both significance levels $\alpha = 0.05$ and 0.01 and accept that the proposed algorithm outperformed the four reference algorithms.

4.3 Individual Component Contribution

We also study the individual contribution of the global search and the local search to the overall performance of the proposed algorithm. For this, we used two reference algorithms obtained from the complete proposed algorithm. One algorithm



(a) Branin (b) Greiwank

(c) Rastrigin (d) Rosenbrock

(e) Schwefel 2.13 (f) Weierstrass

Fig. 5 Bivariate versions of the mathematical test functions

uses only the proposed global search (local search is disabled) and the other uses only the proposed local search (global search is disabled). The two reference algorithms were tested with five test functions: Greiwank and Rastrigin in dimension 10 and Rosenbrock, Schwefel and Weierstrass in dimension 30. A similar analysis to that of the Sect. 4.2 was used, that is we provide the statistics mean, standard deviation, median, best and worst and the Mann–Whitney U statistic.

Table 8 shows test results over the five test functions. First, for the highly multimodal functions (Greiwank, Rastrigin, Weierstrass) the global search reference algorithm performed better than the local search one. In such complicated landscapes an extensive global search finds a better optimum while a local search converges to an inferior optimum typically close to the starting site. An opposite trend exists for the simpler Rosenbrock and Schwefel functions where an extensive local search

Table 5 Results for mathematical tests functions

			Reference algorithms			
Function	Statistic [1,2]	Proposed	(G,10,0)	(S,10,0)	(G,5,5)	(S,5,5)
Branin	Mean	$3.979e-01$	$4.091e-01$	$4.767e-01$	$4.088e-01$	$4.566e-01$
	S.D.	$1.881e-06$	$5.158e-03$	$2.441e-01$	$1.754e-03$	$1.941e-01$
	Median	$3.979e-01$	$4.079e-01$	$4.015e-01$	$4.079e-01$	$3.982e-01$
	Best	$3.979e-01$	$4.079e-01$	$3.979e-01$	$4.079e-01$	$3.979e-01$
	Worst	$3.979e-01$	$4.362e-01$	$1.527e+00$	$4.139e-01$	$1.409e+00$
	U		$5.941e+00$	$5.426e+00$	$5.941e+00$	$4.376e+00$
Hartman 3	Mean	$-3.862e+00$	$-3.860e+00$	$-3.716e+00$	$-3.759e+00$	$-3.818e+00$
	S.D.	$3.510e-04$	$2.112e-03$	$2.242e-01$	$5.526e-03$	$5.532e-02$
	Median	$-3.863e+00$	$-3.861e+00$	$-3.806e+00$	$-3.763e+00$	$-3.841e+00$
	Best	$-3.863e+00$	$-3.862e+00$	$-3.863e+00$	$-3.763e+00$	$-3.863e+00$
	Worst	$-3.862e+00$	$-3.857e+00$	$-3.087e+00$	$-3.751e+00$	$-3.628e+00$
	U		$2.928e+00$	$4.082e+00$	$2.928e+00$	$3.581e+00$
Hartman 6	Mean	$-3.322e+00$	$-3.307e+00$	$-3.284e+00$	$-3.281e+00$	$-3.288e+00$
	S.D.	$5.245e-04$	$4.535e-02$	$5.982e-02$	$6.484e-02$	$7.002e-02$
	Median	$-3.322e+00$	$-3.321e+00$	$-3.321e+00$	$-3.321e+00$	$-3.321e+00$
	Best	$-3.322e+00$	$-3.321e+00$	$-3.321e+00$	$-3.321e+00$	$-3.321e+00$
	Worst	$-3.321e+00$	$-3.178e+00$	$-3.192e+00$	$-3.165e+00$	$-3.127e+00$
	U		$3.254e+00$	$2.712e+00$	$2.495e+00$	$2.712e+00$

[1] S.D. :standard deviation
[2] Reject H_0 at $\alpha = 0.05$ if $U \geqslant 1.644$.
Reject H_0 at $\alpha = 0.01$ if $U \geqslant 2.326$.

finds a better optimum. Second, the Mann–Whitney statistic indicates the full proposed algorithm outperformed the reference algorithms, which shows the benefit of the proposed global–local approach. Lastly, the standard deviation of results of the reference algorithms was typically much larger than for the proposed algorithm which indicates their performance is much less stable. Overall, results show that both the proposed global search and the proposed local search contribute to the optimization search. However, individually they perform well on some functions but worse on others. The proposed algorithm combines both approaches and so it achieves an effective and efficient search over a wide range of functions.

4.4 Real-World Application

As a final test we have also applied the proposed algorithm to the real-world application of airfoil shape optimization. Here we are given an aircraft's flight conditions (speed and altitude) and the goal is to find an airfoil shape which generates the required lift force (L) with a minimum of aerodynamic friction (or drag) force (D). In practice these requirements are not expressed as forces but as aerodynamic coefficients:

$$c_{\mathrm{L}} = \frac{L}{\frac{1}{2}\rho U^2} \quad \text{(lift coefficient)} \tag{23a}$$

$$c_{\mathrm{D}} = \frac{D}{\frac{1}{2}\rho U^2} \quad \text{(lift coefficient)} \tag{23b}$$

where ρ is the air density at the prescribed flight altitude and U is the prescribed flight speed. Figure 6 shows an example.

Table 6 Results for mathematical tests functions–10D

			Reference algorithms			
Function	Statistic [1,2]	Proposed	(G,10,0)	(S,10,0)	(G,5,5)	(S,5,5)
Greiwank	Mean	$1.001e+00$	$1.663e+00$	$1.125e+00$	$1.319e+00$	$1.137e+00$
	S.D.	$1.511e-01$	$9.735e-01$	$1.659e-01$	$4.055e-01$	$1.735e-01$
	Median	$9.955e-01$	$1.247e+00$	$1.101e+00$	$1.253e+00$	$1.113e+00$
	Best	$7.996e-01$	$9.890e-01$	$8.281e-01$	$8.418e-01$	$7.318e-01$
	Worst	$1.373e+00$	$4.466e+00$	$1.811e+00$	$2.873e+00$	$1.567e+00$
	U		$3.654e+00$	$2.967e+00$	$2.936e+00$	$2.905e+00$
Rastrigin	Mean	$4.089e+00$	$7.546e+01$	$2.703e+01$	$5.730e+01$	$2.243e+01$
	S.D.	$3.145e+00$	$2.249e+01$	$1.317e+01$	$2.634e+01$	$1.368e+01$
	Median	$3.980e+00$	$7.950e+01$	$2.686e+01$	$5.933e+01$	$1.940e+01$
	Best	$1.488e-06$	$1.393e+01$	$1.991e+00$	$1.375e+01$	$3.980e+00$
	Worst	$1.393e+01$	$1.238e+02$	$5.771e+01$	$1.120e+02$	$5.970e+01$
	U		$6.653e+00$	$6.209e+00$	$6.623e+00$	$6.165e+00$
Rosenbrock	Mean	$4.292e-01$	$9.697e-01$	$2.902e+00$	$2.078e+00$	$3.531e+00$
	S.D.	$7.117e-01$	$8.911e-01$	$1.284e+00$	$1.529e+00$	$1.546e+00$
	Median	$2.064e-01$	$7.126e-01$	$2.965e+00$	$1.707e+00$	$3.433e+00$
	Best	$3.928e-04$	$4.606e-03$	$6.685e-01$	$4.903e-02$	$2.855e-01$
	Worst	$3.734e+00$	$2.841e+00$	$6.009e+00$	$5.515e+00$	$6.729e+00$
	U		$2.587e+00$	$6.195e+00$	$4.923e+00$	$6.150e+00$
Schwefel 2.13	Mean	$1.082e+04$	$5.461e+04$	$2.776e+04$	$4.682e+04$	$4.102e+04$
	S.D.	$1.461e+04$	$7.234e+04$	$2.465e+04$	$4.876e+04$	$4.042e+04$
	Median	$5.082e+03$	$2.723e+04$	$1.722e+04$	$2.191e+04$	$2.977e+04$
	Best	$2.266e+01$	$4.433e+02$	$3.050e+03$	$2.723e+02$	$1.492e+03$
	Worst	$4.116e+04$	$3.470e+05$	$1.098e+05$	$1.674e+05$	$1.713e+05$
	U		$2.467e+00$	$2.567e+00$	$2.467e+00$	$2.867e+00$
Weierstrass	Mean	$-7.366e+00$	$-3.229e+00$	$-3.752e+00$	$-3.404e+00$	$-4.384e+00$
	S.D.	$1.011e+00$	$5.841e-01$	$7.929e-01$	$6.456e-01$	$1.073e+00$
	Median	$-7.408e+00$	$-3.144e+00$	$-3.821e+00$	$-3.422e+00$	$-4.450e+00$
	Best	$-9.119e+00$	$-4.835e+00$	$-5.208e+00$	$-4.498e+00$	$-7.164e+00$
	Worst	$-5.353e+00$	$-2.111e+00$	$-2.403e+00$	$-2.105e+00$	$-2.113e+00$
	U		$6.653e+00$	$6.653e+00$	$6.653e+00$	$6.357e+00$

[1] S.D. :standard deviation

[2] Reject H_0 at $\alpha = 0.05$ if $U \geqslant 1.644$.
Reject H_0 at $\alpha = 0.01$ if $U \geqslant 2.326$.

Table 7 Results for mathematical tests functions–30D

			Reference algorithms			
Function	Statistic [1,2]	Proposed	(G,10,0)	(S,10,0)	(G,5,5)	(S,5,5)
Greiwank	Mean	$1.003e+00$	$1.098e+00$	$1.060e+00$	$1.999e+00$	$1.063e+00$
	S.D.	$3.421e-02$	$3.175e-02$	$3.597e-02$	$6.687e-02$	$3.833e-02$
	Median	$1.022e+00$	$1.104e+00$	$1.056e+00$	$2.014e+00$	$1.059e+00$
	Best	$9.384e-01$	$9.784e-01$	$9.969e-01$	$1.748e+00$	$9.846e-01$
	Worst	$1.034e+00$	$1.139e+00$	$1.146e+00$	$2.077e+00$	$1.201e+00$
	U		$3.800e+00$	$3.257e+00$	$4.072e+00$	$3.568e+00$
Rastrigin	Mean	$7.734e+00$	$7.776e+01$	$7.483e+01$	$8.859e+01$	$7.369e+01$
	S.D.	$1.112e+01$	$2.514e+01$	$2.669e+01$	$2.739e+01$	$2.159e+01$
	Median	$2.145e+00$	$7.190e+01$	$6.730e+01$	$8.463e+01$	$7.408e+01$
	Best	$9.834e-03$	$4.659e+01$	$3.738e+01$	$5.143e+01$	$2.313e+01$
	Worst	$3.139e+01$	$1.361e+02$	$1.379e+02$	$1.570e+02$	$1.435e+02$
	U		$4.685e+00$	$4.685e+00$	$4.685e+00$	$4.654e+00$
Rosenbrock	Mean	$1.402e+01$	$2.106e+01$	$2.041e+01$	$2.095e+01$	$2.070e+01$
	S.D.	$1.761e+00$	$2.823e+00$	$2.722e+00$	$3.039e+00$	$2.486e+00$
	Median	$1.391e+01$	$2.085e+01$	$2.050e+01$	$2.116e+01$	$2.068e+01$
	Best	$9.863e+00$	$1.505e+01$	$1.418e+01$	$1.389e+01$	$1.751e+01$
	Worst	$1.907e+01$	$2.651e+01$	$2.657e+01$	$3.010e+01$	$2.537e+01$
	U		$6.368e+00$	$6.141e+00$	$6.277e+00$	$6.429e+00$
Schwefel 2.13	Mean	$1.933e+05$	$2.994e+05$	$3.086e+05$	$3.176e+05$	$3.157e+05$
	S.D.	$7.435e+04$	$1.168e+05$	$1.245e+05$	$1.173e+05$	$9.693e+04$
	Median	$1.875e+05$	$2.906e+05$	$2.886e+05$	$2.800e+05$	$3.307e+05$
	Best	$9.325e+04$	$1.079e+05$	$1.021e+05$	$1.540e+05$	$7.321e+04$
	Worst	$2.902e+05$	$6.348e+05$	$5.886e+05$	$5.843e+05$	$4.981e+05$
	U		$2.655e+00$	$2.780e+00$	$2.905e+00$	$3.342e+00$
Weierstrass	Mean	$-2.078e+01$	$-1.512e+01$	$-1.256e+01$	$-1.440e+01$	$-1.573e+01$
	S.D.	$8.117e-01$	$4.363e+00$	$3.090e+00$	$3.532e+00$	$3.591e+00$
	Median	$-2.092e+01$	$-1.434e+01$	$-1.171e+01$	$-1.545e+01$	$-1.490e+01$
	Best	$-2.189e+01$	$-2.577e+01$	$-2.078e+01$	$-2.143e+01$	$-2.372e+01$
	Worst	$-1.973e+01$	$-7.481e+00$	$-8.318e+00$	$-8.356e+00$	$-9.391e+00$
	U		$3.223e+00$	$4.189e+00$	$3.867e+00$	$3.258e+00$

[1] S.D. :standard deviation
[2] Reject H_0 at $\alpha = 0.05$ if $U \geqslant 1.644$.
Reject H_0 at $\alpha = 0.01$ if $U \geqslant 2.326$.

The specific problem we study is that of optimizing the airfoil of a transport aircraft cruising at 35,000 ft and at a Mach number M = 0.8 (that is 80% of the speed of sound at this altitude) with an angle of attack $\alpha = 2^\circ$. The target lift coefficient is $c_L^\star = 1$. Also, the airfoil thickness must be equal to or larger than a minimum value ($t^\star = 0.095$, normalized by the airfoil chord) to ensure the airfoil does not break during flight. The cruise conditions and thickness constraint are based on [29, p.484–487], [85]. The airfoil optimization problem is then

$$
\begin{aligned}
\min\ & c_{\mathrm{D}} \text{ (drag coefficient)} \\
\text{s.t. } & c_{\mathrm{L}}^{\star} = 1 \text{ (required lift coefficient)} \\
& t^{\star} = 0.095 \text{ (min. allowed thickness at 0.2–0.8 of chord)} \\
& \alpha = 2^{\circ} \text{ (cruise angle of attack)} \\
& \mathrm{M} = 0.8 \text{ (cruise Mach number)} \\
& \mathrm{h} = 35{,}000\,\text{ft (cruise altitude)}
\end{aligned} \tag{24}
$$

Table 8 Results for mathematical tests functions–Individual component contribution

			Reference algorithms	
Function	Statistic [1,2]	Proposed	Global Search	Local Search
Greiwank–10D	Mean	$1.001e+00$	$1.765e+05$	$1.771e+08$
	S.D.	$1.511e-01$	$8.430e+04$	$3.753e+07$
	Median	$9.955e-01$	$1.572e+05$	$1.717e+08$
	Best	$7.996e-01$	$7.172e+04$	$1.262e+08$
	Worst	$1.373e+00$	$3.491e+05$	$2.398e+08$
	U		$3.780e+00$	$3.780e+00$
Rastrigin–10D	Mean	$4.089e+00$	$6.543e+00$	$5.199e+01$
	S.D.	$3.145e+00$	$2.567e+00$	$1.290e+01$
	Median	$3.980e+00$	$5.757e+00$	$5.077e+01$
	Best	$1.488e-06$	$3.642e+00$	$2.723e+01$
	Worst	$1.393e+01$	$1.083e+01$	$6.866e+01$
	U		$2.218e+00$	$4.685e+00$
Rosenbrock–30D	Mean	$1.402e+01$	$2.674e+01$	$1.993e+01$
	S.D.	$1.761e+00$	$2.013e+00$	$4.236e+00$
	Median	$1.391e+01$	$2.681e+01$	$2.031e+01$
	Best	$9.863e+00$	$2.269e+01$	$1.339e+01$
	Worst	$1.907e+01$	$2.931e+01$	$2.791e+01$
	U		$4.664e+00$	$3.924e+00$
Schwefel 2.13–30D	Mean	$1.933e+05$	$1.294e+06$	$3.453e+05$
	S.D.	$7.435e+04$	$2.314e+05$	$1.756e+05$
	Median	$1.875e+05$	$1.303e+06$	$3.131e+05$
	Best	$9.325e+04$	$7.783e+05$	$1.415e+05$
	Worst	$2.902e+05$	$1.569e+06$	$7.355e+05$
	U		$3.780e+00$	$2.419e+00$
Weierstrass–30D	Mean	$-2.078e+01$	$-1.857e+01$	$-1.459e+01$
	S.D.	$8.117e-01$	$1.421e+00$	$2.563e+00$
	Median	$-2.092e+01$	$-1.808e+01$	$-1.440e+01$
	Best	$-2.189e+01$	$-2.153e+01$	$-1.910e+01$
	Worst	$-1.973e+01$	$-1.721e+01$	$-1.024e+01$
	U		$2.843e+00$	$3.554e+00$

[1] S.D. :standard deviation

[2] Reject H_0 at $\alpha = 0.05$ if $U \geqslant 1.644$.
Reject H_0 at $\alpha = 0.01$ if $U \geqslant 2.326$.

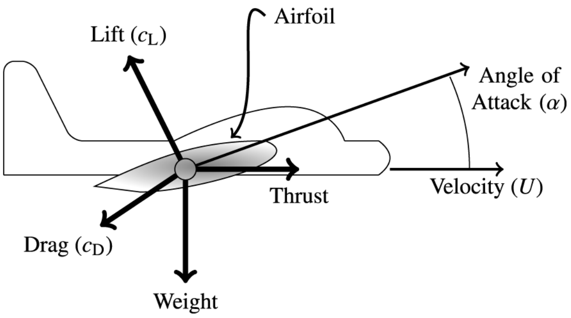
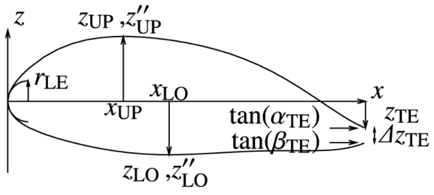


Fig. 6 The forces operating on an aircraft at flight. The angle-of-attack (α) is the measured approximately between the velocity and the airfoil chord.

Fig. 7 PARSEC design variables



Accordingly, we used the objective function

$$f = \frac{|c_L - c_L^\star|}{\max\{c_{L,\max} - c_L^\star, c_L^\star - c_{L,\min}\}} + \frac{c_D}{c_{D,\max}} + \frac{\max\{t^\star - t, 0\}}{t^\star} \tag{25}$$

where $c_{L,\max} = 1.5$, $c_{L,\min} = -0.5$ are the assumed extremal values for the lift coefficient and $c_{D,\max} = 0.2$ is an assumed maximal drag coefficient. For $c_{L,\min}$, $c_{L,\max}$ and $c_{D,\max}$ only rough estimates are needed since they only normalize the objectives.

To generate a candidate airfoil we used the PARSEC parameterization which uses 11 design variables [108]. Figure 7 shows an example of this. We set the bounds of these design variables based on previous studies [41, 86] and Table 9 gives their values. To ensure a closed airfoil shape we set the leading edge gap as $\Delta z_{TE} = 0$. Also, to avoid unrealistic shapes where the lower airfoil curve intersects the upper curve we set the trailing edge angles to satisfy $\beta_{TE} \geqslant \alpha_{TE}$ (effectively $\beta_{TE} = \alpha_{TE} + \theta$ where $\theta \geqslant 0$).

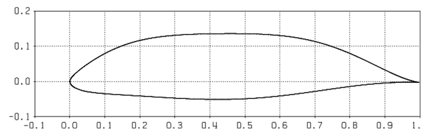
To obtain the lift coefficient and drag coefficient of candidate airfoils the optimization algorithm used XFoil, an analysis code for subsonic isolated airfoils based on the panel method [26]. Each airfoil evaluation required approximately 30 seconds on a desktop computer. We set the limit of function evaluations to $fe_{\max} = 150$.

Figure 8 shows an airfoil found by the proposed algorithm and the variation of the pressure coefficient (c_P) along the upper and lower airfoil curves. The airfoil yields a lift coefficient of $c_L = 1.019$ and a drag coefficient $c_D = 0.023$ and satisfies

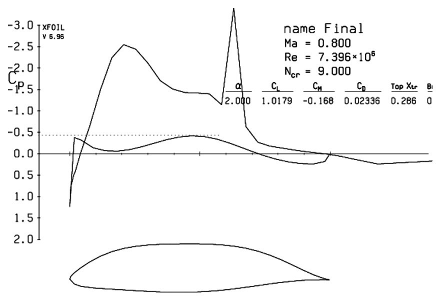
Table 9 PARSEC design variables and their bounds

Variable	Meaning	min.	max.
r_{LE}	leading-edge radius	0.002	0.030
x_{up}	max. upper thickness location	0.2	0.7
z_{up}	max. upper thickness	0.08	0.18
z''_{up}	max. upper curvature	−0.6	0.0
x_{lo}	max. lower thickness location	0.2	0.6
z_{lo}	max. upper thickness	−0.09	0.02
z''_{lo}	max. lower curvature	0.2	0.9
z_{TE}	trailing edge height	−0.01	0.01
Δz_{TE}	trailing edge thickness	0	0
α_{TE}[1]	upper trailing edge angle°	165	180
β_{TE}[1,2]	lower trailing edge angle°	165	190

[1] measured anti-clockwise from the x-axis.
[2] $\beta_{\text{TE}} \geqslant \alpha_{\text{TE}}$ to avoid intersecting curves.



(a) Airfoil geometry



(b) Pressure coefficient change

Fig. 8 An airfoil obtained by the proposed algorithm. (a) shows the airfoil geometry. (b) shows the change of pressure coefficient along the upper and lower airfoil curves and the airfoil is shown below for reference

the minimum thickness requirement (minimum thickness at 0.2–0.8 of chord is t = 0.097). Figure 8(b) shows the pressure coefficient change along the upper and lower airfoil curves. A pressure jump on the upper curve around 0.7 of the chord indicates a shockwave, which is expected due to the high subsonic cruise speed (M = 0.8).

We have also benchmarked the proposed algorithm against the four reference algorithms from the Sect. 4.1 and have performed a statistical analysis as in Sects. 4.2–4.3. Table 10 shows the test statistics from which it follows that also in this real-world application the proposed algorithm outperformed the four reference algorithms.

Table 10 Results for the airfoil shape optimization

Function	Statistic [1,2]	Proposed	(G,10,0)	(S,10,0)	(G,5,5)	(S,5,5)
Airfoil	Mean	$5.674e-01$	$7.336e-01$	$7.757e-01$	$6.934e-01$	$7.571e-01$
	S.D.	$8.098e-02$	$1.336e-01$	$2.583e-01$	$8.974e-02$	$1.701e-01$
	Median	$6.029e-01$	$6.920e-01$	$7.460e-01$	$7.092e-01$	$7.666e-01$
	Best	$3.846e-01$	$4.763e-01$	$4.287e-01$	$5.009e-01$	$4.369e-01$
	Worst	$6.326e-01$	$9.159e-01$	$1.430e+00$	$8.003e-01$	$9.738e-01$
	U		$3.021e+00$	$2.858e+00$	$2.939e+00$	$2.613e+00$

[1] S.D. :standard deviation
[2] Reject H_0 at $\alpha = 0.05$ if $U \geqslant 1.644$.
Reject H_0 at $\alpha = 0.01$ if $U \geqslant 2.326$.

5 Summary

Modern engineering design optimization uses computer simulations and as such it is cast as a problem of optimizing an expensive black-box function. To efficiently and effectively solve such problems we have proposed a new model-assisted memetic algorithm. The proposed algorithm combines several optimization approaches such as: global–local optimization, modelling and memetic optimization. It first trains and selects a global model which is an artificial neural network and seeks the optimum of this model. It then uses a local search to improve this predicted optimum. This sequence is repeated until the number of function evaluations reaches the prescribed limit. Compared to existing studies the proposed algorithm contains three main novelties: a) it selects all models under a unified and statistically-sound framework of model selection and complexity control, and this gives optimal models which are adapted during the search b) it uses an improved trust-region framework to converge to a true optimum while replacing the classical quadratic models with Kriging models and adapting these models during the search and c) it improves global exploration by training the global model with modified sites.

An extensive performance analysis has been provided. Results show the proposed algorithm outperformed four model-assisted EAs on eight well-known mathematical test functions. The individual contribution of the proposed global search and local search component was also studied. While each component performs well on a certain class of problems it also performs poorly on another. This emphasizes the advantage of the global–local approach used. Lastly, the proposed algorithm was also applied to a real-world application of airfoil shape optimization where it also performed better than the reference algorithms.

References

1. Akaike, H.: Information theory and an extension of the maximum likelihood principle. In: Proceedings of the 2*nd* International Symposium on Information Theory, Akadémiai Kiadó, Budapest, pp. 267–281 (1973)

2. Alexandrov, N.M., Lewis, R.M., Gumbert, C.R., Green, L.L., Newma, P.A.: Optimization with variable-fidelity models applied to wing design. In: Proceedings of the 38*th* Aerospace Sciences Meeting and Exhibit, American Institute for Aeronautics and Astronautics, Reston, Virginia (2000)
3. Anderson, D.R., Burnham, K.P., White, G.C.: Comparison of Akaike information criterion and consistent Akaike information criterion for model selection and statistical inference from capture-recapture studies. Journal of Applied Statistics 25(2), 263–282 (1998)
4. Barthelemy, J.F.M., Haftka, R.T.: Approximation concepts for optimum structural design – a review. Structural optimization 5, 129–144 (1993)
5. Bellman, R.E.: Adaptive Control Processes: A Guided Tour. Princeton University Press, Princeton (1961)
6. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, New York (1995)
7. Booker, A.J., Dennis, J.E., Frank, P.D., Serafini, D.B., Torczon, V., Trosset, M.W.: A rigorous framework for optimization of expensive functions by surrogates. Structural Optimization 17(1), 1–13 (1999)
8. Box, G.E.P., Draper, N.R.: Empirical Model Building and Response Surface. John Wiley and Sons, New York (1987)
9. Brent, R.P.: Algorithms for Minimization Without Derivatives, 3rd edn. Dover Publications, New York (2002)
10. Broomhead, D., Lowe, D.: Multivariate functional interpolations and adaptive networks. Complex Systems 2, 321–355 (1988)
11. Buhman, M.D.: Radial Basis Functions Theory and Implementations. Cambridge Monographs on Applied and Computational Mathematics, vol. 12. Cambridge University Press, Cambridge (2003)
12. Burnham, K.P., Anderson, D.R.: Model selection and inference: A Practical Information-theoretic Approach. Springer, New York (1998)
13. Chipperfield, A., Fleming, P., Pohlheim, H., Fonseca, C.: Genetic Algorithm TOOLBOX For Use with MATLAB, Version 1.2. Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield (1994)
14. Conn, A.R., Scheinberg, K., Toint, P.L.: On the convergence of derivative-free methods for unconstrained optimization. In: Iserles, A., Buhmann, M.D. (eds.) Approximation Theory and Optimization: Tributes to M.J.D. Powell, pp. 83–108. Cambridge University Press, Cambridge (1997)
15. Conn, A.R., Scheinberg, K., Toint, P.L.: Recent progress in unconstrained nonlinear optimization without derivatives. Mathematical Programming 79, 397–414 (1997)
16. Conn, A.R., Scheinberg, K., Toint, P.L.: A derivative free optimization algorithm in practice. In: Proceedings of the Seventh AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, American Institute for Aeronautics and Astronautics, Reston, Virginia, AIAA Paper AIAA-1998-4718 (1998)
17. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust Region Methods. SIAM, Philadelphia (2000)
18. Conover, W.: Practical Nonparametric Statistics, 2nd edn. John Wiley and Sons, New York (1980)
19. Cressie, N.A.C.: The origins of Kriging. Mathematical Geology 22(3), 239–252 (1990)
20. Cressie, N.A.C.: Statistics for Spatial Data. Wiley, New York (1993)
21. Dawkins, R.: The Selfish Gene. Oxford University Press, Oxford (1976)

22. Demmel, J.W.: The geometry of ill-conditioning. Computer Journal 3(2), 201–229 (1987)
23. Dennis, J.E., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Classics in Applied Mathematics. SIAM Publishing, Philadelphia (1996)
24. Dixon, L.C.W., Szegö, G.P.: The global optimization problem: An introduction. In: [25], pp. 1–15 (1978)
25. Dixon, L.C.W., Szegö, G.P. (eds.): Towards Global Optimisation 2. North-Holland Publishing Company, Amsterdam (1978)
26. Drela, M., Youngren, H.: XFOIL 6.9 User Primer. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA (2001)
27. Eby, D., Averill, R.C., Punch III, W.F., Goodman, E.D.: Evaluation of injection island GA performance on flywheel design optimization. In: Proceedings of the Third Conference on Adaptive Computing in Design and Manufacturing–ACDM 1998, pp. 121–136. Springer, London (1998)
28. Fang, K.T., Li, R., Sudjinato, A.: Design and Modeling for Computer Experiments. Chapman and Hall, Boca Raton (2006)
29. Filippone, A.: Flight Performance of Fixed and Rotary Wing Aircraft, 1st edn. Elsevier, Amsterdam (2006)
30. Gaspar-Cunha, A., Vieira, A.: A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. International Journal of Computers, Systems and Signals 6(1), 18–36 (2005)
31. Giannakoglou, K.C.: Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. International Review Journal Progress in Aerospace Sciences 38(1), 43–76 (2002)
32. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
33. Griewank, A.O.: Generalized descent for global optimization. Journal of Optimization Theory and Applications 34, 11–39 (1981)
34. Groch, A., Vidigal, L.M., Director, S.W.: A new global optimization method for electronic circuit design. IEEE Transactions on Circuit and Systems 32(2), 160–170 (1985)
35. Hamerly, G., Elkan, C.: Alternatives to the *k*-means algorithm that find better clusterings. In: Munson, E.V., Furuta, R., Maletic, J.I. (eds.) Proceedings of the 2002 ACM Symposium on Document Engineering, in conjunction with the eleventh ACM International Conference on Information and Knowledge Management–CIKM 2002, New York, pp. 600–607 (2002)
36. Hardy, G.H.: Weierstrass's non-differentiable function. Transactions of the American Mathematical Society 17, 301–325 (1916)
37. Hart, W.E., Belew, R.K.: Optimization with genetic algorithm hybrids that use local search. In: Belew, R.K., Mitchell, M. (eds.) Adaptive Individuals in Evolving Populations: Models and Algorithms, Santa Fe Institute Studies in the Sciences of Complexity, ch. 27, pp. 483–496. Addison-Wesley, Reading (1995)
38. Hart, W.E., Krasnogor, N., Smith, J.E.: Special issue on memetic algorithms. Evolutionary Computation 12(3) (2004)
39. Hart, W.E., Krasnogor, N., Smith, J.E.: Recent Advances in Memetic Algorithms. Studies in Fuzziness and Soft Computing, vol. 166. Springer, Heidelberg (2005)
40. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice-Hall, Upper Saddle River (1999)

41. Holst, T.L., Pulliam, T.H.: Aerodynamic shape optimization using a real-number-encoded genetic algorithm. In: Proceedings of the 19th AIAA Applied Aerodynamics Conference, American Institute for Aeronautics and Astronautics, Reston, Virginia, AIAA Paper AIAA-2001-2473 (2001)
42. Hurvich, C.M., Tsai, C.L.: Regression and time series model selection in small samples. Biometrika 76(2), 297–307 (1989)
43. Ingber, L.A., Rosen, B.: Genetic algorithms and very fast simulated reannealing: A comparison. Mathematical and Computer Modelling 16(11), 87–100 (1992)
44. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Transactions on Evolutionary Computation 7, 204–223 (2003)
45. Jin, R., Chen, W., Simpson, T.W.: Comparative studies of metamodeling techniques under multiple modeling criteria. Structural Optimization 23(1), 1–13 (2001)
46. Jin, Y., Olhofer, M., Sendhoff, B.: A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on evolutionary computation 6(5), 481–494 (2002)
47. Johnson, M.E., Moore, L.M., Ylvisaker, D.: Minimax and maximin distance designs. Journal of Statistical Planning and Inference 26(2), 131–148 (1990)
48. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13, 455–492 (1998)
49. de Jong, K.A.: Genetic algorithms are NOT function optimizers. In: Whitley, D.L. (ed.) Foundations of Genetic Algorithms 2, pp. 5–17. Morgan Kaufmann, San Mateo (1993)
50. de Jong, K.A., Spears, W.M.: An analysis of the interacting roles of population size and crossover in genetic algorithms. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 38–47. Springer, Heidelberg (1991)
51. Kansa, E.J., Hon, Y.C.: Circumventing the ill-conditioning problem with multiquadric radial basis functions: Applications to elliptic partial differential equations. Computers and Mathematics with Applications 39(7), 123–137 (2000)
52. Kim, H.S., Cho, S.B.: An efficient genetic algorithm with less fitness evaluation by clustering. In: Proceedings of 2001 IEEE Conference on Evolutionary Computation, pp. 887–894. IEEE, Piscataway (2001)
53. Koehler, J.R., Owen, A.B.: Computer experiments. In: Ghosh, S., Rao, C.R., Krishnaiah, P.R. (eds.) Handbook of Statistics, pp. 261–308. Elsevier, Amsterdam (1996)
54. Krasnogor, N., Smith, J.E.: A tutorial for competent memetic algorithms: Model, taxonomy, and design issues. IEEE Transactions on Evolutionary Computation 9(5), 474–488 (2005)
55. Ku, K., Mak, M., Siu, W.: A study of the Lamarckian evolution of recurrent neural networks. IEEE Transactions on Evolutionary Computation 4, 31–42 (2000)
56. Kullback, S., Leibler, R.A.: On information and sufficiency. The Annals of Mathematical Statistics 22(1), 79–86 (1951)
57. Kushner, H.J.: A versatile stochastic model of a function of unknown and time varying form. Journal of Mathematical Analysis and Applications 5(1), 150–167 (1962)
58. Laslett, G.M.: Kriging and splines: An empirical comparison of their predictive performance in some applications. Journal of the American Statistical Association 89(426), 391–400 (1994)
59. Leontaritis, I.J., Billings, S.A.: Model selection and validation for non-linear systems. International Journal of Control 45(1), 311–341 (1986)

60. Liang, K.H., Yao, X., Newton, C.: Evolutionary search of approximated N-dimensional landscapes. International Journal of Knowledge-Based Intelligent Engineering Systems 4(3), 172–183 (2000)
61. Linhart, H., Zucchini, W.: Model Selection. Wiley Series in Probability and Mathematical Statistics. Wiley-Interscience Publication, New York (1986)
62. Lorentz, G.G.: Approximation of Functions. Rinehart and Winston, New York (1966)
63. Madych, W.R.: Miscellaneous error bounds for multiquadric and related interpolators. Computers and Mathematics with Applications 24(12), 121–138 (1992)
64. Mann, H.B., Whitney, D.R.: On a test whether one of two variables is stochastically larger than the other. The Annals of Mathematical Statistics 18, 50–60 (1947)
65. Marida, K., Marshall, R.: Maximum likelihood estimation of models for residual covariance in spatial regression. Biometrika 71(1), 135–146 (1984)
66. Matheron, C.: Principles of geostatistics. Economic Geology 58, 1246–1266 (1963)
67. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21(2), 239–245 (1979)
68. Meckesheimer, M., Booker, A.J., Barton, R.R., Simpson, T.W.: Computationally inexpensive metamodel assessment strategies. AIAA Journal 40(10), 2053–2060 (2002)
69. Medgyessy, P.: Decomposition of Superpositions of Distribution Functions. Akadémiai Kiadó, Budapest (1961)
70. Michalewicz, Z., Fogel, D.B.: How to Solve It: Modern Heuristics. Springer, Berlin (2004)
71. Mitchell, T.J., Morris, M.D.: Bayesian design and analysis of computer experiments: Two examples. Statistica Sinica 2 (1992)
72. Mockus, J., Vitešis, V., Žilinskas, A.: The application of Bayesian methods for seeking the extremum. In: [25], pp. 117–130 (1978)
73. Moody, J., Darken, C.J.: Fast learning in networks of locally-tuned processing units. Neural Computation 1(2), 281–294 (1989)
74. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms. Tech. Rep. 826, California Institute of Technology, Pasadena, California (1989)
75. Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive models for the breeder genetic algorithm I: Continuous parameter optimization. Evolutionary Computations 1(1), 25–49 (1993)
76. Myers, R.H., Montgomery, D.C.: Response Surface Methodology: Process and Product Optimization Using Designed Experiments. John Wiley and Sons, New York (1995)
77. Norman, M., Moscato, P.: A competitive-cooperative approach to complex combinatorial search. Tech. Rep. 790, California Institute of Technology, Pasadena, California (1989)
78. Ong, Y.S., Keane, A.J.: Meta-Lamarckian learning in memetic algorithm. IEEE Transactions On Evolutionary Computation 8(2), 99–110 (2004)
79. Ong, Y.S., Nair, P.B., Keane, A.J.: Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA Journal 41(4), 687–696 (2003)
80. Ong, Y.S., Nair, P.B., Keane, A.J., Wong, K.W.: Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In: Knowledge Incorporation in Evolutionary Computation. Studies in Fuzziness and Soft Computing, vol. 167, pp. 307–332. Springer, Berlin (2005)

81. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of adaptive memetic algorithms: A comparative study. IEEE Transactions on Systems, Man, and Cybernetics–Part B 36(1), 141–152 (2006)
82. Ong, Y.S., Krasnogor, N., Ishibuchi, H.: Special issue on memetic algorithms. IEEE Transactions on Evolutionary Computation 37(1) (2007)
83. Ong, Y.S., Lim, M.H., Neri, F., Ishibuchi, H.: Special issue on emerging trends in soft computing–memetic algorithms. Journal of Soft Computing (to appear)
84. Owen, A.B.: Orthogonal arrays for computer experiments, integration and visualization. Statistica Sinica 2, 439–452 (1992)
85. Oyama, A., Obayashi, S., Nakahashi, K.: Real-coded adaptive range genetic algorithm and its application to aerodynamic design. International Journal of the Japan Society of Mechanical Engineering 43(2), 124–129 (2000)
86. Oyama, A., Obayashi, S., Nakahashi, T.: Real-coded adaptive range genetic algorithm applied to transonic wing optimization. In: Schoenauer, M. (ed.) The 6*th* International Conference on Parallel Problem Solving from Nature–PPSN VI, pp. 712–721. Springer, Heidelberg (2000)
87. Pawitan, Y.: In All Likelihood: Statistical Modelling and Inference Using Likelihood. Oxford Scientific Publishing, Oxford (2001)
88. Poloni, C., Giurgevich, A., Onseti, L., Pediroda, V.: Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. Computer Methods in Applied Mechanics and Engineering 186(2-4), 403–420 (2000)
89. Powell, M.J.D.: UOBYQA: Unconstrained optimization by quadratic approximation. Mathematical Programming, Series B 92, 555–582 (2002)
90. Quagliarella, D., Vicini, A.: Coupling genetic algorithms and gradient based optimization techniques. In: Quagliarella, D., Périaux, J., Poloni, C., Winter, G. (eds.) Genetic Algorithms in Engineering and Computer Science, ch. 14, pp. 288–309. John Wiley and Sons, Chichester (1997)
91. Ratle, A.: Accelerating the convergence of evolutionary algorithms by fitness landscape approximations. In: Eiben, A.E., Bäck, T., Schwefel, H.P. (eds.) Proceedings of the 5*th* International Conference on Parallel Problem Solving from Nature–PPSN V, pp. 87–96. Springer, Berlin (1998)
92. Ratle, A.: Optimal sampling strategies for learning a fitness model. In: The 1999 IEEE Congress on Evolutionary Computation–CEC 1999, pp. 2078–2085. IEEE, Piscataway (1999)
93. Renderes, J.M., Bersini, H.: Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways. In: Sebald, A., Fogel, L.J. (eds.) Proceedings of the Third Annual Conference on Evolutionary Programming, pp. 312–317. World Scientific, Singapore (1994)
94. Renderes, J.M., Flasse, S.P.: Hybrid methods using genetic algorithms for global optimization. IEEE Transactions on Systems, Man, and Cybernetics–Part B 26(2), 243–258 (1996)
95. Reyes-Sierra, M., Coelle Coello, C.A.: Dynamic fitness inheritance proportion for multi-objective particle swarm optimization. In: Keijzer, M. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference–GECCO 2006, pp. 89–90. Association for Computing Machinery, New York (2006)
96. Rinnooy Kan, A.H.G., Timmer, G.T.: Stochastic global optimization methods part I: Clustering methods. Mathematical Programming 39, 27–56 (1987)

97. Rosenbrock, H.H.: An automated method for finding the greatest of least value of a function. The Computer Journal 3, 175–184 (1960)
98. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical Science 4(4), 409–435 (1989)
99. Santner, T.J., Williams, B.J., Notz, W.: The Design and Analysis of Computer Experiments. Springer Series in Statistics. Springer, New York (2003)
100. Schaback, R.: Multivariate interpolation and approximation by translates of a basis function. In: Chui, C.K., Schumaker, L.L. (eds.) Approximation Theory VIII, pp. 491–514. World Scientific, Singapore (1995)
101. Schwefel, H.P.: Numerical Optimization of Computer Models, Interdisciplinary Systems Research, vol. 26. John Wiley and Sons, Chichester (1981)
102. Sefrioui, M., Périaux, J.: Aerodynamic shape optimization using a hierarchical genetic algorithm. In: European Conference on Computational Methods in Applied Sciences and Engineering–ECCOMAS 2000, European Committee on Computational Methods in Applied Sciences, pp. 1–18 (2000)
103. Seront, G., Bersini, H.: A new GA-local search hybrid for continuous optimization based on multi level single linkage clustering. In: Whitley, D.L., Beyer, H., Cantu-Paz, E., Goldberg, D.E., Parmee, I., Spector, L. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference–GECCO 2000, pp. 90–95. Morgan Kaufmann, San Francisco (2000)
104. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures, 4th edn. Chapman and Hall, Boca Raton (2007)
105. Simkin, J., Trowbridge, C.W.: Optimizing electromagnetic devices combining direct search methods with simulated annealing. IEEE Transactions on Magnetics 28(2), 1545–1548 (1992)
106. Simpson, T.W., Poplinski, J.D., Koch, P.N., Allen, J.K.: Metamodels for computer-based engineering design: Survey and recommendations. Engineering with Computers 17, 129–150 (2001)
107. Smith, R.E., Dike, B., Stegmann, S.: Fitness inheritance in genetic algorithms. In: George, K.M. (ed.) Proceedings of the 1995 ACM Symposium on Applied Computing–ACM 1995, pp. 345–350. Association for Computing Machinery, New York (1995)
108. Sobieckzy, H.: Parametric airfoils and wings. In: Fujii, K., Dulikravich, G.S., Takanashi, S. (eds.) Recent Development of Aerodynamic Design Methodologies: Inverse Design and Optimization, Notes on Numerical Fluid Mechanics, vol. 68, pp. 71–88. Vieweg, Braunschweig (1999)
109. Søren, L.N., Nielsen, H.B., Søndergaard, J.: DACE: A MATLAB Kriging toolbox. Technical Report IMM-TR-2002-12, Informatik and Mathematical Modelling, Technical University of Denmark, Lingby, Copenhagen (2002)
110. Stuckman, B.E.: A global search method for optimizing nonlinear systems. IEEE Transactions on Systems, Man, and Cybernetics 18(6), 965–977 (1988)
111. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report KanGAL 2005005, Nanyang Technological University, Singapore and Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology Kanpur, India (2005), http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/SharedDocuments/Forms/AllItems.aspx
112. Tenne, Y.: Metamodel accuracy assessment in evolutionary optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation–CEC 2008. IEEE World Congress on Computational Intelligence, pp. 1505–1512. IEEE, Piscataway (2008)

113. Tenne, Y., Armfield, S.W.: A memetic algorithm using a trust-region derivative-free optimization with quadratic modelling for optimization of expensive and noisy black-box functions. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. Studies in Computational Intelligence, vol. 51, pp. 389–415. Springer, Berlin (2007)
114. Tenne, Y., Armfield, S.W.: A versatile surrogate-assisted memetic algorithm for optimization of computationally expensive functions and its engineering applications. In: Yang, A., Shan, Y., Thu Bui, L. (eds.) Success in Evolutionary Computation. Studies in Computational Intelligence, vol. 92, pp. 43–72. Springer, Heidelberg (2008)
115. Torczon, V.: On the convergence of pattern search algorithms. SIAM Journal on Optimization 7(1), 1–25 (1997)
116. Törn, A.: Global optimization as a combination of global and local search. In: Proceedings of Computer Simulation Versus Analytical Solutions for Business and Economic Models, School of Business Administration, Göteborg, Göteborg, Sweden. Business Administration Studies–BAS, vol. 17, pp. 191–206 (1973)
117. Törn, A., Žilinskas, A.: Global Optimization. LNCS, vol. 350. Springer, Heidelberg (1989)
118. Törn, A., Ali, M.M., Viitanen, S.: Stochastic global optimization: Problems classes and solution techniques. Journal of Global Optimization 14, 437–447 (1999)
119. Žilinskas, A.: A review of statistical models for global optimization. Journal of Global Optimization 2, 145–153 (1992)
120. Waldorp, L.J., Raoul, P.P.P., Huizenga, H.M.: Goodness-of-fit and confidence intervals of approximate models. Journal of Mathematical Psychology 50, 203–213 (2006)
121. Winfield, D.: Function minimization by interpolation in a data table. Journal of the Institute of Mathematics and its Applications 12, 339–347 (1973)
122. Yen, J., Liao, J.C., Lee, B., Randolph, D.: A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. IEEE Transactions on Systems, Man, and Cybernetics–Part B 28(2), 173–191 (1998)
123. Yun, Y., Gen, M., Seo, S.: Various hybrid methods based on genetic algorithm with fuzzy logic controller. Journal of Intelligent Manufacturing 14, 401–419 (2003)
124. Zhang, B.: Generalized K-harmonic means–dynamic weighting of data in unsupervised learning. Tech. Rep. HPL-2000-137, Hewlett-Packard Labs (2000)
125. Zhang, B., Hsu, M., Dayal, U.: K-harmonic means–a data clustering algorithm. Tech. Rep. HPL-1999-124, Hewlett-Packard Labs, Software Technology Laboratory, HP Laboratories Palo Alto (1999)
126. Zhou, Z., Ong, Y.S., Lim, M.H., Lee, B.: Memetic algorithms using multi-surrogates for computationally expensive optimization problems. Journal of Soft Computing 11(10), 957–971 (2007)
127. Zhou, Z., Ong, Y.S., Nair, P.B., Keane, A.J., Lum, K.Y.: Combining global and local surrogate models to accelerate evolutionary optimization. IEEE Transactions On Systems, Man and Cybernetics-Part C 37(1), 66–76 (2007)

A Self-adaptive Mixed Distribution Based Uni-variate Estimation of Distribution Algorithm for Large Scale Global Optimization

Yu Wang and Bin Li

Abstract. Large scale global optimization (LSGO), which is highly needed for many scientific and engineering applications, is a very important and very difficult task in optimization domain. Various algorithms have been proposed to tackle this challenging problem, but the use of estimation of distribution algorithms (EDAs) to it is rare. This chapter aims at investigating the behavior and performances of uni-variate EDAs mixed with different kernel probability densities via fitness landscape analysis. Based on the analysis, a self-adaptive uni-variate EDA with mixed kernels (MUEDA) is proposed. To assess the effectiveness and efficiency of MUEDA, function optimization tasks with dimension scaling from 30 to 1500 are adopted. Compared to the recently published LSGO algorithms, MUEDA shows excellent convergence speed, final solution quality and dimensional scalability.

1 Introduction

Considered as a kind of classical yet extremely difficult task, large scale global optimization (LSGO) has attracted more and more research interest in recent years [21, 31]. LSGO problems have numerous scientific and engineering applications, such as designing large scale electronic systems, scheduling problems with large number of resources, vehicle routing in large scale traffic networks, gene detection in bioinformatics, etc. Therefore, effective LSGO algorithms are in high demand.

Inherently, the nonlinear characteristics of the practical applications usually include discontinuous prohibited zones, ramp rate limits, and nonsmooth or convex

Yu Wang · Bin Li
Nature Inspired Computation and Application Laboratory (NICAL),
University of Science and Technology of China
e-mail: wyustc@mail.ustc.edu.cn, binli@ustc.edu.cn

cost functions. Historically, a number of algorithms, including both mathematical and evolutionary algorithms, have been proposed to handle LSGO problems [5, 10, 15, 17, 23, 26, 32, 33, 36, 37, 38, 42, 43]. Various evolutionary algorithms (EAs) have been developed, in which significant progress has been observed [20] compared to the mathematical algorithms. The major advantages of these EAs over other classical methods can be summarized as: 1) prior knowledge of the search problem is not necessary for EAs, while for mathematical approaches the highly nonlinear characteristic of the problem must be approximated beforehand; 2) they work with a population of candidate solutions and can handle LSGO problems automatically through a single run. However, almost all of these approaches inevitably suffer from the "curse of dimensionality", which means poor performances on LSGO problems.

Without loss of generality, LSGO problems considered in this chapter can be stated as follows:

$$\begin{aligned}&\text{minimize } F(\mathbf{x}) = f(x_1, x_2, ..., x_D)\\ &\text{subject to } \mathbf{x} \in X,\end{aligned} \tag{1}$$

where $X \subset R^D$ denotes the decision space with D dimensions; $\mathbf{x} = \{x_1, x_2, ..., x_D\} \in R^D$ is the decision variable vector; $f : X \to R$ stands for a real-valued continuous objective function for mapping from D dimensional space to 1 dimensional fitness value $F(\mathbf{x})$. The dimensions of LSGO problems considered in this chapter are more than 100. Hence, the purpose of the approaches is to search for the minimized solution in such a large dimensional space. If X is a closed and connected region in R^D, we call eq. (1) continuous LSGO.

In the previous works on LSGO, developing more effective operators for EAs has attracted much research attention. The successful implementations consist of self-adapting strategies for parameter setting, modification of the classical EA operators, etc. The reason of making these modifications is that the classical operators are usually developed for low-dimensional task and they lose their efficiency for high-dimensional tasks. Their performances on LSGO problems cannot be measured effectively [31]. Recently, this field has attracted increased research attention and the typical approaches include population reduction for differential evolution (DE) [5], Dynamic multi-swarm PSO [43], and estimation of distribution algorithm (EDA) with mixed sampling operator [33]. For these approaches, the LSGO problems are optimized as an entire body, which means no divide-and-conquer methods are used. Actually, the implementation of specific operators is attributed to strengthening the algorithm's capability for higher dimensional tasks.

The classical EDAs have been proven to be effective on most classical test functions with less than 100 dimensions [11]. However, EDAs also suffer from the "curse of dimensionality", which implies that their performances deteriorate quickly as the search space increases in dimensions [33]. In this chapter, the difficulties associated with EDAs in solving LSGO problems will be discussed. Then, a heavy tail distribution based sampling (also called mutation in the evolutionary computation domain) operator will be analyzed, and introduced into a Gaussian based EDA to improve its performance. Compared to classical Gaussian distribution based operators, the heavy tail distribution based operators have demonstrated better exploration and faster convergence speed on most test problems. Some typical examples include fast evolutionary programming (FEP) [40], fast evolution strategy (FES) [41], fast simulated annealing (FSA) [30], evolutionary programming using Lévy mutation (LEP) [13] and estimation of distribution algorithm with heavy tail distribution based sampling (LSEDA-gl) [33]. Due to the success of the above algorithms, the heavy tail distribution based sampling operator is regarded as a promising EA technique to tackle some difficult problems. In this chapter, the evovabilities of different sampling operators are investigated via a technique called fitness landscape portrait (FLP). Based on this analysis, a self-adaptive mixed model uni-variate EDA (MUEDA) is proposed. In order to evaluate the performance of MUEDA, it is tested on typical function optimization problems with dimensionality scaling from 30 to 1500.

The rest of this chapter is organized as follows: In section 2, the principle and a brief review of EDAs are provided. The mathematical characteristics of Gaussian and heavy tail distributions are analyzed. Then, the main contributions of this work are presented. In section 3, a general FLP analysis is carried out to investigate the evolvability of different sampling operators in the low-dimension problems. After that, a self-adaptive uni-variate EDA with mixed kernels is proposed. In section 4, discussion of the experimental studies with dimensions ranging from 30 to 1500 is presented. Following which, the scalability and efficiency of MUEDA are presented. In section 5, several general conclusions are drawn and emphasized. In section 6, the future research directions of this area are outlined.

2 Related Work

2.1 Estimation of Distribution Algorithms

The notion of modeling the search space was first proposed in the statistics and machine learning domain. Recently, many works within the Evolutionary Computation community have employed probabilistic models to describe the solution space [11]. These methods have come to be known as EDAs. It has been reported in various works that EDAs have been applied with significant success to many numerical and combination optimization problems in the past few years. Optimization by EDAs can be summarized into two major steps:

- Model the promising area of solution space of the optimization problem by learning from the superior solutions found thus far;
- Generate the population (i.e., offspring) for the next generation by sampling based on the estimated probabilistic model and then, replace the old population (i.e., parents) partially or fully.

These two steps can be regarded as a population-based version of the classical generate-and-test method [39]. As is shown in Fig. 1, there are no classical crossover or mutation operators in EDAs in contrast to traditional GA. The main operators of EDAs are as follows: selection involves selecting good solutions from the entire

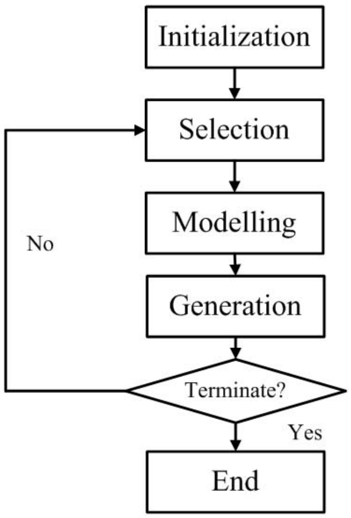


Fig. 1 A general flowchart of EDA

population by truncation or league strategy; modeling involves building the probabilistic model to simulate the landscape of the problem; and generation is to sample the new population based on the probabilistic model. The evolution dynamics of EDAs depend on the distribution of the population directly. Therefore, the major advantage of EDAs is that they can explicitly extract useful structural information to efficiently generate new individuals, which results in faster convergence speed compared to GA [33].

In the EDA domain, modeling the structure of the optimization problem accurately has recently been an area of great concern. To overcome the disadvantages of limited learning ability of uni-variate EDAs, such as population-based incremental learning algorithm (PBIL) [25], stochastic hill climbing by vectors of normal distributions (SHCLVND) [24] and continuous uni-variate marginal distribution algorithm (UMDAc) [12], EDAs whose dependencies are considered in terms of pairwire or multiwire when building probabilistic model have been proposed. Some of the more successful approaches are mutual information maximizing input clustering (MIMIC) [12], estimation of Gaussian networks algorithm by edge exclusion (EGNAee) [12], estimation of Gaussian networks algorithm by BGe metric (EGNA_{BGe}), clustering and estimation of Gaussian distribution algorithm (CEGDA) [16], etc. The learning ability of the EDAs is always considered as the major indicator of the performance for the above-mentioned algorithms. However, the fundamental task of EDAs is to search for the global optimum of the given optimization problem, rather than to simply model the structure of the optimization accurately. Furthermore, the computational cost of constructing a complex model is too expensive for LSGO problems, whose dimensions are usually more than 100. For example, the computational cost of generating the covariance matrix for iterated density estimation algorithm (IDEA) [1, 2, 4, 7] increases exponentially. The reported studies on extending EDAs to LSGO domain are scarce so far. The emphasis of heavy tail based EDA in this work is therefore on extending EDAs to the LSGO domain.

2.2 Mathematical Characteristics of Heavy Tail Distribution

In the above reviewed EDA works, Gaussian probabilistic distribution has been widely adopted in continuous optimization. In this work, a low proportion of heavy tail stable distribution is incorporated in order to strengthen the search ability of EDAs for LSGO problems.

Definition: Consider a process represented by a set Y_i of identically distributed random variables. If the sum of these random variables has the same probability distribution as individual random variables, then this process is called *stable* [13].

The Gaussian process is a typical example of a stable process with finite second moment, which would lead to a characteristic scale and the Gaussian behavior for the probability density through the central limit theorem [13]. The probability density of Gaussian distribution with mean 0 and variance σ is defined as follows:

$$N(0,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \tag{2}$$

Different to Gaussian probability distribution whose variance can be denoted as a finite scalar, a class of probability distributions with an infinite second moment that also yields a stable process were discovered by P. Lévy in the 1930s [14]. The formal representation for such a class of probability can be expressed as follows [6, 14]:

$$Ł_{\alpha,\gamma} y = \frac{1}{\pi} \int_0^\infty e^{-\gamma q^\alpha} \cos(qy) dq \qquad y \in \mathbf{R}, \tag{3}$$

where γ is the scaling factor satisfying $\gamma > 0$, and α controls the shape of the distribution, requiring $0 < \alpha < 2$. More analytic details about the Lévy distribution are available in [6, 13, 14]. Although the analytic form of the integral is still unknown for general α, the shapes generated by Lévy distributions with different α values are known: the length of the tail is inversely proportional to the value of α. The Cauchy probability distribution adopted in FEP, is a special case of the Lévy probability distribution with $\alpha = 1$. For the limit of $\alpha = 2$, the distribution is reduced to the classical Gaussian distribution which is not included in Lévy probability distribution class. The Cauchy density with median 0 and upper quartile τ can be denoted as follows:

$$C(0,\tau) = \frac{1}{\pi} \frac{\tau^2}{x^2 + \tau^2} \tag{4}$$

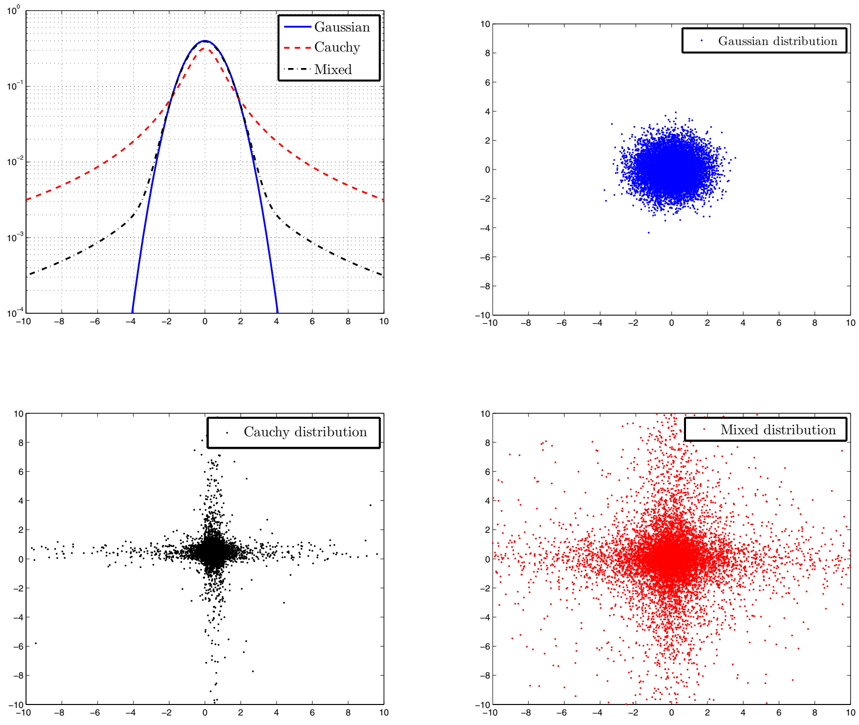


Fig. 2 Comparison among Gaussian, Cauchy and Mixed distributions in terms of the density function (up left) and the distributions of 10000 sampled points

The comparison between Gaussian and Cauchy probability density is shown in Fig. 2. It is apparent that the characteristic of infinite second moment in Cauchy probability provides a much wider distribution.

Since it is rather difficult to generate random numbers under different Lévy probability distributions except Cauchy distribution [13], several works adopt a mixed distribution, which combines the Gaussian distribution with Cauchy distribution by a suitable proportion to simulate the desired distribution. Some successful examples are [27] and [33]. These special mixed sampling density functions can be expressed as:

$$f_M = (1-\eta)\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}} + \eta\frac{1}{\pi}\frac{1}{x^2+1} \tag{5}$$

where η stands for the mixed proportion that can be tuned to any scalar between 0 and 1. In this chapter, $\eta = 0.1$ is used for analysis without statement. After investigating the mixed distribution in [13, 27, 33], it was observed that $\eta = 0.1$ favors more problems than any other value of η. For heavy distribution based sampling operator, the sampling distribution in classical uni-variate EDA ($N(\mu,\sigma) = \mu + \sigma N(0,1)$) is replaced by:

$$N_M(\mu,\sigma) = \mu + \sigma(1-\eta)N(0,1) + \sigma\eta C(0,1) \tag{6}$$

The density of 10000 sampled points by Gaussian, Cauchy and Mixed probability distribution are compared in Fig. 2. In contrast to the widest distribution sampled by Cauchy, it can be observed that the shape of a 2-D Gaussian distribution is like that of a sphere with the density increasing from the periphery towards the core. Similar observation can be made for mixed distribution but the sampling region is much wider. The mixed distribution is expected to achieve a more reasonable balance between exploration and exploitation theoretically.

2.3 Analysis of Heavy Tail Distribution in Evolutionary Computation

2.3.1 Fast Evolutionary Programming

In 1996, Yao X [40] proposed an important EP version named FEP, which replaces the Gaussian mutation in classical EP (CEP) with the Cauchy mutation. This change leads the individuals to make a much longer jump, which results in a significantly faster convergence speed when the global optima is far from the initial point. It was shown that the FEP outperforms CEP in terms of convergence speed in most test functions and the accuracy of the result in the multimodal functions [40].

Due to the excellent performance of FEP, [35] and [40] investigated the differences of expected length between Cauchy mutation and Gaussian mutation. The existing analysis methods of different probability operators focus on the properties of the operators themselves (i.e., properties related to the operators closely only). The expected length of Gaussian mutation jump with $\sigma = 1$ is calculated as follows:

$$E_G(x) = 2\int_0^{+\infty} x\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}dx = \frac{2}{\sqrt{2\pi}} = 0.80, \tag{7}$$

and the expected length of Cauchy mutation jump is calculated as:

$$E_C(x) = 2\int_0^{+\infty} x\frac{1}{\pi}\frac{1}{x^2+1}dx = +\infty \tag{8}$$

Under the 1-D space, it is obvious that the Cauchy probability operator extends the expected sampling region to an infinite area. Therefore, a general conclusion has been made in [40]: Gaussian mutation is much more localized than Cauchy mutation. However, benefiting from the proportion of Cauchy probability incorporated into it, the mixed operator ($E_M(x) = +\infty$) also exhibits the ability of sampling population widely. There is no remarkable difference between mixed sampling operator and Cauchy sampling operator in expected sampling length theoretically. Thus, in the following section, a much more effective analysis tool FLP is adopted to distinguish the differences between different operators.

2.3.2 Heavy Tail Distribution Analysis

An important work which tries to answer the question: *when do the important tail distributions help?* is presented in [9]. The most important contributions of this work come from two hypotheses and their proofs, which can be summarized as: although the heavy tail distributions have stronger ability of maintaining diversity, they inevitably get lost in huge dimensional space, which will be proven to be worth discussing in the following section. The experimental analysis in [9] was carried out to study the behaviors of ES on Rastrigin function optimization with different distributions based mutation operators. Obviously, the viewpoint of the first hypothesis that heavy tail distributions are good at maintaining diversity is absolutely true. In our work, however, the heavy tail distributions are also proven to work in large dimension problems empirically, which is remarkably inconsistent to the second hypothesis.

2.4 Major Contribution

Based on the above review and discussion, the contributions and differences of this chapter compared to the previous work are summarized as follows:

- The tool fitness landscape portrait (FLP) [28] was proposed over over 5 years ago. Besides the random search, it has not attracted much interest in analyzing the specific EA operators, such as crossover, mutation, etc. In our work, this effective analysis tool is implemented to analyze the expected behaviors of different kernel distribution sampling operators in low dimensional spaces. Based on the analysis, some valuable suggestions on designing appropriate sampling operator are presented theoretically.
- LSGO problems are considered as a difficult task in the optimization domain. In this work, an effective algorithm called the self-adaptive MUEDA is proposed for the large scale and complex optimization problems.

3 Fitness Landscape Analysis and Self-adaptive Mixed Probability Distribution Based Uni-Variate Estimation of Distribution Algorithm

In this section, the evolvability analysis of different kernel probability distributions is presented in terms of fitness landscape analysis on low dimensional landscapes. Based on the theoretical analysis, an effective self-adaptive heavy tail based sampling operator is proposed to strengthen the search ability of uni-variate EDAs.

3.1 Fitness Landscape Analysis

In fitness landscape analysis, the optimization problems can be expressed as a set of landscapes containing one or more optima [29, 34]. Based on the number of the optima, we also classify the landscapes into smooth or rugged problems. Evolution can thus be viewed as the movement of the population, represented by a set of points (genotypes), towards lower (fitter) areas of the landscape [28]. In order to explore the evolvability of different probability based operators, we adopt partial FLP technique to test the ability of sampling population on more promising regions by different operators on two typical landscapes.

FLP that was derived from comprehensive local view for sampling is adopted as an effective tool to analyze the evolvability of given operator on special landscapes. The metric selected for describing the evolvability of different operators is defined as follows:

$$flag(x) = \begin{cases} 1, & if\ f(x) \leq f(x_g) \\ 0, & otherwise. \end{cases} \tag{9}$$

$$E_{ev}(x_g) = \frac{\int_{-\infty}^{+\infty} f_{ope}(x) \cdot flag(x) dx}{\int_{-\infty}^{+\infty} f_{ope}(x) dx}, \tag{10}$$

where evolvability $E_{ev}(x_g)$ of solution x_g with fitness $f_{ope}(x_g)$ for the EA operator ope is directly tied to the probability of solution x_g not generating offspring of lower fitness.

Since the difficulty of searching global optimum related to the structure of the fitness landscape closely is now clear, two typical fitness landscapes (i.e. sphere landscape and rugged landscape), which include most of the existing landscapes, are chosen to evaluate the evolvability of Gaussian, Cauchy and mixed probability sampling operators.

3.1.1 Smooth Landscape

The evolvability of sampling operators with $\sigma = 1$ is tested on the region intercepted by $[-10, 10]$ for the sphere landscape shown in Fig. 3 (left). The evolvability metric curve line which is generated for each mean value moving from -10 to 10 by eq.(10) is shown in Fig. 3 (right). The sphere landscape is especially adopted to evaluate the evolvability of different sampling operators on smooth problems.

It can be observed from the evolvability curve that the Gaussian sampling operator provides the best evolvability for the whole region. Accordingly, the evolvability of Cauchy sampling operator decreases sharply while the global optimum is still far away, and this delays the convergence speed significantly. This may be the reason for the unsolved question in [9, 35] that Gaussian leads to a faster convergence speed than Cauchy for Sphere problem. Benefiting from the property of Gaussian distribution, the mixed sampling operator is equipped with similar evolvability as Gaussian sampling operator, which is remarkably better than Cauchy sampling operator by itself.

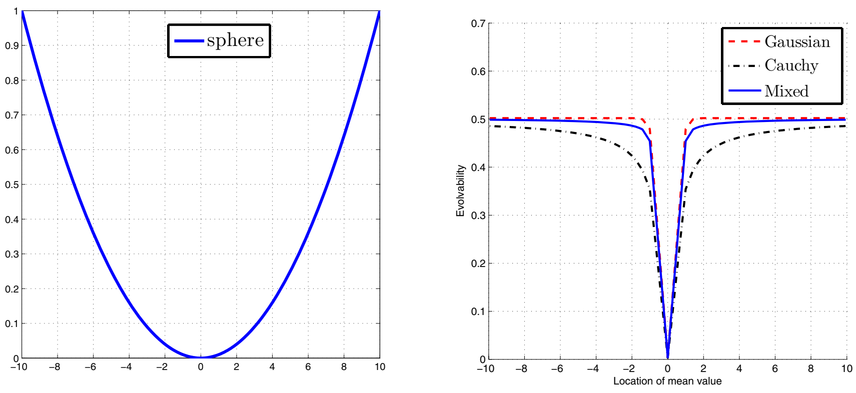


Fig. 3 Sphere landscape on the left and evolvability curve line graph on the right

3.1.2 Nonsmooth Landscape

It has been known that evolvability on rugged landscape mainly depends on the capability of escaping from the local optima. Consider the rugged landscape ($f(x) = \cos(x) - \frac{x}{20}$) shown in Fig. 4 (left). We intercept one local optimal basin [0, 2π] for analysis. Curve lines of evolvability generated by different sampling operators with mean value moving from 0 to 2π are also generated by eq. (10).

It is apparent that the evolvability of Gaussian sampling operator worsens sharply near the local optimum, which means that its evolvability shrinks quickly towards the local optimum. By comparison, Cauchy sampling operator demonstrates the best ability of escaping from the local optima. Therefore, the incorporation of low proportional Cauchy distribution highly improves the evolvability of mixed sampling operator. Therefore, the mixed sampling operator keeps a steady high probability of escaping from the local optima and thus, maintains the evolvability effectively.

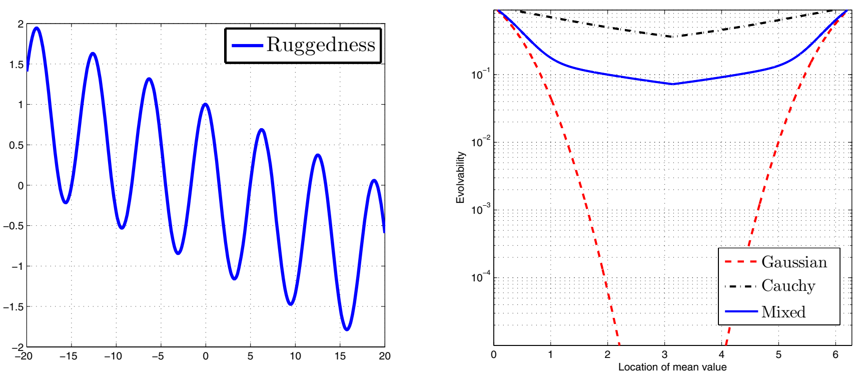


Fig. 4 Sphere landscape on the left and evolvability curve line graph on the right

Based on the above fitness landscape analysis, the mixed operator shows superior ability in both moving quickly towards the global optimum on smooth landscape and maintaining evolvability on rugged landscape globally. Compared to the classical Gaussian sampling operator, the sampling area of the mixed sampling operator exceeds the Gaussian model. In such case, the balance between learning and optimization is well handled. Therefore, it seems to be a promising way to develop more robust EDA by using mixed sampling operator.

3.2 Algorithm

Compared with the mutation operators of GA, ES and EP that mutate the individuals, the sampling operator of EDA mutates the distribution of the whole population in each generation. To strengthen the global search ability, a mixed Gaussian and Lévy distribution is adopted here, which is similar to [33].

Based on the above analysis, a new self-adaptive uni-variate EDA is proposed here. In order to reduce the complexity of conducting the probabilistic model, the uni-variate EDA whose variables are considered independently is adopted in our algorithm. Similar to UMDAc [12], the joint probabilistic distribution over a set of random variables $x = \{x_i\}$ where i = 1, 2... D for D dimensional space is defined as follows:

$$P(\mathbf{x}) = \prod_{i=1}^{D} P(x_i). \tag{11}$$

The probability distribution used to model each variable $P(x_i)$ is a single mixed Gaussian and Cauchy distribution. In contrast to iterated density estimation algorithm (IDEA) [1] developed by Bosman which requires computing all elements of covariance matrix to adapt an arbitrary Gaussian, MUEDA abandons adapting the non-diagonal elements in covariance matrix, which remarkably reduces the computational cost for LSGO problems. The updated rule for $P(x)$ is defined as follows:

$$P_{t+1}(\mathbf{x}) = (1-\theta)\cdot P_t(\mathbf{x}) + \theta \cdot P_t'(\mathbf{x}) \tag{12}$$

where $P_t'(\mathbf{x})$ is exactly the estimated joint probability distribution for the superior solutions of τ generation under the classical Gaussian distribution and θ stands for the learning intensity coefficient. Hence, the candidates for the $t+1$ generation are produced based on $P_t(\mathbf{x})$. Similar to UMDAc, the model for UMDAc is built by the current population only, which means that the learning coefficient θ is 1.

The details of mixed distribution and the strategy for generating each candidate are shown as:

$$randnum = rand;$$

$$P_m = \frac{10 \cdot randnum}{D}$$

$$N_L = \begin{cases} 0.9 \cdot N(0,1) + 0.1 \cdot C, & if\, D < 100 \\ (1-P_m)\cdot N(0,1) + P_m \cdot C, & otherwise, \end{cases} \tag{13}$$

$$X_{ji} = \overline{X}_i + \sqrt{\delta_i} \cdot N_L, \tag{14}$$

In eq.(13), $N(0,1)$ stands for the Gaussian distribution with mean 0 and standard deviation 1; C denotes the Cauchy distribution with $t = 1$; D is the dimensional size of the problem; *randnum* is generated randomly based on uniform distribution between $(0,1)$. The probability distribution for creating each offspring on different dimensional size is self-adapted as shown in eq.(13): a smaller mixed probability P_m is adopted for a larger dimensional size. In eq.(14), the ith element of jth individual is sampled under mixed Gaussian and Lévy distribution model N_L, where $\overline{X}_i$ stands for the ith element of the mean vector. Compared with UMDAc, whose sampling operator is denoted as $X_{ji} = \overline{X}_i + \sqrt{\delta_i} \cdot N(0,1)$, the standard Gaussian probability $N(0,1)$ is replaced with the mixed Gaussian and Lévy distribution N_L.

The flow of MUEDA is described as follows:

Input:

- LSGO problem;
- a termination condition;

Output: The solution with best fitness value.
Flow of MUEDA:
Step 0) Initialization:

- **Step 0.0)** Set population size $NP = (\log(D) - 3) \times 50$, selection size $N = (\log(D) - 3.5) \times 15$ and weight vector W.
- **Step 0.1)** Randomly initialize the population X_0.
- **Step 0.2)** Set $t = 0$.

Step 1) Reproduction and update:

- **Step 1.0) Reproduction:** Sample the new candidates by specific EA operator.
- **Step 1.1)** Set $t = t + 1$.
- **Step 1.2) Selection:** Select N best individuals by truncation strategy.
- **Step 1.3) Update:** Update the model with the selected individuals (eq.(6)).

Step 2) Standard Deviation Control Strategy (STDC) Then, if termination condition is met, go to step 3, else go to step 1.
Step 3) Terminate and output the GO.

It has been observed that one crucial problem that prevents uni-variate Gaussian based EDAs from biasing the search population towards a better region is that the standard deviation of some variables often shrinks to zero quickly while the global optimum is still far away [33]. In step 2, we introduce the standard deviation control strategy (STDC) to improve the exploration ability of uni-variate Gaussian based EDA. The main idea of STDC is to estimate a common threshold of standard deviations for all variables during the optimization process to control their shrinking speeds and therefore, to control the decreasing level of diversity dynamically. In more detail, the variables with lower standard deviation values than the corresponding thresholds will be forced to set their standard deviations to the corresponding thresholds. The weighted mean of the standard deviations of all variables is used as the threshold here. The details of STDC are shown as follows:

```
Pseudo code of STDC
begin STDC
  for j=1:D
    if $\delta(j) < W(j) \times \overline{\delta}$
      $\delta(j) = W(j) \times \overline{\delta}$
      /* $\overline{\delta}$ is the mean of variances */
      /* $W$ is a weight vector */
    end if
  end for
end
```

After analyzing a great deal of W values for each dimensionality setting, we calculate the value of W in an adaptive way shown as in eq. (15) for problems of different Ds:

$$W(i) = 0.55 - e^{lg(\frac{D}{10^5})} \tag{15}$$

$$\text{for} \quad i = 1, 2, ..., D$$

The metric W is determined by D only, which means a larger W is adopted for problems with lower D. It is observed that lower W value is adopted for larger D. Eq. (15) is used because the metric W is generally hard for the inexperienced users to choose.

Some existing works on standard deviation based triggering of variance scaling have been reported, such as adaptive variance scaling [8] and standard deviation ratio [3]. In these approaches, the scaling of standard deviation is determined by the performance of the latest generation. In contrast, the STDC strategy only enlarges the standard deviations of some variables under an adaptive threshold vector. Therefore, STDC is much simpler.

4 Experimental Study

4.1 Classical Function Optimization with Low Dimensionality

The purpose of this experiment is to compare mixed distribution based sampling operator of MUEDA with the Gaussian distribution based classical UMDAc. Moreover, the existing heavy tail distribution based algorithms, including FEP and FES, are adopted to provide the comparison results. The formal definitions of the test functions are summarized in Table 1. Function 1-6 are unimodal problems. Function 7-12 are multimodal problems, whose landscapes are full of local optima. In this chapter, the initialized population are randomly generated within the bounds for all experiments.

Table 1 Classical benchmark problems to be minimized

Num	Problems	Bounds	Objective function	Global location	GO
fun1	**Sphere**	[-100,100]	$f_1(x) = \sum_{i=1}^n x_i^2$	$x_i = 0$	0
fun2		[-10,10]	$f_2(x) = \sum_{i=1}^n \lvert x_i\rvert + \prod_{i=1}^n \lvert x_i\rvert$	$x_i = 0$	0
fun3	**Schwefel**	[-100,100]	$f_3(x) = \sum_{i=1}^n (\sum_{j=1}^i x_j^2)$	$x_i = 0$	0
fun4		[-100,100]	$f_4(x) = \max \lvert x_i\rvert$	$x_i = 0$	0
fun5	**Rochenbrock**	[-100,100]	$f_5(y) = \sum_{i=1}^D (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	$x_i = 1$	0
fun6	**Step**	[-30,30]	$f_6(x) = \sum_{i=1}^n \mid x_i + 0.5 \mid^2$	$x_i = -0.5$	0
fun7	**Noise**	[-1.28,1.28]	$f_7(x) = \sum_{i=1}^n ix_i^4 + random[0,1)$	$x_i = 0$	0
fun8	**Rastrigin**	[-5.12, 5.12]	$f_8(x) = \sum_{i=1}^n (x_i^2) - 10cos(2\pi x_i) + 10$	$x_i = 0$	0
fun9	**Ackley**	[-32,32]	$f_9(x) = -20 \cdot exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2})$ $+e - exp(\frac{1}{n}\sum_{i=1}^n cos(2\pi x_i)) + 20$	$x_i = 0$	0
fun10	**Griewangk**	[-600,600]	$f_{10}(x) = \frac{1}{4000}\sum_{i=1}^n x_i^2 + 1 - \prod_{i=1}^n cos(\frac{x_i}{\sqrt{i}})$	$x_i = 0$	0
fun11	**Penalized**	[-50, 50]	see appendix	$x_i = 1$	0
fun12	**Penalized**	[-50, 50]	see appendix	$x_i = 1$	0

Table 2 Experimental results for classical function optimization

function	MUEDA	UMDAc	FES	FEP	CEP
fun1	**2.8E-160 ± 1.5E-159**	2.0E-03 ± 4.9E-03	2.5E-04 ± 6.8E-05	5.7E-04 ± 1.3E-04	2.2E-04 ± 5.9E-04
fun2	**1.3E-124 ± 3.0E-124**	2.2E-01 ± 8.9E-01	6.0E-02 ± 9.6E-03	8.1E-03 ± 7.7E-04	2.6E-03 ± 1.7E-04
fun3	**0.0E+00 ± 0.0E+00**	1.3E+01 ± 4.6E+01	1.4E-03 ± 5.3E-04	1.6E-02 ± 1.4E-02	5.0E-02 ± 6.6E-02
fun4	**4.2E-72 ± 2.9E-71**	1.3E-03 ± 3.8E-03	5.5E-03 ± 6.5E-04	3.0E-01 ± 5.0E-01	2.0E+00 ± 1.2E+00
fun5	**9.0E-05 ± 2.2E-04**	1.7E+01 ± 5.5E+00	3.3E+01 ± 4.3E+01	5.1E+00 ± 5.9E+00	6.2E+00 ± 1.4E+01
fun6	**0.0E+00 ± 0.0E+00**	0.0E+00 ± 0.0E+00	0.0E+00 ± 0.0E+00	0.0E+00 ± 0.0E+00	5.8E+02 ± 1.1E+03
fun7	**3.0E-03 ± 5.8E-03**	1.1E-02 ± 1.8E-03	1.2E-02 ± 5.8E-03	7.6E-03 ± 2.6E-03	1.8E-03 ± 6.4E-03
fun8	2.1E+01 ± 4.7E+00	1.9E+00 ± 8.7E-03	1.6E-01 ± 3.3E-01	**4.6E-02 ± 1.2E-02**	8.9E+01 ± 2.3E+01
fun9	**4.4E-15 ± 0.0E+00**	1.9E-04 ± 9.6E-03	1.2E-02 ± 1.8E-03	1.8E-02 ± 2.1E-03	9.2E+00 ± 2.8E+00
fun10	**0.0E+00 ± 0.0E+00**	4.7E-03 ± 5.9E-03	3.7E-02 ± 5.0E-02	1.6E-02 ± 2.2E-02	8.6E-02 ± 1.2E-01
fun11	**1.6E-32 ± 0.0E+00**	1.9E-06 ± 3.2E-06	2.8E-06 ± 8.1E-07	9.2E-06 ± 3.6E-06	1.8E+00 ± 2.4E+00
fun12	**1.3E-32 ± 0.0E+00**	4.0E-05 ± 7.0E-05	4.7E-05 ± 1.5E-05	1.6E-04 ± 7.3E-05	1.4E+00 ± 3.7E+00

Statistical experimental results of 50 runs

For fair comparison, we set the parameters as in [40]. The following parameters are used in this experiment: 1) population size 100 for all algorithms; 2) maximum number of generations: 1500 for function 1, function 6, function 9, function 11 and function 12; 2000 for function 2 and function 10; 3000 for function 7, 5000 for function 3, function 4 and function 8. The statistical experimental results of 50

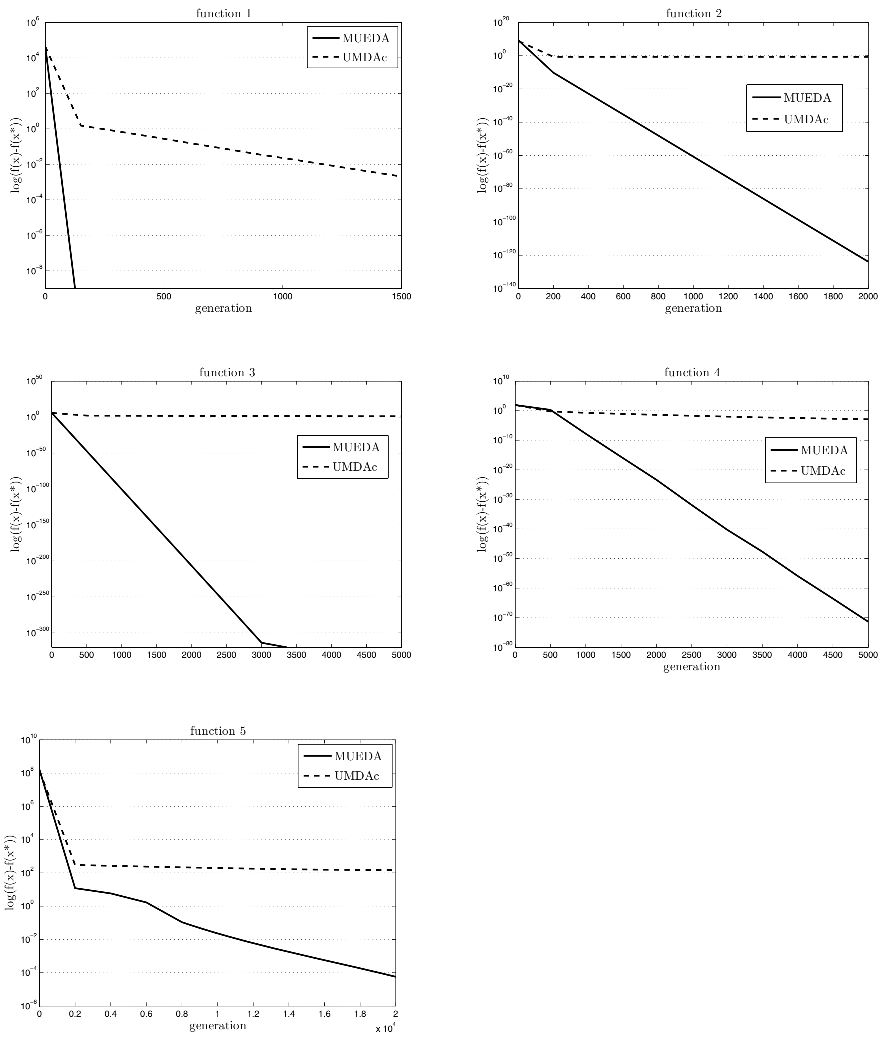


Fig. 5 Evolutionary curves of unimodal problems

independent runs are summarized in Table 2. Fig. 5 and 6 show the optimization curves for the unimodal problems and multimodal problems respectively.

Compared to UMDAc, it is apparent that MUEDA provides significantly better performance in terms of both convergence speed and accuracy of the final result for almost all of the test functions with 30 *D*. For the unimodal problems functions 1-6, MUEDA always provides the fastest convergence speed. It should be noted that, for function 5, a well-known hard test problem, the MUEDA approaches the true global optimum within the fixed number of generations, while the other algorithms are still struck at local optima after the final generation. The global search ability

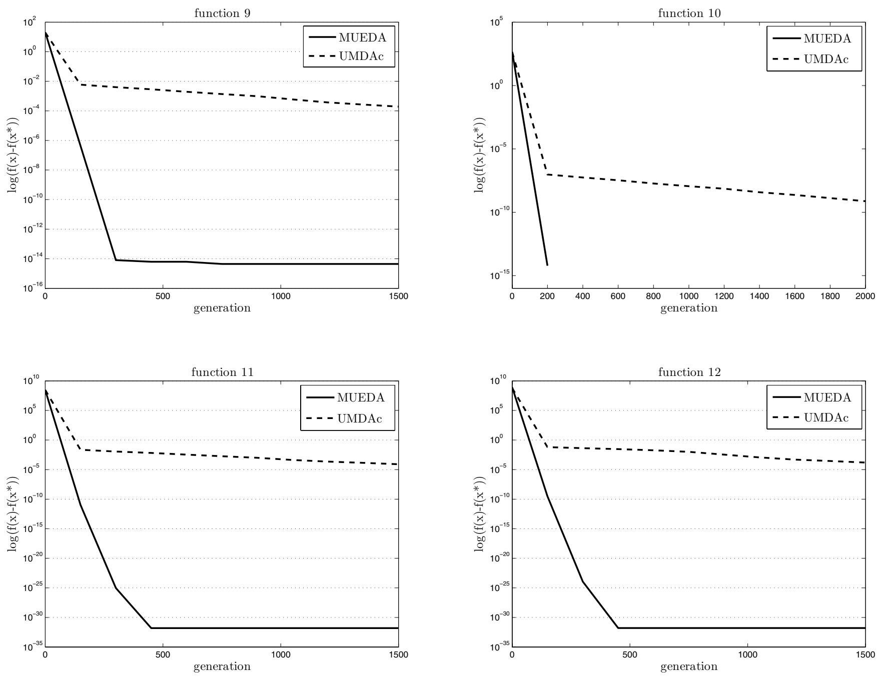


Fig. 6 Evolutionary curves of multimodal problems

of MUEDA is proven via experiments on multimodal test functions (function 7 - function 12). The accurate results (i.e. the error is lower than 10^{-8}) are achieved in all runs on function 9 - function 12. For the problem with noise incorporated function 7, MUEDA also outperforms the other approaches. For function 8, MUEDA cannot achieve satisfactory result, the reason for which will be discussed in the following section. Through this experiment, the advantages of MUEDA on both exploration and exploitation have been clearly demostrated.

4.2 Experiments on LSGO Problems

The benchmark set selected for this experiment consists of 6 test functions defined in [31]. Functions 1 - 3 are unimodal functions and Functions 4 - 6 are multimodal functions. To prevent exploitation of symmetry of the search space and of the typical zero value associated with the global optimum, the local optima of classical functions are shifted to a value different than zero and the function values of the global optima are non-zero. Without loss of generality, maximum fitness evaluation size (MFES) $500 \times D$ is adopted for function 1, 3, 5 and 6 and $5000 \times D$ for function 2 and 4. The details of standard benchmarks are defined in Table 3. The source codes for these functions are available from `http://nical.ustc.edu.cn/cec08ss.php` and `http://www3.ntu.edu.sg/home/EPNSugan/`.

Table 3 Standard benchmark problems of CEC08 to be minimized

Num	Problems	Bounds	Objective function	GO location	GO
funcec1	**Sphere**	[-100,100]	$f_1(y) = \sum_{i=1}^{D} y_i^2 + f_{bias1}$	$y_i = 0$	$f_{bias1} = -450$
funcec2	**Schwefel**	[-100,100]	$f_2(y) = \max(\lvert y_i \rvert) + f_{bias2}$	$y_i = 0$	$f_{bias2} = -450$
funcec3	**Rochenbrock**	[-100,100]	$f_3(y) = \sum_{i=1}^{D}(100(y_i^2 - y_{i+1})^2 + (y_i - 1)^2) + f_{bias3}$	$y_i = 1$	$f_{bias3} = 390$
funcec4	**Rastrigin**	[-5, 5]	$f_4(y) = \sum_{i=1}^{n}(y_i^2) - 10cos(2\pi y_i) + 10 + f_{bias4}$	$y_i = 0$	$f_{bias4} = -330$
funcec5	**Griewangk**	[-600,600]	$f_5(y) = \frac{1}{4000}\sum_{i=1}^{D} y_i^2 + 1 - \prod_{i=1}^{D} cos(\frac{y_i}{\sqrt{i}}) + f_{bias5}$	$y_i = 0$	$f_{bias5} = -180$
funcec6	**Ackley**	[-32,32]	$f_6(x) = -20 \cdot exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2})$ $+e - exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi y_i)) + 20 + f_{bias6}$	$y_i = 0$	$f_{bias6} = -140$

4.2.1 Comparison among Different Cauchy Proportions Based Algorithm

In order to illustrate the impacts of different heavy tail distributions, four Cauchy proportion are adopted for comparison: $\eta = 0$ (UMDAc), $\eta = 0.1$ (constant Cauchy), $\eta = 1$ (only Cauchy), and adaptive η (eq. (13) for MUEDA). For fair comparison, we choose parameters as consistently as possible. For external parameters, we set $NP = (\lvert \log(D) - 3 \rvert \times 50)$ and $N = (\lvert \log(D) - 3.5 \rvert \times 15)$. The statistical experimental results of over 50 runs are summarized in Table 4, in which the best result for each function is in boldface.

Table 4 Experimental results for 100 D cec08 function optimization

Algorithm	Metric	funcec1	funcec2	funcec3	funcec4	funcec5	funcec6
UMDAc	mean	3.81E+02	2.18E+01	2.96E+05	**2.20E+01**	3.24E+00	3.80E+00
$\eta = 0$	std	2.26E+02	3.08E+00	3.33E+05	**4.34E+00**	1.67E+00	1.14E+00
Mixed	mean	7.85E-01	1.72E+01	1.28E+07	8.03E+02	1.00E+00	3.52E+00
$\eta = 0.1$	std	5.53E-01	1.43E+00	2.01E+06	1.89E+01	6.04E-02	2.14E-01
Cauchy	mean	3.68E+05	1.52E+02	3.05E+11	1.91E+03	3.28E+03	2.13E+01
$\eta = 1$	std	2.13E+04	2.79E+00	3.75E+10	4.24E+01	1.70E+02	3.49E-02
MUEDA	mean	**6.82E-14**	**2.21E-13**	**2.89E+03**	1.04E+02	**1.28E-03**	**6.57E-12**
adaptive η	std	**2.32E-14**	**2.50E-14**	**3.69E+03**	2.49E+01	**3.63E-03**	**2.14E-11**

Statistical experimental results of 25 runs

It is apparent that Cauchy only ($\eta = 1$) distribution based algorithm fails in all problems. Therefore, excessive exploitation is not always beneficial for high dimensional problems. Generally speaking, MUEDA outperforms the other algorithms remarkably in most problems and is followed by UMDAc. For the hard task funcec 2, it is interesting to note that the result provided by MUEDA is very accurate while all of the other algorithms fail to get close to the global optima. The performances of all algorithms for Rochenbrock problem deteriorate sharply.

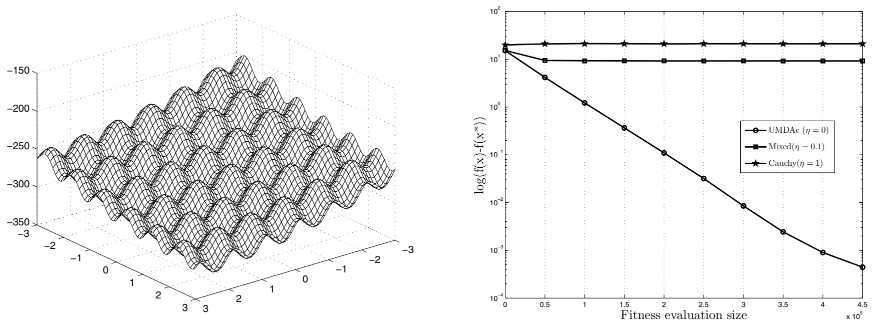


Fig. 7 Rastrigin problem: landscape on the left and convergence process comparison on the right

The 2-D landscape of Rastrigin problem and variance change process comparison are shown in Fig. 7. It is obvious that the landscape is full of local optima. The above experimental results have shown that UMDAc provides the best result for Rastrigin problem. However, the change curve of variance shows that the population of classical UMDAc just shrinks into a local optimum after a small fitness evaluation size. For this reason, the Gaussian distribution exhibits low evolvability and the final result is no longer reasonable. For the heavy tail distribution based operators, there is a high variance level throughout the search (which appears as a random search). Therefore, heavy tail distribution based operators seem more robust in these landscapes, although the final accuracy is unsatisfactory.

4.2.2 Comparison with other LSGO Evolutionary Algorithms

To benchmark MUEDA further, the comparison on larger dimensional (1000 D) problems is carried out. In some recent studies, some algorithms have reported the regular experimental results for the funcec functions. We only take the EA based algorithms into account. The algorithms are listed as follows:

- Efficient Population Utilization Strategy for Particle Swarm Optimizer (EPUS-PSO) [10]
- Unbiased Evolutionary Programming (UEP) [17]
- Self-Adaptive Differential Evolution algorithm (jDEdynNP-F) [5]
- Dynamic multi-swarm particle swarm optimizer (DMS-PSO) [43]
- Multilevel cooperative coevolution (MLCC) [38]
- Differential Evolution with Self-Adaptive cooperative co-evolution (DEwSAcc) [42]

The statistical analysis is shown in Table 5. In Table 6, the t-test results regarding $algorithm1$ vs $algorithm2$ are shown as '+', '-', 's+' and 's-' when $algorithm1$ is insignificantly better than, insignificantly worse than, significantly better than, and significantly worse than $algorithm2$ respectively. For unimodal problems, it is

Table 5 Experimental results for 1000 *D* cec08 function optimization

Algorithm	Metric	funcec1	funcec2	funcec3	funcec4	funcec5	funcec6
MLCC	mean	6.15E-01	1.09E+02	7.91E+03	**1.37E-10**	2.98E-02	2.02E-01
	std	6.58E-01	4.75E+00	1.12E+03	**3.37E-10**	2.26E-02	3.14E-01
EPUS-PSO	mean	3.46E+05	4.66E+01	3.77E+10	7.58E+03	3.09E+03	2.10E+01
	std	1.48E+04	4.00E-01	3.30E+09	1.51E+02	1.30E+02	1.62E-02
jDEdynNP-F	mean	2.92E+05	1.95E+01	7.10E+10	2.17E-04	2.53E+03	1.81E+01
	std	1.22E+04	2.25E+00	5.99E+09	4.06E-04	1.66E+02	1.81E-01
UEP	mean	7.78E+04	1.05E+02	4.43E+09	1.03E+04	7.93E+02	1.99E+01
	std	4.54E+03	7.07E+00	4.54E+08	9.94E+02	5.29E+01	1.19E-02
DEwSAcc	mean	5.68E+05	1.26E+02	2.16E+11	9.46E+03	5.19E+03	2.00E+01
	std	1.06E+05	4.48E+00	2.16E+11	7.15E+01	6.33E+02	3.33E-01
DMS-PSO	mean	**0.00E+00**	9.15E+01	2.94E+11	3.84E+03	**0.00E+00**	1.92E+01
	std	**0.00E+00**	7.14E-01	1.69E+11	1.71E+02	**0.00E+00**	4.06E-02
MUEDA	mean	3.30E-13 *II*	**9.45E-05**	**1.99E+03**	5.00E+03 *IV*	1.71E-13 *II*	**5.92E-08**
	std	2.54E-14 *II*	**9.66E-06**	**2.35E+02**	1.26E+02 *IV*	2.01E-14 *II*	**8.10E-08**

Statistical experimental results of 25 runs

Table 6 The t-test results of comparing MUEDA with the other algorithms

	funcec1	funcec2	funcec3	funcec4	funcec5	funcec6
MUEDA vs MLCC	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$
MUEDA vs EPUS-PSO	$s+$	$s+$	$s+$	$+$	$s+$	$s+$
MUEDA vs jDEdynNP-F	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$
MUEDA vs UEP	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
MUEDA vs DEwSAcc	$s+$	$s+$	$s+$	$+$	$s+$	$s+$
MUEDA vs DMS-PSO	$s-$	$s+$	$s+$	$-$	$s-$	$s+$

observed that MUEDA provides accurate results. For 1000 *D* funcec 1, DMS-PSO outperforms MUEDA in terms of accuracy. For the other algorithms, high-quality results cannot be generated due to the low computational cost. Funcec 2 with high *D* is an extremely hard task because the variables are non-separable. Although the fitness landscape is very smooth, the fact that only one variable with largest absolute value contributes to fitness value makes it almost impossible to be solved by classical approaches. It is evident that only MUEDA succeeds in converging to a solution with accuracy lower than 10^0 to the true global optimum. For the other algorithms, especially cooperative coevolution based ones, the search is stuck badly, which implies that they are not effective enough to deal with the non-separate LSGO problem. These results are not surprising, because only MUEDA considers modelling the rough structure of the search space. Furthermore, the implementation of mixed distribution makes breaking the common restriction possible.

The multimodal problems become much harder as the dimensionality increases to 1000. Funcec 3, whose variables are linked to a neighboring one (i.e. Rocenbrock problem), is the hardest task. For this problem, all algorithms fail in all runs. Another difficulty of this problem is that the valley towards the global optimum is very narrow and none of the algorithms could adaptively control the shrinking speed to suit this nonseparate problem. Amongst the compared algorithms, MUEDA demonstrates the most superior performance. For funcec 4, most of the algorithms fail to obtain GO except MLCC. Funcec 5 and 6 can be solved completely by MUEDA. The dominant convergence ability of MUEDA is strongly indicated by all these results.

4.2.3 Scalability Study

In order to study the scalability of MUEDA, we compare it with cooperative coevolution based LSGO approach Cooperative coevolution differential evolution II (DECC-II) on the 1500 dimensional problems. The reasons of selecting DECC-II in this experiment are as follows: 1) the cooperative coevolution appears to be a very promising method and has becomes very popular in LSGO domain [31]; 2) Compared with the classical cooperative coevolution based algorithms FEPCC [15] and CCGA [23], DECC-II has performed better on most problems [36]. In DECC-II, the variables selected for optimization in one iteration are chosen randomly with constant size 100. The other parameters chosen for experiment are the same as [36]. The comparison of evolution process between MUEDA and DECC-II on 1500 dimensional function optimization is shown in Fig. 8, and the comparison of final error between the best solution and true global optimum is shown in Table 7.

It is apparent that for unimodal function 1 and function 2, the proposed algorithm outperforms DECC-II not only in convergence speed, but also in the accuracy of results obtained. This is especially true for function 2 whose variables are linked. Similar conclusion can be drawn for multimodal functions globally, although the results achieved by DECC-II and proposed algorithm on function 4 (i.e. shift Rastrigin) are comparable. The reasons for this result have been analyzed in the first experiment. It is observed that the proposed algorithm works well on function 5 and 6 within such low computational cost even for 1500 D. In summary, the proposed algorithm provides a steady and accurate performance even for problems scaling to 1500 dimensions.

4.2.4 Efficiency Study

In the traditional analysis of [18, 19], the efficiency of the search method is tested on the problems with different dimensionality setting. Efficiency is defined on number of function evaluations needed to solve the problem [18]. In order to evaluate the efficiency of MUEDA, the results for multimodal problems - Ackley function (function 9 in Table 1) and Giewangk function (function 10 in Table 1) are presented in Table 8, 9 and Fig. 9. Moreover, the effective algorithms DE and PSO are also implemented in order to provide comparison results. For fair comparison, we

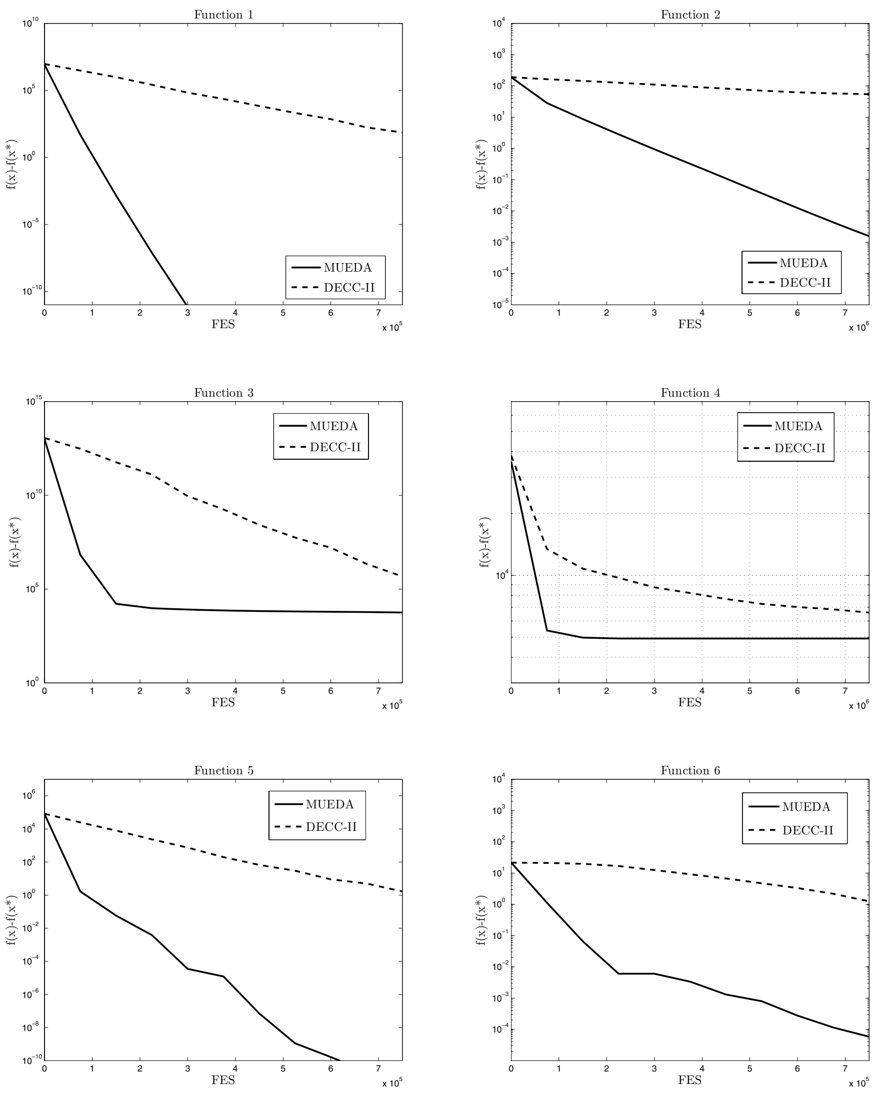


Fig. 8 Evolutionary curves of 1500 *D* problems

choose the same population size 200 for all algorithms and the most commonly used extensive parameters for respective algorithms.

It is apparent that the computational cost of MUEDA tends to increase linearly as the dimensionality arises. For DE and PSO, the efficiencies are much less. This is especially true when *D* is very large. It is evident that handling large scale optimization problems is a difficult task for evolutionary algorithms. Compared to breeder

Table 7 Experimental results for 1500 D cec08 function optimization

Algorithm	Metric	funcec1	funcec2	funcec3	funcec4	funcec5	funcec6
MUEDA	mean	1.83E-09	8.72E-05	5.83E+03	4.76E+03	1.71E-13	2.74E-05
	std	2.51E-09	2.37E-05	2.66E+03	1.41E+02	2.01E-14	2.01E-05
DECC-II	mean	2.24E+04	5.13E+01	1.44E+09	6.73E+03	7.46E+01	9.02E+00
	std	2.66E+03	3.01E+00	4.46E+08	8.60E+01	5.15E+00	3.80E-01

Statistical experimental results of 25 runs

Table 8 Efficiency test on Ackley function

Dimesionality	MUEDA	$166D + 23000$	$28D\ln(D)$	DE	PSO
100	40625	39600	128945	218802	282723
200	65076	56200	148353	fail	532617
300	86008	72800	159706	fail	761572
400	100458	89400	167761	fail	fail
500	113592	106000	174009	fail	fail
600	129110	122600	179114	fail	fail
700	141982	139200	183430	fail	fail
800	159629	155800	187169	fail	fail
900	174234	172400	190467	fail	fail
1000	189410	189000	193417	fail	fail

Termination criterion is 10^{-3}.

Table 9 Efficiency test on Giewangk function

Dimesionality	MUEDA	$156D + 24000$	$26D\ln(D)$	DE	PSO
100	40170	39600	119734	179030	249316
200	67432	55200	137756	fail	472276
300	92017	70800	148298	fail	fail
400	105875	86400	155778	fail	fail
500	119864	102000	161580	fail	fail
600	133825	117600	166320	fail	fail
700	142888	133200	170328	fail	fail
800	153660	148800	173800	fail	fail
900	166801	164400	176862	fail	fail
1000	180451	180000	179602	fail	fail

Termination criterion is 10^{-3}.

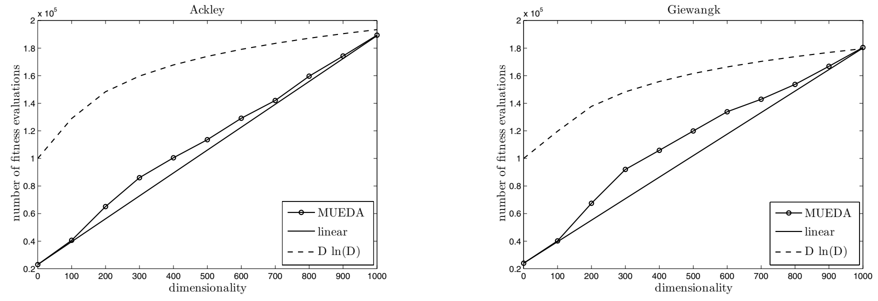


Fig. 9 Efficiency test on Ackley function on the left and Giewangk function on the right

GA scaling like $D\ln(D)$, MUEDA shows excellent efficiency, which scales linearly. Thus, the advantages of MUEDA on LSGO problems are easily observable.

5 Concluding Remarks

In this chapter, there are two major works, which can be summarized as:

- Via fitness landscape analysis, the expected behaviors and evolvability of Uni-variate EDAs with different kernel probability distributions based sampling operators are studied in low dimensional spaces. The evolvability change curve analysis reveals that mixing Gaussian with Cauchy distribution may be a promising way to strengthen the search ability.
- Based on the above analysis, a self-adaptive mixed distribution based uni-variate EDA named MUEDA is proposed for both LSGO problems. For the low dimensional problems, MUEDA provides excellent performance. Experimental evidence of large scale global function optimization is demonstrated to illustrate the merits and demerits of the proposed algorithm. Moreover, some scalability study is also carried out to evaluate MUEDA further.

In summary, this work aims at providing both expected and experimental analysis on improving the performances of uni-variate EDAs by designing a more effective sampling operator. As to the experimental results, MUEDA improves the performance of the uni-variate EDA significantly, and the proposed algorithm is good at both exploration and exploitation simultaneously. Particularly, the adaptive mixed distribution based sampling strategy could be simply incorporated into existing EDAs to accelerate the convergence speed and escape from the local optima.

6 Future Research Direction

There are still many issues that need to be urgently analyzed , of which the major ones are summarized as follows:

- Although many attempts have been carried out to analyze the characteristics of heavy tail distributions, the comprehensive mathematical properties of the heavy tail probability distributions are subject to suggestion.
- Even though the algorithm performed remarkably better than the classical algorithms, there are still some problems that cannot be completely solved by the proposed algorithm.
- In this chapter, the heavy tail distribution based sampling operator has shown good efficiency based not only on theoretical analysis, but also on the experimental analysis. More attempts are needed to prove the efficiency to the adaptive mixed strategy, such as incorporating the strategy to EP or ES.

Acknowledgements. This work was partially supported by the National Natural Science Foundation of China (Grants No. 60401015, U0835002), the Chinese Academy of Science Special Grant for Postgraduate Research, Innovation and Practice, IEEE Walter Kaplus Student Summer Research Grant, and Graduate Innovation Fund of University of Science and Technology of China.

We are deeply grateful to the reviewers for their invaluable comments which help us improve the quality of the chapter. The first author would like to thank Mr. Aimin Zhou for his warm-hearted help and the invaluable suggestions! Moreover, the authors would like to thank Mr. Xiaolei Chen and Ms. Xuexiao Lai, who helped us a lot in this chapter's preparation.

Appendix

Classical test function 11 and 12

u and y_i are defined as follows:

$$u(x_i,a,k,m)=\begin{cases} k(x_i-a)^m, & if\, x_i>a,\\ 0, & -a\le x_i\le a,\\ k(-x_i-a)^m, & if\, x_i<-a,\end{cases}$$

$$y_i=1+\frac{1}{4}(x_i+1).$$

f_{11} Generalized Penalized Functions

$$f_{11}=\frac{\pi}{30}\{10\sin^2+\sum_{i=1}^{2}9(y_i-1)^2\cdot[1+$$

$$10\sin^2(\pi y_{i+1}+(y_n-1)^2)]+\sum_{i=1}^{30}u(x_i,10,100,4)\}$$

f_{12} Generalized Penalized Functions

$$f_{12}=0.1\{\sin^2(\pi 3x_1)+\sum_{i=1}^{2}9(x_i-1)^2\cdot[1+$$

$$\sin^2(3\pi x_{i+1})]+(x_n-1)^2[1+\sin^2(2\pi x_30)]\}+\sum_{i=1}^{30}u(x_i,5,100,4)$$

A List of Terms and Definition

Heavy tail probability distribution - Different from Gaussian probability distribution whose variance can be denoted as a finite scalar, heavy tail probability distribution stands for a class of probability distributions with an infinite second moment that also yields a stable process.

Estimation of distribution algorithms (EDAs) - The methods that make use of the notion of modeling process as applied in statistics and machine learning domain to

Table 10 Abbreviations used in this chapter

abbreviation	full name
EDA	estimation of distribution algorithm
LSGO	large scale global optimization
MUEDA	mixed distribution based uni-variate EDA
EA	evolutionary algorithm
DE	differential evolution
PSO	particle swarm optimization
FEP	fast evolutionary programming
FES	fast evolution strategy
FSA	fast simulated annealing
LEP	Lévy mutation
LSEDA-gl	EDA with mixed Gaussian and Cauchy distribution for LSGO
FLP	fitness landscape portrait
GA	genetic algorithm
PBIL	population-based incremental learning algorithm
SHCLVND	Stochastic hill climbing by vectors of normal distributions
UMDAc	uni-variate marginal distribution algorithm
MIMIC	mutual information maximizing input clustering
EGNAee	estimation of Gaussian networks algorithm by edge exclusion
CEGDA	clustering and estimation of Gaussian distribution algorithm
IDEA	iterated density estimation algorithm
EP	evolutionary programming
CEP	classical EP
ES	evolution strategy
STDC	Standard Deviation Control Strategy
MFES	maximum fitness evaluation size

extract the important information to effectively build the space structure belong to EDAs.

Fitness landscape analysis - Fitness landscape analysis considers optimization problems as a set of landscapes containing one or more optima. Evolution can thus be viewed as the movement of the population, represented by a set of points (genotypes), towards lower (fitter) areas of the landscape.

Large scale global optimization - Large scale global optimization defines a suit of global optimization problems with more than 100 variables to be optimized.

Kernel probability distribution - Consider a probability with mean 0, symmetrical probability distribution and monotonously increasing; then this probability can be used as Kernel probability distribution for EDA.

References

1. Bosman, P.A.N.: Design and application of iterated density-estimation evolutionary algorithms. Ph.D. dissertation, Universiteit Utrecht, The Netherlands (2003)
2. Bosman, P.A.N., Grahl, J.: Matching inductive search bias and problem structure in continuous Estimation-of-Distribution Algorithms. Eur. J. Oper. Res. 185, 1246–1264 (2008)
3. Bosman, P.A.N., Grahl, J., Rothlauf, F.: SDR: A better trigger for adaptive variance scaling in normal EDAs. In: Proc. Genetic and Evolutionary Computation Conference (GECCO), London, United Kingdom, pp. 492–499 (2007)
4. Bosman, P.A.N., Thierens, D.: Expanding from discrete to continuous Estimation Of Distribution Algorithms: The IDEA. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 767–776. Springer, Heidelberg (2000)
5. Brest, J., Zamuda, A., Boskovic, B., Maucec, M.S., Zumer, V.: High-dimensional real-parameter optimization using self-adaptive Differential Evolution algorithm with population size reduction. In: Proc. IEEE Congress on Evolutionary Computation (CEC), Hong Kong, pp. 2032–2039 (2008)
6. Gnedenko, B., Kolmogorov, A.: Limit distribution for sums of independent random variables. Addison Wesley, Cambridge (1954)
7. Grahl, J., Bosman, P.A.N., Minner, S.: Convergence phases, variance trajectories, and runtime analysis of continuous EDAs. In: Proc. Genetic and Evolutionary Computation Conference (GECCO), pp. 516–522 (2007)
8. Grahl, J., Bosman, P.A.N., Rothlauf, F.: The correlation Ctriggered adaptive variance scaling idea. In: Proc. Genetic and Evolutionary Computation Conference (GECCO), pp. 397–404 (2006)
9. Hansen, N., Gemperle, F., Auger, A., Koumoutsakos, P.: When do heavy-tail distributions help? In: Parallel Problem Solving From Nature IX (PPSN IX), pp. 62–71. Springer, Heidelberg (2006)
10. Hsieh, S., Sun, T., Liu, C., Tsai, S.: Solving large scale global optimization using improved particle swarm optimizer. In: Proc. IEEE Congress on Evolutionary Computation (CEC), Hong Kong, pp. 1777–1784 (2008)

11. Larranga, P.: A review on estimation of distribution algorithms. In: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, ch. 3, pp. 57–100. Springer, Heidelberg (2001)
12. Larranga, P., Etxeberria, R., Lozano, J.A., Pena, J.M.: Optimization in continuous domains by learning and simulation of Gaussian networks. In: Proc. Genetic and Evolutionary Computation Conference Workshop Program, pp. 201–204 (2000)
13. Lee, C.Y., Yao, X.: Evolutionary programming using mutations based on the Lévy Probability Distribution. IEEE Trans. Evol. Comput. 8(1), 1–13 (2004)
14. Lévy, P.: Theorie de l'Addition des Veriables Aleatoires. Paris, France, Gauthier-Villars (1937)
15. Liu, Y., Yao, X., Zhao, Q., Higuchi, T.: Scaling up fast evolutionary porgramming with cooperative coevolution. In: Proc. IEEE Congress on Evolutionary Computation (CEC), pp. 1101–1108 (2001)
16. Lu, Q., Yao, X.: Clustering and learning Gaussian Distribution for continuous optimization. IEEE Trans. Systems Man and Cybernetics 35(2), 195–204 (2005)
17. MaCnish, C., Yao, X.: Direction matters in high-dimensional optimisation. In: Proc. IEEE Congress on Evolutionary Computation (CEC), Hong Kong, pp. 2377–2384 (2008)
18. Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive models for breeder genetic algorithm. Evol. Comput. 1(1), 25–49 (1993)
19. Mühlenbein, H., Schomisch, M., Born, J.: The parallel genetic algorithm as function optimizer. Parallel Comput. 17, 25–49 (1991)
20. Panait, L., Luke, S., Wiegand, R.: Biasing coevolutionary search for optimal multiagent behaviors. IEEE Trans. Evol. Comput. 10(6), 629–645 (2006)
21. Pardalos, P.: Large-Scale nonlinear optimization. In: Nonconvex Optimization and Its Applications. Springer, The Netherlands (2006)
22. Potter, A.M., Jong, K.A.D.: A cooperative co-evolutionary approach to function optimization. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994)
23. Potter, A.M., Jong, K.A.D.: Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evol. Comput. 8(1), 1–29 (2000)
24. Rudlof, S., Koppen, M.: Stochastic hill climbing with learning by vectors of normal distributions. In: Proc. First Online Workshop on Soft Computing (WSC1), Nagoya, Japan, pp. 60–70 (1996)
25. Sebag, M., Ducoulombier, A.: Extending population-based incremental learning to continuous search spaces. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 418–427. Springer, Heidelberg (1998)
26. Shi, Y., Teng, H., Li, Z.: Cooperative co-evolutionary differential evolution for function optimization. In: Proc. First International Conference on Natural Computation, pp. 1080–1088 (2005)
27. Sinha, N., Chakrabarti, R., Chattopadhyay, P.K.: Evolutionary programming techniques for economic load dispatch. IEEE Trans. Evol. Comput. 7(1), 83–94 (2003)
28. Smith, T., Husbands, P., Layzell, P., OShea, M.: Fitness landscapes and evolvability. Evol. Comput. 10(1), 1–34 (2002)
29. Stadler, P.F.: Fitness landscapes. In: Biological Evolution and Statistical Physics, pp. 187–207. Springer, Heidelberg (2002)
30. Szu, H., Hartley, R.: Fast simulated annealing. Phys. Lett. A 122(3,4), 157–162 (1987)

31. Tang, K., Yao, X., Suganthan, P.N., MacNish, C., Chen, Y.P., Chen, C.M., Yang, Z.Y.: Benchmark functions for the CEC 2008 special session and competition on large scale global optimization. Technical Report for IEEE Congress of Evolutionary Computation (CEC) (2008) (special issue)
32. Tseng, L., Chen, C.: Multiple trajectory search for large scale global optimization. In: Proc. IEEE Congress on Evolutionary Computation (CEC), pp. 3052–3059 (2008)
33. Wang, Y., Li, B.: A restart uni-variable estimation of distribution algorithms: Sampling under mixed Gaussian and Levy Probability Distribution. In: Proc. IEEE Congress on Evolutionary Computation (CEC), Hong Kong, pp. 3218–3225 (2008)
34. Wright, S.: The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In: Proc. 6th Congr. Genetics, vol. 1, p. 365 (1932)
35. Yang, Z., He, J.S., Yao, X.: Making a difference to differential evolution. In: Michalewicz, Z., Siarry, P. (eds.) Advances in Metaheuristics for Hard Optimization, pp. 397–414. Springer, Heidelberg (2007)
36. Yang, Z., Tang, K., Yao, X.: Differential evolution for high-dimensional function optimization. In: Proc. IEEE Congress on Evolutionary Computation (CEC), Singapore, pp. 3523–3530 (2007)
37. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Inf. Sci. 178(15), 2985–2999 (2008)
38. Yang, Z., Tang, K., Yao, X.: Multilevel cooperative coevolution for large scale optimization. In: Proc. IEEE Congress on Evolutionary Computation (CEC), Hong Kong, pp. 1663–1670 (2008)
39. Yao, X.: An overview of evolutionary computation. Chinese J. Adv. Software Res. 3(1), 12–29 (1996)
40. Yao, X., Liu, Y.: Fast evolutionary programming. In: Proc. Fifth Annual Conference on Evolutionary Programming (EP 1996), pp. 451–460. MIT Press, San Diego (1996)
41. Yao, X., Liu, Y.: Fast evolution strategies. Control and Cybern 26(3), 467–496 (1997)
42. Zamuda, A., Brest, J., Boskovic, B., Zumer, V.: Large scale global optimization using differential evolution with self adaptation and cooperative co-evolution. In: Proc. IEEE Congress on Evolutionary Computation (CEC), Hong Kong, pp. 3719–3726 (2008)
43. Zhao, S., Liang, J.J., Suganthan, P.N., Tasgetiren, M.F.: Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In: Proc. IEEE Congress on Evolutionary Computation (CEC), Hong Kong, pp. 3845–3852 (2008)

Differential Evolution with Fitness Diversity Self-adaptation

Ville Tirronen and Ferrante Neri

Abstract. This chapter proposes the integration of fitness diversity adaptation techniques within the parameter setting of Differential Evolution (DE). The scale factor and crossover rate are encoded within each genotype and self-adaptively updated during the evolution by means of a probabilistic criterion which takes into account the diversity properties of the entire population. The population size is also adaptively controlled by means of a novel technique based on a measurement of the fitness diversity. An extensive experimental setup has been implemented by including multivariate problems and hard to solve fitness landscapes. A comparison of the performance has been conducted by considering both standard DE and modern DE based algorithms, recently proposed in the literature. Available numerical results show that the proposed approach seems to be very promising for some fitness landscapes and still competitive with modern algorithms in other cases. In most cases analyzed the proposed self-adaptation is beneficial in terms of algorithmic performance and can be considered a useful tool for enhancing the performance of a DE scheme.

1 Introduction

Differential Evolution (DE, see [33], [29], and [6]) is a reliable and versatile function optimizer. DE, like most popular Evolutionary Algorithms (EAs), is a population based tool. DE, unlike other EAs, generates offspring by perturbing the solutions with a scaled difference of two randomly selected

Ville Tirronen · Ferrante Neri
Department of Mathematical Information Technology, Agora, University of Jyväskylä, P.O. Box 35 (Agora), FI-40014 University of Jyväskylä, Finland
e-mail: aleator@cc.jyu.fi,neferran@cc.jyu.fi

population vectors, instead of recombining the solutions by means of a probability function. In addition, DE employs a steady state logic which allows replacement of an individual only if the offspring outperforms its corresponding parent. Due to its algorithmic structure, over the optimization process DE generates a super-fit individual which leads the search until an individual with better performance is generated. Therefore, as highlighted in [16], a DE population can be subject to stagnation in such cases where no offspring individuals outperform the corresponding parents for a large number of generations. In order to avoid this undesired effect a proper parameter setting (two parameters in particular) is crucial.

Some empirical studies carried out in the literature, for example [28] and [16], give some hints as to how to perform such settings. A more advanced criterion based on the fitness values is given in [1]. Paper [36] proposed a dynamic population sizing strategy based on self-adaptation, and [19] proposed the employment of a fuzzy controller in order to perform setting of the parameters. In order to avoid execution of the parameter setting, in [30] an adaptive system based on the combined use of two learning strategies has been proposed. Paper [2] proposed a simple probabilistic scheme with a self-adaptive logic for updating parameter values during the evolution. Although the algorithm updates the parameters only by periodically refreshing them by means of random values, the self-adaptive logic carried out seems very robust and shows good performance under various fitness landscapes. This algorithmic philosophy has also been extended in the case of constrained optimization [3] and multi-objective problems [43]. Some hybrid approaches (also known as Memetic Algorithms) consisting of a DE framework and local search components have been proposed in the literature in order to avoid stagnation problems and, more generally, enhance the performance of the DE. Paper [35] proposed a hybrid algorithm based on a combination of DE and an estimation of distribution algorithm. This technique uses a probability model to detect promising areas and then focuses the search process on those regions. Recently, [26] proposed a Memetic Algorithm which integrates the local search hill-climber within variation operators of the DE; [31] proposed a heuristic technique, namely, opposition-based learning (OBL) for population initialization and also for generation jumping. In [42] a complex self-adaptation is proposed in order to significantly enhance the performance of a DE framework. Papers [7] and [8] propose the integration of a neighborhood logic within a DE scheme and the employment of multiple scale factors in the mutation operator. Papers [37] and [38] proposed a Memetic Differential Evolution (MDE) composed of a DE framework and two local search algorithms integrated within the framework and coordinated by a fitness diversity logic.

Fitness diversity adaptation has recently been applied with success in the context of Memetic Algorithms for performing coordination of local search algorithms and parameter setting. Several control parameters have been designed on the basis of the evolutionary framework and optimization problems under analysis (see [4], [23], [21], [24], and [22]). A measurement of the fitness diversity as the difference in performance between the fittest individual and other members of the population has been recently introduced for a MDE scheme [5].

This chapter aims to employ the fitness diversity logic and integrate it within each genotype of a DE population in order to enhance the algorithmic performance. The algorithm proposed in this chapter is composed of a DE structure employing a self-adaptation similar to the one proposed in [2], with a modified probabilistic criterion which is based on a novel measurement of the fitness diversity. In addition, the proposed algorithm contains an adaptive population size determined by variations in the fitness diversity.

Section 2 presents the state-of-the-art regarding DE and five recently published DE based algorithms. Section 3 describes the concept of fitness diversity and the reason for its employment in algorithmic adaptation. Section 4 gives a description of the proposed algorithm, namely Fitness Diversity Self-Adaptive Differential Evolution. Section 5 shows the numerical results. Finally, Section 6 gives the conclusion to this chapter.

2 Background: Differential Evolution Based Algorithms

This section describes the DE and five DE based algorithms recently proposed. In order to clarify the notation used throughout this chapter we refer to the minimization problem of an objective function $f(x)$, where x is a vector of n design variables in a decision space D.

2.1 Differential Evolution

According to its original definition given in [33], the DE consists of the following steps. An initial sampling of S_{pop} individuals is performed pseudo-randomly with a uniform distribution function within the decision space D. At each generation, for each individual x_i of the S_{pop}, three individuals x_r, x_s and x_t are pseudo-randomly extracted from the population. According to DE logic, a provisional offspring x'_{off} is generated by mutation as:

$$x'_{off} = x_t + F(x_r - x_s) \tag{1}$$

where $F \in [0, 1+[$ is a scale factor which controls the length of the exploration vector $(x_r - x_s)$ and thus determines how far from point x_i the offspring should be generated. The mutation scheme shown in eq. (1) is also known as DE/rand/1. Other variants of the mutation rule have been subsequently proposed in literature, see [30]:

- DE/best/1: $x'_{off} = x_{best} + F(x_s - x_t)$
- DE/cur-to-best/1: $x'_{off} = x_i + F(x_{best} - x_i) + F(x_s - x_t)$

- DE/best/2: $x'_{off} = x_{best} + F\,(x_s - x_t) + F\,(x_u - x_v)$
- DE/rand/2: $x'_{off} = x_r + F\,(x_s - x_t) + F\,(x_u - x_v)$

where x_{best} is the solution with the best performance among the individuals of the population, x_u and x_v are two additional pseudo-randomly selected individuals.

Then, to increase exploration, each gene of the new individual x'_{off} is switched with the corresponding gene of x_i with a uniform probability and the final offspring x_{off} is generated:

$$x_{off,j} = \begin{cases} x_{i,j} & \text{if} \quad rand\,(0,1) < CR \\ x'_{off,j} & otherwise \end{cases} \tag{2}$$

where $rand\,(0,1)$ is a random number between 0 and 1; j is the index of the gene under examination.

The resulting offspring x_{off} is evaluated and, according to a one-to-one spawning strategy, it replaces x_i if and only if $f(x_{off}) < f(x_i)$; otherwise no replacement occurs. For sake of clarity, the pseudo-code highlighting the working principles of the DE is shown in Fig. 1.

```
generate $S_{pop}$ individuals of the initial population pseudo-randomly;
while budget condition
  for $i = 1 : S_{pop}$
    compute $f\,(x_i)$;
  end-for
  for $i = 1 : S_{pop}$
    **mutation**
    select three individuals $x_r$, $x_s$, and $x_t$;
    compute $x'_{off} = x_t + F(x_r - x_s)$;
    **crossover**
    $x_{off} = x'_{off}$;
    for $j = 1 : n$
      generate $rand(0,1)$;
      if $rand(0,1) < CR$
        $x_{off,j} = x_{i,j}$;
      end-if
    end-for
    **selection**
    if $f\,(x_{off}) < f\,(x_i)$
      $x_i = x_{off}$;
    end-if
  end-for
end-while
```

Fig. 1 DE pseudocode

2.2 Parameter Setting in Differential Evolution

As highlighted in [16], due to its inner structure, the DE is subject to stagnation problems. Stagnation is that undesired effect which occurs when a population based algorithm does not converge to a solution (even suboptimal) and the population diversity is still high. In the case of the DE, stagnation occurs when the algorithm does not manage to improve upon any solution of its population for a prolonged amount of generations.

In order to avoid this undesired effect, the moving operators of the DE must be properly set. In other words, for successful functioning of the DE, a proper setting of the population size and parameters F and CR (see equations (1)and (2)) must be performed. The population size, analogous to the other Evolutionary Algorithms (EAs), if too small could cause premature convergence and if too large could cause stagnation (see [11]). A good value can be found by considering the dimensionality of the problem similar to what is commonly performed for the other EAs. A guideline is given in [34] where a setting of S_{pop} equal to ten times the dimensionality of the problem is proposed.

On the other hand, the setting of F and CR is neither an intuitive nor a straightforward task but is unfortunately crucial for guaranteeing the algorithmic functioning. Several studies have thus been proposed in literature. The study reported in [16] arrives at the conclusion, after an empirical analysis, that usage of $F = 1$ is not recommended, since according to a conjecture of the authors it leads to a significant decrease in explorative power. Analogously, the setting $CR = 1$ is also discouraged since it would dramatically decrease the amount of possible offspring solutions. In [34] and [17] the settings $F \in [0.5, 1]$ and $CR \in [0.8, 1]$ are recommended. In [17] the setting $F = CR = 0.9$ is chosen on the basis of discussion in [28]. The empirical analysis reported in [44] shows that in many cases the setting of $F \geq 0.6$ and $CR \geq 0.6$ leads to results having better performance.

Several studies, e.g., [13] and [18], highlight that an efficient parameter setting is very prone to problems (e.g., $F = 0.2$ could be a very efficient setting for a certain fitness landscape and completely inadequate for another problem). This result can be seen as a confirmation of the validity of the No Free Lunch Theorem [40] with reference to the DE schemes. In [1], a modified version of DE has been proposed. A system with two evolving populations has been proposed. The crossover rate CR has been set equal to 0.5 after an empirical study. Unlike CR, the value of F is adaptively updated at each generation by means of the following scheme:

$$F = \begin{cases} \max\left\{l_{\min}, 1 - \left|\frac{f_{\max}}{f_{\min}}\right|\right\} & \text{if } \left|\frac{f_{\max}}{f_{\min}}\right| < 1 \\ \max\left\{l_{\min}, 1 - \left|\frac{f_{\min}}{f_{\max}}\right|\right\} & \text{otherwise} \end{cases} \tag{3}$$

where $l_m = 0.4$ is the lower bound of F, f_{min} and f_{max} are the minimum and maximum fitness values over the individuals of the populations.

2.3 Self-adapting Control Parameters in Differential Evolution

In order to avoid the manual parameter setting of F and CR a simple and effective strategy has been proposed in [2]. This strategy is named Self-Adapting Control Parameters in Differential Evolution. The DE algorithm employing this strategy here is called Self-Adaptive Control Parameter Differential Evolution (SACPDE) and consists of the following.

With reference to Fig. 1, when the initial population is generated, two extra values between 0 and 1 are also generated per each individual. These values represent F and CR related to the individual under analysis. Each individual is thus composed (in a self-adaptive logic) of its genotype and its control parameters:

$$x_i = \langle x_{i,1}, x_{i,2}, ..., x_{i,j}, ...x_{i,n}, F_i, CR_i \rangle .$$

In accordance with a self-adaptive logic, see e.g., [32], the variation operations are preceded by the parameter update. More specifically when, at each generation, the i^{th} individual x_i is taken into account and three other individuals are extracted pseudo-randomly, its parameters F_i and CR_i are updated according to the following scheme:

$$F_i = \begin{cases} F_l + F_u rand_1, \text{ if } rand_2 < \tau_1 \\ F_i, \qquad\qquad \text{otherwise} \end{cases} \tag{4}$$

$$CR_i = \begin{cases} rand_3, \text{ if } rand_4 < \tau_2 \\ CR_i, \qquad \text{otherwise} \end{cases} \tag{5}$$

where $rand_j$, $j \in \{1, 2, 3, 4\}$, are uniform pseudo-random values between 0 and 1; τ_1 and τ_2 are constant values which represent the probabilities that parameters are updated, F_l and F_u are constant values which represent the minimum value that F could take and the maximum variable contribution to F, respectively. The newly calculated values of F_i and CR_i are then used for generating the offspring. The variation operators and selection scheme are identical to that of a standard DE (see section 2.1).

For sake of clarity, the pseudo-code highlighting the working principle of the SACPDE is given in Fig. 2.

2.4 Differential Evolution with Adaptive Crossover Local Search

In order to enhance performance of the DE, in [26] a memetic approach, called Differential Evolution with Adaptive Hill Climbing Simplex Crossover (DEahcSPX), has been proposed. The main idea is that a proper balance of the exploration abilities of the DE and the exploitation abilities of a Local

Searcher (LS) can lead to an algorithm with high performance. The proposed algorithm hybridizes the DE described in section 2.1 as an evolutionary framework and a LS is deterministically applied to the individual of the DE population with the best performance (in terms of fitness value).

The LS proposed for this hybridization is Simplex Crossover (SPX) [39]. More specifically, at each generation, that individual having the best fitness value, indicated here with x_b, is extracted and the LS described in Fig. 3 is applied. If the SPX succeeds in improving upon the starting solution, a replacement occurs according to a meta-Lamarckian logic [27].

It should be remarked that ϵ in Fig. 3 is a control parameter of the SPX which has been set equal to 1 in [26]. Finally, the DE framework employed is the standard DE described in Fig. 1.

```
generate $S_{pop}$ individuals of the initial population with related
parameters pseudo-randomly;
*while* budget condition
  *for* $i = 1 : S_{pop}$
    compute $f(x_i)$;
  *end-for*
  *for* $i = 1 : S_{pop}$
    **$F_i$ update**
    generate $rand_1$ and $rand_2$;
    $F_i = \begin{cases} F_l + F_u rand_1, & \text{if } rand_2 < \tau_1 \\ F_i, & \text{otherwise} \end{cases}$;
    **mutation**
    select three individuals $x_r$, $x_s$, and $x_t$;
    compute $x'_{off} = x_t + F_i(x_r - x_s)$;
    **$CR_i$ update**
    generate $rand_3$ and $rand_4$;
    $CR_i = \begin{cases} rand_3, & \text{if } rand_4 < \tau_2 \\ CR_i, & \text{otherwise} \end{cases}$
    **crossover**
    $x_{off} = x'_{off}$;
    *for* $j = 1 : n$
      generate $rand(0, 1)$;
      *if* $rand(0, 1) < CR_i$
        $x_{off,j} = x_{i,j}$;
      *end-if*
    *end-for*
    **selection**
    *if* $f(x_{off}) < f(x_i)$
      $x_i = x_{off}$;
    *end-if*
  *end-for*
*end-while*
```

Fig. 2 SACPDE pseudocode

```
while budget condition OR $f(C) \geq f(x_b)$
select pseudo-randomly $n_p - 1$ individuals from the DE population
compute the center of mass (including $x_b$):
$O = \frac{1}{n_p} \sum_{i=1}^{n_p} x_i$;
for $i = 1 : n_p - 1$
    $r_i = rand(0,1)^{\frac{1}{i+1}}$;
end-for
for $i = 1 : n_p$
    $y_i = O + \epsilon(x_i - O)$;
end-for
$C_1 = 0$;
for $i = 2 : n_p$
    $C_i = r_{i-1}\left(y_{i-1} - y_i + C_{i-1}\right)$;
end-for
$C = C_{n_p} + y_{n_p}$;
end-while
```

Fig. 3 SPX pseudocode

2.5 Opposition Based Differential Evolution

The Opposition Based Differential Evolution (OBDE), proposed in [31], employs logic of the opposition points in order to enhance exploration properties of the DE and test a wide portion of the decision space.

For a given point $x_i = \langle x_{i,1}, x_{i,2}, ..., x_{i,j}, ..., x_{i,n} \rangle$ belonging to a set $D = [a_1, b_1] \times [a_2, b_2] \times ... \times [a_j, b_j] \times ... \times [a_n, b_n]$ its opposition point is defined as: $\tilde{x}_i = \langle a_1 + b_1 - x_{i,1}, a_2 + b_2 - x_{i,2}, ..., a_j + b_j - x_{i,j}, ..., a_n + b_n - x_{i,n} \rangle$. The OBDE consists of a DE framework and two opposition based components: the first after the initial sampling and the second after the survivor selection scheme. While the first opposition based component is always applied after initialization, the second is activated by means of the probability j_r (jump rate). These opposition based components process a set of candidate solutions and generate their opposition points. They then merge the two sets of points (original and opposition) and select those points which have the best performance (as many as there are candidate solutions in the original set).

More specifically, when the initial sampling is pseudo-randomly performed, opposition points of the initial population are calculated and then half of these points (having the best fitness values) are selected to begin the optimization process. Analogously, at the end of each DE generation, when the population has been selected for the subsequent generation, the opposition based component is applied.

For sake of clarity the pseudo-code describing the functioning of the OBDE is shown in Fig. 4.

```
generate $S_{pop}$ individuals of the initial population pseudo-randomly;
while budget condition
  for $i = 1 : S_{pop}$
    compute $f(x_i)$;
  end-for
  **opposition based component**
  for $i = 1 : S_{pop}$
    for $j = 1 : n$
      compute $\tilde{x}_{i,j} = a_j + b_j - x_{i,j}$;
    end-for
    compute $f(\tilde{x}_i)$;
  end-for
  merge original population and opposition based points;
  select the $S_{pop}$ solutions which have the best performance;
  for $i = 1 : S_{pop}$
    **mutation**
    select three individuals $x_r$, $x_s$, and $x_t$;
    compute $x'_{off} = x_t + F(x_r - x_s)$;
    **crossover**
    $x_{off} = x'_{off}$;
    for $j = 1 : n$
      generate $rand(0, 1)$;
      if $rand(0, 1) < CR$
        $x_{off,j} = x_{i,j}$;
      end-if
    end-for
    **selection**
    if $f(x_{off}) < f(x_i)$
      $x_i = x_{off}$;
    end-if
  end-for
  generate $rand(0, 1)$;
  if $rand(0, 1) < j_r$
    **opposition based component**
    for $i = 1 : S_{pop}$
      for $j = 1 : n$
        compute $\tilde{x}_{i,j} = a_j + b_j - x_{i,j}$;
      end-for
      compute $f(\tilde{x}_i)$;
    end-for
    merge original population and opposition based points;
    select the $S_{pop}$ solutions which have the best performance;
  end-if
end-while
```

Fig. 4 OBDE pseudocode

2.6 Differential Evolution with Global and Local Neighborhoods

The Differential Evolution with Global and Local Neighborhoods (DEGL) modifies the mutation operation in the DE, explained in Subsection 2.1, by defining a neighborhood as a portion of the population identified by a radius k, see [7] and [8]. More specifically, individuals of the population are pseudo-randomly sorted and each individual is characterized by a position index i. The neighborhood of the i^{th} individual x_i is given by those individuals $x_{i-k}, ..., x_i, ..., x_{i+k}$.

The concept of neighborhood is used during the mutation operation since the provisional offspring x'_{off} is acquired through the combination of two contributions, the first contribution is given by the neighborhood individuals and the second by the whole population. Thus, in order to perform the mutation, for an individual x_i, the local contribution is calculated as:

$$L_i = x_i + \alpha \left(x_{n-best} - x_i \right) + \beta \left(x_p - x_q \right) \tag{6}$$

where x_{best} is the individual having best performance in the neighborhood, x_p and x_q and two individuals pseudo-randomly selected from the neighborhood. Values α and β are two constant which have a similar role to that of the scale factor F, see eq. (1). The global contribution is given by:

$$G_i = x_i + \alpha \left(x_{p-best} - x_i \right) + \beta \left(x_r - x_s \right) \tag{7}$$

where x_{p_best} is that individual with the best performance out of the entire population, x_r and x_s are two individuals pseudo-randomly selected from the population. The two contributions are then combined by means of:

$$x'_{off} = wG_i + (1 - w) L_i \tag{8}$$

where w_i is a weight factor to be set between 0 and 1.

Regarding the parameter setting, as suggested in [7], it has been set $\alpha = \beta$ equal to constant value. Also the neighborhood radius k has been set as a constant. On the contrary, the weight factor w is updated according to:

$$w = w_{\min} + \left(w_{\max} - w_{\min} \right) \frac{g}{g_{\max}} \tag{9}$$

where $w_{\min}$ and $w_{\max}$ are the lower and upper bounds of the weight factor, respectively. The indexes g and $g_{\max}$ denote the current generation index and the maximum amount of generations, respectively. In other words, at the beginning of the optimization process ($g = 0$) the weight factor is set equal to w_{min} and subsequently linearly varies over time. At the end of the optimization process, the weight factor takes the value $w_{\max}$. The crossover and replacement occur as with a plain DE, see Subsection 2.1.

2.7 Self-adaptive Differential Evolution with Neighborhood Search

The Self-Adaptive Differential Evolution with Neighborhood Search (SaNSDE) [42] is a combination of two different algorithms: Differential Evolution with Neighborhood Search introduced in [41] and Self-Adaptive Differential Evolution (SADE) introduced in [30].

The SaNSDE modifies the DE in the following way. When the mutation is performed, the SaNSDE alternates, with a probabilistic scheme, two mutation strategies. The first strategy is the so called DE/rand/1 shown in eq. (1) while the second one is the DE/current to best/2 characterized by the formula:

$$x'_{off} = x_i + F\left(x_{p-best} - x_i\right) + F\left(x_r - x_s\right), \tag{10}$$

where x_{p_best} is the individual displaying best performance in the population, x_r and x_s are two individuals pseudo-randomly selected.

The probability of selecting the mutation strategy DE/rand/1 is initially 0.5 and subsequently updated (every 50 generations) by:

$$\rho = \frac{s_1(s_2+f_2)}{s_2(s_1+f_1)+s_1(s_2+f_2)}, \tag{11}$$

where s_1, s_2, f_1, f_2 are respectively the number of successful and unsuccessful attempts at generating offspring by the two mutation strategy under investigation. More specifically, s_1 is the number of times the DE/rand/1 led to an offspring outperforming the corresponding parent, s_2 is the number of times the DE/current to best/2 led to an offspring outperforming the corresponding parent, f_1 is the number of unsuccessful attempts at generating an offspring by DE/rand/1 and f_2 is the number of unsuccessful attempts at generating an offspring by DE/current to best/2. The probability of selecting the mutation strategy DE/current to best/2 is given by $1-\rho$.

A self adaptation of the scale factor F is also performed. For each individual a scale factor F_i is associated. Each scale factor is updated according to:

$$F_i = \begin{cases} N_i\left(0.5, 0.3\right) & \text{if} \quad rand < F_\rho \\ \delta_i & \text{otherwise} \end{cases} \tag{12}$$

where $N_i\left(0.5, 0.3\right)$ is a number sampled normal distribution with mean value of 0.5 and standard deviation equal to 0.3, $rand$ is a pseudo-random number sampled from a uniform distribution, δ_i is a random sample from the Cauchy distribution with a scale parameter equal to 1, F_ρ is a probability constructed with the same logic as eq. (11) but related to the success of the scale factor.

In order to perform the crossover, the SaNSDE employs the weighted crossover rate self-adaptation. With each individual a crossover rate CR_i

is associated. Every five generations new CR_i values for each individual are generated for the new population by means of the following equation:

$$CR_i = N\left(CRm, 0.1\right). \tag{13}$$

The value CRm is initially set equal to 0.5 and then updated every 25 generations according to:

$$CRm = \sum_{k=1}^{|CRrec|} w_k CRrec\left(k\right) \tag{14}$$

where $CRrec$ is a vector containing the CR_i values which contributed to the generation of a successful offspring (individual which outperform its parent x_i). Each weight factor w_k is calculated as:

$$w_k = \frac{\Delta f_{rec}\left(k\right)}{\sum_{k=1}^{|\Delta f_{rec}|} \Delta f_{rec}\left(k\right)} \tag{15}$$

where $\Delta f_{rec}\left(k\right)$ is the enhancement in that fitness value corresponding to the application of $CRrec\left(k\right)$.

All remaining operations are performed as shown in Subsection 2.1.

3 Fitness Diversity Adaptation in Evolutionary Algorithms

As mentioned in [10] Exploration and Exploitation are two cornerstones in EAs and a proper balance of these properties seems to be the basic principle for algorithmic success. Unfortunately, every optimization problem has its own features, and thus, this balance must be designed taking into account the class of fitness landscapes which should be handled. In addition, as highlighted in [11], an EA is implicitly dynamic. Thus, in order to design a highly efficient algorithm, the explorative/exploitative pressure should vary over time and adapt to the demands of the optimization process while the search is performed.

A properly designed optimization algorithm should, in principle, be able to explore the decision space and detect the optimal basin of attraction, eventually converging to the global optimum. Unfortunately, the location of the global optimum and optimal basin of attraction is in general unknown a priori and it is impossible to even know during the optimization process whether one candidate solution has fallen within the optimal basin of attraction.

Nevertheless, the behavior of the algorithm (on a fitness landscape) can be observed and indirectly measured: if the population contains a high variety of genotypes (highly diverse), the algorithm explores the decision space and hopefully detects new promising solutions; if the population contains a low variety of genotypes, the algorithm exploits a search direction and converges to a solution. The concept of diversity can then be used to monitor algorithmic behavior and prevent undesired stagnation and premature convergence situations. By means of a parameter adjustment, the algorithm can increase exploitation if the algorithm seems to be too explorative (checking a huge amount of solutions without improving upon the current best) or conversely increase exploration if the algorithm tends to lose the diversity and converge to a suboptimal solution (see for example [14], [11], and [9]). This logic has been implemented in various ways during the latest decades and has been widely used for local search coordination in Memetic Algorithms since their early definition in [20].

A not so trivial problem is how to efficiently measure the population diversity and how to make use of this information for actually obtaining an algorithm which adequately responds to variations. Measurement of the genotypical diversity presents the problem that a comparison among vectors of numbers is somehow required (e.g., distance between pairs of vectors) and a table expressing this comparison is generated; this operation can be computationally very expensive if the fitness is highly multi-variate.

An indirect way to measure the population diversity is through its fitness values. Measurement of the fitness diversity requires handling of only one vector of numbers which is composed of the fitness values of each individual of the population. On the other hand, the presence of plateaus and saddle points can give an inaccurate estimation of the distribution of points in the decision space. Fortunately, this limitation of the fitness diversity schemes is not very severe when this diversity is used for making an adaptive choice on the explorative/ exploitative countermeasures. If fitness diversity is high, the solutions are somehow spread out in the decision space, the algorithm is exploring new search directions and a higher exploitative pressure can help in choosing the most promising search direction; if the fitness diversity is low, either the algorithm is converging towards a solution or the entire population is contained in a plateau or a saddle; in both cases an increase in exploration is required in order to detect new promising solutions outside the current basin of attraction or plateau (or saddle).

The fitness diversity can thus be efficiently employed to measure the state of the algorithm and apply proper countermeasures to avoid stagnation and premature convergence, resulting in a high performance algorithm. In order to use information related to the fitness diversity, an index which estimates the diversity must be defined. In literature, several proposals have been made. Indicating with f_{best}, f_{avg} and f_{worst} respectively the best, average and worst fitness over the individuals of the population, in [4] and [23] the following index has been proposed:

$$\xi = \min\left\{\left|\frac{f_{best} - f_{avg}}{f_{best}}\right|, 1\right\} \tag{16}$$

In [22] and [24] the following parameter is given:

$$\psi = 1 - \left| \frac{f_{avg} - f_{best}}{f_{worst} - f_{best}} \right| \tag{17}$$

In [37] the fitness diversity has been measured by means of the following index:

$$\nu = \min \left\{ \frac{\sigma_f}{|f_{avg}|}, 1 \right\}, \tag{18}$$

where σ_f is the standard deviation over the fitness values of individuals of the population.

In [5] the following parameter is used:

$$\chi = \frac{|f_{best} - f_{avg}|}{max\, |f_{best} - f_{avg}|_k} \tag{19}$$

where f_{best} and f_{avg} are the fitness values of, respectively, the best and average individuals of the population. $max\, |f_{best} - f_{avg}|_k$ is the maximum difference observed (e.g., at the k^{th} generation), beginning from the start of the optimization process.

It must be noted that the four control parameters shown above can take values in the interval $[0, 1]$. These limit conditions 0 and 1 mean no fitness diversity and high fitness diversity respectively. Although all the above mentioned parameters measure fitness diversity of the population, the way this diversity is measured and scored differ much according to the index employed.

Although an in depth analysis of the structure of fitness diversity indexes is beyond the scope of this chapter, it is interesting to note that these parameters can be seen as an answer to four distinct questions related to the algorithms' state during the run. More specifically, ξ can be seen as the answer to the question "How close is the average fitness to the best one?"; ψ is the answer to the question "If we sort all fitness values over a line, which position is occupied by the average fitness?"; ν is the answer to the question "How sparse are the fitness values within the population?"; χ is the answer to the question "How much better is the best individual than the average fitness of the population with respect to the history of the optimization process?".

4 Fitness Diversity Self-adaptive Differential Evolution

This chapter aims to propose an efficient modification of DE which includes within its structure the fitness diversity logic in order to control explorative/exploitative pressure and thus improve upon the DE performance. The proposed algorithm, namely Fitness Diversity Self-Adaptive Differential Evolution (FDSADE) consists of the following steps.

4.1 Initial Sampling and Encoding of the Solutions

An initial sampling of S_{pop} individuals is executed pseudo-randomly with a uniform distribution function. As for the algorithm in [2], each individual x_i is composed of its genotype over the decision space D and its control parameters F_i and CR_i pseudo-randomly sampled between 0 and 1:

$$x_i = \langle x_{i,1}, x_{i,2}, ..., x_{i,j}, ...x_{i,n}, F_i, CR_i \rangle .$$

The fitness of each individual is calculated and the solutions are sorted according to their fitness values from best to worst.

4.2 Fitness Diversity Self-adaptation

At each generation the following fitness diversity index is calculated:

$$\phi = \frac{\sigma_f}{|f_{worst} - f_{best}|} \tag{20}$$

where σ_f is the standard deviation of fitness values over individuals of the populations, f_{worst} and f_{best} are the worst and best fitness values, respectively, of the population individuals.

Analogous to the other fitness diversity indexes listed in section 3, ϕ varies between 0 and 1. When the fitness diversity is high, $\phi \approx 1$; on the contrary when the fitness diversity is low, $\phi \approx 0$. The index ϕ can be seen as a combination of ν in formula (18) and ψ in formula (17) because it represents the distribution of fitness values over individuals of the population with respect to its range of variability. In other words, ϕ is the answer to the question "How sparse are the fitness values with respect to the range of fitness variability at the current generation?". Employment of the standard deviation in the numerator in formula (20) is due to the fact that a DE framework tends to generate an individual with performance significantly above the average (see [37] and [5]) and efficiently continues optimization for several generations. In this sense, an estimation of the fitness diversity of a DE population by means of the difference between best and average fitness values can return a misleading result and each value must be taken into account. Regarding the denominator in formula (20), a normalization to the range of variability of the current population makes the index co-domain invariant (unlike ν in formula (18)) and its estimation is not affected, for example by adding an offset to the fitness function. Thus, the index ϕ can be successfully employed, within a DE framework, on problems of various kinds.

The control parameter F_i is then updated according to the scheme:

$$F_i = \begin{cases} F_l + F_u rand_1, & \text{if } rand_2 < K(1-\phi) \\ F_i, & \text{otherwise} \end{cases} \tag{21}$$

where $rand_1$ and $rand_2$ are pseudo-random numbers generated by means of the uniform distribution, F_l and F_u are the same constant values shown in formula (4) for the SACPDE.

Analogously, the control parameter CR_i is updated according to the scheme:

$$CR_i = \begin{cases} rand_3, \text{ if } rand_4 < K(1-\phi) \\ CR_i, \qquad \text{otherwise} \end{cases} \tag{22}$$

where $rand_3$ and $rand_4$ are pseudo-random numbers generated by means of the uniform distribution. For both equations (21) and (22), the constant value K represents the maximum update probability of the parameters.

In other words, the proposed algorithm hybridizes the self-adaptive scheme proposed in [2] with the fitness diversity logic in order to obtain a high performance DE algorithm. The main idea behind the proposed self-adaptation is that in high diversity conditions ($\phi \approx 1$), the solutions x_i should tend to keep the control parameters F_i and CR_i unvaried and thus exploit the current potential search. On the contrary, when the diversity condition is low ($\phi \approx 0$), the algorithm should try to better explore the decision space by frequently changing intensity of the mutation move F_i and updating the recombination rate CR_i. The proposed logic is thus similar to the fitness diversity based activation of a local search in a MA (see e.g., [22]): if the algorithms need to exploit the genotype no changes to the evolutionary framework are necessary, if the algorithm has poor diversity a change in the exploratory logic and exploration of new search perspectives are recommended (see [15]).

4.3 Recombination Operators

Recombination operations, i.e., mutation and crossover, occur in the FD-SADE, similar to operations performed in the standard DE. For each solution x_i, three individuals x_r, x_s and x_t are pseudo-randomly extracted from the population and a provisional offspring x'_{off} is generated by mutation:

$$x'_{off} = x_t + F_i(x_r - x_s) \tag{23}$$

where F_i is the scale factor corresponding to solution x_i. Each gene of the new individual x'_{off} is then switched with the corresponding gene of x_i with the corresponding uniform probability CR_i and the final offspring x_{off} is generated:

$$x_{off,j} = \begin{cases} x_{i,j} \qquad \text{if} \qquad rand(0,1) < CR_i \\ x'_{off,j} \; otherwise \end{cases} \tag{24}$$

4.4 Adaptive Population Size and Selection

When S_{pop} offspring individuals are generated, the offspring x_{off} is evaluated and, according to standard DE logic, replaces x_i if and only if $f(x_{off}) < f(x_i)$; otherwise no replacement occurs.

```
generate $S_{pop}$ individuals of the initial population with related
parameters pseudo-randomly;
while budget condition
  for $i = 1 : S_{pop}$
    compute $f(x_i)$;
  end-for
  sort the population;
  compute $\phi = \frac{\sigma_f}{|f_{worst} - f_{best}|}$;
  for $i = 1 : S_{pop}$
    **$F_i$ update**
    generate $rand_1$ and $rand_2$;
    $F_i = \begin{cases} F_l + F_u rand_1, & \text{if } rand_2 < K(1-\phi) \\ F_i, & \text{otherwise} \end{cases}$;
    **mutation**
    select three individuals $x_r$, $x_s$, and $x_t$;
    compute $x'_{off} = x_t + F_i(x_r - x_s)$;
    **$CR_i$ update**
    generate $rand_3$ and $rand_4$;
    $CR_i = \begin{cases} rand_3, & \text{if } rand_4 < K(1-\phi) \\ CR_i, & \text{otherwise} \end{cases}$
    **crossover**
    $x_{off} = x'_{off}$;
    for $j = 1 : n$
      generate $rand(0,1)$;
      if $rand(0,1) < CR_i$
        $x_{off,j} = x_{i,j}$;
      end-if
    end-for
    **selection**
    if $f(x_{off}) < f(x_i)$
      $x_i = x_{off}$;
    end-if
  end-for
  **population size adjustment**
  compute $\phi = \frac{\sigma_f}{|f_{worst} - f_{best}|}$;
  $S_{pop}^{old} = S_{pop}$;
  compute $S_{pop} = S_{pop}^{f} + S_{pop}^{v}(1-\phi)$;
  if $S_{pop} < S_{pop}^{old}$
    select the $S_{pop}$ individuals with the best performance;
  else duplicate in the population $S_{pop} - S_{pop}^{old}$
      solutions with the best performance;
  end-if;
end-while
```

Fig. 5 FDSADE pseudocode

The parameter ϕ is then calculated as shown in formula (20) and population size is adjusted for the subsequent generation according to the equation:

$$S_{pop} = S_{pop}^{f} + S_{pop}^{v}\left(1 - \phi\right) \tag{25}$$

where S_{pop}^{f} stands for fixed population size and is the minimum value of the population size, S_{pop}^{v} stands for variable population size and is the maximum value of the variable contribution of the population size. The new value of S_{pop} is then employed for the subsequent generation: if population size is reduced, only those S_{pop} individuals having the best performance are considered for the next steps; if the population is expanded, the best solutions are duplicated in order to have a population composed of S_{pop} individuals.

The meaning of this variable population size is that in high diversity conditions the algorithm should shrink the population and exploit the available genotypes, in low diversity conditions the algorithm should enlarge the population and through the recombination mechanism focus on detecting new promising search directions. In this way the algorithm should be able to adapt to necessities of the fitness landscape and dynamically balance the explorative/exploitative necessities. Similar approaches have been proposed in the literature, e.g., [4], [23], [21] and [24].

For sake of clarity the pseudo-code illustrating the working principles of the FDSADE is given in Fig. 5.

5 Numerical Results

The FDSADE has been tested on a set of various test problems and compared with a plain DE [33], SACPDE [2], DEahcSPX [26], OBDE [31], the DEGL [8], and SaNSDE [42].

- The DE has been run with $F = 0.7$ and $CR = 0.7$ in accordance to the suggestions given in [44].
- The SACPDE has been run, with reference to the formulas (4) and (5), with $F_l = 0.1$, $F_u = 0.9$, $\tau_1 = 0.1$, and $\tau_2 = 0.1$ as shown in [2].
- The DEahcSPX has been run with $F = 0.7$ and $CR = 0.7$ (in accordance with [44]) and, with reference to Fig. 3, $n_p = 10$.
- The DEGL has been run, with reference to formulas (6), (7), (8), and (9), with $k = 5$, $\alpha = \beta = 0.8$, $w_{\min} = 0.4$, $w_{\max} = 0.8$, and $CR = 0.7$
- The SaNSDE has been run as explained in Subsection 2.7
- The FDSADE has been run with reference to formulas (21) and (22), $F_l = 0.1$, $F_u = 0.9$ as suggested in [2] and $K = 0.3$.

Regarding the population size, the FDSADE has been run with $S_{pop}^{f} = 10$, $S_{pop}^{v} = 40$ (see formula (25)) while all the other algorithms have been run with $S_{pop} = 30$. Each algorithm has been run for 30 independent runs, 50000 fitness evaluations each run.

The choice of $K = 0.3$ has been empirically performed after having carried out an analysis of the algorithmic behavior with dependance on K. The main result of our analysis is that the FDSADE has a high algorithmic performance for chosen K values in the interval $[0.1, 0.7]$ and performance of the FDSADE is not very sensitive to the chosen K value within this interval. Nevertheless, it has been observed that the best performance is obtained in correspondence to $K = 0.3$. Fig. 6 shows an example of the FDSADE average performance (over 30 runs) with dependence on various values of K. Fig. 6 refers to the Michalewicz function in 30 dimensions.

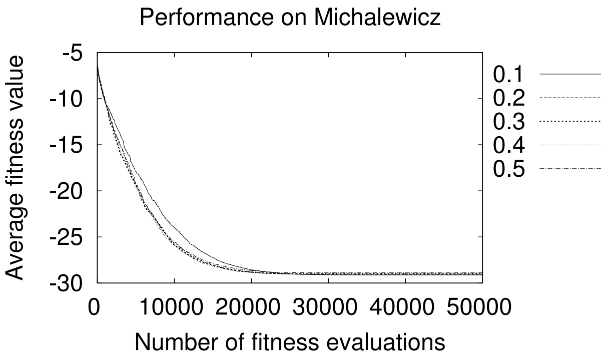


Fig. 6 Algorithmic performance in dependance on K

The test problems under investigation are listed in Table 1. It should be remarked that some rotated problems have been added to our benchmark set. The rotated problems are obtained by means of multiplication of the vector of variables to a randomly generated orthogonal rotation matrix. The test problem indicated with "Tirronen" is generated by means of a novel test function proposed in this chapter. This test function has been included in order to obtain a highly multi-modal landscape which contains a global minimum in an asymmetrical position and a pattern that changes towards the minimum (unlike Rastrigin or Ackley, whose pattern is orthogonal to the axes throughout the entire landscape). Fig. 7 shows the Tirronen function in two dimensions.

Table 2 shows the average final results (after 50000 fitness evaluations) and the related standard deviation, calculated over the 30 available runs. The best results are highlighted in bold face.

It can be noted that only for the Easom and Camelback functions do all algorithms converge to the same value. For the other eighteen test problems, one of the algorithms outperforms all others in terms of final value. Table 2 shows that the DEGL seems to be a very competitive algorithm since it produced the best performance in ten cases; the proposed FDSADE reached the best final value in seven cases out of twenty under analysis, the DE

Table 1 Test Problems

Test Problem	n	Function	Decision Space
Ackley	30	$-20+e+\exp\left(-\frac{0.2}{n}\sqrt{\sum_{i=1}^{n}x_i^2}\right)$ $-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi\cdot x_i)x_i\right)$	$[-1,1]^n$
Alpine	30	$\sum_{i=1}^{n}\lvert x_i\sin x_i+0.1x_i\rvert$	$[-10,10]^n$
Camelback	2	$4x_1^2-2.1x_1^2+\frac{x_1^6}{3}+x_1x_2-4x_2^2+4x_2^4$	
DeJong	30	$\lvert\lvert x\rvert\rvert^2$	$[-5.12,5.12]^n$
DropWave	30	$-\frac{1+\cos\left(12\sqrt{\lvert\lvert x\rvert\rvert^2}\right)}{\frac{1}{2}\lvert\lvert x\rvert\rvert^2+2}$	$[-5.12,5.12]^n$
Easom	2	$\cos x_1\cos x_2\exp\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)$	$[-100,100]^n$
Griewangk	30	$\frac{\lvert\lvert x\rvert\rvert^2}{4000}-\prod_{i=0}^{n}\cos\frac{x_i}{\sqrt{i}}+1$	$[-600,600]^n$
Michalewicz	30	$-\sum_{i=1}^{n}\sin x_i\left(\sin\left(\frac{i\cdot x_i^2}{\pi}\right)\right)$	$[0,\pi]^n$
Pathological	30	$\sum_{i=1}^{n-1}\left(0.5+\frac{\sin^2\left(\sqrt{100x_i^2+x_{i+1}^2}-0.5\right)}{1+0.001*\left(x_i^2-2x_ix_{i+1}+x_{i+1}^2\right)^2}\right)$	$[-100,100]^n$
Rosenbrock	30	$\sum_{i=0}^{n-1}\left(\left(x_{n+1}-x_i^2\right)^2+(1-x)^2\right)$	$[-2.048,2.048]^n$
Rastrigin	30	$10n+\sum_{i=0}^{n}\left(x_i^2-10\cos(2\pi x_i)\right)$	$[-5.12,5.12]^n$
Schwefel	30	$\sum_{i=1}^{n}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	$[-500,500]^n$
Sum of powers	30	$\sum_{i=1}^{n}\lvert x_i\rvert^{i+1}$	$[-1,1]^n$
Tirronen	30	$3\exp\left(-\frac{\lvert\lvert x\rvert\rvert^2}{10n}\right)-10\exp\left(-8\lvert\lvert x\rvert\rvert^2\right)$ $+\frac{2.5}{n}\sum_{i=1}^{n}\cos\left(5(x_i+(1+i \mod 2))cos(\lvert\lvert x\rvert\rvert)\right)$	$[-10,5]^n$
Whitley	30	$\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{y_{i,j}^2}{4000}-\cos\left(y_{i,j}\right)+1\right)$, where $y_{i,j}=\left(100(x_j-x_i)^2+(1-x_i)^2\right)^2$	$[-100,100]^n$
Zakharov	30	$\lvert\lvert x\rvert\rvert^2+\left(\sum_{i=1}^{n}\frac{ix_1}{2}\right)^2+\left(\sum_{i=1}^{n}\frac{ix_1}{2}x_i\right)^4$	$[-5,10]^n$

reached the best final value in five cases, the SaNSDE and the SACPDE reached the best final values for four test problems and the DEahcSPX only with the Easom and Camelback functions. In addition, it can be seen that when the FDSADE does not reach the best value, it will in any case often

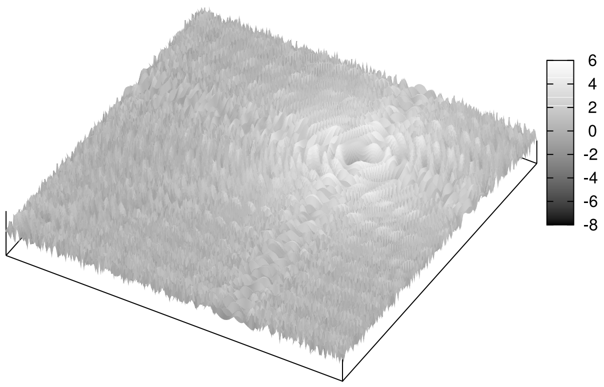


Fig. 7 Tirronen function

return a solution with a high performance (see e.g., the values in De Jong or Rotated Rastrigin). Finally, it must be highlighted that the FDSADE never obtained the worst results over the twenty test problems under analysis and in most cases offers a competitive performance in terms of final solution.

In order to prove statistical significance of the results, the Student's t-test has been applied according to the description given in [25] for a confidence level of 0.95. The final values obtained by the FDSADE have been compared to the final value returned by each algorithm used as a benchmark. Table 3 shows results of the test. Indicated with "+" is the case when the FDSADE statistically outperforms, for the corresponding test problem, the algorithm mentioned in the column; indicated with "=" is the case when pairwise comparison leads to success of the t-test i.e., the two algorithms have the same performance; indicated with "-" is the case when the FDSADE is outperformed.

Table 2 and Table 3 show that DEGL performs the best, in terms of final values, among all the algorithms considered here. However, it must be noticed from Table 3 that the FDSADE is outperformed in only eighteen cases out of the hundred-twenty pairwise comparisons considered, the FDSADE has the same algorithmic performance as the other algorithms in twenty-six cases and outperforms the other algorithms in fifty-six cases. It can thus be concluded that the FDSADE is outperformed in only 15% of the cases and outperforms the other algorithms in 56% of the cases. Therefore, the FDSADE, in a statistical sense, offers good performance in detecting a final solution with high quality and is competitive with modern DE based algorithms. Considering that the experimental setup is composed of a very diverse set of test problems (e.g., some functions are uni-modal, others are highly multi-modal), the FDSADE seems to have a high performance with various kinds of problems. Finally, attention must be paid in observing the comparison between FDSADE and SACPDE since the FDSADE has the same self-adaptive structure of the SACPDE and the same update logic of the control parameters.

Table 2 Final Values, Average ± Standard Deviation

Test Prob.	DE	SACPDE	DEahcSPX	DEGL	SaNSDE	FDSADE
Ackley	6.17e-10±1.82e-10	3.16e-07±1.70e-06	9.42e-02±6.24e-02	**7.19e-15±1.07e-15**	6.21e-02±2.32e-01	1.75e-12±7.22e-12
Alpine	3.35e-02±1.87e-02	3.40e-05±7.30e-05	1.28e+00±1.05e+00	**1.12e-10±5.92e-10**	5.32e-03±2.87e-02	1.24e-04±4.88e-04
Camelback	**-1.03e+00±6.66e-16**	**-1.03e+00±6.66e-16**	-1.03e+00±5.73e-16	**-1.03e+00±6.66e-16**	**-1.03e+00±6.66e-16**	**-1.03e+00±6.66e-16**
De Jong	1.59e-57±1.94e-57	**4.25e-86±1.57e-85**	2.24e-02±4.73e-02	9.66e-50±1.34e-49	9.92e-19±5.16e-18	2.26e-66±2.92e-66
Dropwave	-5.98e-01±4.72e-02	-2.50e-01±6.70e-02	-1.82e-01±4.54e-02	**-7.91e-01±2.70e-02**	-1.73e-01±6.66e-02	-2.60e-01±6.90e-02
Easom	**-1.00e+00±0.00e+00**	**-1.00e+00±0.00e+00**	-1.00e+00±7.21e-24	**-1.00e+00±0.00e+00**	**-1.00e+00±0.00e+00**	**-1.00e+00±0.00e+00**
Griewangk	**2.27e-13±5.36e-13**	3.90e-02±6.13e-02	3.70e+00±3.30e+00	5.75e-04±2.18e-03	4.63e-01±9.26e-01	9.00e-03±1.57e-02
Rot. Griew.	**1.86e-03±4.14e-03**	6.25e-03±8.48e-03	3.71e+00±2.67e+00	2.79e-03±5.27e-03	9.51e-03±1.19e-02	2.71e-03±4.73e-03
Michalew.	-2.89e+01±1.97e-01	-2.89e+01±2.76e-01	-1.48e+01±1.21e+00	-2.10e+01±5.96e-01	-2.45e+01±1.27e+00	**-2.90e+01±2.40e-01**
Rot. Mich.	-1.15e+01±5.85e-01	-1.91e+01±2.16e+00	-1.07e+01±7.98e-01	-9.46e+00±6.47e-01	-1.37e+01±1.50e+00	**-2.14e+01±2.26e+00**
Patholog.	2.45e+01±2.46e-05	2.45e+01±3.20e-06	2.45e+01±4.59e-04	**1.45e+01±5.79e-07**	1.45e+01±3.36e-06	2.45e+01±1.66e-06
Rastrigin	**8.87e+00±3.19e+00**	1.23e+01±3.55e+00	1.25e+02±1.72e+01	5.68e+01±8.56e+00	6.88e+01±1.87e+01	1.13e+01±3.73e+00
Rot. Rast.	4.36e+02±1.91e+01	8.44e+01±1.81e+01	4.00e+02±1.54e+01	1.67e+02±1.56e+01	**7.28e+01±2.37e+01**	9.13e+01±1.90e+01
Rosenbrock	7.01e-10±5.34e-10	4.21e-04±1.32e-03	3.42e+00±1.06e+00	**3.69e-20±1.15e-19**	1.34e-07±2.24e-07	1.16e-05±1.77e-05
Schwefel	-1.19e+04±2.75e+02	-1.14e+04±3.18e+02	-5.62e+03±5.16e+02	-1.04e+04±1.05e+03	-9.39e+03±4.73e+02	**-1.19e+04±2.85e+02**
Rot. Schw.	-7.36e+03±3.61e+02	-1.11e+04±8.25e+02	-5.35e+03±4.29e+02	-6.56e+03±5.30e+02	-8.56e+03±6.63e+02	**-1.41e+04±8.51e+02**
Sum pow.	9.21e-36±1.67e-35	2.52e-07±7.68e-07	1.04e-04±3.97e-04	**3.98e-66±1.03e-65**	6.24e-31±2.29e-30	3.25e-08±1.33e-07
Tirronen	-1.86e+00±5.92e-02	-2.06e+00±1.47e-01	-1.07e+00±9.34e-02	-1.98e+00±6.16e-02	-2.03e+00±1.92e-01	**-2.15e+00±7.88e-02**
Whitley	1.60e+03±1.20e+02	1.66e+12±5.59e+12	1.00e+13±6.93e+14	**3.23e+02±1.43e+02**	3.06e+03±3.27e+03	1.85e+12±6.57e+12
Zakharov	5.90e+02±7.64e+01	2.07e+01±1.30e+01	1.77e+02±5.54e+01	**2.20e+00±1.10e+00**	4.07e+01±5.05e+01	2.20e+01±1.05e+01

Table 3 Results of the Student's t-test

Test Problem	DE	SACPDE	DEahcSPX	DEGL	SaNSDE
Ackley	+	=	+	=	+
Alpine	+	=	+	−	=
Camelback	=	=	=	=	=
De Jong	=	=	+	=	=
Dropwave	−	=	+	−	+
Easom	=	=	=	=	=
Griewangk	−	+	+	−	+
Rot. Griewangk	=	+	+	=	+
Michalewicz	+	+	+	+	+
Rot. Michalewicz	+	+	+	+	+
Pathological	+	+	+	−	−
Rastrigin	−	=	+	+	+
Rot. Rastrigin	+	−	+	+	−
Rosenbrock	−	+	+	−	−
Schwefel	=	+	+	+	+
Rot. Schwefel	+	+	+	+	+
Sum of powers	−	+	+	−	−
Tirronen	+	+	+	+	+
Whitely	−	=	=	−	−
Zakharov	+	=	+	−	+

As shown in Section 4, the FDSADE integrates, in addition to the SACPDE, the fitness diversity philosophy and fitness diversity based variable population size. Integration of the fitness diversity seems to be very promising because, as shown in Table 3, it leads to an improvement of performance in ten cases, the same performance in nine cases and a worse performance in only one case. This analysis confirms that the fitness diversity adaptation can be an efficient instrument in enhancing the effectiveness of an algorithm.

In order to carry out a numerical comparison of the convergence speed performance, for each test problem the average final fitness value returned by the best performing algorithm G has been considered. Subsequently, the average fitness value at the beginning of the optimization process J has also been computed. The threshold value $THR = J - 0.95(G - J)$ has then been calculated. The value THR represents 95% of the decay in the fitness value in the best performing algorithms fitness value. If an algorithm succeeds during a certain run to reach the value THR, the run is said to be successful. For each test problem the average amount of fitness evaluations $\bar{ne}$ required for each algorithm to reach THR has been computed. Subsequently, the Q-test (Q stands for Quality) described in [12] has been applied. For each test problem and each algorithm, the Q measure is computed as $Q = \frac{\bar{ne}}{R}$ where the robustness R is the percentage of successful runs. It is clear that for each test problem the smallest value means the best performance in terms of convergence speed. The value inf means that $R = 0$, i.e. the algorithm never reached the THR. Table 4 shows the Q values in 30 dimensions. The best results are highlighted in bold face.

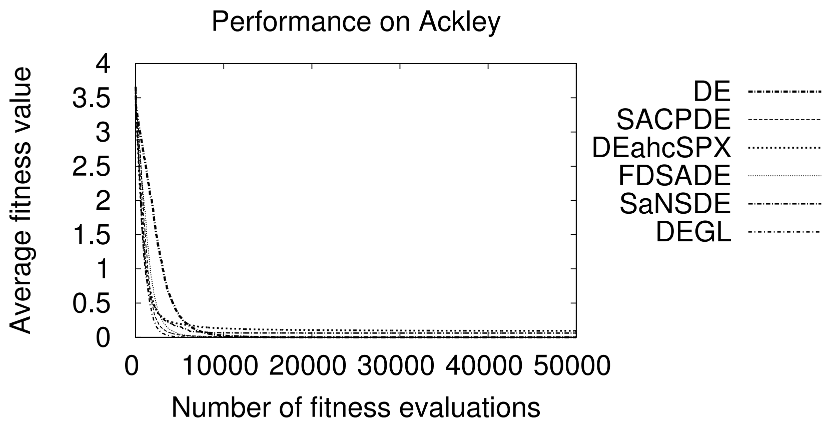
Figures 8 show the average performance trend (over 30 independent runs) of the six algorithms under analysis over some of the test problems listed in Table 1.

It can be observed that the FDSADE fails in detecting the optimum, with respect to the others, only in the case of the Dropwave and Whitley functions. In all other cases the FDSADE displays a high performance; the proposed algorithm is either competitive with another algorithm of the benchmark (see e.g., Rotated Griewangk, Michalewicz, Pathological, and Rotated Rastrigin) or has extraordinarily high performance as shown with Rotated Michalewicz, Schwefel, and Rotated Schwefel. Performance in terms of convergence speed is also competitive with the other algorithms in most of the cases analyzed.

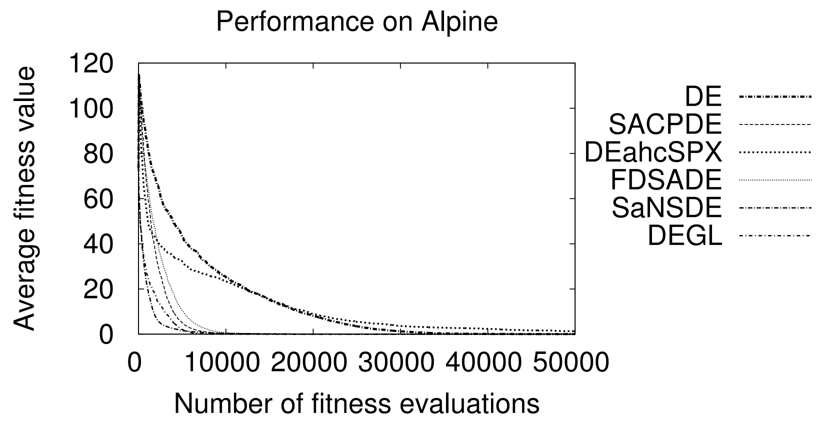
Numerical results in Table 4 show that although the FDSADE does not always have the best performance in terms of convergence velocity, it is in most cases very competitive with the algorithm that has the best performance. In other words, the other algorithms seem to have a high convergence velocity performance with some test problems and a poor performance with others. On the contrary, the FDSADE demonstrates a performance close to the best in almost all test problems. In addition, it must be remarked that the FDSADE does not reach THR in only two cases (Dropwave and Pathological

Table 4 Results of the Q-test

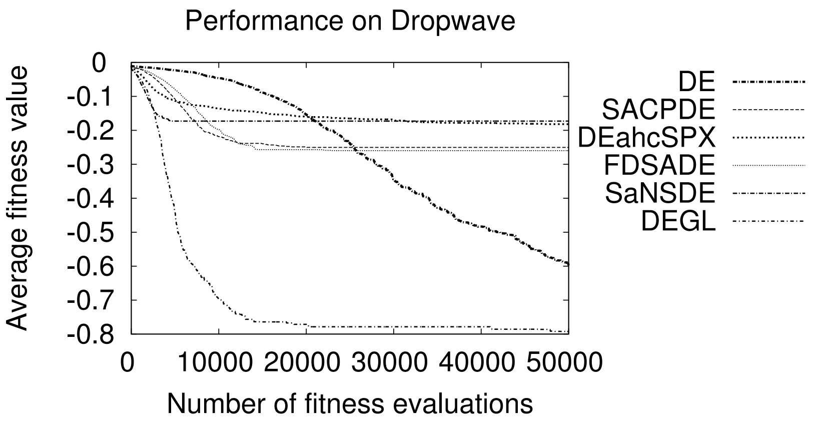
Test Problem	DE	SACPDE	DEahcSPX	DEGL	SaNSDE	FDSADE
Ackley	6.0e+01	2.8e+01	6.4e+01	**2.3e+01**	3.4e+01	3.4e+01
Alpine	2.3e+02	5.0e+01	2.4e+02	4.1e+01	**2.7e+01**	6.4e+01
Camelback	2.5e+00	2.5e+00	1.5e+01	2.4e+00	**1.6e+00**	2.6e+00
De Jong	8.0e+00	**5.3e+00**	7.0e+00	1.6e+01	2.6e+01	6.7e+00
Dropwave	inf	inf	inf	**1.3e+02**	inf	inf
Easom	1.7e+01	9.8e+00	8.6e+01	1.5e+01	**5.9e+00**	1.4e+01
Griewangk	3.5e+01	1.6e+01	1.4e+01	**1.3e+01**	1.7e+01	2.0e+01
Rotated Griewangk	3.4e+01	1.6e+01	1.4e+01	**1.3e+01**	1.5e+01	2.0e+01
Michalewicz	3.1e+02	**1.1e+02**	inf	inf	inf	1.4e+02
Rotated Michalewicz	inf	2.1e+03	inf	inf	inf	**5.2e+02**
Pathological	inf	inf	inf	**4.0e-01**	**4.0e-01**	inf
Rastrigin	2.8e+02	**5.7e+01**	inf	inf	inf	6.5e+01
Rotated Rastrigin	inf	2.5e+02	inf	inf	5.1e+01	**2.3e+02**
Tirronen	inf	7.4e+02	inf	6.4e+03	**1.7e+02**	4.9e+02
Rosenbrock	5.3e+01	**2.6e+01**	9.6e+01	1.9e+01	2.8e+01	3.1e+01
Schwefel	2.2e+02	1.3e+02	inf	1.8e+03	inf	**8.2e+01**
Rotated Schwefel	inf	inf	inf	inf	inf	**2.5e+02**
Sum of powers	2.1e+01	7.0e+00	5.2e+00	4.4e+00	**4.2e+00**	7.3e+00
Whitely	2.7e+01	8.1e+00	5.7e+00	3.6e+00	**3.2e+00**	9.7e+00
Zakharov	1.2e+00	5.7e+00	1.5e+01	**4.0e-01**	**4.0e-01**	1.5e+00



(a)

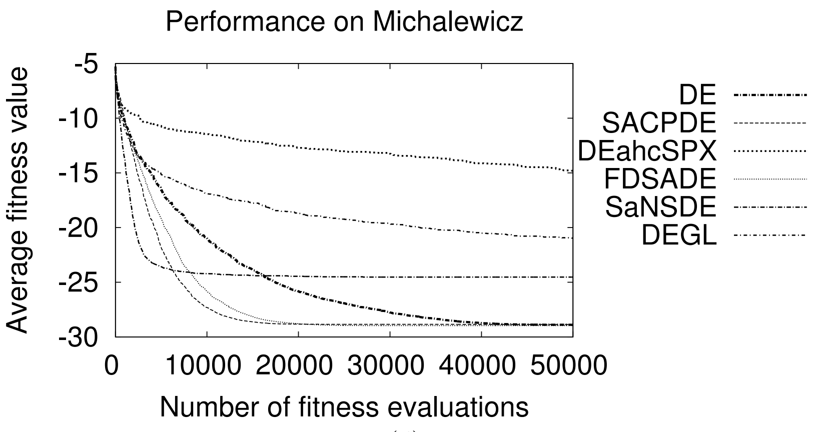


(b)

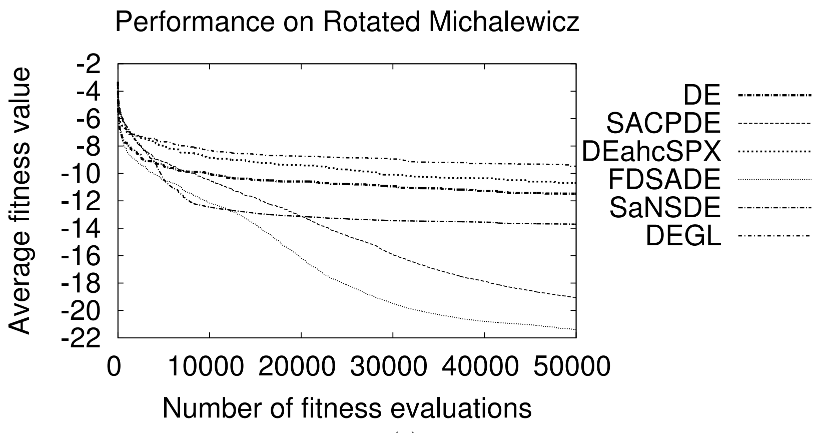


(c)

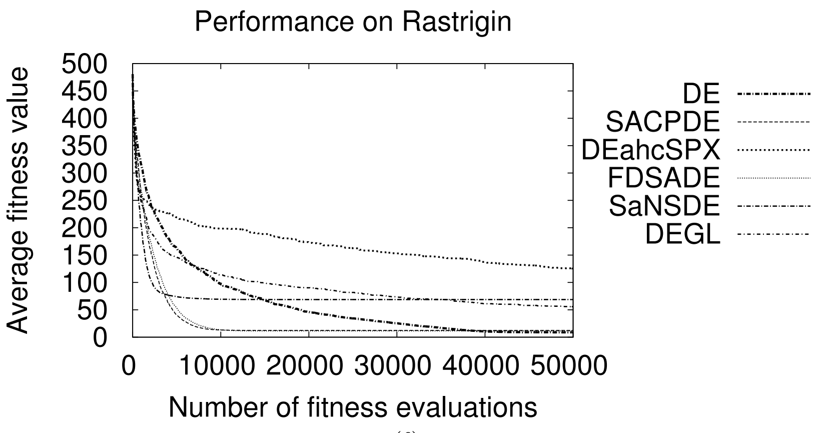
Fig. 8 Algorithmic performance over selected test problems



(d)

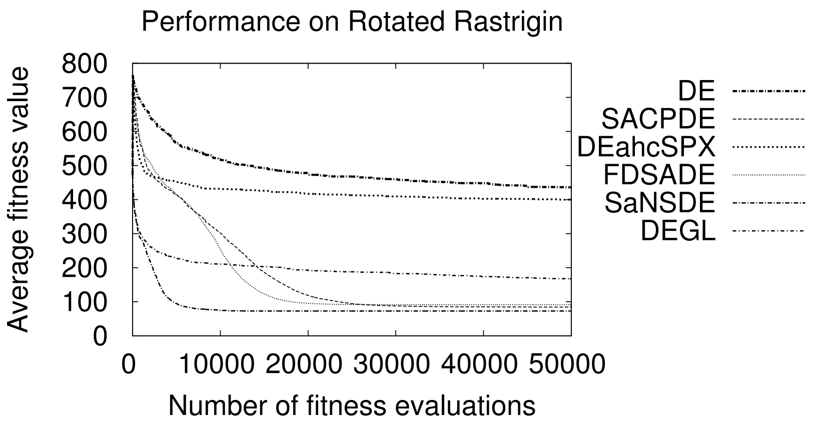


(e)

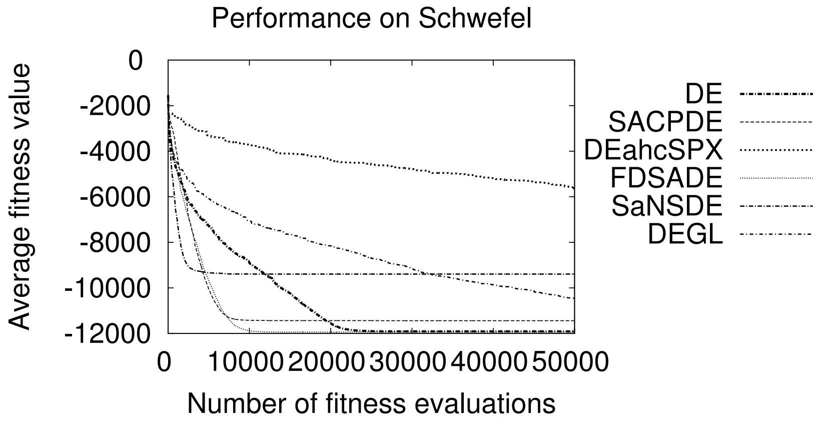


(f)

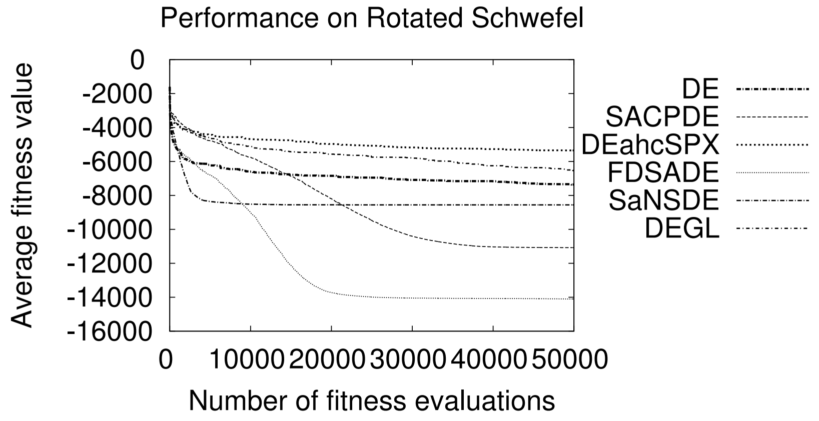
Fig. 8 (*continued*)



(g)



(h)



(i)

Fig. 8 (*continued*)

functions); in all other cases it reaches the threshold in at least one case. This result confirms that the FDSADE has a robust behavior over a set of various test problems. It can be seen that, in this sense, the FDSADE is the best algorithm (it is the algorithm with the fewest number of inf values) among the six under examination.

5.1 Numerical Results for (Relatively) Large Scale Problems

The previous six algorithms have also been run in order to minimize ten of the functions in Table 1 for $n = 100$. The algorithms have been run with the same parameter setting mentioned above except population size. The FDSADE has been run with $S^f_{pop} = 30$, $S^v_{pop} = 120$ while all the other algorithms with $S_{pop} = 100$. For each algorithm, 30 independent runs have been performed for 50000 fitness evaluations. Table 5 shows the average final results (after 50000 fitness evaluation) and the related standard deviation, calculated over the 30 available runs.

It must be observed that the algorithmic performance is dramatically different with respect to the numerical results in 30 dimensions. An important fact is that, in accordance with the No Free Lunch Theorem [40], the algorithmic performance in high dimensions is very problem-dependant, i.e., none of the algorithms examined seems to be, in general, clearly superior to the others. The performance comparison reported here is valid only for the parameter settings used in this chapter. The effect of variation of parameters has not been investigated.

However, it can be noticed that the DE and SACPDE never have the best performance in terms of quality of the final solution. Performance of the DEGL is not so promising as the one displayed in the low dimensional case. On the contrary, the SaNSDE seems to be rather promising for large scale optimization problems. The FDSADE is competitive with the SaNSDE and reaches quite good results in all the test problems considered.

Table 6 shows results of the Student's t-test in 100 dimensions.

The t-test in Table 6 shows that the FDSADE is outperformed nine times and has the same performance in only two cases out of the fifty pairwise comparisons considered. Thus, the FDSADE is outperformed in only 18% of the cases and outperforms the other algorithms in 78% of the cases. Finally, it must be observed that in 100 dimensions, the FDSADE either obtained results similar to the SACPDE or succeeded in improving upon the SACPDE performance, thus confirming the efficiency of the approach proposed.

Table 7 shows the Q-test results for the 100 dimension case.

Regarding convergence speed and robustness of the algorithms, since in the high dimensional space the various algorithms tend to reach very different values, for each test problem that algorithm which reaches final values with

Table 5 Final Values, Average ± Standard Deviation in 100 dimensions

Test Problem	DE	SACPDE	DEahcSPX	DEGL	SaNSDE	FDSADE
Rot. alpine	2.00e+02±9.59e+00	1.29e+02±1.38e+01	**6.83e-01±4.98e-01**	6.11e+00±7.93e+00	4.04e+01±9.14e+00	1.16e+01±6.78e+00
Dropwave	-9.71e-03±2.14e-04	-1.20e-02±0.00e+00	**-4.78e-01±0.00e+00**	-3.67e-01±0.00e+00	-4.91e-02±1.02e-02	-1.82e-01±6.72e-03
Michalewicz	-3.62e+01±0.00e+00	-2.82e+01±1.88e+00	-2.84e+01±4.95e-01	-3.65e+01±1.04e+00	**-7.24e+01±2.22e+00**	-5.66e+01±7.54e-01
Rot. Michalewicz	-1.51e+01±7.42e-01	-1.46e+01±7.88e-01	-1.63e+01±5.06e-01	-1.69e+01±1.16e+00	**-2.31e+01±5.79e+00**	-1.84e+01±1.46e+00
Rastrigin	9.44e+02±3.07e+01	5.34e+01±5.44e+01	8.07e+02±6.23e+01	6.57e+02±1.89e+01	2.80e+02±4.28e+01	**5.08e+01±8.82e+00**
Rot. Rastrigin	1.26e+03±3.29e+01	5.08e+02±4.80e+01	8.72e+02±3.81e+00	8.95e+02±2.65e+01	**2.87e+02±4.58e+01**	4.12e+02±7.62e+01
Rosenbrock	8.25e+01±7.88e+00	8.98e+01±1.33e+01	3.05e-01±6.24e-02	**1.41e-01±3.20e-02**	3.74e+00±1.78e+00	1.33e+00±1.87e-01
Schwefel	-1.72e+04±6.80e+02	-3.69e+03±1.31e+03	-9.18e+03±5.14e+02	-1.32e+04±5.42e+02	-2.91e+04±1.18e+03	**-3.99e+04±5.86e+02**
Rot. Schwefel	-1.03e+04±7.34e+02	-9.44e+03±6.07e+02	-8.56e+03±6.19e+02	-1.00e+04±6.80e+02	**-2.94e+04±1.54e+03**	-1.85e+04±3.74e+03
Zakharov	2.09e+03±1.20e+02	6.98e+10±3.56e+11	1.29e+03±1.92e+02	1.12e+03±9.45e+01	1.66e+03±2.39e+02	**5.53e+02±2.51e+02**

Table 6 Results of the Student's t-test in 100 dimensions

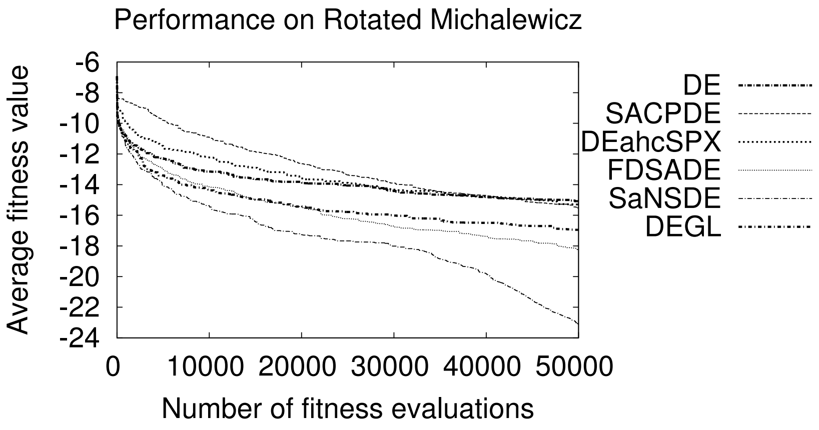
TestProblem	DE	SACPDE	DEahcSPX	DEGL	SaNSDE
Rotated alpine	+	+	−	−	+
Dropwave	+	+	−	−	+
Michalewicz	+	+	+	+	−
Rotated Michalewicz	+	+	+	+	−
Rastrigin	+	=	+	+	+
Rotated Rastrigin	+	+	+	+	−
Rosenbrock	+	+	−	−	+
Schwefel	+	+	+	+	+
Rotated Schwefel	+	+	+	+	−
Zakharov	+	=	+	+	+

Table 7 Results of the Q-test in 100 dimensions

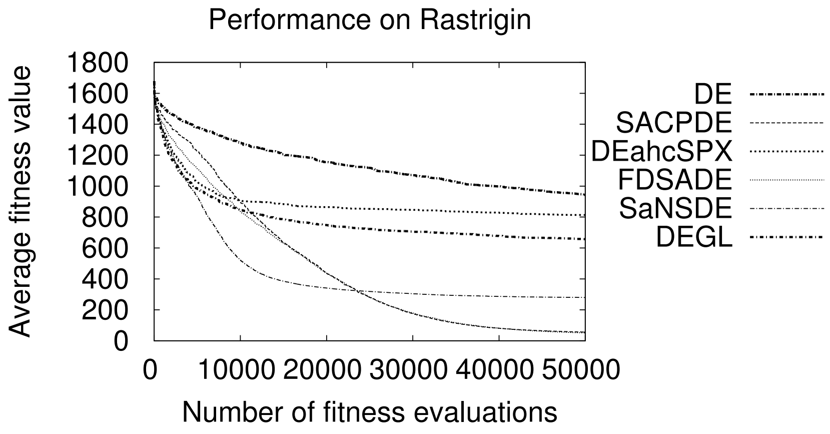
Test Problem	DE	SACPDE	DEahcSPX	DEGL	SaNSDE	FDSADE
Rotated alpine	inf	4.4e+02	**3.1e+02**	5.1e+02	inf	7.1e+02
Dropwave	inf	inf	**4.6e+03**	inf	inf	inf
Michalewicz	inf	inf	inf	inf	**4.2e+02**	inf
Rotated Michalewicz	inf	inf	inf	inf	**1.0e+03**	inf
Rastrigin	inf	**3.4e+02**	inf	inf	inf	**3.4e+02**
Rotated Rastrigin	inf	2.4e+03	inf	inf	**3.3e+02**	1.4e+03
Rosenbrock	inf	1.9e+02	1.4e+02	**1.2e+02**	1.7e+02	1.9e+02
Schwefel	inf	1.4e+04	inf	inf	inf	**3.9e+02**
Rotated Schwefel	inf	inf	inf	inf	**3.3e+02**	inf
Zakharov	5.1e+00	5.5e+01	2.9e+01	3.6e+00	**2.0e+00**	5.2e+00

a best performance is usually the one which has the best performance in terms of Q measures. In this sense, numerical results in Table 7 confirm the findings in Table 5 and Table 6, i.e., the SaNSDE has very good performance in highly dimensional problems and the FDSADE is, in any case, quite competitive. On the other hand, behavior of the algorithms in terms of robustness is worthwhile commenting on. As shown in Table 7, the DE succeeds in reaching a competitive value (reaches the threshold value) for only one test problem, the DEGL in three cases, the DEahcSPX in four cases, SACPDE, SaNSDE and FDSADE in six cases out of ten test problems considered. This fact means there is not an algorithm which has an extraordinarily high performance in terms of robustness among the ones considered. However the FDSADE maintains the good robustness performance of the SACPDE and tends to improve upon this in late stages of the optimization process.

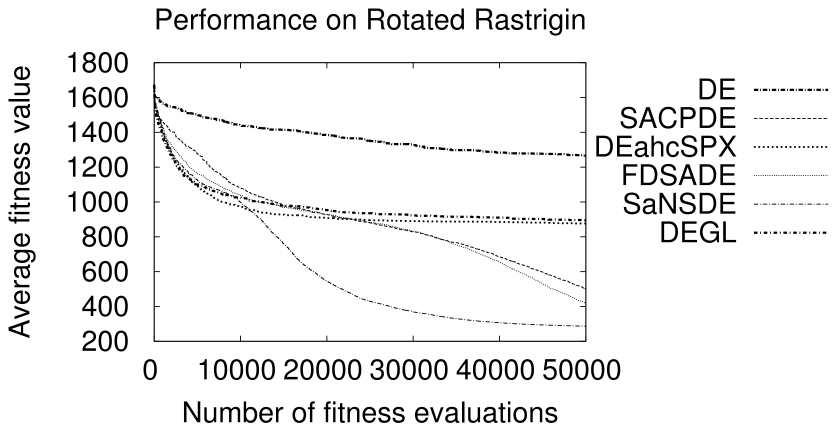
For the sake of completeness, some performance trends are shown in Figures 9.



(a)

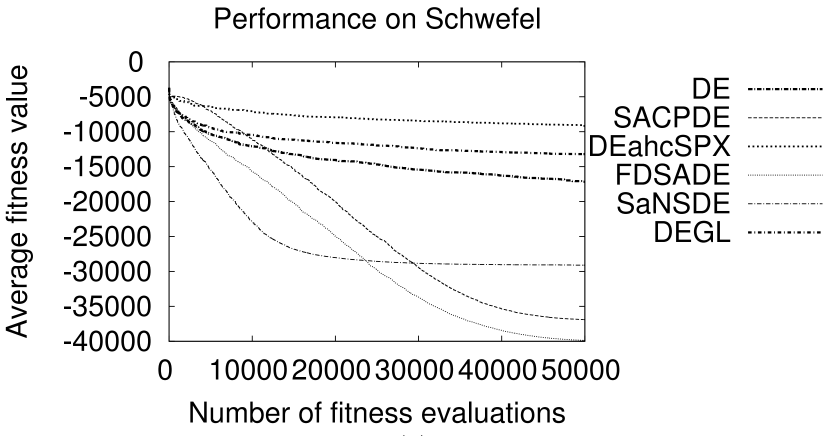


(b)

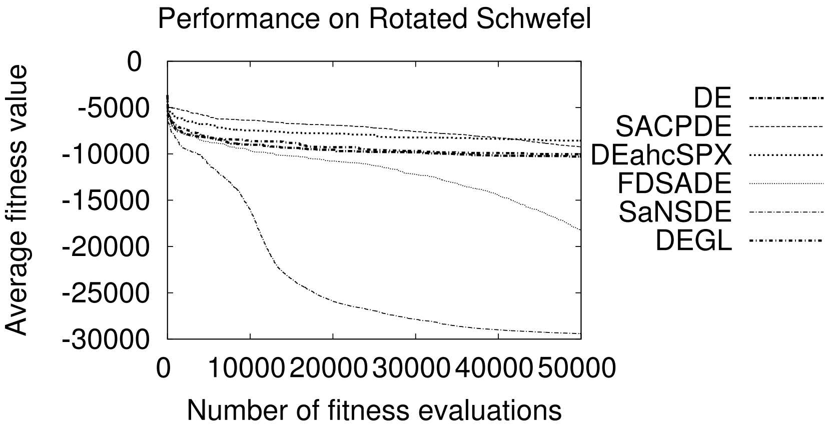


(c)

Fig. 9 Algorithmic performance over selected test problems in 100 dimensions



(d)



(e)

Fig. 9 (*continued*)

6 Conclusion

This chapter proposes integration of fitness diversity logic within the DE frameworks in order to self-adaptively affect the control parameter update and adaptively perform a dynamic population sizing. This integration is pursued by the definition of a novel index that measures the fitness diversity of a DE population and a novel algorithmic implementation.

The proposed algorithm has been tested on a set of twenty test problems and compared with a standard DE and four recently proposed DE based algorithms. Numerical results show that the proposed algorithm is very efficient in detecting solutions having high performance and that it has a robust behavior over a various set of test problems. The performance in convergence speed is also competitive with those algorithms that represent the state-of-the-art in DE based implementation.

The fitness diversity logic is thus very efficient at monitoring the state of the algorithm during the optimization process and dynamically foreseeing algorithmic necessities. It is important to remark that the fitness diversity (self-)adaptation must aim at coordinating the exporative/exploitative pressure by increasing exploration when the diversity is low and exploitation when diversity is high. This mechanism tends to prevent the undesired effects of premature convergence and stagnation, therefore efficiently performing the optimization towards high quality solutions (as numerical results confirm). This aim is, in this chapter, pursued by frequently updating the scale factor and crossover rate and enlarging the population size in low diversity conditions; these operations give the algorithm a better chance of testing unexplored areas of the decision space and of, hopefully, detecting new promising solutions. Conversely, in high diversity conditions the population is shrunk and the control parameters kept constant; in this way the available genotypes and potential search directions (e.g., by means of the scale factor values) are exploited until the diversity decreases.

A remaining issue is that a proper fitness diversity index must be designed on the basis of the algorithmic structure (i.e., variation operators and selection mechanisms). The index proposed in this chapter seems to be very efficient in monitoring the state of the population of a DE, regardless of the optimization problem under examination. Therefore, we suggest its employment for designing (self-)adaptation within a DE framework.

References

1. Ali, M.M., Törn, A.: Population set based global optimization algorithms: Some modifications and numerical studies. Computers and Operations Research 31(10), 1703–1725 (2004)
2. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Žumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation 10(6), 646–657 (2006)
3. Brest, J., Zumer, V., Maucec, M.: Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 215–222 (2006)
4. Caponio, A., Cascella, G.L., Neri, F., Salvatore, N., Sumner, M.: A fast adaptive memetic algorithm for on-line and off-line control design of PMSM drives. IEEE Transactions on System Man and Cybernetics-part B, special issue on Memetic Algorithms 37(1), 28–41 (2007)
5. Caponio, A., Neri, F., Tirronen, V.: Super-fit control adaptation in memetic differential evolution frameworks. Soft Computing-A Fusion of Foundations, Methodologies and Applications 13(8), 811–831 (2009)
6. Chakraborty, U.K. (ed.): Advances in Differential Evolution. Studies in Computational Intelligence, vol. 143. Springer, Heidelberg (2008)

7. Chakraborty, U.K., Das, S., Konar, A.: Differential evolution with local neighborhood. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2042–2049 (2006)
8. Das, S., Abraham, A., Chakraborty, U.K., Konar, A.: Differential evolution with a neighborhood-based mutation operator. IEEE Transactions on Evolutionary Computation (to appear, 2009)
9. Deb, K.: Multi-objective Optimization using Evolutionary Algorithms, pp. 147–149. Wiley and Sons LTD, Chichester (2001)
10. Eiben, A.E., Schippers, C.A.: On evolutionary exploration and exploitation. Fundamenta Informaticae 35(1-4), 35–50 (1998)
11. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computation. Springer, Berlin (2003)
12. Feoktistov, V.: Differential Evolution in Search of Solutions, pp. 83–86. Springer, Heidelberg (2006)
13. Gämperle, R., Müller, S.D., Koumoutsakos, P.: A parameter study for differential evolution. In: NNA-FSFS-EC, WSEAS, pp. 293–298 (2002)
14. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Co., Reading (1989)
15. Krasnogor, N.: Toward robust memetic algorithms. In: Hart, W.E., Krasnogor, N., Smith, J.E. (eds.) Recent Advances in Memetic Algorithms. Studies in Fuzzines and Soft Computing, pp. 185–207. Springer, Berlin (2004)
16. Lampinen, J., Zelinka, I.: On stagnation of the differential evolution algorithm. In: Osmera, P. (ed.) Proceedings of 6th International Mendel Conference on Soft Computing, pp. 76–83 (2000)
17. Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. In: Proceedings of the 17th IEEE region 10 international conference on computer, communications, control and power engineering, vol. I, pp. 606–611 (2002)
18. Liu, J., Lampinen, J.: On setting the control parameter of the differential evolution algorithm. In: Proceedings of the 8th international Mendel conference on soft computing, pp. 11–18 (2002)
19. Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. Soft Computing - A Fusion of Foundations, Methodologies and Applications 9(6), 448–462 (2005)
20. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Tech. Rep. 790 (1989)
21. Neri, F., Mäkinen, R.A.E.: Hierarchical evolutionary algorithms and noise compensation via adaptation. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. Studies in Computational Intelligence, ch. 15, pp. 345–369. Springer, Heidelberg (2007)
22. Neri, F., Toivanen, J., Cascella, G.L., Ong, Y.S.: An adaptive multimeme algorithm for designing HIV multidrug therapies. IEEE/ACM Transactions on Computational Biology and Bioinformatics, Special Issue on Computational Intelligence Approaches in Computational Biology and Bioinformatics 4(2), 264–278 (2007)
23. Neri, F., Toivanen, J., Mäkinen, R.A.E.: An adaptive evolutionary algorithm with intelligent mutation local searchers for designing multidrug therapies for HIV. Applied Intelligence, Special Issue on Computational Intelligence in Medicine and Biology 27(3), 219–235 (2007)

24. Neri, F., Kotilainen, N., Vapa, M.: A memetic-neural approach to discover resources in P2P networks. In: van Hemert, J., Cotta, C. (eds.) Recent Advances in Evolutionary Computation for Combinatorial Optimization. Studies in Computational Intelligence, ch. 8, pp. 119–136. Springer, Heidelberg (2008)
25. NIST/SEMATECH, e-Handbook of Statistical Methods (2003), `http://www.itl.nist.gov/div898/handbook/`
26. Noman, N., Iba, H.: Accelerating differential evolution using an adaptive local search. IEEE Transactions on Evolutionary Computation 12(1), 107–125 (2008)
27. Ong, Y.S., Keane, A.J.: Meta-lamarkian learning in memetic algorithms. IEEE Transactions on Evolutionary Computation 8(2), 99–110 (2004)
28. Price, K., Storn, R.: Differential evolution: A simple evolution strategy for fast optimization. Dr Dobb's J. Software Tools 22(4), 18–24 (1997)
29. Price, K.V., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Heidelberg (2005)
30. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 2, pp. 1785–1791 (2005)
31. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.: Opposition-based differential evolution. IEEE Transactions on Evolutionary Computation 12(1), 64–79 (2008)
32. Rechemberg, I.: Evolutionstrategie: Optimierung Technisher Systeme nach prinzipien des Biologishen Evolution. Fromman-Hozlboog Verlag (1973)
33. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. Rep. TR-95-012, ICSI (1995)
34. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal on Global Optimization 11, 341–359 (1997)
35. Sun, J., Zhang, Q., Tsang, E.: DE/EDA: A new evolutionary algorithm for global optimization. Information Science (169), 249–262 (2004)
36. Teo, J.: Exploring dynamic self-adaptive populations in differential evolution. Soft Computing–A Fusion of Foundations, Methodologies and Applications 10(8), 673–686 (2006)
37. Tirronen, V., Neri, F., Kärkkäinen, T., Majava, K., Rossi, T.: A memetic differential evolution in filter design for defect detection in paper production. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 320–329. Springer, Heidelberg (2007)
38. Tirronen, V., Neri, F., Kärkkäinen, T., Majava, K., Rossi, T.: An enhanced memetic differential evolution in filter design for defect detection in paper production. Evolutionary Computation 16(4), 529–555 (2008)
39. Tsutsui, S., Yamamura, M., Higuchi, T.: Multi-parent recombination with simplex crossover in real coded genetic algorithms. In: Proceedings of the Genetic Evol. Comput. Conf (GECCO), pp. 657–664 (1999)
40. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)
41. Yang, Z., He, J., Yao, X.: Making a difference to differential evolution. In: Michalewicz, Z., Siarry, P. (eds.) Advances in Metaheuristics for Hard Optimization, pp. 397–414 (2008)

42. Yang, Z., Tang, K., Yao, X.: Self-adaptive differential evolution with neighborhood search. In: Proceedings of the World Congress on Computational Intelligence, pp. 1110–1116 (2008)
43. Zamuda, A., Brest, J., Boskovic, B., Zumer, V.: Differential evolution for multiobjective optimization with self adaptation. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 3617–3624 (2007)
44. Zielinski, K., Weitkemper, P., Laur, R., Kammeyer, K.D.: Parameter study for differential evolution using a power allocation problem including interference cancellation. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1857–1864 (2006)

Central Pattern Generators: Optimisation and Application

Leena N. Patel

Abstract. This chapter addresses optimisation of a class of biological neural networks, called Central Pattern Generators (CPGs), with a view to providing autonomous, reactive control to otherwise non-adaptive operators. CPGs are self-contained neural circuits which govern rhythmic motor activities such as locomotion, breathing and digestion. Neurons in this system interact to produce rhythmic oscillations without requiring sensory or central input. These phasic firing patterns can be adaptively adjusted, through neuromodulation, and in response to fluctuations in the environment. Thus, CPGs provide autonomous, self-modulatory control and are an ideal candidate to evolve and utilise for practical engineering solutions. An empirical study is described which generates CPG controllers with a wider range of operation than their counterparts. This work is precursory to producing controllers for marine energy devices with similar locomotive properties. Neural circuits are evolved using genetic algorithm techniques. The lamprey CPG, responsible for swimming movements, forms the basis of evolution, and is optimised to operate with a wider range of frequencies and speeds. Results demonstrate that simpler versions of the CPG network can be generated, whilst outperforming the swimming capabilities of the original network [34].

1 Introduction

Rhythmic motor behaviour plays a major role in any living organism, producing actions such as the regular gait of walking, or the slithering snake's body as it bends, alternating from side-to-side, or even the coordinated limb movement of an eight-legged spider. These rhythmic patterns are also evident in non-locomotive behaviours such as swallowing, respiration and digestion.

Leena N. Patel
The University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh, EH9 3JL, UK
e-mail: `L.Patel@ed.ac.uk`

The continuous, repetitive and voluntary nature of this class of movement distinguishes it from others, such as involuntary (and instant) reflexes (blinking, pupil dilation), stimulus-level responses (orgasm, sneezing) which are triggered once a threshold is reached and direct voluntary control (such as stretching, grasping).

The actuation of rhythmic movements relies on a central pattern generator or CPG, which is a specialised circuit, characterised by its generation of oscillatory or alternating motor patterns. It produces these regular patterns of output endogenously (i.e. without rhythmic sensory or central input). With *in vivo* preparations, this pattern of activity performs what is termed *fictive* locomotion, where the motorneurons fire in such a way that if they were still attached to their muscles, movements would occur.

This chapter describes the underlying structure, behaviour and performance of CPG architectures in biology. It provides several examples of these neural circuits and the functions they drive. Artificial representations of CPGs and their areas of application are then covered. A specific model is presented in more detail with an empirical study of the lamprey's spinal CPG. This is followed by a discussion on how this network is regenerated using evolutionary techniques, to increase the range in which the simulated lamprey swims. Based on the theory of natural evolution, a genetic algorithm is used to evolve alternative CPG configurations. Measures of fitness which steer the evolutionary process are constructed to reduce connectivity and produce efficient swimmers which can operate with a larger range of frequencies and speeds. This forms an initial step in our overall aim to develop bio-inspired reactive controllers for a very challenging engineering task in the area of marine energy where a wider range of operation is essential. Network evolution to drive reactive control of wave energy converters is discussed in the final part of this chapter.

2 Central Pattern Generators and Neuromodulation

A CPG is a self-contained network where populations of neurons interact to produce phasic (periodic on and off cycles) temporal and spatial activity. CPGs produce these rhythmic motor patterns, even in isolation from motor and sensory feedback. This characteristic was recognised as early as 1911 by Graham-Brown, where basic stepping was produced in the absence of descending or afferent inputs to the isolated spinal cord of a cat [18]. This important concept and discovery of the CPG has shaped and driven research on the neuronal substrates of invertebrate and vertebrate motor systems, including observations on humans with spinal cord injuries [8].

CPGs vary in anatomy and physiology, but in general, two conditions classify this rhythmic generator: 1) that two or more processes interact, passing activity between them through sequential increase or decrease of activity, 2) the system returns repeatedly to its starting condition through this interaction. A self-sustaining pattern of behaviour is thus produced. These enable precise timings of motor commands which actuate muscles or processes that operate in a synchronised manner (i.e. contracting or stretching, off or on). The following are examples of functions which operate with rhythmic patterns of activity and their underlying CPG networks.

2.1 Locomotion

In the analysis and artificial reproduction of locomotive control, CPG research plays a major role [1, 38]. The primary function of locomotor-CPG networks is to provide oscillatory motor commands with precise timings to coordinate efficient movement of related joints and muscles. The first modern evidence of such a neural network was produced by Wilson in 1961. He isolated the locust nervous system and demonstrated that it produces rhythmic output resembling the insect's flight patterns [47]. Since then, evidence has arisen for the presence of intra-spinal CPG networks which drive and coordinate locomotion in many animals. For instance, Sqalli-Houssaini et. al induce rhythmic locomotor-like activity by adding an excitatory amino acid receptor agonist (N-methyl-D,L-aspartate, NMA) to *in vitro* spinal cord preparations of neonatal rats. They demonstrate that even at birth, oscillatory patterns of activity are produced by these spinal neural networks with connections already established between peripheral sensory afferents and the CPG [45]. There is also evidence of this type of network in frog embryos [39]. What is interesting is that from a very early stage of development CPG networks are already functioning, interactive units of control.

Even though, the actual architecture of the CPG network is seldom observable *in vivo*, important aspects of their structure can be inferred by stimulation and observation of reactionary components. Many studies have been conducted with decerebrate cats (e.g. [18, 43]), all demonstrating the same principle rhythmic patterns of behaviour and control of different gaits, such as walking, trotting and galloping through altered levels of stimulation [43]. Studies have even found the presence of a human locomotive CPG, which is extremely robust and highly adaptable. The clearest evidence comes from Calancie et al. who witnessed step-like movements in a male subject who suffered a cervical spinal cord injury. Initially, he suffered total paralysis below the neck, but eventually regained some movement in his lower limbs. Still unable to support his own weight, when the subject lay down with extended hips, his lower extremities underwent step-like movements. *The movements (i) involved alternating flexion and extension of his hips, knees, and ankles; (ii) were smooth and rhythmic; (iii) were forceful enough that the subject soon became uncomfortable due to excessive muscle 'tightness' and an elevated body temperature; and (iv) could not be stopped by voluntary effort [8].*

A detailed example of a locomotor network is given in section 5 as it forms the basis of our wider research. This system is the spinal neural network of the lamprey, responsible for rhythmic swimming patterns by alternating motion from one side of its body to the other.

2.2 Respiration Pattern Generators

Breathing is a non-locomotive function governed by a CPG in many species. The amphibian brainstem/spinal cord preparation has been widely used to examine the mechanisms of respiratory rhythm generation (e.g. [46, 5]. It is a good example of a respiratory CPG, especially as there is evidence to suggest that the mechanisms which regulate rhythmogenesis and respiratory motor output in amphibians, share many common features with mammals [15].

Larval amphibians accomplish gas exchange mainly through rhythmic ventilation of the gills, but as they develop into mature frogs, lung ventilation assumes a greater role in gas exchange [7]. This is a most interesting neural network as it performs a transitional function, from an aquatic to a terrestrial respiratory system, involving a shift from gill to lung ventilation [7, 46, 15]. Worthwhile comparisons can be made because both the tadpole and adult frog can be studied using identical experimental techniques at all stages of development.

A study by Broch, et al. isolated brainstem preparations of larval (tadpole) and adult bullfrogs. Respiratory motor output from each CPG, measured as neural activity from cranial nerve roots, was associated with *fictive* gill ventilation and lung ventilation in the tadpole and with only lung ventilation in the adult [5].

With controlled conditions, typical neural activity is recorded from tadpole and adult preparations (shown in fig.1 (reproduced from [5]). Bursts of activity are clearly distinguishable between the two generations. Tadpole preparations demonstrate *fictive* gill bursts of low amplitude and high frequency while the bullfrog

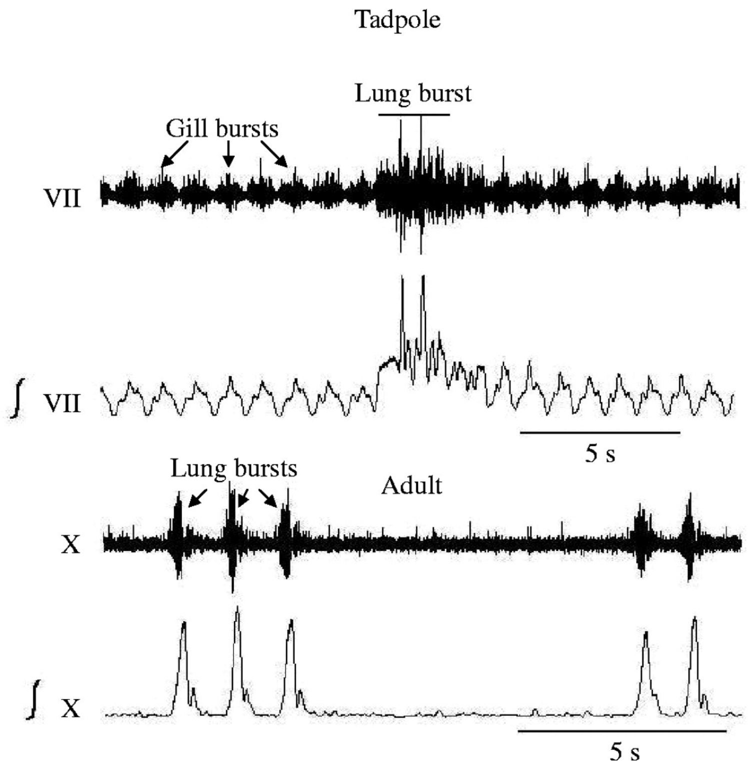


Fig. 1 Representative recordings from [5] of tadpole and bullfrog (*Rana catesbeiana*) respiratory brainstem preparations. Raw and integrated (∫) gill bursts and a lung burst are shown in the first two neural recordings and lung bursts of activity in the lower two.

tests present single (episodic) neural bursts of activity (high amplitude, low frequency) indicative of lung-related activity (see [46, 37]). Their results suggest that both mechanisms are dependent upon conventional chloride-mediated synaptic inhibition and that there may be a developmental change in the fundamental process driving lung ventilation in amphibians [5].

2.3 Heartbeat CPGs

Network-based rhythmicity is shown clearly by the leech heartbeat CPG. Two tubes pump blood through the leech's circulatory system, each alternatively constricting and relaxing. The CPG consists of eight pairs of interneurons. Of these, five pairs regulate the timing and rhythm of the heartbeat; they can reset and entrain the system, while the remaining pairs of interneurons coordinate motorneuron activity.

In fig. 2b, each heart neuron (HN) is indexed according to the extent along which its soma lies on the side of the leech's body. For instance, HN(L,2) is the neuron on the left hand side at segment two along the body of the leech. Notice that burst activity of neurons HN(R,4) and HN(L,4) (fig. 2b) fire out of phase with each other.

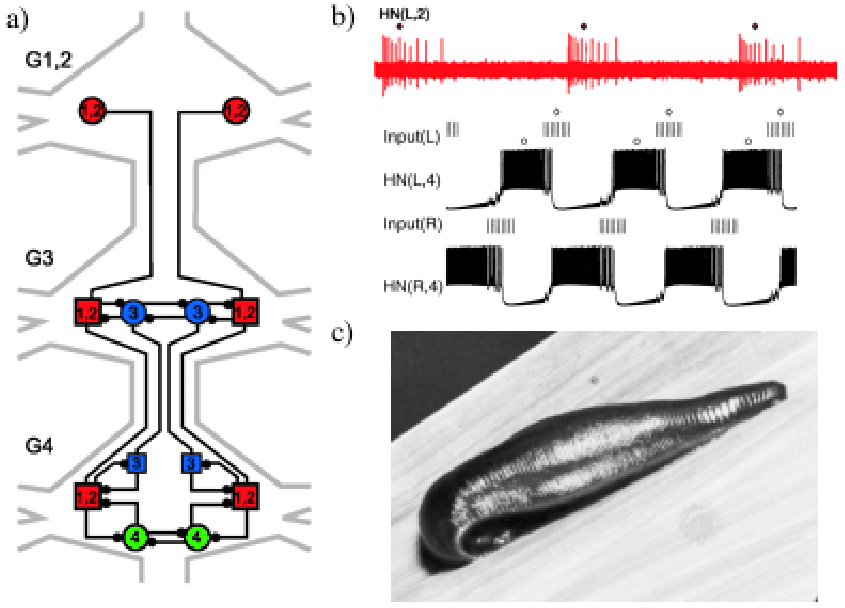


Fig. 2 a) The CPG network modulating heartbeat regulation in the leech (reproduced from [27]). Open circles represent cell bodies, open squares are sites of spike initiation and small filled circles represent inhibitory synapses. b) Neural recordings from some interneurons (HN(L,2) from [27] and HN(L,4), HN(R,4) from [21] c) the intact medicinal leech, *hirudo medicinalis*.

This is the kind of behaviour that produces antiphase patterns between two bilateral functions (inhaling or exhaling). Bursts of activity of each heart neuron also demonstrate the typical active and inactive cycles, which govern rhythmic muscle movement.

2.4 Swallowing Pattern Generators

A final example is the CPG responsible for swallowing patterns of activity. Swallowing involves the coordinated contraction of more than 25 pairs of muscles in the larynx, oesophagus and oropharynx. This complex interaction depends on a CPG located in the medulla oblongata, which involves several brain stem motor nuclei and two main groups of interneurons: a dorsal swallowing group (DSG) and a ventral swallowing group (VSG). Neurons in the DSG generate the swallowing pattern, while those in the VSG distribute commands to the various motorneuronal pools [26]. The swallowing CPG is an interesting one because of its flexibility. Some of its neurons can belong to several CPGs (e.g. the swallowing and respiratory control networks) and thus perform multifunctional roles [10].

2.5 Neuromodulation

Organisms must adapt their behaviour to meet the needs of their internal and external environments. As well as governing centrally-generated base rhythms, CPGs can be modulated to produce several different physical actions depending on the immediate needs of the animal. This family of different motor outputs results from internal and external innervation. Internally, neurotransmitters act on the CPG system to produce appropriate changes in its activity. Evidence suggests that related, but distinct functions can be performed dependant upon the level and type of neurotransmitter released. For example, in a type of sea slug called the *Tritonia diomedea* (fig. 3), a CPG modulates escape swimming, reflexive withdrawal and crawling whereby one function is unaffected by neuromodulation of another. Reflexive withdrawal is actuated in response to weak sensory input, escape swimming with strong sensory input [35] and crawling occurs after escape swimming has ceased. Dorsal swim interneurons (DSIs) within the pattern generator release serotonin to convert to swim mode, while the application of serotonergic antagonists prevents the swim pattern.

The ability to rapidly convert from one mode to another is a fascinating mechanism of CPG networks. Feedback from the environment cause chemical reactions which in turn enable the neural system to respond and change its mode or rate of behaviour. This type of self-regulatory control has gained much interest in the Artificial Intelligence community, particular in the area of robotics where autonomy is crucial for developing responsive systems.

3 Artificial Central Pattern Generators

Although CPGs are generally discussed in relation to biological entities, they can also be replicated artificially for robotic applications [36, 1, 38]. The only requirement is that they produce continuous rhythms of behaviour after an initial stimulus. Examples of artificial neural networks (ANNs) based on CPGs include biophysical models, connectionist models and systems of coupled oscillators. Biophysical models are detailed depictions incorporating chemical properties of individual neurons in the system such as ion pumps and channels (e.g. [20]). They tend to closely resemble actual biological networks. Connectionist models comprise networks of simplified neuron units (e.g. [12]). These networks demonstrate the typical activity of the system using less realistic models of neurons. A detailed example is provided in section 5 where Ekeberg's connectionist network of the lamprey CPG is detailed. These representations demonstrate that complicated neuronal mechanisms are not necessary to produce oscillatory patterns and that modelling connectivity itself is sufficient. Finally, at the most abstract level, coupled oscillator networks

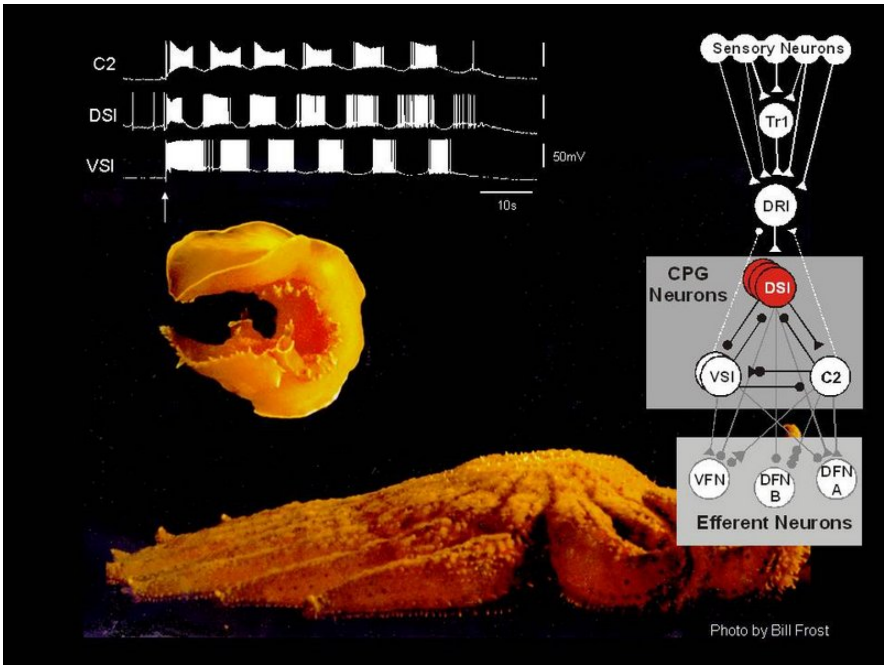


Fig. 3 The sea slug *Tritonia Diomedea* escaping from a sea star, *Pycnopodia*. The top of the diagram shows simultaneous intracellular electrophysiological recordings taken from an isolated brain from the three central pattern generator neurons: C2, DSI and VSI. A body wall nerve was stimulated at the arrow, producing oscillatory discharges with activity alternating between DSI and VSI neuron groups. These result in dorsal and ventral body flexions indicative of escape type swimming. The right hand side displays the CPG circuit from sensory neurons to efferent output. Image reproduced from [28].

model only the behaviour and dynamics of neural populations. They focus on properties of the entire network rather than just individual neurons or sets of neurons to produce phase relations. For example, see [25] where coupled nonlinear oscillators are used to construct a salamander-type robot with a CPG for its body coupled with limb CPGs to enable swimming in water and a trotting gait on land.

Typically, ANNs, inspired by biology and its architecture, form the basis for autonomous control in numerous applications [4, 42, 29]. Such technology is developed to provide a degree of intelligent control to an otherwise non-adaptive operator. For example, sensory robots can navigate and explore inhospitable areas such as the oceans [2] and space [14, 44]. Although they behave in a complex manner, they are designed according to simple control principles from biological exemplars such as stick insects and lobsters.

Unlike conventional approaches, these biomimetic systems are not reliant upon error-prone and expensive reprogramming or fine-tuning by the operator. As a result, the engineered solutions are often more efficient, productive and independent, whilst less labour and time intensive.

Optimum performance of neural circuitry can be generated with evolutionary techniques such as genetic algorithms. However, despite their ability to find superior solutions, evolutionary techniques are not frequently deployed to increase the performance of CPGs (with the exception of [24, 32, 33]). Instead, mainly in the robotics domain, they are used to computationally calculate parameters of neural controllers for locomotion such as biped walking [41], hexapod limb coordination [3] and *anguiliform* swimming [23] where manual, intelligent configuration is virtually impossible. The motivation to design better artificial intelligence (AI) systems for real-world engineering problems underpins the work of this research, and ultimately uses genetic algorithms to generate task-specific, optimised CPG controllers for wave energy devices. As with most AI solutions, inspiration is provided by models of real biological networks and these form the basis of evolution.

4 Optimisation with Evolutionary Algorithms

Evolutionary algorithms (EAs) are search and optimisation techniques for finding optimal solutions to a given problem. They include methods such as:

1. Particle Swarm Optimisation (PSO) [30] which is based on the flocking behaviour of birds (or swarming behaviour of bees).
2. Ant Colony Optimisation (ACO) [11] based on how ants leave pheromone trails along the shortest route to food. These trails diffuse with time to enable newer, shorter routes to dominate.
3. Estimation of Distribution Algorithms (EDAs) [31] which determine fit solutions according to probabilities of where good solutions lie in the solution space.
4. Genetic Algorithms (GAs), based on survival of the fittest mechanisms in nature where good parent solutions are paired to produce child solutions which are tweaked and then evaluated.

Each technique may differ but their overall aim is to find optimal solutions. Of these methods, the genetic algorithm is the most commonly used.

Inspired by Darwin's theory of natural selection and genetics, a Genetic Algorithm (GA) [16, 17, 13] computationally encodes candidate solutions as chromosomes, within which genes represent evolvable elements. The search is directed towards better solutions by the careful construction of an evaluation function. Individuals that score well are more likely to survive and be chosen for the basis of subsequent generations.

Evolution typically starts with a randomly generated population of individuals, covering the entire solution space (although sometimes solutions can be "seeded"). It cycles through several generations, evaluating the fitness of each individual in the population. Three operations are applied each generation, which are selection, variation and elimination. *Selection* involves choosing pairs of parent chromosomes, based on their fitness ranking. Generally, fitter individuals have a greater chance of being selected depending on the level of elitism adopted. *Varying* some of the genes of some chosen parents is the next stage in the GA process. This results in new child candidate solutions which are evaluated in the subsequent generation. It is considered that children will possess good quality genes from their parents and with some tweaking, may result in better fitness. The level of genetic modification and number of cells to vary can be stipulated with probability ratios. Finally, *elimination* involves rejecting the worst solutions, being replaced by higher ranked new candidates to maintain a consistent overall population size.

If well constructed, this evolutionary approach is resistant to problems of local minima that beset other algorithms. Ideally, a good combination of exploration and honing is required. At the appropriate level of search, the GA should converge to find optimal solutions. Furthermore, it is a common and efficient choice for optimisation and exploring the space spanned by a model. For these reasons, this tool is used to re-evolve alternative, wider functioning and more efficient swimming CPGs based on an invertebrate called the lamprey.

5 Modelling the Lamprey's CPG Network, Musculature and Environment

The lamprey (shown in fig. 4a) is an eel-like fish which propels itself by propagating an undulatory wave from its head to its tail. A CPG network (schematically modelled in fig. 4b), along its spine, governs this swimming module by causing rhythmic activity of motorneurons. These actuate muscles which cause motion to alternate between the two sides of the fish's body.

This vertebrate's CPG has been mapped thoroughly after careful analysis and innervation of reactionary components [19] and modelled artificially [12]. This has been possible because the intact spinal cord can survive *in vitro* for several days after being removed, because it is a relatively simple network, with few neurons, and it can be stimulated to produce the *fictive* swimming (without tonic input) motion indicative of a CPG.

a)

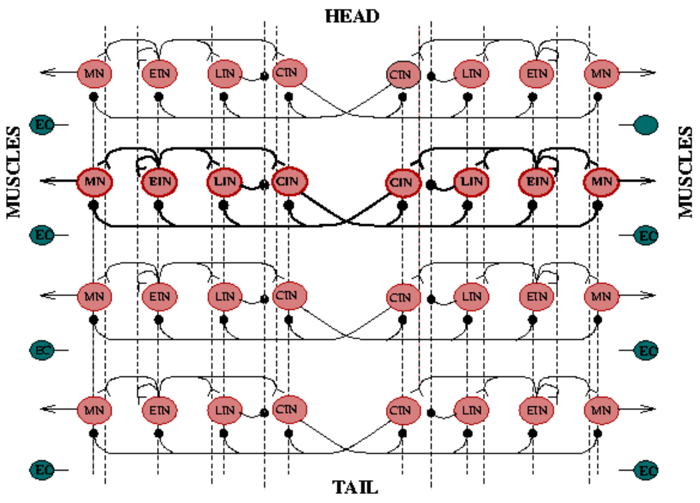
b)


Fig. 4 a) the lamprey, a vertebrate belonging to the family *Petromyzontiformes*; b) the connectionist model of the lamprey's spinal CPG. Excitatory connections are shown as closed circles and inhibitory input as open forks.

Several copies of an oscillatory neural circuit (one is highlighted amongst the four shown in fig. 4b) are interconnected along the fish's spine. On each side of a single network there are four types of neuron governing rhythmic patterns as follows:

1. On the dominantly active side, the excitatory interneuron (EIN) group excites neurons on its side (ipsilaterally). Meanwhile, contralateral inhibitory interneurons (CINs) inhibit all cells on the opposite side (contralaterally). This results in the ipsilateral motorneuron (MN) activating the muscles.
2. After a short delay, a burst terminating mechanism causes control to switch sides. Burst termination is caused by the lateral inhibitory interneurons (LINs) becoming active later in the cycle, which suppresses the active CIN, relinquishing control from one side and building it up on the other.

This ensures that only one side is active at a time, with periodic transfer of control between sides. This behaviour continues while the segment receives base excitation from the brainstem.

Tonic (i.e. non-oscillating) signals (global excitation) from the brainstem (or from neurotransmitters when *in vitro*) regulate the frequency of oscillation. This principle of control is reported in [43], where different levels of stimulation on a decerebrate cat's brainstem results in walking, trotting or galloping. In the lamprey, oscillation frequency and speed of swimming can be adjusted by adding neurotransmitter agonists such as amino acids [9] or L-Dopa [40]. A further tonic input, referred to as extra excitation, is applied to the CPG's headmost segments to invoke a phase lag along the length of the lamprey's body and this causes forward motion. This is achieved *in vitro* by applying a greater concentration of neurotransmitter to the rostral section of the network. For clarity, these tonic inputs are not shown in fig. 4b.

Finally, edge cells (ECs) seen in the schematic (fig. 4b), are external sensors, which inhibit contrateral activity and excite ipsilateral activity. They provide feedback to the circuit from external forces, and invoke adjustments in activity, which maintain straight line swimming.

The lamprey's CPG network, described here, can be reduced to a simplified connectionist model. Neurons are non-spiking and belong to a population of similarly functioning nerve cells. The CPG receives delayed excitatory and inhibitory input and its output is calculated from first order differential equations:

$$\dot{\xi}_+ = \frac{1}{\tau_D}\Big(\sum_{i\varepsilon\Psi_+} u_i w_i - \xi_+\Big), \tag{1}$$

$$\dot{\xi}_- = \frac{1}{\tau_D}\Big(\sum_{i\varepsilon\Psi_-} u_i w_i - \xi_-\Big), \tag{2}$$

$$\dot{\vartheta} = \frac{1}{\tau_A}(u - \vartheta), \tag{3}$$

$$u = max(0, 1 - \exp\{(\Theta - \xi_+)\Gamma\} - \xi_- - \mu\vartheta) \tag{4}$$

In this set of equations, output u represents the mean firing frequency of each neuron population. A time delay (τ_D) is applied to summed excitatory (ξ_+) and inhibitory (ξ_-) inputs, Ψ_+ and Ψ_- represent groups of presynaptic inputs (excitatory and inhibitory respectively) and w_i is the weight associated with each input (eqns. 1-2). The term u_i denotes inputs received from neurons within the single network and from neurons of connected segments, whereas u refers to output of a single neuron. A transfer function (eqn. 4) provides saturation for high levels of excitatory input. A leak is included as delayed negative feedback (eqn. 3) and is subject to a time delay (τ_A). The parameters threshold (Θ), gain (Γ) and adaptation rate (μ) in eqn. 4 are tuned to match observed characteristics in some real neurons (see [12]).

Assymmetric initialisation of these equations leads to out-of-phase bursts of activity and involves setting $\xi_+(0)$=1, $\xi_-(0)$=0 for all left neurons and $\xi_+(0)$=0, $\xi_-(0)$=0 for all right neurons. This enables calculation of the initial output value(s) (u) which are used in subsequent differentiations. Weights, neural parameters and time delay values of the biological model are shown in table 2, section 6.3, originating from [12].

Individual CPG networks are coupled to their neighbours via interneural connections towards the head (rostrally) and tail (caudally). These are depicted as vertical dotted lines in fig. 4b. Interconnections are important as they coordinate longitudinal movement by generating a time delay between successive CPG units. Phase lags are 1% of the period of oscillation and so a single wavelength can be maintained along the length of the body independent of swimming velocity.

Details of intersegmental connectivity in the real lamprey remain unknown, thus, in the original model Ekeberg [12] applied symmetrical connections in both directions. He achieves this by dividing each synaptic weight value by the number of CPG units it is linked to. Since neurons at each end of the complete CPG have fewer afferent connections, their synaptic weights are calibrated accordingly.

A complete simulated CPG interacts with a model of its body in water to demonstrate the expected *anguiliform* swimming behaviour [12, 24, 34]. The mechanical body comprises ten rigid links, each 30mm long, and corresponding to ten neural segments. Their movement is constrained, forcing them to stay connected, by joints with one degree of freedom. Width (generally 30mm) and mass of the lamprey decrease at the caudal end (i.e. the tail narrows). As in Ijspeert's model [24], mass and inertia of each link is calculated by assuming that the density of the lamprey is constant and equal to the surrounding water. Muscles connecting each link are modelled as a combination of springs and dampers. The forces acting upon each link are:

1. *Water forces* apply both horizontal and vertical pressure to the body. These depend on the speed of the body relative to the water and in the model they can be reasonably approximated by considering the water as stationary and applying a 3D water force vector on each link.
2. *Inner forces* exert pressure from neighbouring units. These joint constraints ensure links remain connected together at all times.
3. Muscle *torque forces* prevent links from bending in both directions at once. A linear relationship can be considered to exist between motorneuron activity, these forces and resulting muscular spring constants. Torque forces function as feedback from the neural CPG to the mechanical model. This feedback loop is completed with stretch sensitive edge cells providing information about the local curvature of muscles (assumed to be equal to the length of the body) to the CPG.

The entire lamprey swimming model was first defined in [12], and refined in [32, 33, 34] to more realistically fit physical data. It characterises the biological lamprey's swimming network with some accuracy [12], provides a tool for further exploration of network connectivity and activity (e.g. [34]) and offers potential for developing systems for more complex control. Achieving this type of unmanaged, responsive control in unpredictable conditions is a major challenge in engineering. Such problems can be resolved through CPG architecture coupled with the power and speed of evolutionary computing. The flexibility of the CPG network parameters and swimming capabilities are explored further with the aim to develop a solution in the area of marine energy (discussed in section 7). An initial goal towards this solution is to develop CPGs capable of swimming in wider operative ranges. To achieve this, both the neural CPG and mechanical model of the network interacting in water are modelled (as described in section 5) and then further enhanced with genetic algorithms.

6 SuperLamprey Controllers: Optimised to Increase Swim Ranges

In order to remodel the circuitry for a new and complex control task, the flexibility of the lamprey CPG requires further exploration. Detailed analyses of candidate biological neural systems are essential and must explore whether nature's evolved configurations are unique, whether simpler versions perform effectively, and whether their operation range can be optimised for similar mechanical engineering tasks.

Two GA processes are implemented to enhance the capabilities of the simulated lamprey CPG: the first evolves synaptic weights and neural parameters of an independent neural module and the second generates interconnections between the best solutions (of the first phase) to produce complete multi-segment controllers. The goal of the first GA is to generate a single rhythmic oscillator [32] which operates over a wider range and is less complex, the latter property is important for eventual silicon reproduction. The second GA takes improved CPG units and determines longitudinal connections between neighbouring segments, with optimum performance signified by their capacity to control swimming at different speeds, oscillation frequencies and phase lags between segments.

The decision to implement the evolutionary process in two stages is for the following reasons: (1) computational efficiency - invoking the GA in one process would result in the assessment of far fewer candidates yet over a much longer period, wasting valuable resources, (2) to avoid lengthy testing of linked controllers which have already failed to oscillate in isolation, (3) reducing the problem into subgoals is a widely used and accepted method of developing plausible solutions, (4) each network component (single-segment oscillators) must function, even in isolation [12, 32, 33] and our approach guarantees this. The condition of isolated segments operating independently is also imperative for our engineering solution if continued operation is to be maintained even when part of the system fails.

A random initial population is generated for each experiment of an evolutionary process. They loop through the standard operations of selection, variation and rejection, each generation. Selection involves a fixed number of parents being chosen according to rankbased probability. An elitist procedure is adopted, selecting the fittest individuals of each generation to create offspring. Two-point crossover, mutation and pruning (for isolated CPGs) are applied to vary candidates. Finally, the worst solutions (denoted by their fitness ranking) are rejected, being replaced by higher ranked new solutions. Parameters of both GA procedures are outlined in table 1.

Probability rates and ranges in table 1 describe the degree to which chromosomes are altered. For example, crossover (where substrings of paired parent chromosomes

Table 1 Genetic evolution parameters for generating single rhythmic controller solutions and complete, multilinked swim modules

	Unitary CPG (GA1)	Multilinked CPG (GA2)
population size:	100	60
number of children:	30	18
crossover probability:	0.5	0.5
mutation probability:	0.4	0.4
mutation range	0.2	0.2
pruning probability	0.1	-
pruning range	1.0	-
number of generations	500	50

are swapped) occurs with 50% chance. Pruning is a non-standard GA procedure, where every connection is considered independently for removal (setting it to 0) with a probability of 10%. This is to explore solutions with fewer connections. Of course, their calculated fitness determines the success of this arbitrary removal of connections on a solution-by-solution basis. In addition to this arbitrary pruning through the lifetime of the GA, weak connections of the final population which do not affect neural activity are eliminated through a final prune. This is applied by setting any weight below 0.1 to 0, provided that doing so does not diminish this individual's fitness value. If the pruned candidate is inferior, the original value is reinstated. This procedure is repeated with decrements of 0.02 until all ineffectual or weak connections are eliminated.

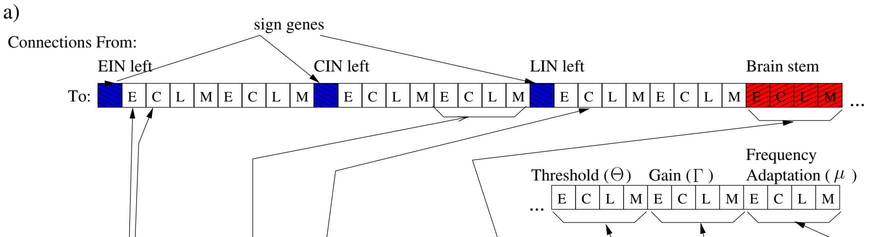
The properties in table 1 are held consistent with [24], to ensure confounds are not introduced and because they generate satisfactory results within a reasonable process time. The following sections outline the distinct characteristics of each GA, including their genetic composition and fitness criteria.

6.1 GA1 - Evolving Independent CPG Oscillators

The primary GA optimises independent lamprey CPGs, seeking solutions with improved performance ranges and low-level system complexity. Ekeberg's artificial network [12] featured hand-tuned network values, developed through measurement, trial and error. Ijspeert [24] used a genetic algorithm to evolve synaptic inputs. In my work [34], these are generated together with neuron-specific parameters (threshold, gain and adaptation rate) that describe the dynamics of the model neurons, exploring their diversity, while testing the true flexibility of the modelled CPG. However, the functional form and circuit structure of Ekeberg's original model is maintained to ensure consistency with the underlying biology.

A real value GA is used, comprising decimal numbers rather than traditional binary digits. Individual solutions are encoded as fixed length strings of 43 genes. Each gene corresponds directly to one evolvable parameter of the neural configuration as shown in fig. 5.

a)



b) Table of synaptic weight values

from: / to:	EINl	CINl	LINl	EINr	CINr	LINr	Brain Stem	Θ	Γ	μ
EINl	0.4				−2.0		2.0	−0.2	1.8	0.3
CINl	3.0		−1.0		−2.0		7.0	0.5	1.0	0.3
LINl	13.0				−1.0		5.0	8.0	0.5	0.0
MNl	1.0				−2.0		5.0	0.1	0.3	0.0
EINr		−2.0		0.4			2.0	−0.2	1.8	0.3
CINr		−2.0		3.0		−1.0	7.0	0.5	1.0	0.3
LINr		−1.0		13.0			5.0	8.0	0.5	0.0
MNr		−2.0		1.0			5.0	0.1	0.3	0.0

Fig. 5 a) An example chromosome solution for the GA evolving a single oscillating network. b) Table showing values corresponding to each evolved gene in the chromosome.

Fig. 5 visually depicts the structure of each solution chromosome with corresponding gene values relating to weights within the CPG (shown in the table of fig. 5b). Each chromosome is a vector of values representing: synaptic connections from EIN (E), CIN (C), LIN (L), Brain Stem Input, Threshold, Gain and Adaptation (as labelled above the chromosome) to E, C, L, MN (labelled within the chromosome). The symmetric nature of the network means that only half of the values require coding into the chromosome. Note that the values in this particular example correspond to Ekeberg's original model. Finally, the sign (excitatory or inhibitory) of each neuron group is contained in three chromosome units. These determine whether the connection is excitatory or inhibitory. Motorneurons only connect to muscles and so their outputs are not evolved. The single CPG unit GA guides solutions according to a fitness function (detailed in [34]), designed to select candidates which favour effective oscillatory behaviour. The following are the objectives incorporated into this evaluation together with justification for their inclusion:

1. Frequency is controllable by simple tonic excitation from the brainstem. It should increase monotonically with input levels. This is to enable variable control of oscillation frequency which in turn varies extensor or flexor phases of muscles.
2. Oscillations must be regular and have only one peak of activity each period. An imperative feature of CPGs is regular activity to ensure cyclic phases of on and off states so that the muscle contracts once it has reached the extent of its stretching phase. This makes this a necessary condition of new solutions.
3. Motorneuron activity must alternate between left and right sides of the CPG. Out-of-phase activity is crucial for rhythmic swimming in the lamprey. This ensures

only one side of each particular section of its body is active at any time. Again, this is a necessary condition for CPG function.
4. Oscillators operating over a wider frequency range are highly favoured. In order to demonstrate improved control, it was considered important to have controllers which performed outside the limits of the biological model. This is also a goal for our intended application in wave energy, where a wider operation bandwidth will be required.
5. The biological frequency range must be included within the operating range of the new solution. Although not essential for marine energy solutions, this was an important aspect to enable a basis for comparison between the biological model and newly evolved CPGs. The evolved solution should perform with wider control ranges than Ekeberg's original solution.
6. Low connectivity is desirable. For converting any CPG solution into silicon, simplicity in network configuration is of major importance. This would make the system more robust, easier to implement and cheaper to manufacture and maintain. Low connectivity is encouraged via a pruning operator (described in section 6).

6.2 GA2 - Evolving Linked Oscillators

As described in section 5, the lamprey comprises several interconnected oscillatory segments. For the whole body to coordinate movements, oscillators need to be linked to their immediate neighbours. Therefore, the function of this GA is to evolve the extent of interlinking connections to coordinate efficient swimming. Still retaining the basic architecture of Ekeberg's model, intersegmental connections between 100 copies of a fixed segmental network are generated. The best segmental oscillators of the previous evolutionary stage (section 6.1 and in [32]) are used and five discrete evolutionary experiments invoked.

Candidate solutions are coded into integer-valued chromosomes, with genes depicting the extent of connections in rostral and caudal directions. Each chromosome comprises 51 genes. Owing to Left-Right symmetry, A CPG's 96 rostral/caudal interconnects are coded as 48 of these. Each has a value between 1 and 12 to incorporate biological prototype values. The other three genes denote whether the inputs are excitatory or inhibitory. These sign genes are preassigned according to the value this connection held in the unitary oscillator and therefore not evolved.

The fitness function (detailed in [34]) rewards solutions based on their ability to control swimming with wide operation bandwidths. These include large ranges of speed, oscillation frequency and phase lags between segments. Stated as objective criteria, multilinked controllers should:

1. be able to alter the oscillation frequency monotonically (with global excitation) and wavelength of undulation (with extra excitation) independently,
2. generate stable oscillations within each CPG unit, with coordinated phase differences to enable travelling undulations of the body, and
3. be able to change the speed of swimming by altering the CPG's oscillation frequency or the wavelength of undulations [33].

These are implemented to ensure efficient, smooth and linear control and to remain within the scope of the biological mechanism for swim control. Also, in line with the biological CPG, emphasis is placed on controllers which invoke swimming with a wavelength corresponding to the length of the fish's body (i.e. phase lag of 1% per segment). Resulting CPG controllers from both evolutionary processes are described in the following section.

6.3 Results of Single- and Multi-Segment GA Phases

Neural weights and parameters of an independent CPG module are evolved as a first-phase genetic algorithm (GA). A second GA takes the best of these solutions and evolves interconnects between neighbouring segments.

Fifty percent (20 experiments) of the first process generate improved oscillators than in [12, 24]. At the second GA stage, five of the best are chosen and four of these demonstrate wider swim ranges when interconnected as complete swim modules. The decision to terminate processes at 500 and 50 generations, for each GA phase respectively, is because most of the populations are stable by this point. Simulation times for the second evolutionary algorithm are significantly greater and so extra process time for little gain seemed unnecessary.

Most evolved solutions in our study demonstrate improved control. The statistics and neural configuration of the best of these is compared (table 2) with the original CPG prototype [12] (where all values were hand-tuned) and the best fixed parameter controller of [24] (where neuron-specific parameters threshold, gain and adaptation rate were hand-tuned values of [12]).

An interpretation of the results presented in table 2 is as follows:

Fitness - evolved CPG oscillators (of the first GA process) produce fitness values of 0.15 to 0.8. Of these, 90% outperform Ekeberg's prototype (fitness 0.11) and 30% out-evolve the fixed parameter (FP) networks (best of [24] is 0.31). Therefore, generating both neural weights and neuron specific parameters proves crucial to the development of high-performance networks.

Linking these via interconnections (the purpose of the second GA) also demonstrates improved swimming performance, with the controller of our study receiving a fitness value of 0.51 compared to 0.2 (biological model) and 0.16 (FP).

In one case, an improved CPG unit failed when it was interlinked to form a multisegment controller. This was due to poor independent control of oscillation frequency and phase lag. This demonstrates the importance of the second phase GA; and more generally that a good oscillator does not necessarily mean it will operate well when cross-coupled.

It is worth noting that Ijspeert's best segmental oscillator (shown in table 2) did not perform as well as the biological prototype when coupled to its neighbours, also confirmed by his results [24] and that another controller superceded it. However, this controller is still not as effective as the best lamprey CPG evolved in our experiments.

Table 2 Comparison of statistics and CPG configurations of the biological [12], fixed parameter [24] and evolved controller [34]. In column order, for each test, the table shows the CPG and level of intra-CPG connectivity (Conn), resulting objective values (GA1 and GA2), operative ranges of frequency, speed and phase lag, synaptic weights with the extent of cross-coupling in square brackets, brainstem input (BS), and finally, evolved neural parameters of threshold (θ), gain (Γ) and adaptation rate (μ). Due to the symmetrical nature of these controllers only half the inputs need to be shown. The complete weight set is derived by substituting l (left) with r(right) and vice versa.

CPG and Conn[a]	**FitV**[b] **GA1, GA2**	**Frq**[c] **Range (Hz) Spd**[d] **Range (m/s) Lag Range (%)**	**Synaptic Intra-CPG weights and Inter-CPG connection extent [rostral, caudal]**								**Neural Parameters**		
			From: To:	EINl	CINl	LINl	EINr	CINr	LINr	BS	θ	Γ	μ
bio	0.11,	1.74 - 5.56	EINl	0.4	-	-	-	-2.0	-	2.0	-0.2	1.8	0.3
	0.2	0.01 - 0.45		[2,2]				[1,10]					
26		0 - 1.165	CINl	3.0	-	-1.0	-	-2.0	-	7.0	0.5	1.0	0.3
				[2,2]	-	[5,5]	-	[1,10]	-				
			LINl	13.0	-	-	-	-1.0	-	5.0	8.0	0.5	0
				[5,5]				[1,10]					
			MNl	1.0	-	-	-	-2.0	-	5.0	0.1	0.3	0
				[5,5]				[5,5]					
FP	0.31,	1.2 - 8.0	EINl	-0.8	-3.8	-	-0.9	-0.7	-	0.8	-0.2	1.8	0.3
	0.16	0.06 - 0.41		[12,4]	[12,10]		[5,10]	[1,10]					
22		0.73 - 1.37	CINl	-	-	-	-3.5	-3.7	-	13.0	0.5	1.0	0.3
							[2,2]	[9,9]					
			LINl	-	-	-	-	-	-	-	8.0	0.5	0
			MNl	-0.4	-3.2	-	-	-	-	3.8	0.1	0.3	0
				[9,2]	[8,1]	-	-	-	-				
Evo	0.8,	0.99 - 12.67	EINl	-	-4.6	-	-	-	-	3.06	-1	0.7	0
	0.51	-0.01 - 0.6			[3,4]								
16		0 - 1.59	CINl	5.53	-	-	-	-2.9	-	-1.18	-1	0.48	0
				[1,8]				[10,1]					
			LINl	-	-	-	-	-	-	-5.0	-1	0.7	0
			MNl	-	-4.3	-	-	-	-	10.8	-1	0.27	0
					[8,6]	-	-	-	-				

[a] Conn = Connection Density, [b] FitV = Fitness Value, [c] Frq = Frequency, [d] Spd = Speed.

Connection density - sparse connectivity is far more efficient computationally and thus a very important consideration for silicon reproduction. This is especially the case when there are several copies of that same unit (as with multilinked controllers). Compared to the former models, the least densely connected CPG unit is produced in our results (with 16 vs. 22 and 26 intra-connections).

Frequency range - the range of frequencies covered by the best evolved controller is 0.99 - 12.67 Hz. This is substantially greater than the frequencies covered

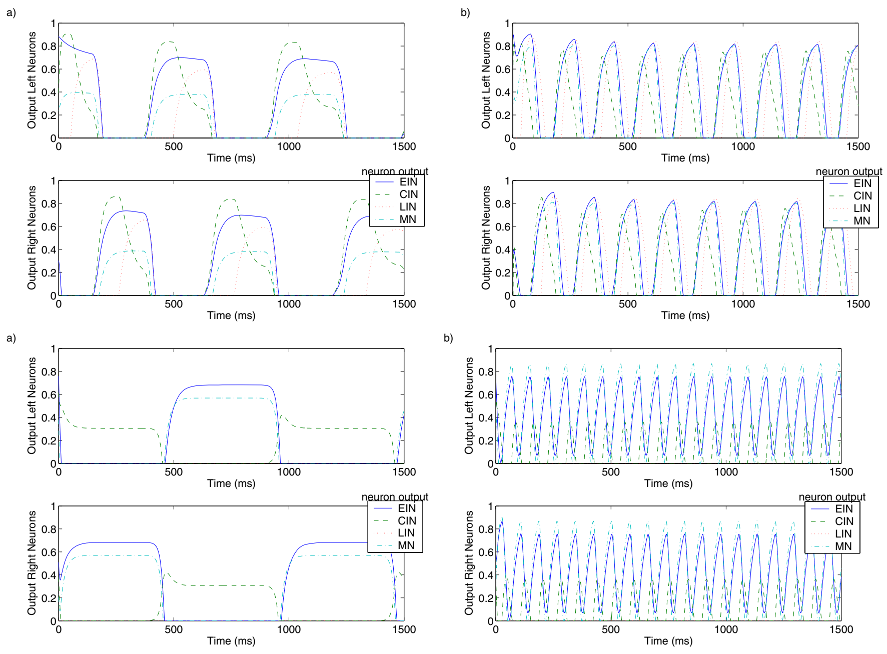


Fig. 6 Neuron behaviour of Ekeberg's biological network [12] (top four graphs) compared to our evolved controller [32] (bottom four graphs): part a) is the lowest oscillation level and part b) the highest frequency, for each CPG network. Network operation is simulated for a fixed duration of 3000ms (for clarity, oscillations for only 1500ms are shown in the charts), with asymmetric initial conditions (all left neurons excited).

by the biological and FP networks (1.74 - 5.56 Hz and 1.2 - 8 Hz respectively). This demonstrates over 100% of a performance increase over prior work where key variables remained static. Frequency is modulated by varying the tonic input, termed global excitation (as it is applied to the whole network). Lowest and highest frequencies are displayed in fig. 6b comparing the biological CPG and our best evolved solution.

The sets of graphs in fig. 6 demonstrate activity of Ekeberg's CPG (top fig. 6 a and b) with our evolved network (bottom fig. 6 a and b). It is evident from them that the evolved network operates with a broader frequency bandwidth than Ekeberg's model. Each set of graphs displays activity of the left neurons (top of each set) and the right neurons (bottom of each set). In all cases, the left-neurons operate antiphase to right-neurons; therefore only one side is active at any time as per stipulated conditions for *fictive* swimming. Other characteristics developed into the fitness evaluator include regular, oscillatory activity (see objective 2 in section 6.1), which is also demonstrated by each solution.

Speed range - The multilinked controller, when interacting with the environment swims within a greater range of speeds than the other networks; numerically,

-0.01 - 0.6 m/s (compared with 0.01 - 0.45 m/s (biological) and 0.06 - 0.41 m/s (FP)). The negative speed recording (-0.01) is due to the kind of wriggling the lamprey performs and not considered an adverse effect.

Lag range - The phase lag between interconnected units ranges from 0 - 1.59% (compared to 0 - 1.165% (biological) and 0.73 - 1.37% (FP)). This is recorded at the midrange of oscillation frequency and by monotonically altering the extra excitation tonic input.

Synaptic weights and interconnections - The magnitude of possible configurations due to connection permutations and synapse strengths can produce very diverse solutions. This is exemplified by the notable differences in type, quantity, sign and weights of active neurons in each solution shown in table 2. It can also be seen graphically by the chart activity shown in fig. 6 where different neurons interact to produce the eventual motorneuron burst patterns. Unlike the biological prototype, oscillatory activity of the best evolved solution occurs by opposing CIN neurons inhibiting each other, while the active EIN excites the CIN ipsilaterally. The non-dormant CIN also suppresses the EIN and MN neurons on its side, thus MN (and EIN) activity is asynchronous with the CIN on each side.

Although the neuron-naming scheme has been kept for comparison purposes, it is worth noting that each neural population loses its functional meaning and even the sign it had in the biological model. This is even true of prior evolved controller networks (i.e. [24]). The only feature they retain are the original dendritic time delays of τ_D = 30ms, 20ms, 50ms and 20ms for the neuron types EIN, CIN, LIN and MN respectively and τ_A = 400ms and 200ms for EIN and CIN. These accord the original solution in [12].

Neural parameters - As with synaptic weights and interconnections, a distinct pattern does not emerge for neuron-specific parameters (threshold, gain and adaptation rate) when solutions are compared. The evolved network parameters seem to bear no commonalities with the fixed parameter CPGs. Furthermore, the best evolved solution is simpler through the elimination of frequency adaptation (μ = 0), removing the need for the leak (eqn. 3, section 5) without affecting preferred swimming capabilities. Note that this parameter's behaviour should not be confused with the role of tonic input changes or edge cell feedback, which perform frequency modulation of the interlinked swim system. Rather, parameter μ relates to an individual neuron group generating time-changing rather than constant output. Since the system does not seem to need this feature to meet its objectives, there is no reason to force its existence. This variation between CPGs demonstrates diversity in the solution set and suggests that there is a spectrum of continuous models as opposed to a distinct number of species.

6.4 Discussion

This study demonstrates that Ekeberg's CPG model [12] of the lamprey spinal controller is not a unique solution and that many simpler versions with wider operative ranges can be generated. Evolved networks operate with a wider frequency, phase lag and speed range with independency of control. Improvements of over 100% are achieved. Our methodology builds upon previous work [24], but improves *anguiliform* swimming performance substantially by relaxing some constraints and exploring variables (threshold, gain and adaptation rate) previously fixed in value. Therefore, the true flexibility of the CPG network is assessed.

In terms of system connectivity, evolved networks are vastly simpler than the biological prototype [12] and fixed parameter solutions [24]. They have reduced parameter sets, which also simplify the original equation set. This is desirable if an integrated VLSI controller is to be developed, especially where this network is one of many functional units of a complete, dynamic system. Furthermore these improvements do not come at the expense of performance.

Since these controllers are developed with specific goals in mind they do not necessarily incorporate all the functionality of the biological prototype. For example, constraints of the natural lamprey may include attributes for mating, searching for food and / or escape swimming. However, since these actions are not essential in a wave power device, it is not worthwhile building them into the new control unit just to keep them in line with the biological model. Instead, targeting specific and necessary behaviours (i.e. rhythmic patterns and adaptation) within the architecture and basic routines of the lamprey CPG, produces streamlined, better solutions.

Additionally, there is absolutely no reason why a natural evolved solution should be optimal even in its own multifunctional capabilities, since evolution does not work that way. Natural lamprey parameter choices could be a result of historical contingency, that is, they are what the genome could build given what it had available at that time. The important point is that it is not possible to know why the biological lamprey neurons use the parameters they do, but if the lamprey could freely optimise for performance, perhaps it would choose different ones. This form of behaviour is intriguing in the context of real biological systems. It is potentially of enormous importance when seeking bio-inspired advances in engineering applications, where the fitness function is different and the rules imposed by the biological substrate are absent.

A large motivator of this work is to develop high performance mechanical controllers based upon, but not limited by or linked slavishly to the underlying biology. Our work therefore out-evolves the natural organism's operation range rather than out-evolving nature. Improving the range of operation is fundamental to developing bio-inspired solutions for alternative control tasks.

In summary, our experiments show that, by relaxing some of the constraints associated with a biological exemplar, controllers (and potentially other computational structures) can be evolved that can capture the strengths of biological computation in a simpler, or perhaps more effective manner. This is intrinsically interesting, as a contribution to understanding the naturally-evolved performance of real organisms. It is also an enormously encouraging first step towards re-evolving other CPG controllers, and potentially other biological processors, for different tasks, using non-biological computing substrates.

7 Towards Controlling Wave Energy Devices and Improving Power Capture Efficiency – A Bio-inspired Solution

Marine energy devices operate with similar rhythmic routines, locomotion and in the same environment as the lamprey. Adding reactive control to these machines can result in autonomy, improved efficiency and increased productivity in the marine energy sector.

The depletion of natural energy resources and the need to reduce carbon dioxide emissions has generated a huge interest in renewable energy. Significant power is stored in the motion of the seas. However, harnessing this energy effectively remains a very difficult challenge. This is due to the highly unpredictable and dynamic nature of seas which are influenced by factors such as wind strength, wind direction, drag forces, as well as superposition and counteracting incoming waves, with different frequencies and velocities.

Wave energy converters (WECs), such as the one displayed in fig. 7, harness some of this energy but cannot adapt autonomously to irregular and changing sea conditions. Instead they rely on past wave data to make inaccurate predictions of future waves or use compromise operational settings until manually reset. As a result, they operate at sub-optimal efficiency. An active and adaptive approach would provide the currently lacking, but necessary, self-regulatory control, thus producing more power and under more robust conditions. Biology already invokes this kind of adaptive control (and does it very well) in the swim module of the lamprey and inspires application of a similar mechanism for marine energy devices.

The general underlying mechanism of WEC operation is to perform managed movements in the oceans, converting wave energy into usable electricity. Locomotion is usually oscillatory, and devices try to match the complex characteristics of wave frequencies or forces. Again, this bears similarity with the type of locomotion governed by lamprey CPG circuits.

In this chapter, the wave power solution has not been addressed directly as there are many contributory components that require attention. This chapter reports on issues that must be resolved with the bio-inspired model, which in turn will be used to advance wave technology. The work assesses the flexibility of the biological architecture intended for use with wave power devices to determine how much it can

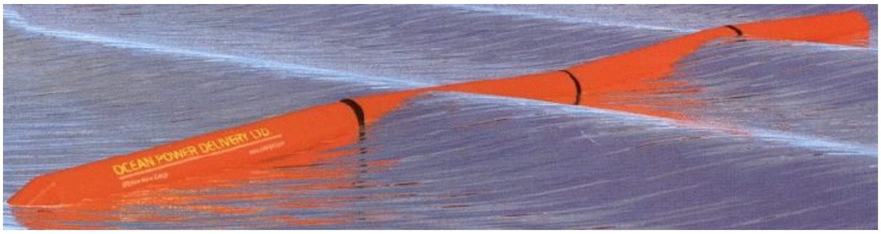


Fig. 7 The Pelamis, a wave energy converter developed by Pelamis Wave Power Ltd. It resembles the lamprey (in fig. 4a), both visually and in locomotion.

be stretched to accommodate the wider range of operability required for the engineering solution. The following section discusses other aims in developing this bio-inspired solution for wave energy.

8 Working towards Wave Power

The aim is to develop adaptive controllers based on the lamprey's CPG architecture to boost the efficiency of wave power devices (both articulated devices and single-point absorbers) operating in irregular sea states. The intention is to increase the renewable power these devices extract from the sea in a reactive rather than static, currently inefficient manner. The lamprey CPG model is an ideal control architecture within which to work. Initially, the biological model was explored without the constraints imposed by the biological substrate. Redevelopment will focus on tuning it to power-extraction elements of WECs, replacing swimming efficiency with power efficiency in the fitness function.

The flexibility and operational boundaries (ranges) of the network were explored by evolving the CPGs interneuron control parameters. The majority of the conditions built into the fitness functions of these genetic programs are also requisite for wave energy control (WEC) solutions. Some promote efficient, streamlined and cost-effective outcomes, such as simplification of the network. Others may require tweaking such as ranges of operation to match requirements of wave conditions. Other factors will require complete implementation such as operations relating to the measurement and dissipation of power. Other developmental goals will include:

1. Complete remodelling of the mechanical body, its effect on surrounding water, and of the waves as they interact with the device.
2. Evolution of sensory input cells - edge cells will play a big role, with fluctuations in wave conditions being fed back to the neural controller in order to modulate or alter the system's behaviour. This will involve a further GA process and related fitness functions to evolve sensory feedback components.
3. Further reorganisation of the network - if the tonic inputs are to serve a more direct reactive role (with inputs feeding directly back to the network rather than modulating patterns of activity from a higher command node).
4. Alternative control strategies - although reactive control is the main aim, other control strategies will be investigated and compared (e.g. latched control [22]) to ascertain their efficiency and resource requirements.
5. A longer term goal, once the concept has been proven - individual implementation of each wave power device, related fitness functions and evolution according to device-specific criteria.

The evolution of improved lamprey CPGs has been a crucial step towards achieving these next stages of WEC application control.

9 Concluding Remarks

This chapter has discussed a class of neural networks called Central Pattern Generators (CPGs), responsible for rhythmic patterns of behaviour. CPGs comprise organised neuronal populations which function collectively to coordinate activity of several cells to produce oscillatory output. CPG modules do not require sensory input to generate rhythmic behaviour, but temporal and phasic signals from afferent sensory inputs can modulate its intrinsic activity.

It has been shown that CPGs control a broad range of functions in animals. Furthermore, they are widely variable and adaptable with age, environment and behaviour. Although anatomical details of CPG circuits are known in only a few cases, most originate from vertebrate spinal cords which are generally small autonomous networks which govern rhythmic patterns of behaviour. A model of the lamprey's (an eel-like fish) CPG is described in detail.

This neural circuit's ability to self-regulate behaviour to meet the needs of a changing environment, and the fact that the system produces the same *fictive* swimming when implemented artificially, make it an ideal candidate for providing similar artificial intelligence to other real tasks where automation would result in increased efficiency and productivity.

Evidence has been presented to demonstrate the flexibility of this network with genetically evolved, more superior controllers (in terms of their operation ranges). These will be further evolved and implemented with wave energy devices to boost the energy they extract from unpredictable and everchanging seas; a task that requires similar rhythmic locomotion and self-regulation that the lamprey's swim module displays. Thus this provides a bio-inspired solution to a challenging engineering task.

Finally, inspiration does not end here; there are many other CPG-driven tasks that could benefit from bio-inspired technology. These include heart-pacemakers, responsive, for example, to changes in the level of physical activity, robotic locomotion (much research is already evident in this area), and hearing-aid modulators (an area not previously considered).

Acknowledgements. This research was supported by the SHEFC INCITE Grant and EPSRC Platform Grant (EP/C516583/1). I would also like to thank Prof. Alan Murray (University of Edinburgh) for his continued advice and support.

References

1. Arena, P.: A mechatronic lamprey controlled by analog circuits. In: Proc 9th IEEE Mediterr Conf. Control Autom (2001)
2. Ayers, J.: Underwater walking. Anthropod Struct Dev 33(4), 347–360 (2004)
3. Barfoot, T.D., Earon, E.J.P., D'Eleuterio, G.M.T.: Experiments in learning distributed control for a hexapod robot. Robot Auton Syst. 54(10), 864–872 (2006)
4. Blanchard, M., Rind, F., Vershure, F.: Collision avoidance using a model of locust LGMD neuron. Robot Auton Syst. 30, 17–38 (2000)

5. Broch, L., Morales, R.D., Sandoval, A.V., et al.: Regulation of the respiratory central pattern generator by chloride-dependent inhibition during development in the bullfrog (Rana catesbeiana). J. Exp. Biol. 205, 1161–1169 (2002)
6. Buchanan, J.T., Grillner, S.: Newly identified glutamate interneurons and their role in locomotion in the lamprey spinal cord. Sci. 236, 312–314 (1987)
7. Burggren, W.W., West, N.H.: Changing importance of gills, lungs and skin during metamorphosis in the bullfrog Rana catesbeiana. Respir Physiol. 47, 151–164 (1982)
8. Calancie, B., Needham-Shropshire, B., Jacobs, P., et al.: Involuntary stepping after chronic spinal cord injury: Evidence for a central rhythm generator for locomotion in a man. Brain 117(5), 1143–1159 (1994)
9. Cohen, A.H., Wallén, P.: Fictive swimming induced in an in vitro preparation of the lamprey spinal cord. Exp. Brain Res. 41(1), 11–18 (1980)
10. Dick, T.E., Oku, Y., Romaniuk, J.R., et al.: Interaction between CPGs for breathing and swallowing in the cat. J. Physiol. 465, 715–730 (1993)
11. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: Optimization by a colony of cooperating Agents. IEEE Trans. Syst. Man Cybern. 26B(1), 29–41 (1996)
12. Ekeberg, Ö.: A combined neuronal and mechanical model of fish swimming. Biol. Cybern. 69, 363–374 (1993)
13. Fogel, D.B.: Evolutionary computation: toward a new philosophy of machine intelligence, 3rd edn. IEEE Press, Piscataway (2006)
14. Frik, M., Guddat, M., Karatas, M., et al.: A novel approach to autonomous control of walking machines. In: Proc. 2nd Int. Conf. Climbing and Walking Robot (CLAWAR), pp. 333–342 (1999)
15. Gdovin, M.J., Torgerson, C.S., Remmers, J.E.: The fictively breathing tadpole brainstem preparation as a model for the development of respiratory pattern generation and central chemoreception. Comp. Biochem. Physiol. 124A, 275–286 (1999)
16. Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading (1989)
17. Goldberg, D.E.: The design of innovation: lessons from and for competent genetic algorithms. Addison-Wesley, Reading (2002)
18. Graham-Brown, T.: The intrinsic factors in the act of progression in the mammal. Proc. R Soc. Lond, Biol Sci. 84, 308–319 (1911)
19. Grillner, S., McClellan, A., Sigvardt, K., et al.: Activation of NMDA receptors elicits fictive locomotion in lamprey spinal cord in vitro. Acta Physiol. Scand 113, 549–551 (1981)
20. Grillner, S., Wallén, P., Hill, R., et al.: Ion channels of importance for the locomotor pattern generation in the lamprey brainstem-spinal cord. J. Physiol. 533(1), 23–30 (2001)
21. Hill, A.A.V., Masino, M.A., Calabrese, R.L.: Model of intersegmental coordination in the leech heartbeat neuronal network. J. Neurophysiol. 87(3), 1586–1602 (2002)
22. Hoskin, R.E., Nichols, N.K.: Latching control of a point absorber. In: 3rd Int Symp. Wave, Tidal, OTEC small scale Hydro Energy, pp. 317–329 (1986)
23. Ijspeert, A.J., Kodjabachian, J.: Evolution and development of a central pattern generator for the swimming of a lamprey. Artif. Life 5(3), 247–269 (1999)
24. Ijspeert, A.J., Hallam, J., Willshaw, D.: Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology. Adapt Behav. 7(2), 151–172 (1999)
25. Ijspeert, A.J., Crespi, A., Ryczko, D., et al.: From swimming to walking with a salamander robot driven by a spinal cord model. Sci. 315(5817), 1416–1420 (2007)
26. Jean, A.: Brain stem control of swallowing: neuronal network and cellular mechanisms. Physiol. Rev. 81, 929–969 (2001)

27. Jezzini, S.H., Hill, A.A.V., Kuzyk, P., et al.: Detailed model of intersegmental coordination in the timing network of the leech heartbeat central pattern generator. J. Neurophysiol. 91, 958–977 (2004)
28. Katz, P.S.: Tritonia. Scholarpedia 2(6), 3504 (2007)
29. National Aeronautics and Space Administration (NASA), Intelligent flight control system. Fact Sheet FS-076 (2005), `http://www.nasa.gov/centers/dryden/news/FactSheets/FS-076-DFRC.html`
30. Kennedy, J., Eberhart, R.C.: Swarm intelligence. Morgan Kaufmann, San Francisco (2001)
31. Lozano, J.A., Larrañga, P., Inza, I., et al. (eds.): Towards a new evolutionary computation. Advances in estimation of distribution algorithms. Springer, Heidelberg (2006)
32. Patel, L.N., Murray, A., Hallam, J.: Increased swimming control with evolved lamprey CPG controllers. In: Int. Jt. Conf. Neural Netw. (IJCNN), pp. 2195–2200 (2005)
33. Patel, L.N., Murray, A., Hallam, J.: Evolving multi-segment superlamprey CPG's for increased swimming control. In: Eur. Symp. Artif. Neural Netw. (ESANN), pp. 461–466 (2006)
34. Patel, L.N., Murray, A.F., Hallam, J.: Super-lampreys and wave energy: Optimised control of artificially-evolved, simulated swimming lamprey. Neurocomputing 70(7-9), 1139–1154 (2007)
35. Popescu, I.R., Frost, W.N.: Highly dissimilar behaviors mediated by a multifunctional network in the marine mollusk tritonia diomedea. J. Neurosci. 22(5), 1985–1993 (2002)
36. Randall, D., Beer, R.D., Chiel, H.J., et al.: A distributed neural network architecture for hexapod robot locomotion. Neural Comput. 4(3), 356–365 (1992)
37. Reid, S.G., Milsom, W.K.: Respiratory pattern formation in the isolated bullfrog (*Rana catesbeiana*) brainstem-spinal cord. Respir Physiol. 114, 239–255 (1998)
38. Righetti, L., Ijspeert, A.J.: Pattern generators with sensory feedback for the control of quadruped locomotion. In: Proc. IEEE Int. Conf. Robot Autom. (ICRA), pp. 819–824 (2008)
39. Roberts, A., Tunstall, M.J.: Mutual re-excitation with post-inhibitory rebound: a simulation study on the mechanisms for locomotor rhythm generation in the spinal cord of xenopus embryos. Eur. J. Neurosci. 2(1), 11–23 (1990)
40. Rovainen, C.M.: Neurobiology of lampreys. Physiol. Rev. 59, 1007–1077 (1979)
41. Russell, A., Orchard, G., Etienne-Cummings, R.: Configuring of spiking central pattern generator networks for bipedal walking using genetic algorthms. In: IEEE Int. Symp. on Circuits and Syst. (ISCAS), pp. 1525–1528 (2007)
42. Sayed, R., Eskandarian, A.: Unobtrusive drowsiness detection by neural network learning of driver steering. Proc. Inst. Mech. Eng., J. Automob. Eng. 215D(K4), 969–975 (2001)
43. Shik, M.L., Severin, F.V., Orlovskii, G.N.: Control of walking and running by means of electrical stimulation of the midbrain. Biofiz 11, 659–666 (1966)
44. Spenneberg, D., Kirchner, F., de Gea, J.: Ambulating robots for exploration in rough terrain on future extraterrestrial missions. Anthropod Struct. Dev. 33(4), 347–360 (2004)
45. Sqalli-Houssaini, Y., Cazalets, J.R., et al.: Oscillatory properties of the central pattern generator for locomotion in neonatal rats. J. Neurophysiol. 70(2), 803–813 (1993)
46. Torgerson, C.S., Gdovin, M.J., Remmers, J.E.: Fictive gill and lung ventilation in the pre- and postmetamorphic tadpole brain stem. ibition during development in the bullfrog (Rana catesbeiana) J. Neurophysiol. 80(4), 2015–2022 (1998)
47. Wilson, D.M.: The central nervous control of flight in a locust. J. Exp. Biol. 38, 471–490 (1961)

Fish School Search

Carmelo J.A. Bastos Filho, Fernando B. de Lima Neto, Anthony J.C.C. Lins, Antônio I.S. Nascimento, and Marília P. Lima

Abstract. Real world problems are packed with complex issues often hard to be computed. Searching for parameters or candidate solutions is frequently associated with these complexities. The reason for that is chiefly related to the large dimensionalities of some search spaces. In general, problems involving large search spaces use traditional computer intensive methods that are, quite often, expensive (i.e. resource consuming). Nature-inspired algorithms, on the other hand, are able to deal reasonably well with the abovementioned difficulties. In this chapter, we provide an overview of a novel approach for searching in high-dimensional spaces based on the behaviors of fish schools. As any other intelligent technique based on population, Fish School Search (FSS) greatly benefits from the collective emerging behavior that increases mutual survivability. Broadly speaking, FSS is composed of operators that can be grouped in the following categories: feeding, swimming and breeding. Together, these operators provide computing behavior such as: (i) high-dimensional search ability, (ii) automatic selection between exploration and exploitation, and (iii) self-adaptable guidance towards sought solutions. This chapter seeks to explain the main ideas behind FSS to researchers and practitioners. In addition, we include examples and simulations aimed at clarifying the simplicity and potentials of FSS.

1 Introduction

Several oceanic fish species, as with other animals, present social behavior. This phenomenon's main purpose is to increase mutual survivability and may be viewed in two ways: (i) for mutual protection and (ii) for synergistic achievement of other collective tasks. By protection we mean reducing the chances of being caught by predators; and by synergy, we refer to an active means of achieving collective goals such as finding food.

Carmelo J.A. Bastos Filho · Fernando B. de Lima Neto · Anthony J.C.C. Lins · Antônio I.S. Nascimento · Marília P. Lima
Department of Computing and Systems, Polytechnic School of Pernambuco, University of Pernambuco (UPE), Recife, Brazil
e-mail: {cjabf,fbln,ajccl,aisn,mpl}@dsc.upe.br

Apart from debating whether the emergent behavior of a fish school is due to learning or genetic reasons, it is important to note that some fish species live their entire lives in schools. This reduces individual freedom in terms of swimming movements and increases competition in regions with scarce food. However, fish aggregation is a fact and the benefits largely outweigh the drawbacks. This chapter aims at presenting a novel computational intelligent search technique inspired by the above-mentioned behavior.

Along with the development of this technique we have taken great care not to depart from the original inspiration source, but FSS contains a few abstractions and simplifications that have been introduced to afford efficiency and usability to the algorithm. The main characteristics derived from real fish schools and incorporated into the core of our approach are sound. They are grouped into two observable categories of behaviors as follows:

- Feeding: inspired by the natural instinct of individuals (fish) to find food in order to grow strong and to be able to breed. Notice that food here is a metaphor for the evaluation of candidate solutions in the search process. We have considered that an individual fish can lose as well as obtain weight, depending on the regions it swims in;
- Swimming: the most elaborate observable behavior utilized in our approach. It aims at mimicking the coordinated and the only apparent collective movement produced by all the fish in the school. Swimming is primarily driven by feeding needs and, in the algorithm, it is a metaphor for the search process itself.

2 Background

2.1 Search Problems and Algorithms

Although there are several approaches for searching, there is, unfortunately, no general optimal search strategy [1]. Thus, solving search problems is sometimes more of an art form than an engineering practice. Although custom-made algorithms are valuable options for specific problems, a more generalized automatic search engine would be a great bonus for tackling problems of high dimensionality.

Search problems can be highly varied. For example, they can be classified into two groups with regard to the structure of their search-space: structured or unstructured. For the former, there are many traditional techniques that are, on average, quite efficient. The same observation does not apply to the latter, that is, there is no overall good approach for search spaces on which there is no prior information.

We think that FSS might be a valuable option for searching in high dimensional and unstructured spaces.

2.2 Population-Based Algorithm (PBA)

Many nature-inspired algorithms such as genetic algorithms (GA) [2, 3], artificial immune systems (AIS) [4], ant colony optimization (ACO) [5, 6] and particle swarm optimization (PSO) [7, 8, 9] are based on the concept of populations. In all these approaches, the computing discrimination power and memorization ability of past experiences are distributed among the individuals of the population in varying degrees.

Distributed representation and computation are interesting features to incorporate into search algorithms because of the parallelization they provide. The obvious trade-off is the cost of control (*i.e.* communication among individuals), as opposed to the lower costs associated with centralized control.

In recent years, PSO has produced good results for search problems with high dimensionality. It is an intelligent computational technique proposed by Kennedy and Eberhart in 1995 [7]. This technique is commonly used to solve optimization problems of nonlinear functions. It is inspired by the social behavior of bird flocks. The idea behind PSO is to create particles that simulate the movements of birds to achieve a specific goal within the search space. It explores the social behavior of an organized group of individuals and the group's communication capacity. Each particle represents a solution in a high-dimensional space. The entire swarm uses a specific communication mechanism. The candidate solutions emerge by flocking behavior around more successful individuals. Particles in PSO utilize the notion of adjustable speed according to the degree of success achieved. In the most common PSO implementations, particles move through the search space using a combination of the attraction to the best solution that they have found individually, and the attraction to the best solution that any particle in the neighborhood has found. A neighborhood is the part of the swarm which a particle is able to communicate with. Bratton *et al.* [9] proposed a standard for performance comparison of PSO implementations. Many velocity equations and communication mechanisms were proposed in recent years [10, 11, 12, 13, 14]. However, the PSO technique struggles in some multimodal problems.

3 Fish-School Search (FSS)

3.1 FSS Computational Principles

The search process in FSS is carried out by a population of limited-memory individuals – the fish. Each fish represents a possible solution to the problem. Similar to PSO or GA, search guidance in FSS is driven by the success of some individual members of the population.

The main feature of the FSS paradigm is that all fish contain an innate memory of their successes – their weights. In comparison to PSO, this information is highly relevant because it can obviate the need to keep a log of the best positions visited by all individuals, their velocities and other competitive global variables.

Another major feature of FSS is the idea of evolution through a combination of some collective swimming, *i.e.* "operators" that select among different modes of operation during the search process, on the basis of instantaneous results.

As for dealing with the high dimensionality and lack of structure of the search space, the authors believe that FSS should at least incorporate principles such as the following:

(i) Simple computation in all individuals;
(ii) Various means of storing distributed memory of past computation;
(iii) Local computation (preferably within small radiuses);
(iv) Low communication between neighboring individuals;
(v) Minimum centralized control (preferably none); and
(vi) Some diversity among individuals.

A brief rationale for the above-mentioned principles is given, respectively: (i) this reduces the overall computation cost of the search; (ii) this allows for adaptive learning; (iii), (iv) and (v) these keep computation costs low as well as allowing some local knowledge to be shared, thereby speeding up convergence; and finally, (vi) this might also speed up the search due to the differentiation/specialization of individuals. These principles incorporated in FSS lead the authors to believe that FSS can deal with multimodal problems better than the PSO approaches.

3.2 *Overview of the New Approach*

The inspiration mentioned in Section I, together with the principles just stated above, are incorporated in our approach in the form of two operators that comprise the main routines of the FSS algorithm. To understand the operators, a number of concepts need to be defined.

The concept of food is related to the function to be optimized in the process. For example, in a minimization problem the amount of food in a region is inversely proportional to the function evaluation in this region. The “aquarium” is defined by the delimited region in the search space where the fish can be positioned.

The operators are grouped in the same manner in which they were observed when drawn from the fish school. They are as follows:

- Feeding: food is a metaphor for indicating to the fish the regions of the aquarium that are likely to be good spots for the search process;
- Swimming: a collection of operators that are responsible for guiding the search effort globally towards subspaces of the aquarium that are collectively sensed by all individual fish as more promising with regard to the search process.

3.3 *FSS Operators*

3.3.1 The Feeding Operator

As in real situations, the fish of FSS are attracted to food scattered in the aquarium in various concentrations. In order to find greater amounts of food, the fish in the school can move independently (see individual movements in the next section).

As a result, each fish can grow or diminish in weight, depending on its success or failure in obtaining food. We propose that fish's weight variation be proportional to the normalized difference between the evaluation of fitness function of previous and current fish position with regard to food concentration of these spots. The assessment of 'food' concentration considers all problem dimensions, as shown in 1,

$$W_i(t+1) = W_i(t) + \frac{f[x_i(t+1)] - f[x_i(t)]}{\max\{|f[x_i(t+1)] - f[x_i(t)]|\}}, \tag{1}$$

where $W_i(t)$ is the weight of the fish i, $\mathbf{x}_i(t)$ is the position of the fish i and $f[\mathbf{x}_i(t)]$ evaluates the fitness function (*i.e.* amount of food) in $\mathbf{x}_i(t)$.

A few additional measures were included to ensure rapid convergence toward rich areas of the aquarium, namely:

- Fish weight variation is evaluated once at every FSS cycle;
- An additional parameter, named weight scale (W_{scale}) was created to limit the weight of a fish. The fish weight can vary between "1" and W_{scale}.
- All the fish are born with weight equal to $\frac{W_{scale}}{2}$.

3.3.2 The Swimming Operators

A basic animal instinct is to react to environmental stimulation (or sometimes, the lack of it). In our approach swimming is considered to be an elaborate form of reaction regarding survivability. In FSS, the swimming patterns of the fish school are the result of a combination of three different causes (*i.e.* movements).

For fish, swimming is directly related to all the important individual and collective behaviors such as feeding, breeding, escaping from predators, moving to more livable regions of the aquarium or, simply being gregarious.

This panoply of motivations to swim away inspired us to group causes of swimming into three classes: (i) individual, (ii) collective-instinct and (iii) collective-volition. Below we provide further explanations on how computations are performed on each of them.

3.3.2.1 Individual Movement

Individual movement occurs for each fish in the aquarium at every cycle of the FSS algorithm. The swim direction is randomly chosen. Provided the candidate destination point lies within the aquarium boundaries, the fish assesses whether the food density there seems to be better than at its current location. If this is not the case or if the step-size is not possible (i.e. it lies outside the aquarium or is blocked by, say, reefs), the individual movement of the fish does not occur. Soon after each individual movement, feeding occurs, as detailed above.

For this movement, we define a parameter to determine the fish displacement in the aquarium called individual step ($step_{ind}$). Each fish moves $step_{ind}$ if the new position has more food than the previous position. Actually, to include more randomness in the search process we multiply the individual step by a random number generated by a uniform distribution in the interval [0,1]. In our simulation we decrease the individual step linearly in order to provide exploitation abilities in later iterations.

Fig. 1 shows an illustrative example of this swimming operator. One can note that just the fish that found spots with more food had moved.

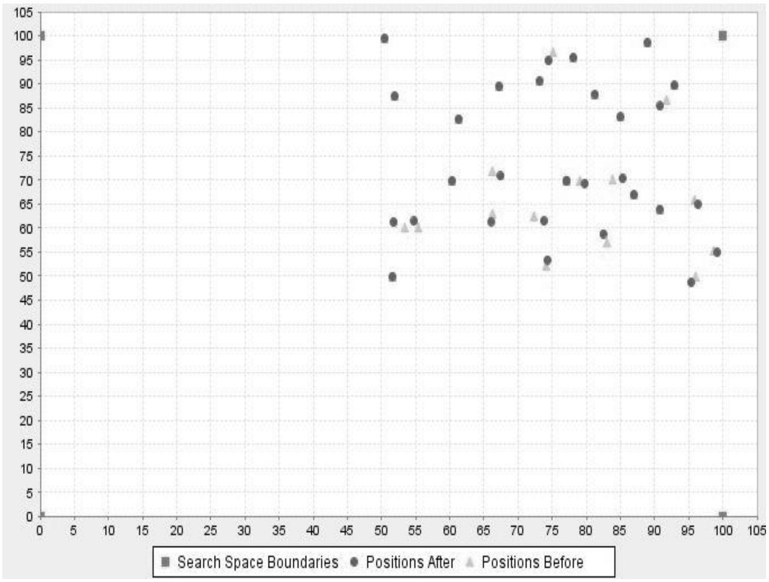


Fig. 1 Individual movement is illustrated here before and after its occurrence; circular dots are fish positions after and triangular dots are the same fish before individual movement

3.3.2.2 Collective-Instinctive Movement

After all fish have moved individually, a weighted average of individual movements based on the instantaneous success of all fish of the school is computed. This means that fish that had successful individual movements influence the resulting direction of movement more than the unsuccessful ones. When the overall direction is computed, each fish is repositioned. This movement is based on the fitness evaluation enhancement achieved, as shown in 2.

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \frac{\sum_{i=1}^{N} \Delta\mathbf{x}_{indi}\{f[x_i(t+1)] - f[x_i(t)]\}}{\sum_{i=1}^{N}\{f[x_i(t+1)] - f[x_i(t)]\}} \tag{2}$$

where $\Delta\mathbf{x}_{indi}$ is the displacement of the fish i due to the individual movement in the FSS cycle.

Fig. 2 shows the influence of the collective-instinctive movement in the example presented in Fig. 1. One can note that in this case all the fish had their positions adjusted.

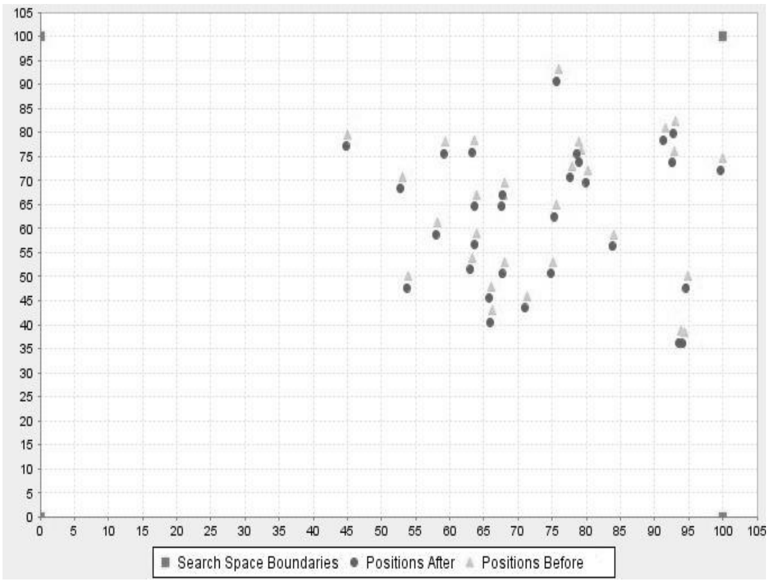


Fig. 2 Collective-instinctive movement is illustrated here before and after its occurrence; circular dots are fish positions after and triangular dots are the same fish before collective-instinctive movement

3.3.2.3 Collective-Volitive Movement

After individual and collective-instinctive movements are performed, one additional positional adjustment is still necessary for all fish in the school: the collective-volitive movement. This movement is devised as an overall success/failure evaluation based on the incremental weight variation of the whole fish school. In other words, this last movement will be based on the overall performance of the fish school.

The rationale is as follows: if the fish school is putting on weight (meaning the search has been successful), the radius of the school should contract; if not, it should dilate. This operator is deemed to help greatly in enhancing the exploration abilities in FSS. This phenomenon might also occur in real swarms, but the reasons are as yet unknown.

The fish-school dilation or contraction is applied as a small step drift to every fish position with regard to the school's barycenter. The fish-school's barycenter is obtained by considering all fish positions and their weights, as shown in 3.

Collective-volitive movement will be inwards or outwards (in relation to the fish-school's barycenter), according to whether the previously recorded overall weight of the school has increased or decreased in relation to the new overall weight observed at the end of the current FSS cycle.

$$Bari(t) = \frac{\sum_{i=1}^{N} \mathbf{x}_i(t) W_i(t)}{\sum_{i=1}^{N} W_i(t)} \tag{3}$$

For this movement, we also define a parameter called volitive step ($step_{vol}$). We evaluate the new position as in 4 if the overall weight of the school increases in the FSS cycle; if the overall weight decreases, we use 5.

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) - step_{vol}.rand.\left[\mathbf{x}_i(t) - Bari(t)\right], \tag{4}$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + step_{vol}.rand.\left[\mathbf{x}_i(t) - Bari(t)\right], \tag{5}$$

where *rand* is a random number uniformly generated in the interval [0,1]. We also decreased the linear $step_{vol}$ along the iterations.

Fig. 3 shows the influence of the collective-volitive movement in the example presented in Fig. 1 after individual and collective-instintive movements. In this case, as the overall weight of the school had increased, the radius of the school diminished.

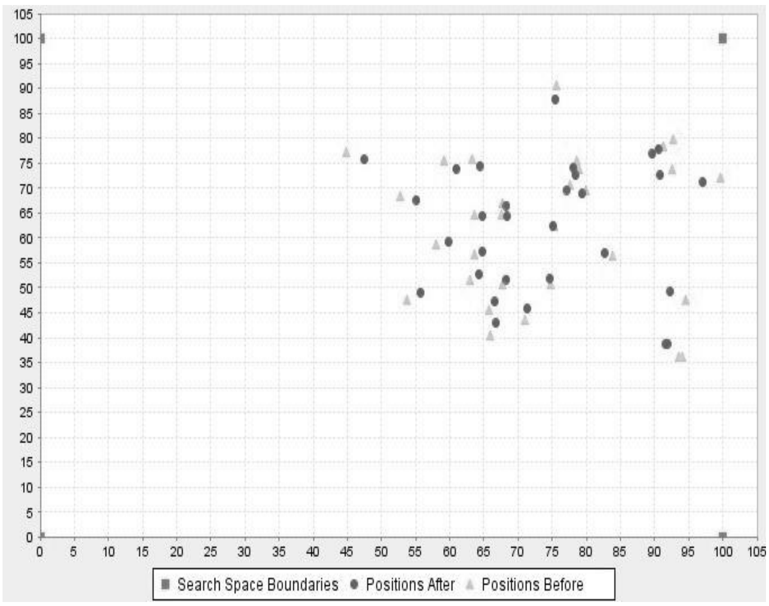


Fig. 3 Collective-volitive movement is illustrated here before and after its occurrence; circular dots are fish positions after and triangular dots are the same fish before collective-volitive movement

3.4 FSS Cycle and Stop Conditions

The FSS algorithm starts by randomly generating a fish school according to parameters that control fish sizes and their initial positions.

Regarding dynamics, the central idea of FSS is that all bio-inspired operators perform independently of each other across the three conceived classes.

The search process (*i.e.* FSS at work) is enclosed in a loop, where invocations of the previously presented operators will occur until at least one stop condition is met.

As of now, stop conditions conceived for FSS are as follows: limitation of the number of cycles (the stopping condition of all experiments in this chapter), time limit, maximum school radius, minimum school weight and maximum fish number.

We present below the pseudo-code for the Fish School Search Algorithm. In the initialization step, each fish in the swarm has its weight initialized with the value $\frac{W_{scale}}{2}$ and its position in each dimension initialized randomly in the search space.

```
Algorithm Fish School Search
1. Initialize fish in the swarm
2. While maximum iterations or stop criteria is not attained do
3.  for each fish i in the swarm do
      a. update position applying the individual operator
            $\Delta \mathbf{x}_i(t+1) = step_{\text{ind}}(t) \cdot 2 \cdot rand \cdot direction$
            $\overrightarrow{temp}_i = \mathbf{x}_i(t) + \Delta \mathbf{x}_i(t+1)$
         calculate fish fitness $f_i(\overrightarrow{temp}_i)$
         if $f(\overrightarrow{temp}_i) < f(\mathbf{x}_i(t))$
            $\mathbf{x}_i(t+1) = \overrightarrow{temp}_i$
            $f_i^{(t+1)} = f_i(\overrightarrow{temp}_i)$
         else
            $\mathbf{x}_i(t+1) = \mathbf{x}_i(t)$
            $f_i^{(t+1)} = f_i^{(t)}$
      b. apply feeding operator
         update fish weight according to 1
      c. apply collective-instinctive movement
         update fish position according to 2
      d. apply collective-volitive movement
         if overall weight of the school increases in the cycle
         update fish position using 4
         elseif overall weight of the school decreases in the cycle
         update fish position using 5
    end for  decrease the individual and volitive steps linearly
end while
```

4 Illustrative Example

This section presents an illustrative example aimed at better understanding how FSS can be used and, ultimately, how it works. The selected example considers a small school and a very simple problem that is three fish are set to find the global optimum of the sphere function in two dimensions. The sphere function is presented in 6 and its parameters are: (i) feasible space [-10,10], (ii) number of iterations equal to 10, (iii) w_{scale}= 10, (iv) initial $step_{ind}$ = 1, (v) final $step_{ind}$ = 0.1, (vi) initial $step_{vol}$ = 0.5, (vii) final $step_{vol}$ = 0.05. Table 1 includes initial values associated with the experimental fish school; Fig. 4a presents start-up loci of all fish.

$$F_{sphere}(x) = \sum_{i=1}^{n-1} (x_i)^2, \tag{6}$$

Table 1 Initial conditions for the three fish in the sphere example

Fish	**Initial conditions**		
	weight	***position***	***fitness***
# 1	5	(9,7)	130
# 2	5	(5,6)	61
# 3	5	(8,4)	80

After initialization, all fish are free to check for new candidate positions that are generated by the individual movement operator. Lets assume that these positions are $\mathbf{x}_1 = (9.6, 6.2)$, $\mathbf{x}_2 = (4.6, 4.4)$ and $\mathbf{x}_3 = (6.2, 4.2)$, and the associated fitnesses are $f(\mathbf{x}_1) = 130.6$, $f(\mathbf{x}_2) = 40.52$ and $f(\mathbf{x}_3) = 56.08$. One should notice that fish #2 and fish #3 found best positions, whereas fish #1 did not move. The positions after the individual movement are then $\mathbf{x}_1 = (9, 7)$, $\mathbf{x}_2 = (4.6, 4.4)$ and $\mathbf{x}_3 = (6.2, 4.2)$. Fig. 4b illustrates the individual movement of the three fish in search space for the sphere problem.

According to our model, the next operator to be computed is feeding. As fish #1 remained in the same position, it will not change its weight. The weight of fish #2 and fish #3 will change according to 1. The weight variation depends on the maximum fitness change. The maximum fitness variation in this case was achieved by fish #3 and is equal to 23.92. As a result, fish #3 increased its weight by 1 unit and its new weight became 6. The fitness variation of fish #2 was 20.48. Dividing the fitness variation of fish #2 by maximum fitness change, we conclude that the weight variation of fish #2 is 0.86. The new weight of fish #2 is then 5.86.

Following our model, the third operator to be computed is the collective-instinctive one. This operator evaluates the collective displacement of the fish school considering the individual fitness variations and the individual movement according to 2. As fish #1 stayed in the same position, it will not influence the overall calculation. Considering the values obtained in this iteration, the displacement is (-1.2,-0.6). This vector is applied to all the fish (including fish #1), so the new positions, after third operator computations, are $\mathbf{x}_1 = (7.8, 6.4)$, $\mathbf{x}_2 = (3.4, 3.8)$ and $\mathbf{x}_3 = (5, 3.6)$.

Then, the fitnesses regarding new positions recalculations are 101.8, 26 and 37.96 for fish #1, #2 and #3, respectively. The individual displacement of all fish due to collective-instinctive operator is presented in Fig. 4c. Reader may find it interesting to compare Fig. 4b and Fig. 4c.

The last operator to be considered in this example is the collective-volitive one. For that, one has to obtain the instantaneous value of the barycenter of the fish school according to 3. In this case, the barycenter is (4.96,4.25). Notice that the weight of whole school has increased, therefore a contraction instead of a dilatation is the implicit decision of the school (*i.e.* collective-volitive). By means of using 4, the new positions are $\mathbf{x}_1 = (5.81, 4.89)$, $\mathbf{x}_2 = (4.02, 3.98)$ and $\mathbf{x}_3 = (4.98, 3.92)$. The barycenter and the collective-volitive movement for this step are presented in Fig. 4d.

At this point the algorithm tests if valid stop-conditions are met. Obviously it is not the case yet, thus a new cycle begins as explained above. If one compares the initial and final positions illustrated in Fig. 4, after this first iteration, the reader can observe that all fish are closer to the optimum point (0,0).

Of course the optimum point is unknown to the algorithm. However, in a very peculiar manner the FSS model assures fast convergence towards it (*i.e.* the goal for the search process) because of the above mentioned natural principles instantiated in the FSS algorithm.

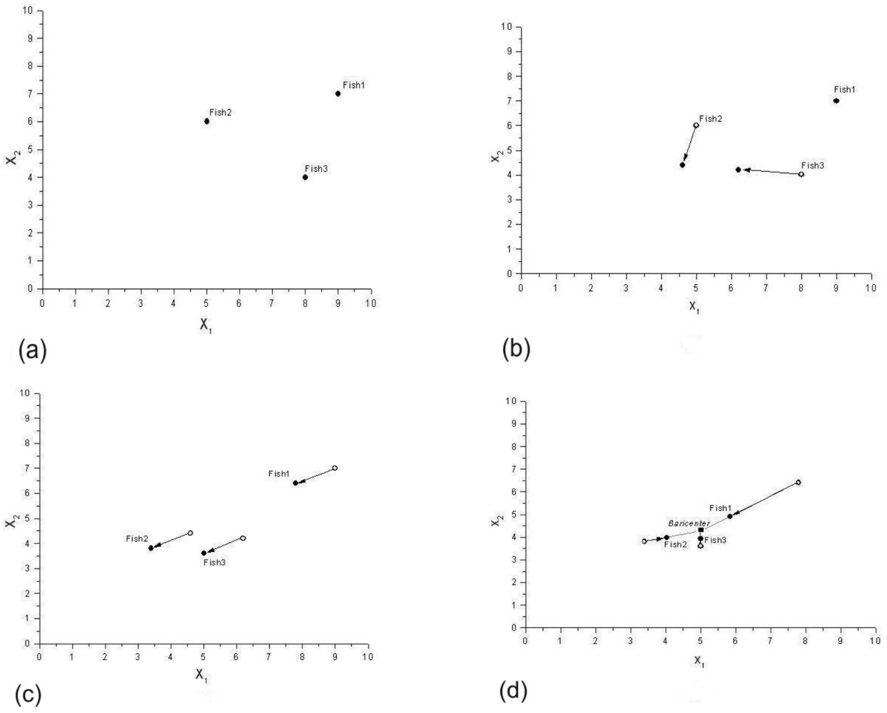


Fig. 4 Example with three fish in the sphere example: (a) Initial position, (b) individual movement, (c) instinctive collective movement and (d) volitive collective movement

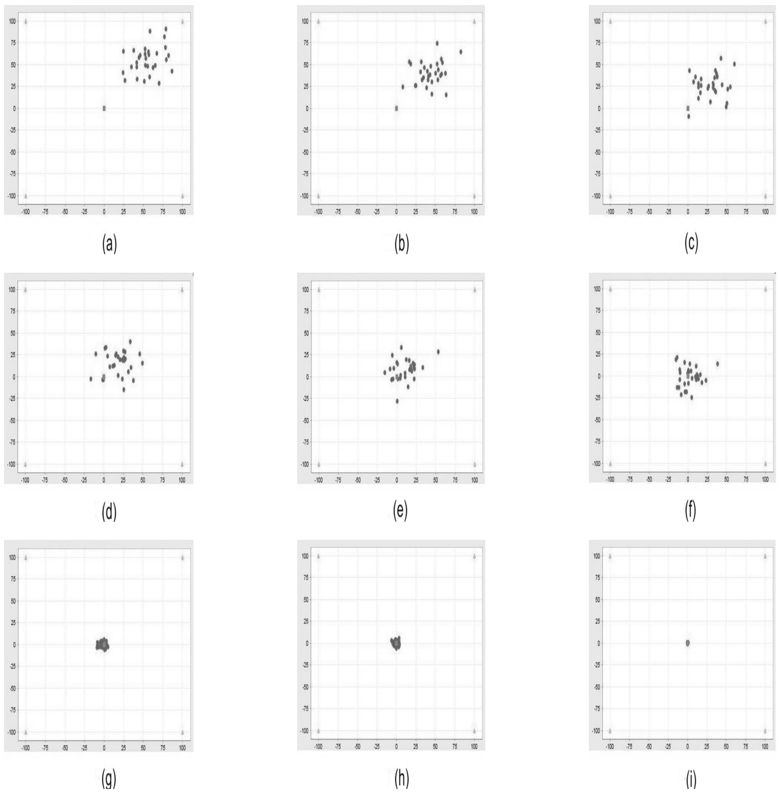


Fig. 5 Fish school evolution after iteration (a) 1, (b) 50, (c) 100, (d) 200, (e) 300, (f) 400, (g) 500, (h) 750 e (i) 1000 for sphere function with 30 fish

In order to illustrate the convergence behavior of the fish school along the iterations, we present the simulation results for the sphere function. In these simulations we used 30 fish, [-100,100] in the two dimensions, initialization range [0,100] in the two dimensions, w_{scale}= 500, initial $step_{ind}$ = 10, final $step_{ind}$ = 0.1, initial $step_{vol}$ = 5, final $step_{vol}$ = 0.5. Fig. 5. shows the fish positions after iteration (a) 1, (b) 50, (c) 100, (d) 200, (e) 300, (f) 400, (g) 500, (h) 750 e (i) 1000, respectively. One can note that the school was attracted to the optimum point (0,0).

5 Comparative Examples

5.1 Experimental Setup

Five benchmark functions were used to carry out simulations and are described in 7, 8, 9, 10, and 11. Table 2 shows the search space, the initialization range, and the optimum for each function. All searches were carried out in 30 dimensions. All five functions are used for minimization problems. Two of these functions namely, Rosenbrock and Schwefel 1.2, represent simple unimodal problems; the other three, Rastrigin, Griewank, and Ackley, are highly complex multimodal functions that contain many local optima. Considered functions in comparisons are:

$$F_{Rosenbrock}(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right], \tag{7}$$

$$F_{Rastrigin}(x) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i)\right], \tag{8}$$

$$F_{Griewank}(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right), \tag{9}$$

$$F_{Ackley}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e, \tag{10}$$

Table 2 Function parameters

Function	**Parameters**		
	Search space	***Initialization***	***Optima***
Rosenbrock	$-30 \leq xi \leq 30$	$15 \leq xi \leq 30$	1.0D
Rastrigin	$-5.12 \leq xi \leq 5.12$	$2.56 \leq xi \leq 5.12$	0.0D
Griewank	$-600 \leq xi \leq 600$	$300 \leq xi \leq 600$	0.0D
Ackley	$-32 \leq xi \leq 32$	$16 \leq xi \leq 32$	0.0D
Schwefel 1.2	$-100 \leq xi \leq 100$	$50 \leq xi \leq 100$	0.0D

and

$$F_{Schwefel1.2}(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 . \tag{11}$$

A factorial planning of experiments was performed to find suitable parametric combination of individual and volitive steps at both initial and final limits (*i.e.* $step_{ind,initial}$, $step_{ind,final}$, $step_{vol,initial}$, $step_{vol,final}$). We have associated the individual and volitive steps as percentages of the actual search space. Percentage values considered for initial and final limits were, respectively, as follows: 10; 1; 0.1 and 0.1; 0.01; 0.001; 0.0001. W_{scale} was set as 5000; this is half of the number of considered iterations.

All FSS simulations were performed using 30 fish and 10,000 iterations. Only after 30 trials the mean and the standard deviation were recorded. All the fish were randomly initialized in areas of the aquarium that are far from the optimal solution regarding every dimension. This initialization process is carried out in order to measure the ability of the fish school in locating the optimum solution outside the initialization space.

We compared our results with PSO simulation results presented in an earlier landmark paper [9]. Three PSO approaches were considered for comparisons: original PSO with the G_{best} topology, constriction PSO with the G_{best} topology and constriction PSO with the L_{best} topology. All the PSO simulations included 30 trials, each of which performed 300,000 evaluations, and simulations that considered 30 dimensions and used 30 particles. Thus, we considered that a fair convergence analysis between the FSS and PSO approaches could be made.

5.2 Simulation Results

Tables 3, 4, 5, 6 and 7 show the best simulation results for function Rosenbrock, Rastrigin, Griewank, Ackley and Schwefel 1.2, respectively. Only the best six results sorted by the fitness average for each function are presented here. The highlighted values are indications of success for all search performed by FSS.

Table 8 presents the comparison between the FSS and PSO approaches. It contains the best results achieved for the five benchmark functions used to evaluate the performance of the four algorithms.

Table 3 Simulation Results for the Rosenbrock Function – Fitness (average and standard deviation) for 30 trials

$step_{ind,initial}$	$step_{ind,final}$	$step_{vol,initial}$	$step_{vol,final}$	$fitness_{(average)}$	$fitness_{(stddev)}$
0.1	**0.001**	**1**	**0.01**	**16.1183**	**0.729559**
0.1	0.0001	1	0.01	16.4036	0.853030
0.1	0.0001	1	0.001	16.4470	0.770458
0.1	0.001	1	0.001	16.4629	0.797471
10	0.01	1	0.001	44.7585	7.785530
10	0.1	0.1	0.001	46.4926	5.676689

Table 4 Simulation Results for the Rastringin Function – Fitness (average and standard deviation) for 30 trials

$step_{ind,initial}$	$step_{ind,final}$	$step_{vol,initial}$	$step_{vol,final}$	$fitness_{(average)}$	$fitness_{(stddev)}$
10	**0.01**	**10**	**0.1**	**13.3868**	**4.005888**
10	0.1	10	0.01	13.7376	2.889882
10	0.1	10	0.1	14.1285	3.680864
10	0.01	10	0.01	14.5193	2.780258
10	0.01	0.1	0.0001	200.225	22.09435
10	0.01	0.1	0.001	200.652	19.42133

Table 5 Simulation Results for the Griewank Function – Fitness (average and standard deviation) for 30 trials

$step_{ind,initial}$	$step_{ind,final}$	$step_{vol,initial}$	$step_{vol,final}$	$fitness_{(average)}$	$fitness_{(stddev)}$
1	**0.001**	**1**	**0.01**	**0.00270**	**0.002291**
0.1	0.0001	1	0.001	0.00373	0.004015
1	0.001	10	0.01	0.00377	0.002375
0.1	0.001	1	0.001	0.00445	0.003982
0.1	0.0001	1	0.01	0.00499	0.004313
0.1	0.001	1	0.01	0.00603	0.004149

Table 6 Simulation Results for the Ackley Function – Fitness (average and standard deviation) for 30 trials

$step_{ind,initial}$	$step_{ind,final}$	$step_{vol,initial}$	$step_{vol,final}$	$fitness_{(average)}$	$fitness_{(stddev)}$
10	**0.01**	**10**	**0.1**	**0.04004**	**0.020568**
10	0.01	10	0.01	0.08393	0.041568
10	0.01	1	0.01	0.15836	0.032344
10	0.01	1	0.001	0.16337	0.031565
10	0.1	10	0.1	0.18650	0.038461
10	0.1	10	0.01	0.20383	0.039025

Table 7 Simulation Results for the Schwefel 1.2 Function – Fitness (average and standard deviation) for 30 trials

$step_{ind,initial}$	$step_{ind,final}$	$step_{vol,initial}$	$step_{vol,final}$	$fitness_{(average)}$	$fitness_{(stddev)}$
1	**0.001**	**1**	**0.01**	**0.08085**	**0.022414**
1	0.001	1	0.001	0.09159	0.032054
1	0.01	1	0.001	0.09478	0.031902
1	0.01	1	0.01	0.09720	0.026483
1	0.001	0.1	0.001	0.37266	0.264715
1	0.001	10	0.01	0.61065	0.139308

Notice that the FSS algorithm outperforms the original PSO in all the cases. Moreover, FSS achieved excellent results for notoriously hard multimodal functions such as the Rastrigin, Griewank, and Ackley.

Table 8 Overall comparison of results between algorithms – Fitness (average and standard deviation) for 30 trials

Function	**Fitness (average and standard deviation)**			
	Orig. PSO	***Constricted PSO (G_{best})***	***Constricted PSO (L_{best})***	***FSS***
Rosenbrock	54.6867 (2.8570)	**8.1579 (2.7835)**	12.6648 (1.2304)	16.118 (0.729)
Rastrigin	400.7194 (4.2981)	140.4876 (4.8538)	144.8155 (4.4066)	**13.386 (4.005)**
Griewank	1.0111 (0.0031)	0.0308 (0.0063)	**0.0009 (0.0005)**	0.0027 (0.002)
Ackley	20.2769 (0.0082)	17.6628 (1.0232)	17.5891 (1.0264)	**0.0400 (0.020)**
Schefel 1.2	5.4572 (0.1429)	**0.0 (0.0)**	0.1259 (0.0178)	0.0808 (0.022)

6 Discussion and Conclusions

In this chapter, we have detailed the general ideas and principles embedded in FSS. This novel search algorithm is quite promising as a search tool for dealing with multimodal high dimensional problems, as it may be concluded from the examples provided in previous sections.

The performance of FSS on some multimodal functions was surprisingly good, especially when compared to monomodal ones.

Although previous works [15, 16, 17] have similar titles and motivations, our approach is quite different as it considers bio-inspired operators to directly guide the search process. Additionally, FSS presents an interesting balance between exploration and exploitation abilities, self-adapts quite swiftly out of local minima (towards sought solutions), and self-regulates the search granularity.

We foresee that FSS will most likely receive a great number of extensions in the near future, namely, sea currents, springs, predators, reefs, corals and other barriers to the school progression; all of them, situations to be avoided or taken advantage of. Altogether, these extensions may allow FSS to deal with noise, attractors, repulsors and no-go regions. Finally, breeding is another bio-inspired feature that ought to be considered further in the near future.

References

1. Michalewicz, Z., Fogel, D.: How to solve it - modern heuristics, 2nd edn. Springer, Heidelberg (2004)
2. Holland, J.: Adaptation in natural and artificial systems. University of State of Michigan Press, Ann Arbor (1975)
3. Eiben, A., Smith, J.: Introduction to evolutionary computing. Springer, Heidelberg (2003)

4. de Castro, L., Timmis, J.: Artificial immune systems - a new computational intelligence approach. Springer, Heidelberg (2002)
5. Dorigo, M., Gambardella, L.: Ant colony systems: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
6. Dorigo, M., Blum, C.: Ant colony optimization theory: A survey. Theoretical Computer Science 344, 243–278 (2005)
7. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948 (1995)
8. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39–43 (1995)
9. Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: Proceedings of the IEEE Swarm Intelligence Symposium (SIS 2007), Hawaii, USA, pp. 120–127 (2007)
10. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
11. Carvalho, D.F., Bastos-Filho, C.J.A.: Clan particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2008), Hong Kong, pp. 3044–3051 (2008)
12. Mendes, R., Kennedy, J., Neves, J.: Watch thy neighbor or how the swarm can learn from its environment. In: Proceedings of the IEEE Swarm Intelligence Symposium (SIS 2003), Indiana, USA, pp. 88–94 (2003)
13. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Hawaii, USA, pp. 1671–1676 (2002)
14. Suganthan, P.N.: Particle swarm optimiser with neighbourhood operator. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999), Washington, DC, USA, pp. 1958–1962 (1999)
15. Oboshi, T., Kato, S., Mutoh, A., Itoh, H.: Effectiveness of strays on ability of fish school to escape from predator. In: Proceedings of the 7th Asia-Pacific Conference on Complex Systems, Cairns, Australia (2004)
16. Luo, F.F., Chen, G.L., Guo, W.Z.: An improved 'fish-search' algorithm for information retrieval. In: Proceedings of IEEE International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE 2005), Wuhan, China, pp. 523–528 (2005)
17. Hersovici, M., Jacovi, M., Maarek, Y., Pelleg, D., Shtalhaim, M., Ur, S.: The shark-search algorithm. An application: tailored Web site mapping. Computer Networks and ISDN Systems 30(1-7), 317–326 (1998)

Magnifier Particle Swarm Optimization

Ying Tan and Junqi Zhang

Abstract. In this chapter, after a brief introduction to the Particle Swarm Optimization (PSO), a novel PSO algorithm based on magnification transformation called Magnifier Particle Swarm Optimization (MPSO) is presented. In the MPSO, the range around each generation's best individual is enlarged akin to using a magnifier, while the velocity of particles stays unchanged. This way, the MPSO achieves much faster convergence speed and better optimization solving capability than the Standard Particle Swarm Optimization (SPSO) by a number of simulations. In the context, the proposed MPSO algorithm is described and explained in detail by comparing it with the SPSO. Simulations on thirteen benchmark test functions are conducted to verify the effectiveness of the MPSO. Our experimental results show that the proposed MPSO not only speeds up the convergence tremendously but also maintains a strong capability of searching for the global solution with high accuracy. The application to spam detection shows that the proposed MPSO gives a promising result.

1 Introduction

PSO is a stochastic global optimization technique inspired by the social behavior of bird flocking or fish schooling [12, 33]. In the conventional PSO, each particle in a swarm population adjusts its position in the search space according to the best position it has found so far and also the overall best position found so far by the whole swarm. The essence of PSO is to use particles with best known positions to guide the swarm population to converge to a single optimum in the search space. Compared to other population based evolutionary algorithms, i.e., genetic algorithms, PSO does not need genetic operators such as crossover and mutation. Thus it has the advantages of easy implementation, fewer parameters to be adjusted, strong capability to

Ying Tan · Junqi Zhang
Key Laboratory of Machine Perception (Ministry of Education), Peking University, and Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, P.R. China
e-mail: `ytan@pku.edu.cn`, `zhangjunqi@cis.pku.edu.cn`

escape from local extrema as well as rapid convergence. Since the PSO comprises a very simple concept and can be implemented easily, it has been successfully utilized in many practical engineering fields such as function optimization, artificial neural network training, fuzzy system control, blind source separation as well as machine learning. Furthermore, the PSO has also been found to be robust and fast in solving nonlinear, nondifferentiable and multimodal problems. A review of PSO application is presented in [37] by identifying and analyzing around 700 PSO application papers stored in IEEE Xplore database at the time of writing. In this chapter, a simple magnification transformation is introduced into the PSO, resulting in a novel magnifier PSO (MPSO).

2 Theoretical Analysis and Variants of PSO

Most research on PSO concentrates on the theoretical analysis and the development of variants of PSO. Actually, the theoretical analysis and the development of variants of PSO are usually promoted by each other. Several theoretical analyses of the dynamics of particle swarms have been offered over the last decade. The first theoretical model of the PSO was presented in [3, 4]. Similar assumptions were used in [25] to analyze the trajectory of particles. One particle, without randomness and during stagnation, was modeled in [5]. An eigen-value analysis of the resulting dynamic system [23] was performed to determine the parameter settings that lead to system stability, and the different classes of particle's behaviors possible. How the spatial extent of a particle swarm varies over time is investigated in [39]. Analyses of a 4-parameter family of particles' model and regions identified in the parameter space are provided in [8]. The distribution of velocities of one particle controlled by the update rule of the PSO with inertia term and stochastic forces are analyzed in [26] for better understanding of the behavior of the PSO during phases of stagnation. The stability of particles in the presence of stochasticity was studied in [41] by using Lyapunov stability analysis. The velocity term of PSO is discussed in detail in [44, 14, 16].

Recently, the probability distributions were analyzed in [14, 15, 16, 17]. In [14, 15], the described probabilistic models select the next point solely based on the previous bests, using a random number generator to produce a candidate problem solution vector from a probability distribution. The previous position of a particle is not taken into account. In [17], the particle's movement over time is defined as a series of points, each selected from the previous one. Empirical trials show that the effective probability distribution is a truncated triangle, with uniform probability across the middle and decreasing in the tails. The "truncated triangle" was termed "Maya pyramid" by James Kennedy and discussed in [27] in detail. The sampling distribution of the PSO is analyzed under the assumption of stagnation in [34, 35, 36, 37]. In [34], a discrete markov chain model of the bare bone PSO was built using finite element grid technique. In [35, 36], the characteristics of a PSO's sampling distribution and their change over generations was determined in the presence of stochasticity under the assumption of stagnation.

Chapter 5 of Holland's book [7], "optimum allocation of trials" reveals the delicate balance between conservative testing of known regions versus risky exploration of the unknown. It appears that the PSO can allocate trials almost optimally [12]. Trelea has given an explanation of the exploitation and exploration for the PSO: "The PSO algorithm includes some parameters that greatly influence its performance, often stated as the exploration-exploitation trade-off: Exploration is the ability to test various regions in the problem space in order to locate a good optimum, hopefully the global one. Exploitation is the ability to concentrate the search around a promising candidate solution in order to locate the optimum precisely." [8]. A parameter of inertia weight was introduced into the original particle swarm optimizer to increase the tendency of particles to explore the unknown search space [44]. A complete theoretical analysis of the algorithm has been given by [25]. Based on this analysis, the authors derived a reasonable set of parameters. The exploration-exploitation trade-off is discussed and illustrated in [8]. Despite these recent efforts, it is still hard to explain how the particle swarm works. Furthermore, selection of the algorithm's parameters remains empirical to a large extent. Simple user-oriented guidelines for the parameter selection for a specific problem are not straightforward.

Since its invention, PSO has attracted extensive attention and interest of researchers from different scientific and engineering domains. Many researchers have worked on improving its performance in various ways, which generally involve balancing the exploitation and exploration of the particles in the swarm. A clever technique for creating a discrete binary version of the PSO introduced by Kennedy and Eberhart [13] in 1997 uses the concept of velocity as a probability that takes on either one or zero. By analyzing the convergence behavior of the PSO, a variant of the PSO with a constriction factor was introduced by Clerc and Kennedy [25], which guarantees convergence and at the same time improves the convergence speed sharply. Parsopoulos and Vrahatis proposed a unified particle swarm optimizer (UPSO) which combines both the global version and local version together [24]. A cooperative particle swarm optimizer was also proposed in [6]. Furthermore, El-Abd and Kamel proposed a Hierarchal Cooperative Particle Swarm Optimizer [28]. In [40], Peram et al. proposed a fitness-distance ratio based particle swarm optimization (FDR-PSO), by defining the "neighborhood" of a particle as the n closest particles of all particles in the population. Very recently, a comprehensive learning particle swarm optimizer (CLPSO) was proposed to greatly improve the performance of the original PSO on multi-modal problems by a novel learning strategy [20]. In [43], Tan proposed a novel strategy of PSO called Clonal PSO (abbreviated as CPSO), which clones and mutates the best particles of certain generations to join the swarm, and then selects the better ones to continue evolving. A stretching technique was introduced into PSO by Parsopoulos and Plagianakos in [29], which applies a two-stage transformation to the shape of the fitness function that eliminates undesired local minima, but preserves the global ones. An ARC-PSO [21] was proposed by introducing the advance and retreat strategy based on the clonal mechanism from [43]. A RBH-PSO [22] was introduced by randomly attracting the particles to the best known area, by which the exploitation of particles is increased.

3 Principle and Analysis of PSO

In [2], Daniel Bratton and James Kennedy defined a Standard PSO model (abbreviated as SPSO), which includes a local ring topology, the constricted update rules in Eqs. (1) and (2), 50 particles, non-uniform swarm initialization, and boundary conditions wherein a particle is not evaluated when it exits the feasible search space. This version is designed to be a straightforward extension of the original algorithm while taking into account more recent developments that are expected to improve performance on standard measures. This standard algorithm is intended for use both as a baseline to test the performance of improvements introduced to the technique, as well as to represent PSO to the wider optimization community. The constriction coefficient χ in this case is defined in Equation (3). It was found that when $\varphi < 4$, the swarm would slowly "spiral" toward and around the best found solution in the search space with no guarantee of convergence, while for $\varphi \geq 4$ convergence would be quick and guaranteed. While it is possible to weigh the velocity update equation to favor the best position of the individual particle or the best position of the entire swarm by adjusting the values of c_1 and c_2, most implementations of constricted particle swarms use equal values for both parameters for the sake of simplicity. Using the constant $\varphi = 4.1$ to ensure convergence, the values $\chi \approx 0.72984$ and $c_1 = c_2 = 2.05$ are obtained.

$$\hat{V}_{id}(t+1) = \chi(V_{id}(t) + c_1\varepsilon_1(P_{iBd}(t) - X_{id}(t)) + c_2\varepsilon_2(P_{gBd}(t) - X_{id}(t))), \quad (1)$$
$$X_{id}(t+1) = X_{id}(t) + \hat{V}_{id}(t+1). \quad (2)$$

$$\chi = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}, \varphi = c_1 + c_2. \quad (3)$$

The other update rule with inertia weight as in Eqs (4) and (5) is algebraically equivalent to a PSO with constriction [25].

$$\hat{V}_{id}(t+1) = wV_{id}(t) + c_1 r_1(P_{iBd}(t) - X_{id}(t)) + c_2 r_2(P_{gBd}(t) - X_{id}(t)), \quad (4)$$
$$X_{id}(t+1) = X_{id}(t) + \hat{V}_{id}(t+1). \quad (5)$$

In Eqs. (4) and (5), $i = 1, 2, \cdots, n$, n is the number of particles in the swarm, $d = 1, 2, \cdots, D$, and D is the dimension of solution space. The learning factors c_1 and c_2 are nonnegative constants, r_1 and r_2 are random numbers uniformly drawn from the interval $[0, 1]$, which are all scalar quantities for each particle in each dimension. The parameter $w \in [-1, 1]$ in Eq. (4) is the inertia weight for the inertia velocity. P_{iBd} and P_{gBd} are the locations of the best positions found so far by particle i and its neighbors in dimension d. The termination for iterations in the PSO is determined according to whether it reaches the designated fitness value or the fixed maximum number of fitness evaluations.

Intuitively, the more particles, the faster the search will be in terms of the number of iterations. However, this iteration count is not really a relevant criterion. Rather, the number of times that the function must be evaluated is significant, because this evaluation requires a considerable time in the majority of real problems. The number of evaluations is obviously equal to the number of particles in the swarm. If one wants to reduce the total number of evaluations needed to find a solution, one is tempted to decrease the size of the swarm. However, too small a swarm is likely to take longer to find a solution or even fail to find a solution. Therefore, a compromise must be reached. Empirically, sizes of about 20 to 50 particles have been proposed, which, indeed, proved entirely sufficient to solve almost all classic test problems [27]. It should, however, be noted that such small value does not facilitate comparison with other methods. Using 100 genes as in genetic algorithms would not be as effective in terms of the number of evaluations as this gene size is too large for PSO. Similarly, if only 20 genes are used for genetic algorithms, the solutions are not always found.

The position of every particle in each dimension is updated independently. The only link between the dimensions in problem space is introduced by the objective function, via P_{iBd} and P_{gBd}. Thus, for the purpose of analysis, without any loss of generality, the algorithmic description can be reduced to the one-dimensional particle case by dropping the subscript d and i, as shown in Eqs. (6) and (7),

$$V(t+1) = wV(t) + c_1r_1(P_{pB}(t) - X(t)) + c_2r_2(P_{gB}(t) - X(t)), \tag{6}$$
$$X(t+1) = X(t) + V(t+1), \tag{7}$$

where $P_{pB}(t)$ denotes the historical best position of this particle, and $P_{gB}(t)$ denotes the historical best position in this particle's neighborhood in the current t-th generation. It is clear that, in each generation, the PSO uses these two factors to guide the particles to evolve.

For the sake of convenience, we make the following denotations,

$$d_1 = P_{pB}(t) - X(t), \tag{8}$$
$$d_2 = P_{gB}(t) - X(t). \tag{9}$$

the update function of the PSO in Eqs (6) and (7) can be rewritten as follows,

$$X(t+1) = X(t) + wV(t) + Z(t). \tag{10}$$

where $Z(t)$ is the hybrid uniform distribution.

We consider that, in each generation, the PSO tries to guide all particles to perform two jobs, *local search* in the space around P_{gB} and P_{pB} (exploitation), and *random search* in the rest of the space (exploration). Here, we first give reasonable definitions of the exploitation and exploration areas in the one-dimensional search space according to the PSO model. Next, we provide exact calculations of the probabilities of the exploitation and exploration jobs using the exact density function of $X(t+1)$. When we define the exploitation and exploration, we share the same basic idea with [8], and improve them in three aspects. First, our definitions are quantified and can be calculated accurately. Second, our definitions can be illustrated and analyzed. Third, our definitions can help users understand the PSO and tune the PSO in an easier way.

Definition 1 (Exploitation Area): The sampling area of a particle X for exploitation is defined as

$$A(exploitation) = [(P_{pB} - |d_1|), (P_{pB} + |d_1|)] \bigcup [(P_{gB} - |d_2|), (P_{gB} + |d_2|)]. \quad (11)$$

Definition 2 (Exploration Area): The sampling area of a particle X for exploration is defined as

$$A(exploration) = R - A(exploitation). \quad (12)$$

where R denotes a set of all real numbers.

Definition 3 (Exploitation Probability): The probability of a particle X landing in A(exploitation) is defined as

$$P(exploitation) = \int_{A(exploitation)} f_X(x)dx. \quad (13)$$

Definition 4 (Exploration Probability): The probability of a particle X landing in A(exploration) is defined as

$$P(exploration) = \int_{A(exploration)} f_X(x)dx. \quad (14)$$

For convenience, we use the abbreviations $A(ita)$, $A(ra)$, $P(ita)$ and $P(ra)$ for A(exploitation), A(exploration), P(exploitation) and P(exploration), respectively. As can be seen by Eqs. (13) and (14), there exists

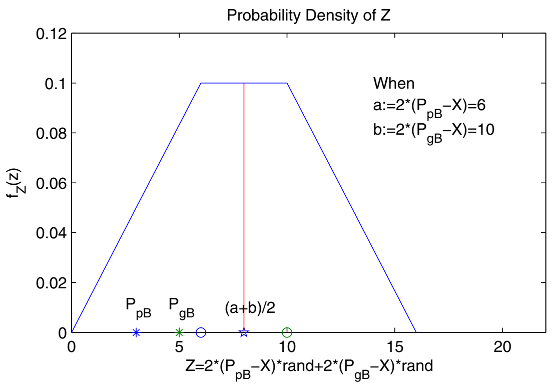
$$P(ra) + P(ita) = 1, \quad (15)$$

so, we focus solely on the $P(ita)$ sequences in the following sections.

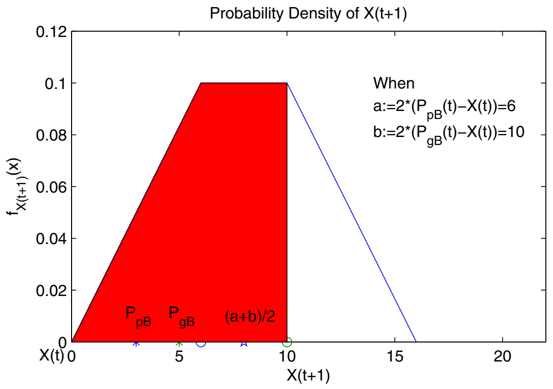
As in Eq. (10), at time t, $X(t)$ and $wV(t)$ are constants, which only give $Z(t)$ translations. So, given the probability density function of $Z(t)$, we can calculate the exact one-step transition probability density function of $X(t+1)$ of every particle in each dimension. The probability density function of $Z(t)$ can be calculated by assuming $Z(t)$ to be Z for simplification. The $f_Z(z)$ can be calculated as illustrated in Figure 1(a).

According to the above analyses, given the information of $t-th$ generation, we can calculate the exact $P(ita)$ and $P(ra)$. First, we consider the PSO without the inertia velocity term for its facility in each generation. In this case, $w = 0$, so

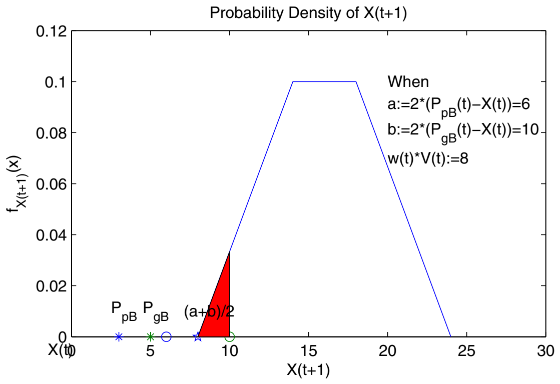
$$X(t+1) = X(t) + Z(t). \quad (16)$$



(a) The distribution density of the hybrid uniform distribution Z(t).



(b) The allocation of P(ita) (the dark coloured area) and P(ra) (the remaining area inside the trapezia) when there is no inertia velocity part.



(c) The allocation of P(ita) (the dark coloured area) and P(ra) (the remaining area inside the trapezia) with inertia velocity part.

Fig. 1 The density function of Z, and the allocation of P(ita)

We set the current position of a particle $X(t)$ at the origin, and only consider one-step movement of the particle in each dimension. The corresponding allocations of the $P(ita)$ and $P(ra)$ are illustrated in detail in Figure. 1(b). The dark area is the $P(ita)$, and the rest of the area inside the trapezia is the $P(ra)$.

Similarly, we consider the PSO with the inertia velocity term, as shown in Eq. (10). The area of the sampling distribution of $X(t+1)$ will be shifted due to the $wV(t)$ term. The allocations of the $P(ita)$ and $P(ra)$ are illustrated in Figure. 1(c).

As can be seen from these figures, the introduction of the inertia velocity term results in the displacement of $X(t+1)$, which increases the probability of $X(t+1)$ landing farther from $X(t)$, makes the allocation change acutely, and facilitates the exploration job.

4 Magnifier PSO

4.1 Magnification Transformation

Magnification transformation is a simple but very useful strategy, which is inspired by the use of a convex lens to observe objects precisely. The essence of this transformation is to set a magnifier around a point of interest, so that the range around the point could be inspected more carefully and precisely. For example, this transformation is regularly used in building screen magnifiers to enlarge the information presented on a visual display in a computer system [31]. Transformation strategies can be divided into two categories: Linear transformations and Non-Linear transformations (which include Fisheye Zoom, Hyperbolic, 3D Pliable Surfaces) [38]. The essence of the SPSO is to use particles with best known positions to guide the swarm to converge to a single optimum in the search space. However, the way the best known individual affects other particles in the swarm is a critical issue. This becomes even more acute when the problem to be solved has multiple optima since the entire swarm or population could be potentially misled to many local optima. In [29], attention was paid to the top part of the fitness function to eliminate undesired local minima by a two-stage transformation. The original function was changed in the stretching PSO to make search easier. Conversely, we will focus on the bottom of the fitness landscape since the range around the best individual deserves a better check, and the probability of the actual global best particle lying in that range is probably greater than others in the search space. The sketch map of MPSO on a composition function is shown in Figure 2. In MPSO, the original function is not changed; just the range mentioned above is enlarged via magnification transformation, while keeping the velocity of particles unchanged. At this speed, it will speed up the local search while maintaining MPSO'S global search capability.

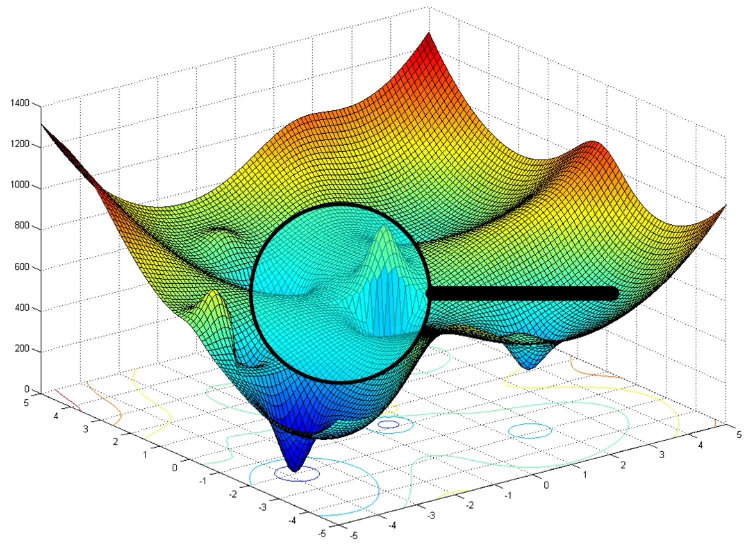


Fig. 2 Sketch map of the MPSO on a composition test function

4.2 MPSO Algorithm

Based on the above discussion, we propose a magnifier operator for the SPSO. First, in each generation, a range around the best individual is set up in each dimension. If the particles in the swarm pass through the range in the next generation, the magnifier operator would enlarge the range without changing the velocity of particles. Thus the particles would have a better chance of landing in the range and check the area around the current best individual more precisely. For those particles who were already going to land in the range, we do not use the magnifier operator on them, because they have already shown interest in the range. On the other hand, we keep the velocity of the particles unchanged so that they are able to fly out of the range after certain generations for maintaining the global search ability of the SPSO. In MPSO, the position transition process of a particle x from the t-th to $(t+1)$-th generation in each dimension can be schematically expressed as four situations in Figs. 3(a)- 3(d), respectively. For each situation, the position after the application of the magnifier operator in the MPSO will be calculated by Eqs. (17) - (20), respectively:

$$\tilde{x}(t+1) = x(t+1) - (2*r/s - 2*r)$$
$$if \quad x(t) < L \quad and \quad x(t+1) > R, \tag{17}$$
$$\tilde{x}(t+1) = x(t+1) + (2*r/s - 2*r)$$
$$if \quad x(t) > R \quad and \quad x(t+1) < L, \tag{18}$$
$$\tilde{x}(t+1) = x(t+1) - [(R - x(t))/s - (R - x(t))]$$

$$if \quad L < x(t) < R \quad and \quad x(t+1) > R, \tag{19}$$

$$\tilde{x}(t+1) = x(t+1) + [(x(t) - L)/s - (x(t) - L)]$$
$$if \quad L < x(t) < R \quad and \quad x(t+1) < L. \tag{20}$$

In Figs. 3(a)- 3(d), x_{gB} is the position of the current global best-fit particle, $x(t)$ is the position of the particle in the current *t-th* generation, $x(t+1)$ is the position that x is supposed to be in the next generation without the magnifier operation, $\tilde{x}(t+1)$ is its actual position in the next generation after the magnifier operation has been applied, and *range* is the interval containing the current best particle.

In Eqs. (17) - (20), r is the radius of the interval whose left and right boundaries are indicated by L and R; s is the scale which decides the on the degree of magnification to enlarge the range.

In Figs. 3(a) and 3(b), the particle $x(t)$ is supposed to pass through the range completely, and the actual position $\tilde{x}(t+1)$ in our MPSO would be calculated by Eqs. (17) and (18), respectively. In Figs. 3(c) and 3(d), the particle $x(t)$ is supposed to pass through the range partially, which denotes the particles already in the range. The actual position $\tilde{x}(t+1)$ can be computed using Eqs. (19) and (20), respectively. Moreover, r should reduce along with the growth of generations, because we want the best individuals to converge inside the range. So, we should fix an initial value for r, from which it reduces linearly to zero. The iterative equation of r is expressed by Equ. (21).

$$r = r * (1 - k/M), \tag{21}$$

where k is the current iteration number and M is the maximum iteration number we set.

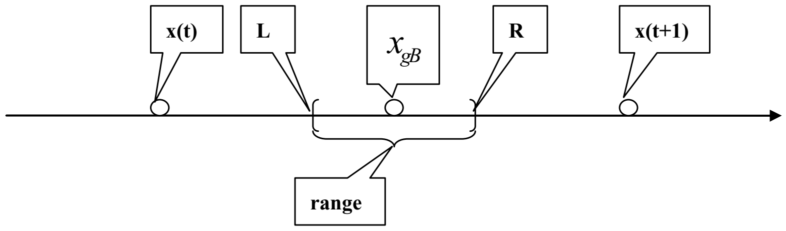
Briefly, the MPSO algorithm has been summarized in Algorithm 1.

Algorithm 1. MPSO Algorithm

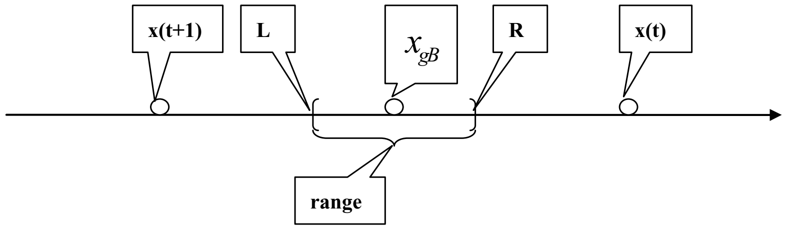
Step 1: Initialization. Assume $c_1 = 2$, $c_2 = 2$, and w be from 0.9 to 0.4 linearly.
Step 2: The state evolution of particles is iteratively updated according to Eqs. (4) and (5).
Step 3: Find the current best-fit particle X_{gB}, and set a range around it in each dimension. The radius of the range is decided by r.
Step 4: Magnification operation. For every particle except for X_{gB}, for each situation mentioned in Figs. 3(a)- 3(d), update its position using equations (17)- (20), respectively.
Step 5: Termination. The algorithm can be terminated by a given maximum number of fitness value evaluations or a preset solution accuracy. In our experiments, we adopt the former stop criterion, i.e. a maximum number of fitness value evaluations, which is 1,200,000 in this study. If the termination condition is not met, go to step 2.

Note that the basic idea about MPSO was also reported in [30] in advance.

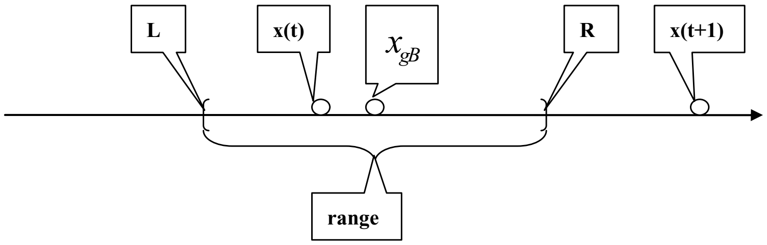
Since, in real-life applications, the optimization cost is usually dominated by evaluations of the objective function, the expected number of fitness evaluations is retained as the main algorithm performance criterion.



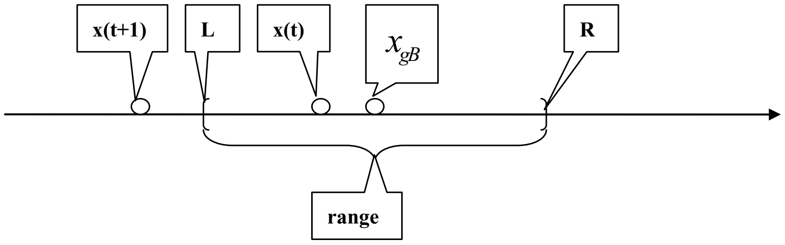
(a) $x(t)$ pass through the range completely along right direction



(b) $x(t)$ pass through the range completely along left direction



(c) $x(t)$ pass through the range partially along right direction



(d) $x(t)$ pass through the range partially along left direction

Fig. 3 Situation 1 - 4: $x(t)$ pass through the range along left and right directions

4.3 Analysis and Discussion

In each generation of the SPSO, we search the space according to Eqs. (4) and (5), and find a current best position x_{gB}. According to our analysis, SPSO does not make a good use of the information from x_{gB} which guides us to search the space around it more carefully. So, the range around x_{gB} is enlarged to give it a more precise search, which in turn speed up the local search. On the other hand, the velocity of particles remains unchanged during the iterations which guarantees the global search capability.

The essence of MPSO is to adjust the particles to search the solution space more pertinently. We increase the probability of particles landing into the range around x_{gB}, maintain the probability of particles flying far from x_{gB}, and decrease the probability of particles wandering around the range containing x_{gB}. In such a way, the particles wandering around the range of x_{gB} are forced to join the range to enhance the local search. In other words, all the particles are given only two choices, either they land very close to x_{gB} to enhance the local search ability or land far from x_{gB} to keep the global search capability. So, MPSO simply makes the particles search the space more pertinently and efficiently to improve both the convergent speed and global search performance without adding much computational cost.

CPSO [43] has a similar mechanism of searching the range around the positions x_{gB} more carefully as MPSO. However, CPSO is more complex by the introduction of the mutation and selection operations, has more computational cost and requires much more memory. On the contrary, the MPSO has a very simple operation like SPSO, which is easier to conduct and understand.

The stretching PSO in [29] makes an indirect search of the optima by introducing a two-stage transformation, which changes the test function, and greatly increases the total number of function evaluations. When the exact shape of the function is unknown, this strategy will run into trouble.

In summary, our MPSO is simpler and more effective than the current improved algorithms in SPSO.

5 Simulations

In order to compare the performance of MPSO and SPSO, thirteen benchmark functions from the $CEC'05$ Test Functions [32] were chosen to be the objective functions, as shown in Table 1. In our simulations, we adopt the number of fitness value evaluations as a criterion of comparison. The stop criterion, i.e. the maximum number of fitness evaluations is set to 50,000 for this study. FEs denote the number of the fitness evaluations. In addition, the MPSO and the SPSO have the same features for the sake of comparison. They include a local ring topology, the constricted update rules in Eqs. (1) and (2), 50 particles, non-uniform swarm initialization, and boundary conditions wherein a particle is not evaluated when it exits the feasible search space.

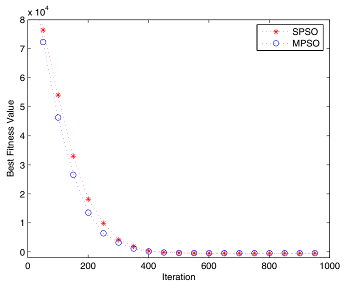
Table 1 List of thirteen benchmark test functions and their parameters from CEC'05

Functions	Exp.	Search Space	f_{bias}	*Type*
Shifted Sphere	F_1	$[-100,100]^D$	-450	unimodal
Shifted Schwefel's Problem 1.2	F_2	$[-100,100]^D$	-450	unimodal
Shifted Rotated High Conditioned Elliptic	F_3	$[-100,100]^D$	-450	unimodal
Shifted Schwefel's Problem 1.2 with Noise in Fitness	F_4	$[-100,100]^D$	-450	unimodal
Schwefel's Problem 2.6 with Global Optimum on Bounds	F_5	$[-100,100]^D$	-310	unimodal
Shifted Rosenbrock	F_6	$[-100,100]^D$	390	basic multimodal
Shifted Rotated Griewanks Function without Bounds	F_7	$[0,600]^D$	-180	basic multimodal
Shifted Rotated Ackleys Function with Global Optimum on Bounds	F_8	$[-32,32]^D$	-140	basic multimodal
Shifted Rastrigin's Function	F_9	$[-5,5]^D$	-330	basic multimodal
Shifted Rotated Rastrigin	F_{10}	$[-5,5]^D$	-330	basic multimodal
Shifted Rotated Weierstrass	F_{11}	$[-0.5,0.5]^D$	90	basic multimodal
Schwefel's Problem 2.13	F_{12}	$[-\pi,\pi]^D$	-460	basic multimodal
Expanded Extended Griewanks plus Rosenbrocks Function (F8F2)	F_{13}	$[-3,1]^D$	-130	extended multimodal

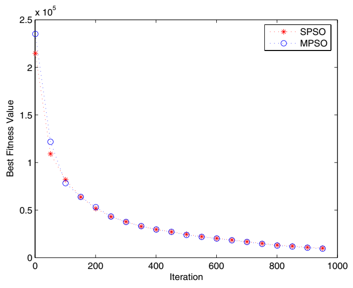
The parameters involved in the proposed MPSO include s and r, which denote the scale of the magnifier operator and the radius of the range in each dimension, respectively. It is probably impossible to find a unique set of algorithm parameters that work well in all cases because the best tradeoff between exploration and exploitation, depends strongly on the function being optimized. The best values of these papramaters may vary for different problems. According to experiments, one good combination is $s = r = 0.1$. So in the following experiments, we let s be 0.1 and let r reduce from 0.1 to 0 linearly along with the iteration number.

The performance comparisons between the proposed MPSO algorithm and SPSO algorithm on thirteen typical benchmark test functions have been shown in Figs. 4 and 5. The convergent curves have been drawn from the averaged values of 20 independent runs. This way, these curves can demonstrate the stable performances of the MPSO and SPSO algorithms completely and reliably. It can be observed from these figures that our proposed MPSO algorithm has much greater speed of convergence than that of the SPSO algorithms on most of the benchmark test functions.

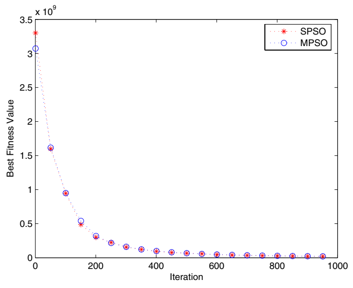
Furthermore, in order to verify the effectiveness and efficiency of our MPSO, the statistical means and standard deviations of the obtained solutions of the thirteen benchmark test functions listed in Table 1 are provided in Table 2 by using MPSO and SPSO, over 20 independent runs. FEs denote the number of fitness value



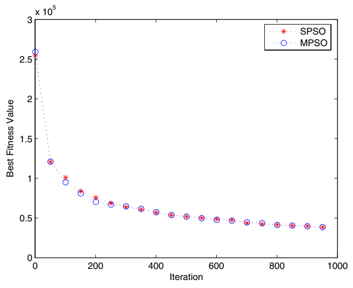
(a) F_1



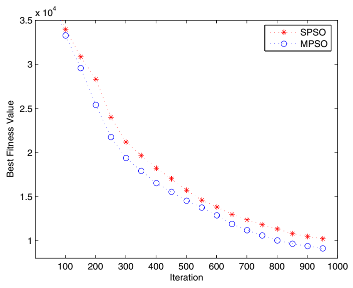
(b) F_2



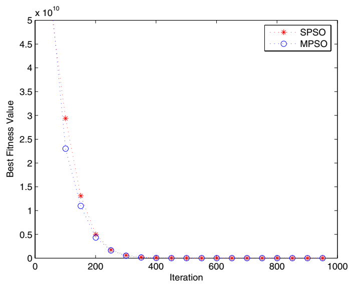
(c) F_3



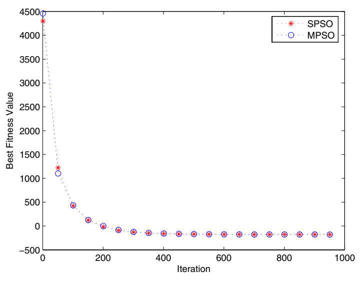
(d) F_4



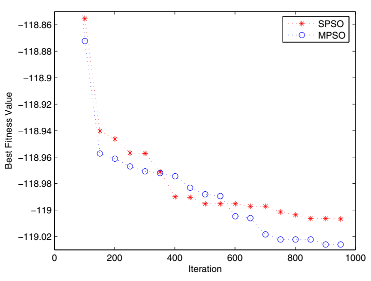
(e) F_5



(f) F_6

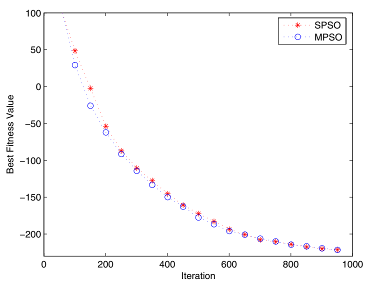


(g) F_7

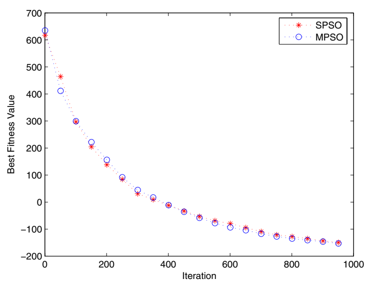


(h) F_8

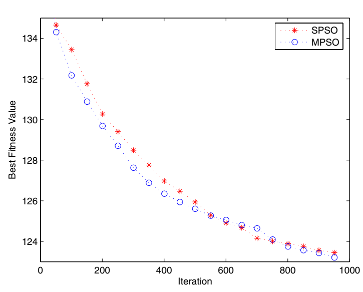
Fig. 4 The averaged performances of the MPSO and SPSO on $F_1 - F_8$ in Table 1 on 20 independent runs with 40 particles in a swarm



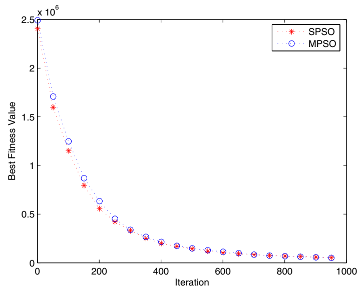
(a) F_9



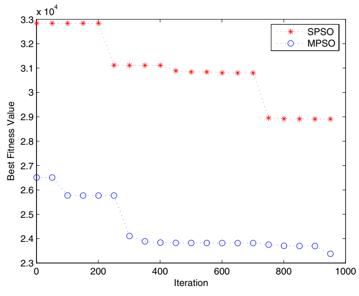
(b) F_{10}



(c) F_{11}



(d) F_{12}



(e) F_{13}

Fig. 5 The averaged performances of the MPSO and SPSO on $F_9 - F_{13}$ in Table 1 on 20 independent runs with 40 particles in a swarm

evaluations of the swarm. It can been seen from the averaged solutions that our proposed MPSO outperforms the SPSO on most of the functions except the F_3, F_4 and F_7 functions.

It can be concluded from the comparisons of performances between MPSO and SPSO that the MPSO not only has a much faster convergence speed, but also has a more accurate optimal solution than the SPSO on most of the thirteen benchmark functions used in the simulations.

Table 2 Statistical means and standard deviations of the solutions of thirteen benchmark test functions, listed in Table 1, given by the MPSO and the SPSO over 20 independent runs

Func	**MPSO's M ± S**	**SPSO's M ± S**
F_1	**-449.9977± 0.0024**	-449.9969 ± 0.0022
F_2	**8.7017e+003 ±2.5447e+003**	8.8899e+003± 4.4882e+003
F_3	1.8750e+007±6.3259e+006	**1.8492e+007± 8.6569e+006**
F_4	3.8060e+004±8.5856e+003	**3.6913e+004±8.4731e+003**
F_5	**8.8096e+003±1.4042e+003**	1.0019e+004±2.2322e+003
F_6	**866.4773 ± 438.7067**	1.0046e+003±849.1794
F_7	-178.8568±0.1176	**-178.8719± 0.0748**
F_8	**-119.0347± 0.0836**	-119.0067± 0.0740
F_9	**-223.0912± 31.4580**	-222.6391±23.8404
F_{10}	**-157.3146±31.4300**	-154.7425±33.4417
F_{11}	**122.8890± 2.9707**	123.2119± 2.2154
F_{12}	**4.4655e+004± 2.0974e+004**	4.8664e+004 ± 2.1744e+004
F_{13}	**2.3361e+004±2.3592e+004**	2.8910e+004± 1.8748e+004

6 Application of MPSO to Spam Detection

Spam, which is usually defined as unsolicited commercial email (UCE), unsolicited bulk email (UBE), or uninterested email from the perspective of an individual email user, has been considered as an increasingly serious problem to the infrastructure of Internet. According to the statistics from ITU (International Telecommunication Union), about 70% to 80% of the present emails sent over the Internet are spam. They not only occupy valuable communications bandwidth and storage space, but also threaten the network security when used as a carrier of viruses and malicious codes. Meanwhile, spam decreases the receiver's productivity considerably as precious time is wasted in tackling them.

Many solutions have been put into practice for solving spam problems so far. [1] investigates how PSO algorithm can help select features relevant for spam email classification. Here, we use the taxonomy of current methods to summarize these solutions, i.e, simple approaches, intelligent approaches and hybrid approaches. Simple approaches include munging, listing, aliasing and challenging. These techniques are easy to implement but can be quite easily deceived. Intelligent approaches play an increasingly important role in anti-spam in recent years because of the self-learning ability and good performance. They include Naive Bayes, Support Vector Machine (SVM) [42], Artificial Neural Network (ANN), Artificial Immune System (AIS) and DNA Computing. An anti-spam shield with one technique alone can be easily intruded in practice. Consequently, several hybrid approaches by combining two or more techniques together have been proposed in an attempt to improve overall performance whilst overcoming the shortcomings of each single approach. SVM has already proved its superiority in pattern recognition for its generalization performance. AIS has some desirable properties for spam detection, including pattern recognition, dynamically changing coverage and noise tolerance. Thus, we utilize some of the characteristics of these algorithms in our algorithm. SVM is a classification algorithm based on the Structural Risk Minimization principle from statistical learning theory formulated by Vapnik. The goal of SVM is to find an optimal hyperplane for which the lowest true error can be guaranteed. A concentration based feature construction (CFC) approach, inspired by the human immune system, is employed to characterize each email through a two-element feature vector. In the CFC approach, self concentration and none-self concentration are constructed by using self gene library and none-self gene library, respectively, and are subsequently used to form a two-element concentration vector which characterizes the email efficiently.

SVM is used as the classifier for spam classification. Two corpus used to test our proposed CFC approaches are the PU1 corpus [9] and Ling corpus [10]. PU1 corpus consists of 1,099 messages, with spam rate 43.77%, while the Ling corpus consists of 2,893 messages, with spam rate 16.63%. All the messages in both corpus have header fields, attachment and HTML tags removed, leaving only subject line and mail body text. In PU1, each token is mapped to a unique integer to ensure the privacy of the content while keeping its original form in Ling. Each corpus is divided into ten partitions with approximately equal amount of messages and spam rate. The version with stop-word removal is used in our simulations. LIBSVM software package is used for the implementation of SVM. Polynomial kernel with three parameters, i.e., gamma, coef0 and degree, is adopted. Together with the cost parameter C, there are four parameters to be optimized. Proposed MPSO is employed to tune the above four parameters. A corresponding test function model with four parameters as input and classification accuracy as output is established. The classification accuracy measured by 10-fold cross validation serves as the fitness evaluation in optimization process of above PSOs. The PSOs terminate when the fitness of the global best particle has not changed for 50 consecutive generations.

Comparisons of the performance of MPSO, Naive Bayesian, Linger-V and SVM-IG are made as shown in Table 3, which shows the accuracy of the MPSO, Naive Bayesian, Linger-V and SVM-IG on corpus PU1 and Ling. Naive Bayesian, Linger-V and SVM-IG are reported in [9, 10, 19, 11]. Linger-V is a NN-based system for automatic e-mail classification. For Naive Bayesian, the version of the corpus adopted in the simulations is the original version. For Linger-V and SVM-IG, the stemming versions are used. All these results are obtained by using 10-fold validation. For Naive Bayesian, 50 words with the highest mutual information scores are selected. LINGER-V and SVM-IG use variance (V) and information gain (IG) as feature selection criteria respectively and the 256 best scoring features are chosen.

Table 3 Performances of MPSO, Naïve Bayesian (NB), Linger-V and SVM-IG on corpus PU1 and Ling, using 10-fold cross validation

Data Sets	MPSO (%)	**NB** (%)	**Linger-V**(%)	**SVM-IG**(%)
PU1	**99.09**	91.07	93.45	93.18
Ling	**99.82**	96.40	98.2	96.85

It can be seen that the proposed MPSO can be used to tune the classifier to promote the accuracy of the classification.

7 Conclusions

On the basis of the theoretical analysis of hybrid uniform distribution of PSO, a variant, named as MPSO was proposed and implemented based on a magnification transformation in this chapter. By enlarging the range around the best individual of every generation and keeping the velocity of particles unchanged, the MPSO is characterized by a better optimization solving capability and convergence performance than the SPSO. The experimental results on thirteen benchmark test functions have demonstrated that the proposed MPSO algorithm is able to speed up the evolution process and improve the performance of the global optima greatly. The application of MPSO on spam detection shows that the proposed MPSO gives a promising result. The better performance of MPSO is mostly credited to the magnification transformation which increased the exploitation of the PSO around the best position of the swarm. However, the developing of a PSO variant which can increase the exploitation and exploration simultaneously is not clear yet. Raising the efficiency of the trade-off between exploitation and exploration in PSO is one area of our future research interest.

Acknowledgements. This work was supported by the National High Technology Research and Development Program of China (863 Program), with Grant No.2007AA01Z453, and partially supported by the National Natural Science Foundation of China under Grant No. 60875080, 60803065, and 60673020. This work was also financially supported by the Research Fund for the Doctoral Program of Higher Education (RFDP) in China and China Postdoctoral Science Foundation with Grant No.20070420257.

References

1. Lai, C.C., Wu, C.H.: Particle swarm optimization-aided feature selection for spam email classification. In: Proceedings of the Second International Conference on Innovative Computing, Information and Control, pp. 165–168 (2007)
2. Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: Proceedings of the IEEE Swarm Intelligence Symposlum (SIS), pp. 120–127 (2007)
3. Ozcan, E., Mohan, C.: Analysis of a simple particle swarm optimization system. In: Proceedings of the IEEE International Conference on Intelligent Engineering Systems Through Artificial Neural Networks, pp. 253–258 (1998)
4. Ozcan, E., Mohan, C.: Particle swarm optimization: surfing the waves. In: Proceedings of the IEEE Congress on Evolution Computation, pp. 1939–1944 (1999)
5. van den Bergh, F.: An analysis of particle swarm optimizers. PhD thesis, Department of Computer Science, University of Pretoria, South Africa (2002)
6. van den Bergh, F., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. IEEE Trans. on Evolutionary Computation 8, 225–239 (2004)
7. Holland, J.H.: Adaption in natural and artificial systems. MIT Press, Cambridge (1975)

8. Trelea, I.C.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85, 317–325 (2003)
9. Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Spyropoulos, C.D.: An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 160–167 (2000)
10. Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G., Spyropoulos, C.D.: An evaluation of naive Bayesian anti-spam filtering. In: Potamias, G., Moustakis, V., Someren, M.V. (eds.) Proceedings of the workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, pp. 9–17 (2000)
11. Koprinska, I., Poon, J., Clark, J., Chan, J.: Learning to classify e-mail. Information Science 177, 2167–2187 (2007)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
13. Kennedy, J.: The particle swarm: social adaption of knowledge. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 13–16 (1997)
14. Kennedy, J.: Bare bones particle swarms. In: Proceedings of the IEEE International Conference on Swarm Intelligence Symposium, pp. 24–26 (2003)
15. Kennedy, J.: Probability and dynamics in the particle swarm. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 340–347 (2004)
16. Kennedy, J.: Why does it need velocity. In: Proceedings of the IEEE International Conference on Swarm Intelligence Symposium, pp. 38–44 (2005)
17. Kennedy, J.: Dynamic-probabilistic particle swarms. In: Proceedings of the IEEE International Conference on Genetic and Evolutionary Computation Conference, pp. 201–207 (2005)
18. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 1671–1676 (2002)
19. Clark, J., Koprinska, I., Poon, J.: A neural network based approach to automated E-mail classification. In: Proceedings of IEEE International Conference on Web Intelligence, pp. 702–705 (2003)
20. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimi-zation of multimodal functions. IEEE Trans. on Evolutionary Computation 10, 281–296 (2006)
21. Zhang, J.Q., Liu, K., Tan, Y., He, X.G.: Hybrid particle swarm optimizer with advance and retreat strategy and clonal mechanism for global numerical optimiza-tion. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 2059–2066 (2008)
22. Zhang, J.Q., Liu, K., Tan, Y., He, X.G.: Random black hole particle swarm optimization. In: Proceedings of International Conference on Neural Networks and Signal Processing, pp. 359–365 (2008)
23. Yasuda, K., Ide, A., Iwasaki, N.: Adaptive particle swarm optimization. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, pp. 1554–1559 (2003)
24. Parsopoulos, K.E., Vrahatis, M.N.: UPSO-a united particle swarm optimization scheme. In: Proceedings of the International Conference of Computational Methods in Sciences and Engineering, pp. 868–873 (2004)
25. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. on Evolutionary Computation 6(1), 58–73 (2002)
26. Clerc, M.: Stagnation analysis in particle swarm optimization or what happens when nothing happens. Technical Report CSM-460 (2006)

27. Clerc, M.: Particle swarm optimization. In: Proceedings of International Scientific and Technical Encyclopedia (2006)
28. El-Abd, M., Kamel, M.S.: A hierarchal cooperative particle swarm optimizer. In: Proceedings of Swarm Intelligence Symposium, pp. 43–47 (2006)
29. Parsopoulos, K.E., et al.: Stretching technique for obtaining global minimizers through particle swarm optimization. In: Proceedings of the Workshop on Particle Swarm Optimization, pp. 22–29 (2001)
30. Zhang, J.Q., Liu, K., Tan, Y., He, X.G.: Magnifier particle swarm optimization for numerical optimization. In: Proceedings of ACM SIGEVO Gentic and Evolutionary Computation Conference (GECCO 2008), pp. 167–168 (2008)
31. Blenkhorn, P., Evans, D.G.: A screen magnifier using 'high level' implementation techniques. IEEE Trans. on Neural Systems and Rehabilitation Engineering 14(4), 501–504 (2006)
32. Suganthan, P.N., et al.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. In: Proceedings of the IEEE Swarm Intelligence Symposlum (SIS) (2005)
33. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the 6th Int. Symp. Mcro Machine Human Science, pp. 39–43 (1995)
34. Poli, R., Langdon, W.B.: Markov chain models of bare-bones particle swarm optimizers. In: Proceedings of the IEEE International Conference on Genetic And Evolutionary Computation Conference, pp. 142–149 (2007)
35. Poli, R., Broomhead, D.: Exact analysis of the sampling distribution for canonical particle swarm optimiser and its convergence during stagnation. In: Proceedings of the IEEE International Conference on Genetic and Evolutionary Computation Conference, pp. 134–141 (2007)
36. Poli, R.: On the moments of the sampling distribution of particle swarm optimizers. In: Proceedings of the IEEE International Conference on Genetic and Evolutionary Computation Conference, pp. 2907–2914 (2007)
37. Poli, R.: Dynamics and stability of the sampling distribution of particle swarm optimisers via moment analysis. Journal of Artificial Evolution and Applications 1, 1–10 (2008)
38. Keahey, T.A., Robertson, E.L.: Techniques for nonlinear magnification transformations. In: Proceedings of the IEEE Symposium on Information Visualization, pp. 38–45 (1996)
39. Blackwell, T.M.: Particle swarms and population diversity. In: Proceedings of IEEE International Conference on Soft Computing, pp. 793–802 (2005)
40. Peram, T.: Fitness-distance-ratio based particle swarm optimization. In: Proceedings of Swarm Intelligence Symposium, pp. 174–181 (2003)
41. Kadirkamanathan, V., et al.: Stability analysis of the particle dynamics in particle swarm optimizer. IEEE Trans. Evolutionary Computation 10(3), 245–255 (2006)
42. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
43. Tan, Y., Xiao, Z.M.: Clonal particle swarm optimization and its applications. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 2303–2309 (2007)
44. Shi, Y.H., Eberhart, R.: A modified particle swarm optimizer. In: Proceedings of IEEE World Congress on Computational Intelligence, pp. 69–73 (1998)

Improved Particle Swarm Optimization in Constrained Numerical Search Spaces

Efrén Mezura-Montes and Jorge Isacc Flores-Mendoza

Abstract. This chapter presents a study about the behavior of Particle Swarm Optimization (PSO) in constrained search spaces. A comparison of four well-known PSO variants used to solve a set of test problems is presented. Based on the information obtained, the most competitive PSO variant is detected. From this preliminary analysis, the performance of this variant is improved with two simple modifications related with the dynamic control of some parameters and a variation in the constraint-handling technique. These changes keep the simplicity of PSO i.e. no extra parameters, mechanisms controlled by the user or combination of PSO variants are added. This Improved PSO (IPSO) is extensively compared against the original PSO variants, based on the quality and consistency of the final results and also on two performance measures and convergence graphs to analyze their on-line behavior. Finally, IPSO is compared against some state-of-the-art PSO-based approaches for constrained optimization. Statistical tests are used in the experiments in order to add support to the findings and conclusions established.

1 Introduction

Nowadays, it is common to find complex problems to be solved in diverse areas of human life. Optimization problems can be considered among them. Different sources of difficulty can be associated in their resolution e.g. a very high number of possible solutions (very large search spaces), hard-to-satisfy constraints and a high nonlinearity. Mathematical Programming (MP) offers a set of techniques to solve different type of problems like numerical, discrete or combinatorial optimization problems. This chapter focuses only on numerical (continuous) optimization problems. MP techniques are always the first option to solve optimization problems. In

Efrén Mezura-Montes · Jorge Isacc Flores-Mendoza
Laboratorio Nacional de Informática Avanzada (LANIA A.C.), Rébsamen 80, Centro, Xalapa, Veracruz, 91000, México
e-mail: `emezura@lania.mx`, `jflores@lania.edu.mx`

fact, they provide, under some specific conditions to be accomplished by the problem, convergence to the global optimum solution. However, for some real-world problems, MP techniques are either difficult to apply (i.e. a problem transformation may be required), they cannot be applied or they get trapped in local optimum solutions. Based on the aforementioned, the use of heuristics to solve optimization problems has become very popular in different areas. Tabu Search [11], Simulated Annealing [16] and Scatter Search [12] are examples of successful heuristics commonly used by interested practitioners and researchers to solve difficult search problems. There is also a set of nature-inspired heuristics designed for optimization problem-solving and they comprise the area of Bio-inspired optimization. Two main groups of algorithms can be distinguished: (1) Evolutionary algorithms (EAs) [7] and (2) Swarm Intelligence algorithms (SIAs) [9]. EAs are based on the theory of evolution and the survival of the fittest. A set of complete solutions of a problem are represented and evolved by means of variation operators and selection and replacement processes. There are three main paradigms in this area: (1) Evolutionary Programming [10], Evolution Strategies [37] and Genetic Algorithms [14]. There are other important EAs proposed such as Genetic Programming [17] where solutions are represented by means of nonlinear structures like trees and its aim is oriented to symbolic optimization and Differential Evolution [34], designed to solve numerical optimization problems by using vector differences as search directions coupled with an EA framework.

On the other hand, SIAs emulate different social and cooperative behaviors found in animals or insects. The two original paradigms are the following: (1) Particle Swarm Optimization (PSO) [15] and (2) Ant Colony Optimization (ACO) [6]. PSO is based on the cooperative behavior of bird flocks, whereas ACO models social behaviors of ants e.g. the foraging behavior as to solve mainly combinatorial optimization problems.

These Bio-Inspired Algorithms (BIAs), such as genetic algorithms, evolutionary programming, evolution strategies, differential evolution and particle swarm optimization, share some features. They work with a set of complete solutions for the problem (usually generated at random). These solutions are evaluated in order to obtain a quality measure, i.e. fitness value, for each one of them. A selection mechanism is then implemented as to select those solutions with a better fitness value. These best solutions will be utilized to generate new solutions by using variation operators. Finally, a replacement process occurs, where the size of the population (which was increased) is trimmed as to always maintain a fixed population size.

In their original versions, BIAs are designed to solve unconstrained optimization problems. Then, there is a considerable amount of research dedicated to designing constraint-handling techniques to be added to BIAs. There are some classifications for constraint-handling techniques based on the way they incorporate feasibility information in the quality of a given solution [4, 30]. For the purpose of this chapter, a simple taxonomy is proposed, because the main goal of the current study is not to design a novel constraint-handling mechanism. Instead, the aim is to propose the analysis of the behavior of a BIA (PSO in this case) as a first step in designing a competitive approach to solve Constrained Numerical Optimization Problems (CNOPs).

As a result, the simplicity of PSO is maintained i.e. no additional mechanisms and/or parameters controlled by the user are considered.

This chapter is organized as follows: Section 2 contains the statement of the problem of interest and some useful optimization concepts. Section 3 introduces PSO in more detail, considering its main elements and variants. A brief introduction to constraint-handling techniques is summarized in Section 4. In Section 5, the approaches which use PSO to solve CNOPs are detailed and discussed. Section 6 presents the empirical comparison of PSO variants and a discussion of results. After that, Section 7 details the modifications made to the most competitive PSO variant obtained from the previous study, all of them in order to improve its performance when solving CNOPs. An in-depth study of the behavior of this novel PSO and a comparison against state-of-the-art PSO-based approaches to solve CNOPs are presented in Section 8. The chapter ends with a conclusion and a discussion of future work in Section 9.

2 Constrained Optimization Problems

The optimization process consists of finding the best solution for a given problem under certain conditions. As it was mentioned before, this chapter will only consider numerical optimization problems in presence of constraints. Without loss of generality a CNOP can be defined as to:

Find $\mathbf{x}$ which minimizes

$$f(\mathbf{x}) \tag{1}$$

subject to

$$g_i(\mathbf{x}) \leq 0, \ \ i = 1, \ldots, m \tag{2}$$

$$h_j(\mathbf{x}) = 0, \ \ j = 1, \ldots, p \tag{3}$$

where $\mathbf{x} \in \mathbf{R}^n$ is the vector of solutions $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ and each x_i, $i = 1, ..., n$ is bounded by lower and upper limits $L_i \leq x_i \leq U_i$ which define the search space $\mathscr{S}$, $\mathscr{F}$ comprises the set of all solutions which satisfy the constraints of the problems and it is called the feasible region; m is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or nonlinear). Equality constraints are transformed into inequalities constraints as follows: $|h_j(\mathbf{x})| - \varepsilon \leq 0$, where ε is the tolerance allowed (a very small value).

As multiobjective concepts will be used later in the chapter, the multiobjective optimization problem will be also introduced. Without loss of generality, a Multiobjective Optimization Problem (MOP) is defined as:

Find $\mathbf{x}$ which minimizes

$$f(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_k(\mathbf{x})]^T \tag{4}$$

subject to

$$g_i(\mathbf{x}) \ \leq \ 0, \ i = 1, .., m \tag{5}$$

$$h_j(\mathbf{x}) \ = \ 0, \ j = 1, ..., p \tag{6}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the vector of solutions $\mathbf{x} = [x_1, x_2, ..., x_n]^T$ and each x_i, $i = 1, ..., n$ is bounded by lower and upper limits $L_i \leq x_i \leq U_i$ which define the search space $\mathscr{S}$, $\mathscr{F}$ is the feasible region; m is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or nonlinear).

A vector $\mathbf{u}= (u_1, ..., u_k)$ is said to dominate another vector $\mathbf{v} = (v_1, ..., v_k)$ (denoted by $\mathbf{u} \preceq \mathbf{v}$) if and only if $\mathbf{u}$ is partially less than $\mathbf{v}$, i.e. $\forall i \in \{1, ..., k\}$, $u_i \leq v_i \wedge \exists i \in \{1, ...k\} : u_i < v_i$.

3 Particle Swarm Optimization

Kennedy and Eberhart [15] proposed PSO, which is based on the social behavior of bird flocks. Each individual "i", called particle, represents a solution to the optimization problem i.e. a vector of decision variables $\mathbf{x}_i$. The particle with the best fitness value is considered the leader of the swarm (population of particles), and guides the other members to promising areas of the search space. Each particle is influenced on its search direction by cognitive (i.e. its own best position found so far, called $\mathbf{x}_{pbest_i}$) and social (i.e. the position of the leader of the swarm named $\mathbf{x}_{gBest}$) information. At each iteration (generation) of the process, the leader of the swarm is updated. These two elements: $\mathbf{x}_{pbest_i}$ and $\mathbf{x}_{gBest}$, besides the current position of particle "i" $\mathbf{x}_i$, are used to calculate its new velocity $\mathbf{v}_i(t+1)$ based on its current velocity $\mathbf{v}_i(t)$ (search direction) as follows:

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1 r_1 (\mathbf{x}_{pbest_i} - \mathbf{x}_i) + c_2 r_2 (\mathbf{x}_{gBest} - \mathbf{x}_i). \tag{7}$$

where c_1 and c_2 are acceleration constants to control the influence of the cognitive and social information respectively and r_1, r_2 are random real numbers between 0 and 1 generated with an uniform distribution.

After each particle updates its corresponding velocity, the flight formula is used to update its position:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1). \tag{8}$$

where $\mathbf{x}_i$(t) is the current position of the particle, $\mathbf{x}_i(t+1)$is the new position of this particle and $\mathbf{v}_i(t+1)$ is its recently updated velocity (search direction).

Based on Equation 7, two main different approaches have been proposed to update the velocity of a particle. The aim is to improve the usefulness of the search direction generated and to avoid premature convergence: (1) PSO with inertia weight and (2) PSO with constriction factor.

3.1 PSO with Inertia Weight

Proposed by Shi and Eberhart [38], the inertia weight was added to the velocity update formula (Equation 7) as a mechanism to control PSO's exploration and exploitation capabilities. Its goal is to control the influence of the previous velocity of a given particle. The inertia weight is represented by w and scales the value of the current velocity $\mathbf{v}_i(t)$ of particle "i". A small inertia weight value promotes local exploration, whereas a high value promotes global exploration. Shi and Eberhart [38] suggested w=0.8 when using PSO to solve unconstrained optimization problems. The modified formula to update the velocity of a particle by using the inertia weight value is the following:

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1 r_1 (\mathbf{x}_{pbest_i} - \mathbf{x}_i) + c_2 r_2 (\mathbf{x}_{gBest} - \mathbf{x}_i) \tag{9}$$

3.2 PSO with Constriction Factor

With the aim of eliminating velocity clamping and encouraging convergence, Clerc and Kennedy [3] proposed, instead of a inertia weight value, a constriction coefficient. This constriction factor is represented by k. Unlike the inertia weight, the constriction factor affects all values involved in the velocity update as follows:

$$\mathbf{v}_i(t+1) = k[\mathbf{v}_i(t) + c_1 r_1 (\mathbf{x}_{pbest_i} - \mathbf{x}_i) + c_2 r_2 (\mathbf{x}_{gBest} - \mathbf{x}_i)] \tag{10}$$

3.3 Social Network Structures

There are two basic PSO variants depending of the social network structure used [9]: (1) global best and (2) local best PSO. In the global best variant the star social structure allows each particle to communicate with all the remaining particles in the swarm, whereas in the local best PSO, the ring social structure allows each particle to communicate only with those particles in its neighborhood. Therefore, in the global best PSO, there is a unique leader of the swarm. On the other hand, in local best PSO, there is a leader for each neighborhood. There are differences expected in the behavior of these two PSO variants due to the way particles communicate among themselves. In global best PSO a faster convergence is promoted as the probability of being trapped in local optima is increased. However, in local best

```
Begin
    GEN = 0
    Generate a swarm of random solutions ($\mathbf{x}_i$), $i = 1,2,...,SWARM\_SIZE$.
    Initialize for each particle, $\mathbf{x}_{pbest_i}$ = $\mathbf{x}_i$ , and $\mathbf{v}_i(t) = 0$.
    Evaluate the fitness of each particle in the swarm.
    Do
        Select the leader ($\mathbf{x}_{gBest}$) of the swarm.
        For each particle, update its velocity with (7).
        For each particle, update its position with (8).
        Evaluate the fitness of the new position for each particle.
        Update the $\mathbf{x}_{pbest_i}$ (memory) value for each particle.
        GEN=GEN+1
    Until GEN = Gmax
End
```

Fig. 1 Global best PSO pseudocode

PSO, a slower convergence usually occurs while a better exploration of the search space is encouraged.

A pseudocode for the global best PSO is presented in Figure 1.

A pseudocode for the local best PSO is presented in Figure 2.

```
Begin
    GEN = 0
    Generate a swarm of random solutions ($\mathbf{x}_i$) $i = 1,2,...,SWARM\_SIZE$.
    Divide the swarm in $n$ neighborhoods.
    Assign equal number of particles to each neighborhood.
    Initialize for each particle, $\mathbf{x}_{pbest_i} = \mathbf{x}_i$, and $\mathbf{v}_i(t) = 0$.
    Do
        Evaluate the fitness of the particle in each neighborhood.
        Select the leader ($\mathbf{x}_{lBest_i}$) of each neighborhood.
        For each particle, update its velocity with (7).
            by using the corresponding leader of each neighborhood $\mathbf{x}_{lBest_i}$
        For each particle, update its position with (8).
        Evaluate the fitness of the new position for each particle.
        Update the $\mathbf{x}_{pbest_i}$ (memory) value for each particle.
        GEN=GEN+1
    Until GEN= Gmax
End
```

Fig. 2 Local best PSO pseudocode

4 Constraint-Handling

As it was mentioned in the introduction to the chapter, EAs and SIAs were originally designed to solve unconstrained optimization problems. Constraint-handling techniques are required to add feasibility information in the fitness calculation of a solution [4, 30]. Roughly, constraint-handling techniques can be divided in two groups:

1. Those based on the fitness penalization of a solution i.e. a combination of the objective function value (Equation 1) and the sum of constraint violation (Equations 2 and 3) .
2. Those based on the separated use of the objective function value (Equation 1) and the sum of constraint violation (Equations 2 and 3) in the fitness value of a solution.

In the first group penalty functions are considered, which is in fact the most popular constraint-handling mechanism. They transform a constrained problem into an unconstrained problem by punishing, i.e. decreasing, the fitness value of infeasible solutions in such a way that feasible solutions are preferred in the selection/replacement processes. However, an important drawback is the definition of penalty factor values, which determine the severity of the penalization. If the penalty is too low, the feasible region may never be reached. On the other hand, if the penalty is too high, the feasible region will be reached so fast, mostly at random and the probability of getting trapped in local optimum might be very high [4].

The second group includes constraint-handling techniques based on Deb's feasibility rules [5], Stochastic Ranking [35], multiobjective concepts [28], lexicographic ordering [36], the α-constrained method [40], Superiority of Feasible points [33] among others.

Different search engines have been used on the above mentioned approaches: Genetic Algorithms [5, 28], Evolution Strategies [35], Differential Evolution [40]. However, to the best of the authors' knowledge, the research usually focuses on adapting a constraint-handling mechanism to a given search engine, but the studies to analyze the performance of a search engine in constrained search spaces are scarce [29].

5 Related Work

This section presents PSO-based approaches proposed to solve CNOPs. Toscano and Coello [42] proposed a global best PSO with inertia weight coupled with a turbulence (mutation) operator, which affects the velocity vector of a particle as follows: $\mathbf{v}_i = \mathbf{v}_j^{\Phi}(t) + r_3$, where $\mathbf{v}_i$ is the current velocity of particle i, $\mathbf{v}_j^{\Phi}(t)$ is the current velocity of its nearest neighbor and r_3 is a random value. The use of this turbulence operator is calculated with a dynamic adaptation approach. The idea is to use more of the turbulence operator in the first part of the search. The constraint-handling technique used was a group-2 approach [5].

Parsopoulos and Vrahatis [32] used their Unified Particle Swarm Optimization (UPSO) to solve CNOPs. The UPSO combines the exploration and exploitation abilities of two basic PSO variants (local best and global best together, both with constriction factor). The scheme of UPSO is the following: A weighted sum of the two velocity values (from the local and global variants) is computed, where a parameter $(0 \leq u \leq 1)$ represents the unification factor and controls the influence of each variant in the final search direction. Finally a typical flight formula with this unified velocity is used to update the position of a particle. A group-1 constraint-handling technique i.e. static penalty function, was used in this approach where the number of violated constraints as well as the amount of constraint violation were taken into account.

Liang and Suganthan [23] proposed a PSO-based approach to solve CNOPs by using dynamic multi-swarms (DMS-PSO). The DMS-PSO was implemented using a local best PSO with inertia weight, where the size and particles of the sub-swarms change periodically. Two concepts are modified from the original PSO in DMS-PSO: (1) Instead of just keeping the best position found so far, all the best positions reached for a particle are recorded to improve the global search and (2) a local search mechanism, i.e. sequential quadratic programming method, was added. The constraints of the problems are dynamically assigned, based on the difficulty to be satisfied for each sub-swarm. Moreover, one sub-swarm will optimize the objective function. As the function and constraints are handled separately, they use a group-2 constraint-handling. The sequential quadratic programming method is applied to the pbest values (not to the current positions of the particles) as to improve them.

Li, Tian and Kong [21] solved CNOPs by coupling an inertia weight local best PSO with a mutation strategy. Their constraint-handling mechanism belongs to group 2 and was based on the superiority of feasible points [33]. The mutation strategy used a diversity metric for population diversity control and for convergence improvement. When the population diversity was low (based on a defined value), the swarm is expanded through the mutation strategy. This mutation strategy consisted of a random perturbation applied to the particles in the swarm. Li, Tian and Min [22] used a similar constraint-handling mechanism, but without mutation strategy and using instead a global best PSO variant to solve Bilevel Programming Problem (BLPP).

Lu and Chen [25] implemented a group-2 constraint-handling technique by using a global best PSO with inertia weight and velocity restriction. The original problem (CNOP) is transformed into a bi-objective problem using a Dynamic-Objective Strategy (DOM). DOM consists of the following: if a particle is infeasible, its unique objective is to enter the feasible region. On the other hand, if the particle is feasible its unique objective is now to optimize the original objective function. This process is dynamically adjusted according to the feasibility of the particle. The bi-objective problem is defined as: minimize $F(\mathbf{x}) = (\phi(\mathbf{x}), f(\mathbf{x}))$. $\phi(\mathbf{x})$ is the sum of constraint violations and $f(\mathbf{x})$ is the original function objective. Based on the feasibility of $\mathbf{x}_{gBest}$ and $\mathbf{x}_{pbest_i}$, the values of important parameters like c_1 and c_2 are defined to promote feasible particles to remain feasible. The formula to update the velocity is modified in such a way that the positions of the pbest and gbest are mixed in the search direction defined.

Cagnina, Esquivel and Coello [2] used a group-2 constraint-handling technique, Deb's rules [5], in a combination of global-local best PSO particle swarm optimizer to solve CNOPs. The velocity update formula and also the flight formula are changed as to include information of the global and local best leaders and to use a Gaussian distribution to get the new position for the particle, respectively. Furthermore, a dynamic mutation operator is added for diversity promotion in the swarm.

Wei and Wang [43] presented a global best PSO with inertia weight which transformed the problem, as in [25], into a bi-objective problem (group-2 constraint-handling). The original objective function was the second objective and the first one was the degree of constraint violation: $\min(\delta(\mathbf{x}), f(\mathbf{x}))$. Deb's feasibility rules were used as selection criteria. A new three-parent crossover operator (TPCO) is also added to the PSO. Finally, a dynamic adaptation for the inertia weight value was included to encourage a correct balance between global and local search.

Krohling and dos Santos Coelho [18] proposed a global best PSO with constriction factor and a co-evolutionary approach to solve CNOPs. This problem is transformed into a min-max problem. The Lagrange-based method (group-1 constraint-handling) is used to formulate the problem in terms of a min-max problem. Two swarms are used: The first one moves in the space defined by the variables of the problem, whereas the second swarm optimizes the Lagrange multipliers.

He, Prempain and Wu [13] proposed a ‘fly-back’ mechanism added to a global best PSO with inertia weight to solve CNOPs. In their approach, the authors also solved mixed (i.e. continuous-discrete) optimization problems. Discrete variables were handled by a truncation mechanism. The initial swarm must be always located in the feasible region of the search space, which may be a disadvantage when dealing with problems with a very small feasible region. The ‘fly-back’ mechanism keeps particles from flying out of the feasible region by discarding those flights which generate infeasible solutions. Then, the velocity value is reduced and a new flight is computed.

Based on the related work, some interesting modifications were found regarding PSO for solving CNOPs: (1) Mutation, crossover operators or even local search are added to PSO to promote diversity in the swarm [2, 21, 23, 42, 43], (2) there is a tendency to mix global and local best PSO variants into a single one [2, 32], (3) the original CNOP is transformed into a multiobjective problem [23, 25, 43], and finally, (4) the original velocity update and flight formulas are modified [2, 25].

6 Motivation and Empirical Comparison

Unlike the previous research, the motivation of this work is two-fold: (1) to acquire more knowledge about the behavior of PSO in its original variants when solving CNOPs and (2) after considering this knowledge as a first step of design, to propose simple modifications to PSO in order to get a competitive approach to solve CNOPs by maintaining PSO’s simplicity.

In this section, two original PSO variants (inertia weight and constriction factor) combined with two social network structures (star and ring) are compared. In the remaining of this chapter, each combination of variant-social network will be called as variant. They are selected based on the following criteria:

- They are the most used in the approaches reported in the specialized literature on numerical constrained optimization (Section 5).
- As mentioned in the beginning of this Section, the motivation of this work is to acquire knowledge about the behavior of PSO in its original variants i.e. variants without additional mechanisms.

The four variants are: (1) global best PSO with inertia weight, (2) global best PSO with constriction factor, (3) local best PSO with inertia weight and (4) local best PSO with constriction factor.

In order to promote a fair analysis of the four PSO variants and not add extra parameters to be fine-tuned, a group-2 (objective function and constraints handled separately) parameter-free constraint-handling technique is chosen for all the variants. This technique consists of a set of three feasibility rules proposed by Deb [5]. They are the following: (1) If two solutions are feasible, the one with the best value of the objective function is preferred, (2) if one solution is feasible and the other one is infeasible, the feasible one is preferred and (3) if two solutions are infeasible, the one with the lowest normalized sum of constraint violation is preferred.

24 test problems (all minimization problems) were taken from the specialized literature [24] and used to test the performance of the four PSO variants. These problems are an extension of the well-known benchmark used to test BIAs in constrained search spaces. In fact, these problems were used to evaluate state-of-the-art approaches in the IEEE Congress on Evolutionary Computation (CEC 2006). Details of the problems can be found in [24]. A summary of their features can be found in Table 1.

As can be noted, the problems have different characteristics such as dimensionality, type of objective function, type and number of constraints and active constraints at the optimum (i.e. the solution lies in the boundaries between the

Table 1 Details of the 24 test problems. "n" is the number of decision variables, $\rho = |F| / |S|$ is the estimated ratio between the feasible region and the search space, LI is the number of linear inequality constraints, NI the number of nonlinear inequality constraints, LE is the number of linear equality constraints and NE is the number of nonlinear equality constraints. a is the number of active constraints at the optimum.

Prob.	***n***	**Type of function**	ρ	**LI**	**NI**	**LE**	**NE**	***a***
g01	13	quadratic	0.0111%	9	0	0	0	6
g02	20	nonlinear	99.9971%	0	2	0	0	1
g03	10	polynomial	0.0000%	0	0	0	1	1
g04	5	quadratic	52.1230%	0	6	0	0	2
g05	4	cubic	0.0000%	2	0	0	3	3
g06	2	cubic	0.0066%	0	2	0	0	2
g07	10	quadratic	0.0003%	3	5	0	0	6
g08	2	nonlinear	0.8560%	0	2	0	0	0
g09	7	polynomial	0.5121%	0	4	0	0	2
g10	8	linear	0.0010%	3	3	0	0	6
g11	2	quadratic	0.0000%	0	0	0	1	1
g12	3	quadratic	4.7713%	0	1	0	0	0
g13	5	nonlinear	0.0000%	0	0	0	3	3
g14	10	nonlinear	0.0000%	0	0	3	0	3
g15	3	quadratic	0.0000%	0	0	1	1	2
g16	5	nonlinear	0.0204%	4	34	0	0	4
g17	6	nonlinear	0.0000%	0	0	0	4	4
g18	9	quadratic	0.0000%	0	12	0	0	6
g19	15	nonlinear	33.4761%	0	5	0	0	0
g20	24	linear	0.0000%	0	6	2	12	16
g21	7	linear	0.0000%	0	1	0	5	6
g22	22	linear	0.0000%	0	1	8	11	19
g23	9	linear	0.0000%	0	2	3	1	6
g24	2	linear	79.6556%	0	2	0	0	2

feasible and infeasible regions). Therefore, they present different challenges to the algorithms tested.

This first experiment is designed as follows: 30 independent runs were computed per PSO variant per test problem. Statistical results (best, mean and standard deviation) were calculated from the final results. They are presented, for the first twelve problems in Table 2 and for the last twelve in Table 3. The parameters used in this experiment are the following: 80 particles and 2000 generations (160,000 total evaluations), $c_1 = 2.7$ and $c_2 = 2.5$ for all PSO variants. For the two local best variants 8 neighborhoods were used, $w = 0.7$ for both inertia weight variants and $k = 0.729$ [3] for both constriction factor variants. The tolerance for equality constraints was set to ε =0.0001 for all variants.

These parameter values were defined by a trial and error process. The population size was varied from low values (40) to higher values (120), however no improvement was reported. c_1 and c_2 values required unusually higher values to provide competitive results. w and k values were taken as recommended in previous research [39, 38, 3] where the performance was the most consistent. In fact, PSO presented a high sensitivity to w and k values. Higher or lower values for these parameters decreased the performance of the variants, which, at times, were unable to reach the feasible region of the search space in some problems, despite slightly improving the results in other test functions.

Lower L_i and upper U_i limits for each decision variable i are handled in the flight formula (Equation 8) as follows: After the flight, if the new value $\mathbf{x}_i(t+1)$ is outside the limits, the velocity value $\mathbf{v}_i(t+1)$ is halved until the new position is within the valid limits. In this way, the search direction is maintained.

The results will be discussed based on quality and consistency. Quality is measured by the best solution found from the set of 30 independent runs. Consistency is measured by the mean and standard deviation values, i.e. a mean value closer to the best known solution and a standard deviation value close to zero indicate a more consistent performance of the approach.

In order to have more statistical support, nonparametric statistical tests were applied to the samples presented in Tables 2 and 3. Kruskal-Wallis test was applied to pair of samples with the same size (30 runs) and Mann-Whitney test was applied to samples with different sizes (<30 runs) as to verify if the differences shown in the samples are indeed significant. Test problems where no feasible solutions were found for all the algorithms e.g. g20 and g22, or when just the one variant found feasible results e.g. g13 and g17 are not considered in these tests. The results obtained confirmed the differences shown in Tables 2 and 3, except in the following cases, where the performance of the compared approaches is considered similar in problems g03 and g11 for the global best PSO with inertia weight and the local best PSO with constriction factor, in problem g11 for the local best PSO with inertia weight and the local best PSO with constriction factor and in problems g02, g03, g08 and g24 for both (global and local) constriction factor variants.

The results in Tables 2 and 3 suggest that the local best PSO with constriction factor (last column in Tables 2 and 3) provides the best performance overall. With

Table 2 Statistical results of 30 independent runs on the first 12 test problems for the four PSO variants compared."(n)" means that in only "n" runs feasible solutions were found. Boldface remarks the best result per function. "-" means that no feasible solutions were found in any single run.

STATISTICS FROM 30 INDEPENDENT RUNS FOR THE PSO VARIANTS

Problem & best-known solution		global best (w=0.7)	global best (k=0.729)	local best (w=0.7)	local best (k=0.729)
g01	Best	-14.961	-14.951	-14.999	**-15.000**
-15.000	Mean	-11.217	-11.947	-12.100	**-13.363**
	St. Dev.	2.48E+00	1.81E+00	3.05E+00	**1.39E+00**
g02	Best	-0.655973	-0.634737	-0.614785	**-0.790982**
-0.803619	Mean	-0.606774	-0.559591	-0.543933	**-0.707470**
	St. Dev.	2.64E-02	3.03E-02	**2.00E-02**	5.92E-02
g03	Best	-0.080	-0.019	-0.045	**-0.126**
-1.000	Mean	-9.72E-03	-1.72E-03	-1.00E-02	**-1.70 E-02**
	St. Dev.	1.60E-02	**4.64E-03**	1.20E-02	2.70E-02
g04	Best	-30655.331	-30665.439	**-30665.539**	**-30665.539**
-30665.539	Mean	-30664.613	-30664.606	**-30665.539**	**-30665.539**
	St. Dev.	5.70E-01	5.40E-01	**7.40E-012**	**7.40E-012**
g05	Best	-	-	5126.646 (18)	**5126.496**
5126.498	Mean	-	-	6057.259	**5140.060**
	St. Dev.	-	-	232.25E+00	**15.52E+00**
g06	Best	-6959.517	-6959.926	-6958.704	**-6961.814**
-6961.814	Mean	-6948.937	-6948.121	-6941.207	**-6961.814**
	St. Dev.	6.31E+00	6.41E+00	9.05E+00	**2.67E-04**
g07	Best	43.731	38.916	41.747	**24.444**
24.306	Mean	68.394	64.186	59.077	**25.188**
	St. Dev.	40.69E+00	17.15E+00	7.65E+00	**5.9E-01**
g08	Best	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**
-0.095825	Mean	-0.095824	**-0.095825**	**-0.095825**	**-0.095825**
	St. Dev.	1.75E-07	7.25E-08	**4.23E-17**	**4.23E-17**
g09	Best	692.852	693.878	696.947	**680.637**
680.630	Mean	713.650	708.274	728.730	**680.671**
	St. Dev.	12.96E+00	10.15E+00	15.80E+00	**2.10E-02**
g10	Best	8024.273	8769.477	8947.646	**7097.001**
7049.248	Mean	8931.263	9243.752	9247.134	**7641.849**
	St. Dev.	39.0.6E+01	22.94E+01	**18.4.7E+01**	36.14E+01
g11	Best	**0.749**	**0.749**	0.750	**0.749**
0.749	Mean	0.752	0.755	0.799	**0.749**
	St. Dev.	9.27E-03	1.40E-02	5.70E-02	**1.99E-03**
g12	Best	-0.999	-0.999	-0.999	**-1.000**
-1.000	Mean	-0.999	-0.999	-0.999	**-1.000**
	St. Dev.	6.96E-07	5.13E-07	2.59E-05	**0.00E+00**

Table 3 Statistical results of 30 independent runs on the last 12 test problems for the four PSO variants compared."(n)" means that in only "n" runs feasible solutions were found. Boldface remarks the best result per function. "-" means that no feasible solutions were found in any single run.

STATISTICS FROM 30 INDEPENDENT RUNS FOR THE PSO VARIANTS					
Problem & best-known solution		global best (w=0.7)	global best (k=0.729)	local best (w=0.7)	local best (k=0.729)
g13	Best	-	-	-	**8.10E-02**
0.053949	Mean	-	-	-	**0.45**
	St. Dev.	-	-	-	**2.50E-01**
g14	Best	-	-	-41.400 (9)	**-41.496** (3)
-47.764	Mean	-	-	-38.181	**-40.074**
	St. Dev.	-	-	2.18E+00	**1.45E+00**
g15	Best	-	-	967.519 (5)	**961.715**
961.715	Mean	-	-	970.395	**961.989**
	St. Dev.	-	-	2.62E+00	**3.9E-01**
g16	Best	-1.904	-1.903	-1.904	**-1.905**
-1.905	Mean	-1.901	-1.901	-1.904	**-1.905**
	St. Dev.	1.46E-03	1.37E-03	1.51E-04	**5.28E-11**
g17	Best	-	-	-	**8877.634**
8876.981	Mean	-	-	-	**8932.536**
	St. Dev.	-	-	-	**29.28E+00**
g18	Best	-	-	-0.450967 (3)	**-0.866023**
-0.865735	Mean	-	-	-0.287266	**-0.865383**
	St. Dev.	-	-	1.40E-01	**8.65E-04**
g19	Best	36.610	36.631	36.158	**33.264**
32.656	Mean	42.583	43.033	39.725	**39.074**
	St. Dev.	7.05E+00	4.30E+00	**2.30E+00**	6.01E+00
g20	Best	-	-	-	-
0.188446	Mean	-	-	-	-
	St. Dev.	-	-	-	-
g21	Best	-	-	800.275 (3)	**193.778**
193.778	Mean	-	-	878.722	**237.353**
	St. Dev.	-	-	10.64E+01	**35.29E+00**
g22	Best	-	-	-	-
382.902	Mean	-	-	-	-
	St. Dev.	-	-	-	-
g23	Best	-3.00E-02 (5)	-228.338 (20)	**-335.387** (20)	-98.033 (16)
-400.003	Mean	107.882	**-20.159**	159.312	134.154
	St. Dev.	14.03E+01	**13.23E+01**	25.47E+01	17.99E+01
g24	Best	-5.507	-5.507	**-5.508**	**-5.508**
-5.508	Mean	-5.507	-5.507	**-5.508**	**-5.508**
	St. Dev.	2.87E-04	1.87E-04	**9.03E-16**	**9.03E-16**

respect to the global best PSO with inertia weight, the local best PSO with constriction factor obtains results with better quality and consistency in twenty test problems (g01, g02, g04, g05, g06, g07, g08, g09, g10, g12, g13, g14, g15, g16, g17, g18, g19, g21, g23 and g24). With respect to the local best PSO with inertia weight, the local best PSO with constriction factor provides better quality and consistency results in sixteen problems (g01, g02, g05, g06, g07, g09, g10, g12, g13, g14, g15, g16, g17, g18, g19 and g21). Finally, with respect to the global best PSO with constriction factor, the local best PSO with constriction factor presents better quality and consistency results in seventeen problems (g01, g04, g05, g06, g07, g09, g10, g11, g12, g13, g14, g15, g16, g17, g18, g19 and g21).

Table 4 Comparison of results provided by two state-of-the-art PSO-based approaches and the two local best PSO variants. "(n)" means that in only "n" runs feasible solutions were found. Boldface remarks the best result per function. "-" means that no feasible solutions were found in any single run.

PSO VARIANTS AND STATE-OF-THE-ART ALGORITHMS						
Problem & best-known solution		local best ($w = 07$)	local best ($k = 0.729$)	Toscano & Coello [42]	Lu & Chen [25]	Cagnina et al. [2]
g01	Best	-14.999	**-15.000**	**-15.000**	**-15.000**	**-15.000**
-15.000	Mean	-12.100	-13.363	**-15.000**	-14.418	**-15.000**
g02	Best	-0.614785	-0.790982	**-0.803432**	-0.664	-0.801
-0.803619	Mean	-0.543933	-0.707470	**-0.790406**	-0.413	0.765
g03	Best	-0.045	-0.126	-1.004	**-1.005**	-1.000
-1.000	Mean	-1.00E-02	-1.70 E-02	**-1.003**	-1.002	-1.000
g04	Best	**-30665.539**	**-30665.539**	-30665.500	**-30665.659**	**-30665.659**
-30665.539	Mean	**-30665.539**	**-30665.539**	-30665.500	**-30665.539**	-30665.656
g05	Best	5126.646 (18)	5126.496	5126.640	**5126.484**	5126.497
5126.498	Mean	6057.259	**5140.060**	5461.081	5241.054	5327.956
g06	Best	-6958.704	**-6961.814**	-6961.810	-6961.813	-6961.825
-6961.814	Mean	-6941.207	**-6961.814**	-6961.810	-6961.813	-6859.075
g07	Best	41.747	24.444	24.351	**24.306**	24.400
24.306	Mean	59.077	25.188	25.355	**24.317**	31.485
g08	Best	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**
-0.095825	Mean	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	-0.095800
g09	Best	696.947	680.637	680.638	**680.630**	680.636
680.630	Mean	728.730	680.671	680.852	**680.630**	682.397
g10	Best	8947.646	7097.001	7057.900	**7049.248**	7052.852
7049.248	Mean	9247.134	7641.849	7560.047	**7049.271**	8533.699
g11	Best	0.750	**0.749**	**0.749**	**0.749**	**0.749**
0.749	Mean	0.799	**0.749**	0.750	**0.749**	0.750
g12	Best	-0.999	**-1.000**	**-1.000**	**-1.000**	**-1.000**
-1.000	Mean	-0.999	**-1.000**	**-1.000**	**-1.000**	**-1.000**
g13	Best	-	8.10E-02	0.068	**0.053**	0.054
0.053949	Mean	-	**0.45**	1.716	0.681	0.967

The local best PSO with inertia weight provides the "best" quality result in one problem (g23).

The global best PSO with constriction factor obtains more consistent results in one problem (g23). Problems g20 and g22 could not be solved by any PSO variant; these problems have several equality constraints and are the most difficult to solve [23].

Comparing the global best variants (third and fourth columns in Tables 2 and 3) with respect to those local best PSOs (fifth and sixth columns in Tables 2 and 3) the results suggest that the last ones perform better in this sample of constrained search spaces i.e. the global best variants have problems finding the feasible region in some problems where the local best variants indeed find it (g05, g14, g15, g18 and g21). Finally, when comparing inertia weight variants (third and fifth columns in Tables 2 and 3) with respect to constriction factor variants (fourth and sixth columns in Tables 2 and 3), there is no clear superiority. However, both variants (inertia weight and constriction factor) perform better coupled with local best PSO (ring social network).

The overall results from this first experiment suggest that the local best PSO with constriction factor is the most competitive approach (based on quality and consistency) in this set of test CNOPs. Besides, some important information regarding the behavior of PSO in constrained search spaces was obtained and discussed.

As an interesting comparison, in Table 4 the two most competitive PSO variants from this experiment (local best PSO with constriction factor and inertia weight) are compared with three state-of-the-art PSO-based approaches. The results show that these two variants are competitive in some test problems (g04, g08, g11 and g12). However, they are far from providing a performance like those presented by the state-of-the-art algorithms. Therefore, the most competitive PSO variant (local best PSO with constriction factor) will be improved in the next Section of this chapter.

7 Simple Modifications to the Original PSO

Besides the results presented, the experiment in the previous Section provided valuable information regarding two issues related to PSO for constrained search spaces. (1) The local best PSO with constriction factor presents a lower tendency to converge prematurely when solving CNOPs and (2) all PSO variants compared have problems dealing with test functions with equality constraints. Therefore, two simple modifications are proposed to this most competitive variant to improve its performance. This new version will be called Improved PSO (IPSO).

7.1 Dynamic Adaptation

Based on the velocity update formula (Eq. 10) two parameters were detected as the most influential in this calculation: (1) k, which affects the entire value of the velocity and (2) c_2, which has more influence in the calculation because, most of the time, the pbest value is the same as the current position of the particle i.e. this term in Eq. 10 may be eliminated, whereas the gbest value is different from the position of the particle in all the search process (except for the leader). Moreover, PSO has a tendency to prematurely converge [9]. Then, a dynamic (deterministic) adaptation mechanism [8] for these two parameters k and c_2 is proposed to start with low velocity values for some particles and to increase these values during the search process as follows: A dynamic value for k and c_2, based on the generation number will be used for a (also variable) percentage of particles in the swarm. The remaining particles in the swarm will use the fixed values for these two parameters. It is important to note that, at each generation, the particles which will use the dynamic values will be different e.g. a given particle may use the fixed values at generation "t" and the dynamic values at generation "$t+1$". The aim is to let, at each generation, some particles (those which use the dynamic values) to move at a slower velocity with respect to the remaining ones. The expected behavior is to slow down convergence and, as a result, better performance i.e. better quality and consistent results.

Based on the strong tendency of PSO to converge fast, a dynamic variation was chosen in such a way that in the first part of the process (half of total generations) k and c_2 values would remain low, and in the second half of the process they would increase faster. Then, the following function was chosen: $f(y) = y^4$, where $y = GEN/Gmax$. This function is presented in Figure 3, where it is noted that very low values are generated before 0.5 in the x-axis i.e. in the first half of the search process. This means that the values for the adapted parameters will be also low. However, in the second part of the search (0.5 to 1.0) the parameter values increase faster to reach their original values.

The expressions to update both parameter values at each generation "$t+1$" are defined as follows: $k^{t+1} = k \cdot f(y)$ and $c_2^{t+1} = c_2 \cdot f(y)$, where k and c_2 are the static values for these parameters. The initial values are small values close to zero e.g.

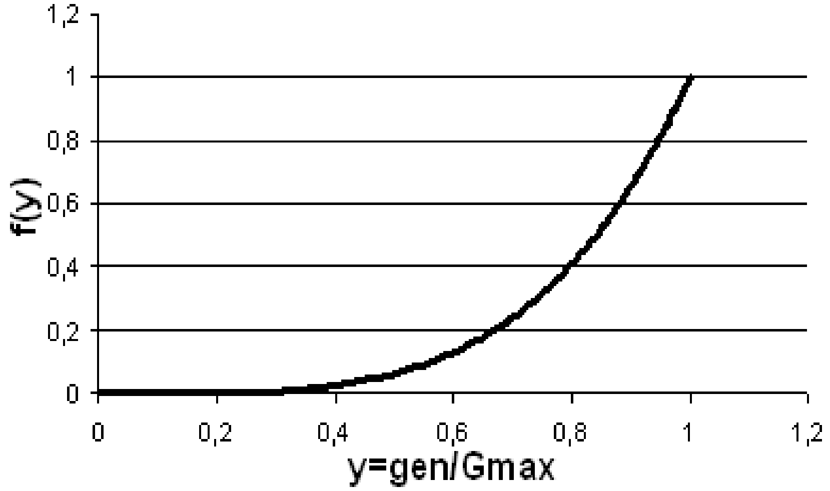


Fig. 3 Function used to dynamically adapt k and c_2 parameters. In the first half of the search low values are generated, while in the second half the values increase very fast.

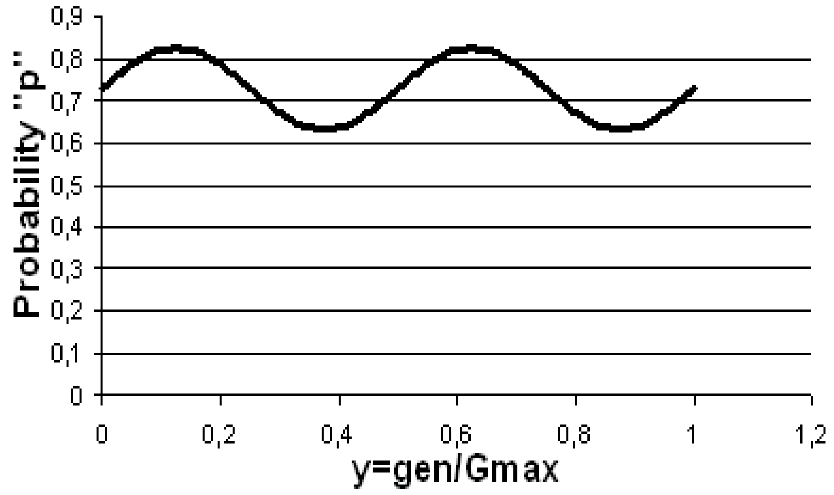


Fig. 4 Oscillatory percentage of particles that will use the fixed values for k and c_2. The remaining particles will use the dynamic values.

4E-13, and the values at the last generation will be exactly the fixed values ($k = 0.729$ and $c_2 = 2.5$).

As it was mentioned before, the number of particles which will use these dynamic values is also dynamic. In this case, based on observations considering the best performance, an oscillatory percentage of particles was the most suited. Therefore, a probability value is computed as to decide if a given particle will use either the static or the dynamic values: $p = k + \frac{(\sin(4\pi y))}{10.3}$, where k is the fixed value for this parameter ($k = 0.729$) and $y = GEN/Gmax$. The constant value 10.3 defines the maximum and minimum values p can take ($p \in [0.62, 0.82]$). A higher constant value decreases this range and a lower value increases it. The value suggested (10.3) worked well in all the experiments performed. The percentage of particles which will use the fixed parameters is modified as shown in Figure 4.

The main advantage of the dynamic mechanism proposed in this chapter over the addition of extra parameters (e.g. mutation operators), the combination of PSO variants or the modification of the original problem, all of them to keep PSO from converging prematurely (as shown on previous approaches in Section 5), is that the user does not need to fine-tune additional parameter values i.e. this work is done in IPSO by the own PSO. Even though the dynamic approach seems to be more complicated with respect to the addition of a parameter, this additional mechanism maintains the simplicity of PSO from the user's point of view.

7.2 Modified Constraint-Handling

The third feasibility rule proposed by Deb [5] selects, from two infeasible solutions, the one with the lowest sum of normalized constraint violation:

$$s = \sum_{i=1}^{m} \max(0, g(\mathbf{x})) + \sum_{j=1}^{p} \max(0, (|h(\mathbf{x})| - \varepsilon)) \tag{11}$$

As can be noted from Equation 11, the information of the violation of inequality and equality constraints are merged into one single value s. Besides, in the specialized literature there is empirical evidence that equality constraints are more difficult to satisfy than inequality constraints [27, 40, 31].

Based on the way s is computed, some undesired situations may occur when two infeasible solutions a and b are compared e.g. the s value from one of them (called s_a) can be lower than the the other one (s_b), but the violation sum for equality constraints can be higher in s_a. Therefore, it may be more convenient to handle these sums separately as to provide the search with more detailed information in the selection process:

$$s_1 = \sum_{i=1}^{m} \max(0, g(\mathbf{x})) \tag{12}$$

$$s_2 = \sum_{j=1}^{p} \max(0, (|h(\mathbf{x})| - \varepsilon)) \tag{13}$$

After that, a dominance criterion (as defined in Section 2) is used to select the best solution by using the vector [*s1*,*s2*] for both solutions to be compared. The solution which dominates the other is chosen. If both solutions are nondominated between them, the age of the solution is considered i.e for the leader selection and for the pbest update there will be always a current solution and a new solution to be compared, if both solutions do not dominate each other, the older solution is kept. In this way, the solutions will be chosen/updated only if one amount of violation is decreased without increasing the other or if both amounts are decreased. The expected effect is detailed in Figure 5, where the current solution (white circle) must be replaced by a new one. Three candidate solutions are available: $P1$, $P2$

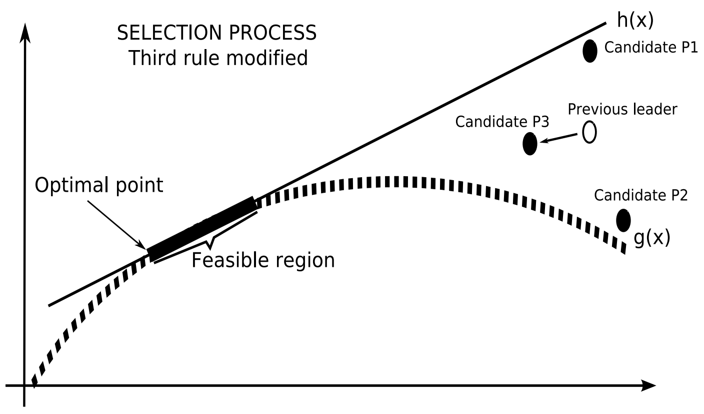


Fig. 5 Expected behavior on the modified constraint-handling mechanism

and *P*3 (all black circles). *P*1 is discarded because it decreases the violation of the equality constraint but also increases the violation of the inequality constraint. *P*2 is also discarded because it decreases the violation amount of the inequality constraint but also increases the violation of the equality constraint. *P*3 is chosen because both violation amounts are decreased i.e. *P*3 dominates *P*1 and *P*2.

IPSO is then based on the local best PSO with constriction factor as a search engine, coupled with the dynamic adaptation mechanism for k and c_2 parameters and the modification to the third rule of the constraint-handling technique. No additional operators, parameters, local search, problem re-definition, PSO variants mixtures nor original velocity or flight formulas modifications were considered on IPSO, whose details are found in Figure 6 (modifications are remarked) and its behavior and performance is analyzed in the next Section. The same mechanism explained in Section 6 to generate values within the allowed boundaries for the variables of the problem is also used in IPSO.

```
Begin
    GEN = 0
    Generate a swarm of random solutions ($\mathbf{x}_i$) $i = 1,2,...,SWARM\_SIZE$.
    Divide the swarm in $n$ neighborhoods.
    Assign equal number of particles to each neighborhood.
    Initialize for each particle, $\mathbf{x}_{pbest_i} = \mathbf{x}_i$, and $\mathbf{v}_i(t) = 0$.
    Evaluate the fitness of the particle in each neighborhood.
    Do
        Select the leader ($\mathbf{x}_{lBest_i}$) of each neighborhood.
            by using the modified feasibility rules
        For each particle, update its velocity with (10).
            by using the corresponding leader of each neighborhood $\mathbf{x}_{lBest_i}$
            Depending of the $p$ value use the fixed values for k and $c_2$
            Otherwise use the dynamic values for these parameters
        For each particle, update its position with (8).
        Evaluate the fitness of the new position for each particle.
        Update the $\mathbf{x}_{pbest_i}$ (memory) value for each particle.
            by using the modified feasibility rules
        GEN=GEN+1
    Until GEN = Gmax
End
```

Fig. 6 Improved PSO pseudocode. Modifications are underlined

8 Experiments and Results

In this Section, four aspects of IPSO are analyzed: (1) The quality and consistency of its final results, (2) its online behavior by using two performance measures found in the specialized literature [27], (3) its convergence behavior by analyzing convergence graphs and (4) its performance compared to those provided by state-of-the-art

PSO-based approaches to solve CNOPs. The same 24 test problems used in the preliminary experiment are considered in this Section.

8.1 Quality and Consistency Analysis

A similar experimental design to that used in the comparison of PSO variants is considered here. IPSO is compared against two PSO original variants: global best and local best PSO, both with constriction factor, as to analyze the convenience of the two modifications proposed. The parameter values are the same used in the previous experiment. 30 independent runs were performed and the statistical results are summarized in Tables 5 and 6 for the 24 test problems.

Like in the previous experiment, the nonparametric statistical tests were applied to the samples summarized in Tables 5 and 6. For the following problems, no significant differences were found among the results provided by the three algorithms: g04, g08 and g24. Besides, no significant difference in performance is found in problems g05, g13 and g17 when the local best PSO with constriction factor and IPSO are compared, and in problem g23 when the global and local best PSOs, both with constriction factor, are also compared. In all the remaining comparisons, the differences are significant. IPSO provides better quality and more consistent results in five problems (g03, g07, g10, g14 and g21), better quality results in two problems (g02 and g18) and it also obtains more consistent results in six problems (g01, g06, g09, g11, g19 and g23), all with respect to the local best PSO with constriction factor, which is the variant in which IPSO is based. The original PSO with constriction factor presents the best performance in problems g15. Also, it is more consistent in problems g02 and g18 and it finds the "best" quality result in problems g09 and g23. The global best PSO with constriction factor is not better in any single problem.

The overall analysis of this experiment indicates that the two simple modifications made to a competitive PSO variant lead to an improvement in the quality and mostly in the consistency of the final results e.g. in problems with a combination of equality and inequality constraints such as g21 and g23, IPSO provided a very consistent and good performance. The exception was g05, where, despite the better results in the samples for IPSO, the statistical test considered the differences as not significant.

8.2 On-Line Behavior Analysis

Two performance measures will be used to compare the two PSO variants and IPSO to know: (1) how fast the feasible region is reached and (2) the ability of each PSO to move inside the feasible region (difficult for most BIAs as analyzed in [27]).

1. Evals: Proposed by Lampinen [19]. It counts the number of evaluations (objective function and constraints) required to generate the first feasible solution. Then, a lower value is preferred because it indicates a faster approach to the feasible region of the search space.

Table 5 Statistical results of 30 independent runs on the first 12 test problems for IPSO and the two PSO variants with constriction factor compared. "(n)" means that in only "n" runs feasible solutions were found. Boldface remarks the best result per function.

STATISTICS FROM 30 INDEPENDENT RUNS				
Problem & best-known solution		global best (k)	local best (k)	IPSO
g01	Best	**-15.000**	**-15.000**	**-15.000**
-15.000	Mean	-10.715	-13.815	**-15.000**
	St. Dev.	2.54E+00	1.58E+00	**0.00E+00**
g02	Best	-0.612932	-0.777758	**-0.802629**
-0.803619	Mean	-0.549707	**-0.717471**	-0.713879
	St. Dev.	**2.39E-02**	4.32 E-02	4.62 E-02
g03	Best	-0.157	-0.426	**-0.641**
-1.000	Mean	-0.020	-0.037	**-0.154**
	St. Dev.	**3.00E-02**	9.20E-02	1.70 E-01
g04	Best	**-30665.539**	**-30665.539**	**-30665.539**
-30665.539	Mean	**-30665.539**	**-30665.539**	**-30665.539**
	St. Dev.	**7.40E-12**	**7.40E-12**	**7.40E-12**
g05	Best	6083.449 (12)	5126.502	**5126.498**
5126.498	Mean	6108.013	5135.700	**5135.521**
	St. Dev.	9.78E+00	**9.63E+00**	1.23E+01
g06	Best	-6957.915	**-6961.814**	**-6961.814**
-6961.814	Mean	-6943.444	-6961.813	**-6961.814**
	St. Dev.	9.45E+00	4.66E-04	**2.81E-05**
g07	Best	45.633	24.463	**24.366**
24.306	Mean	60.682	25.045	**24.691**
	St. Dev.	7.50E+00	5.10E-01	**2.20E-01**
g08	Best	**-0.095825**	**-0.095825**	**-0.095825**
-0.095825	Mean	**-0.095825**	**-0.095825**	**-0.095825**
	St. Dev.	**4.23E-17**	**4.23E-17**	**4.23E-17**
g09	Best	705.362	**680.635**	680.638
680.630	Mean	736.532	680.675	**680.674**
	St. Dev.	1.58E+01	**2.90E-02**	3.00E-02
g10	Best	8673.098	7124.709	**7053.963**
7049.248	Mean	9140.877	7611.759	**7306.466**
	St. Dev.	2.36E+02	3.22E+02	**2.22E+02**
g11	Best	**0.749**	**0.749**	**0.749**
0.749	Mean	0.794	**0.753**	**0.753**
	St. Dev.	5.90E-02	1.00E-02	**6.53E-03**
g12	Best	-0.999	**-1.000**	**-1.000**
-1.000	Mean	-0.999	**-1.000**	**-1.000**
	St. Dev.	2.77E-05	**0.00E+00**	**0.00E+00**

Table 6 Statistical results of 30 independent runs on the last 12 test problems for IPSO and the two PSO variants with constriction factor compared. "(n)" means that in only "n" runs feasible solutions were found. Boldface remarks the best result per function. "-" means that no feasible solutions were found in any single run.

STATISTICS FROM 30 INDEPENDENT RUNS				
Problem & best-known solution		global best (k)	local best (k)	IPSO
g13	Best	-	0.127872	**0.066845**
0.053949	Mean	-	0.520039	**0.430408**
	St. Dev.	-	**2.30E+00**	**2.30E+00**
g14	Best	-47.394 (7)	-45.062 (4)	**-47.449**
-47.764	Mean	-38.619	-43.427	**-44.572**
	St. Dev.	4.95E+00	1.59E+00	**1.58E+00**
g15	Best	967.519 (5)	**961.715**	**961.715**
961.715	Mean	969.437	**961.963**	962.242
	St. Dev.	2.62E+00	**3.20E-01**	6.20E-01
g16	Best	-1.904	**-1.905**	**-1.905**
-1.905	Mean	-1.904	**-1.905**	**-1.905**
	St. Dev.	1.46E-04	5.28E-11	**2.42E-12**
g17	Best	-	**8853.721**	8863.293
8876.981	Mean	-	8917.155	**8911.738**
	St. Dev.	-	3.17E+01	**2.73E+01**
g18	Best	-0.295425 (5)	-0.865989	**-0.865994**
-0.865735	Mean	-0.191064	**-0.864966**	-0.862842
	St. Dev.	1.20E-01	**1.38E-03**	4.41E-03
g19	Best	37.568	**33.939**	33.967
32.656	Mean	40.250	38.789	**37.927**
	St. Dev.	3.97E+00	3.97E+00	**3.20E+00**
g20	Best	-	-	-
0.188446	Mean	-	-	-
	St. Dev.	-	-	-
g21	Best	666.081 (7)	193.768	**193.758**
193.778	Mean	896.690	237.604	**217.356**
	St. Dev.	1.21E+02	3.60E+01	**2.65E+01**
g22	Best	-	-	-
382.902	Mean	-	-	-
	St. Dev.	-	-	-
g23	Best	-98.033 (16)	**-264.445**	-250.707
-400.003	Mean	134.154	70.930	**-99.598**
	St. Dev.	1.79E+02	2.58E+02	**1.20E+02**
g24	Best	**-5.508**	**-5.508**	**-5.508**
-5.508	Mean	**-5.508**	**-5.508**	**-5.508**
	St. Dev.	**9.03E-16**	**9.03E-16**	**9.03E-16**

2. Progress Ratio: Proposed by Mezura-Montes & Coello [27], it is a modification of Bäck's original proposal for unconstrained optimization [1]. It measures the improvement inside the feasible region by using the objective function values of the first feasible solution and the best feasible solution reached at the end of the process. The formula is the following: $\Pr = \left| \ln \sqrt{\frac{f_{\min}(G_{ff})}{f_{\min}(T)}} \right|$, where $f_{\min}\left(G_{ff}\right)$ is the objective function value of the first feasible solution found and $f_{\min}(T)$ is the objective function value of the best feasible solution found in all the search so far. A higher value means a better improvement inside the feasible region.

30 independent runs for each PSO variant, for each test problem, for each performance measure were computed. Statistical results are calculated and summarized in Tables 7 and 8 for the Evals performance measure and in Tables 9 and 10 for the Progress Ratio. The parameter values for the three PSOs are the same utilized in the previous experiment.

Regarding the Evals measure, some test problems are not considered in the discussion because feasible solutions were found in the initial swarm generated randomly. This was due to the size of the feasible region with respect to the whole search space (Table 1, fourth column). These problems are g02, g04, g08, g09, g12, g19 and g24. Problems g20 and g22 are also excluded because none of the algorithms could find a single feasible solution.The nonparametric tests applied to the samples of the remaining problems confirmed the significance of differences for all of them, with the exception of problem g11 for the three algorithms and in the comparison between the global best PSO and IPSO in problem g23.

The global best PSO with constriction factor is the fastest and also the most consistent variant to reach the feasible region in four problems (g01, g06, g07, g10). It is also the fastest (but not the most consistent) in problem g03 and it is the most consistent in problem g16. However, it failed to find a single feasible solution in problems g13 and g17 and it is able to find feasible solutions in just some runs (out of 30) in problems g05 (12/30), g14 (17/30), g15 (5/30), g18 (5/30), g21 (7/30) and g23 (16/30). The local best PSO with constriction factor provides the fastest and more consistent approach to the feasible region in problem g23 and it is the fastest (but not the most consistent) in problems g16, g18 and g21. Finally, IPSO presents the fastest and more consistent approach to the feasible region in four problems (g05, g13, g15 and g17) and it is the most consistent in four problems (g03, g14, g18 and g21).

The overall results for the Evals performance measure show that the global best PSO with constriction factor, based on its fast convergence, presents a very irregular approach to the feasible region, being the fastest in some problems, but failing to find feasible solutions in others. The local best PSO with constriction factor is not very competitive at all, whereas IPSO provides a very consistent performance, while not the fastest. However, IPSO has a good performance in problems g05 and g21, both with a combination of equality and inequality constraints. The exception in this regard is problem g23.

Table 7 Statistical results for the EVALS performance measure based on 30 independent runs in the first 12 test problems for IPSO and the two PSO variants with constriction factor. "(n)" means that in only "n" runs feasible solutions were found. Boldface remarks the best result per function.

EVALS				
Problem		global best (k)	local best (k)	IPSO
g01	Best	**162**	246	252
	Mean	**306**	368	419
	St. Dev.	6.40E+01	**5.41E+01**	7.77E+01
g02	Best	**0**	**0**	**0**
	Mean	**0**	**0**	**0**
	St. Dev.	**0.00E+00**	**0.00E+00**	**0.00E+00**
g03	Best	**189**	366	457
	Mean	3568	2118	**1891**
	St. Dev.	3.50E+03	1.35E+03	**9.82E+02**
g04	Best	**0**	**0**	**0**
	Mean	4	**2**	**2**
	St. Dev.	5.14E+00	3.11E+00	**2.92E+00**
g05	Best	33459 (12)	16845	**13087**
	Mean	86809	23776	**17037**
	St. Dev.	4.53E+04	3.68E+03	**2.21E+03**
g06	Best	**180**	256	254
	Mean	**440**	513	562
	St. Dev.	2.73E+02	2.58E+02	**1.86E+02**
g07	Best	**178**	484	812
	Mean	**873**	1164	1316
	St. Dev.	7.11E+02	4.01E+02	**2.52E+02**
g08	Best	4	**0**	3
	Mean	**56**	78	76
	St. Dev.	**3.97E+01**	5.21E+01	5.79E+01
g09	Best	**5**	8	17
	Mean	**88**	94	107
	St. Dev.	5.01E+01	**4.81E+01**	6.91E+01
g10	Best	**242**	579	522
	Mean	**861**	972	1202
	St. Dev.	3.29E+02	**2.69E+02**	4.56E+02
g11	Best	**85**	249	364
	Mean	1662	1152	**1009**
	St. Dev.	2.10E+03	8.32E+02	**5.72E+02**
g12	Best	2	**0**	1
	Mean	19	**15**	25
	St. Dev.	**1.60E+01**	1.77E+01	2.17E+01

Table 8 Statistical results for the EVALS performance measure based on 30 independent runs in the last 12 test problems for IPSO and the two PSO variants with constriction factor. "(n)" means that in only "n" runs feasible solutions were found. Boldface remarks the best result per function. "-" means that no feasible solutions were found in any single run.

EVALS				
Problem		global best (k)	local best (k)	IPSO
g13	Best	-	11497	**8402**
	Mean	-	1.75E+04	**1.28E+04**
	St. Dev.	-	2.82E+03	**1.82E+03**
g14	Best	**7820** (17)	9481 (4)	8353
	Mean	33686	13485	**12564**
	St. Dev.	2.47E+03	3.13E+03	**3.07E+03**
g15	Best	24712 (5)	7299	**5228**
	Mean	68444	11805	**8911**
	St. Dev.	4.16E+04	2.43E+03	**1.72E+03**
g16	Best	164	**32**	106
	Mean	**309**	442	493
	St. Dev.	**1.28E+02**	2.26E+02	2.23E+02
g17	Best	-	21971	**16489**
	Mean	-	29458	**22166**
	St. Dev.	-	5.07E+03	**2.73E+03**
g18	Best	110395 (5)	**2593**	2614
	Mean	125303	5211	**4479**
	St. Dev.	2.31E+04	1.04E+03	**8.87E+02**
g19	Best	**0**	**0**	**0**
	Mean	**2**	**2**	**2**
	St. Dev.	2.15E+00	**2.13E+00**	2.90E+00
g20	Best	-	-	-
	Mean	-	-	-
	St. Dev.	-	-	-
g21	Best	43574 (7)	**11617**	13403
	Mean	82594	27978	**19652**
	St. Dev.	3.45E+04	7.11E+03	**3.51E+03**
g22	Best	-	-	-
	Mean	-	-	-
	St. Dev.	-	-	-
g23	Best	8499 (16)	**2081**	18304
	Mean	32661	**17797**	28764
	St. Dev.	2.58E+04	1.35E+04	**5.34E+03**
g24	Best	**0**	**0**	**0**
	Mean	**1**	**1**	2
	St. Dev.	1.95E+00	**1.88E+00**	2.36 E+00

Table 9 Statistical results for the PROGRESS RATIO performance measure based on 30 independent runs in the first 12 test problems for IPSO and the two PSO variants with constriction factor. "(n)" means that in only "n" runs feasible solutions were found. Boldface remarks the best result per function.

PROGRESS RATIO				
Problem		global best (k)	local best (k)	IPSO
g01	Best	0.302	0.346	**0.368**
	Mean	0.196	0.266	**0.295**
	St. Dev.	5.90E-02	5.00E-02	**3.80E-02**
g02	Best	**1.388**	1.373	1.218
	Mean	0.884	**1.015**	1.013
	St. Dev.	1.20E-01	1.00E-01	**9.00E-02**
g03	Best	**0.346**	**0.346**	0.334
	Mean	0.037	0.026	**0.067**
	St. Dev.	**6.50E-02**	7.25E-01	8.10E-02
g04	Best	0.110	0.120	**0.124**
	Mean	0.070	**0.080**	0.071
	St. Dev.	**2.30E-02**	**2.30E-02**	2.50-02
g05	Best	4.273E-07 (12)	**0.087**	**0.087**
	Mean	1.250E-07	**0.056**	0.036
	St. Dev.	**1.64E-07**	3.70E-02	3.20E-02
g06	Best	0.799	**0.807**	0.772
	Mean	0.306	**0.348**	0.296
	St. Dev.	2.00E-01	**1.80E-01**	1.90E-01
g07	Best	2.117	**2.504**	2.499
	Mean	1.656	1.919	**1.963**
	St. Dev.	3.60E-01	3.60E-01	**3.50E-01**
g08	Best	0.494	0.451	**0.556**
	Mean	0.317	0.304	**0.356**
	St. Dev.	9.10E-02	9.20E-02	**7.20E-02**
g09	Best	4.685	4.394	**4.768**
	Mean	**2.622**	2.209	2.510
	St. Dev.	1.29E+00	**1.12**E+00	1.24E+00
g10	Best	0.598	0.665	**0.678**
	Mean	0.360	**0.482**	0.468
	St. Dev.	1.30E-01	**1.00E-01**	1.10E-01
g11	Best	**0.143**	**0.143**	**0.143**
	Mean	0.088	**0.113**	0.101
	St. Dev.	**4.70E-02**	4.90E-02	5.20E-02
g12	Best	**0.342**	0.281	0.285
	Mean	**0.136**	0.119	0.096
	St. Dev.	7.20E-02	**6.70E-02**	7.40E-02

Table 10 Statistical results for the PROGRESS RATIO performance measure based on 30 independent runs in the last 12 test problems for IPSO and the two PSO variants with constriction factor. "(n)" means that in only "n" runs feasible solutions were found. Boldface remarks the best result per function. "-" means that no feasible solutions were found in any single run.

PROGRESS RATIO

Problem		global best (k)	local best (k)	IPSO
g13	Best	-	1.165	**2.327**
	Mean	-	0.410	**0.549**
	St. Dev.	-	**3.40E-01**	5.70E-01
g14	Best	7.564E-04 (7)	0.077 (4)	**0.167**
	Mean	3.253E-04	0.028	**0.061**
	St. Dev.	**2.78E-04**	3.50-02	3.90E-02
g15	Best	1.520E-06 (5)	**5.460E-03**	5.419E-03
	Mean	6.866E-07	**2.805E-03**	2.571E-03
	St. Dev.	**5.79E-07**	1.82E-03	1.67E-03
g16	Best	0.379	**0.509**	0.412
	Mean	0.224	0.217	**0.246**
	St. Dev.	**7.20E-02**	9.20E-02	8.90E-02
g17	Best	-	**0.023**	0.022
	Mean	-	**8.342E-03**	6.015E-03
	St. Dev.	-	7.17E-03	**6.50E-03**
g18	Best	0.660 (5)	**1.540**	1.297
	Mean	0.348	**0.883**	0.690
	St. Dev.	**2.70E-01**	3.30E-01	2.90E+00
g19	Best	3.463	3.470	**3.580**
	Mean	2.975	**3.062**	3.050
	St. Dev.	2.70E-01	**1.80E-01**	3.2E-01
g20	Best	-	-	-
	Mean	-	-	-
	St. Dev.	-	-	-
g21	Best	7.252E-03 (7)	**0.819**	**0.819**
	Mean	1.036E-03	0.628	**0.646**
	St. Dev.	**2.74E-03**	1.60E-01	1.40E-01
g22	Best	-	-	-
	Mean	-	-	-
	St. Dev.	-	-	-
g23	Best	2.697E-03 (16)	0.691	**0.847**
	Mean	3.835E-04	0.139	**0.240**
	St. Dev.	**6.19E-04**	1.90E-01	2.10E-01
g24	Best	**1.498**	1.211	1.062
	Mean	**0.486**	0.443	0.481
	St. Dev.	3.50E-01	2.60E-01	**2.40E-01**

The behavior, regarding Evals, presented by IPSO is somehow expected, because the approach to the feasible region might be slower because some particles will use lower parameter values in the velocity update, mostly in the first half of the search.

The nonparametric tests applied to the samples of results for the Progress Ratio showed no significant differences for the three algorithms in problems g04, g08, g12, g16 and g24 and in the comparison of the local best PSO with constriction factor and IPSO in problems g13 and g17. Problems g20 and g22 were discarded because no feasible solutions were found. In the remaining problems, the differences are significant.

Despite being very fast in reaching the feasible region, the global best PSO obtains the "best" improvement within the feasible region only in problem g02 and it is the most consistent in problem g09. The local best PSO obtains the "best" quality and most consistent results in problems g05, g06, g15, g17 and g18. Also, it presents the best result in problem g07 and the most consistent improvement in the feasible region in problems g10, and g19. IPSO is the best approach, based on quality and consistency in problems g01, g13, g14, g21 and g23. Besides, it presents the "best" quality results in problems g10 and g19 and it is the most consistent approach in problem g07.

As a conclusion for the Progress Ratio measure, IPSO does not significantly improve PSO's ability of moving inside the feasible region. However, IPSO is very competitive in problems with a combination of equality and inequality constraints (g21 and g23), but it is surpassed by the local best PSO with constriction factor in problem g05 as well as in other problems. A last finding to remark is the poor results obtained for the global best PSO with constriction factor. It seems that its fast approach to the feasible region leads to an inability to improve solutions inside it.

8.3 Convergence Behavior

The convergence behavior of the two original PSO variants and IPSO is graphically compared by plotting the run located in the mean value from a set of 30 independent runs. Problems where the behavior is very similar are omitted. The graphs are grouped in Figure 7. Based on the behavior found in those graphs, IPSO is able to converge faster than the two PSO variants in problems g01, g02 and g10, even local best PSO with constriction factor achieves similar results but in more generations. Global best PSO with constriction factor is trapped in a local optimum solution. In problem g03, the local best PSO provides the best convergence while IPSO and the global best PSO with constriction factor are trapped in local optima solutions. Finally, IPSO clearly shows a better convergence in problems g14, g17, g21 and g23. It is worth reminding that problems g21 and g23 have a combination of equality and inequality constraints. Therefore, the graphs suggest that the modified constraint-handling mechanism helps PSO in this kind of problems.

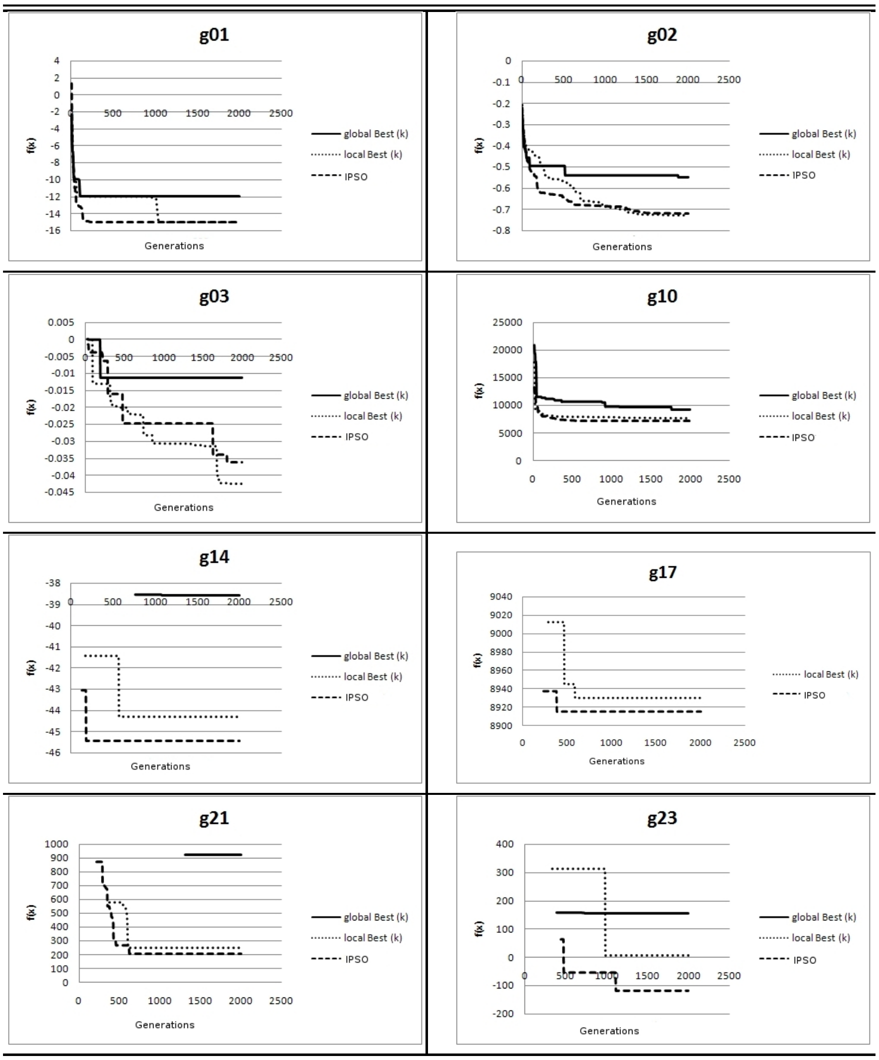


Fig. 7 Representative convergence graphs for the two compared PSO variants and IPSO

8.4 Comparison with State-Of-The-Art PSO-Based Approaches

As a final comparison, IPSO's final results are compared with respect to those reported by four state-of-the-art PSO-based approaches. The approaches are the PSO algorithms proposed by Toscano and Coello [42], Li et al. [20], Lu and Chen [25] and Cagnina et al. [2]. These approaches are selected because they were tested against the same set of test problems. Statistical results (best, mean and worst values) are shown in Table 11. Test problems g14 to g24 are omitted because no results are reported by the compared approaches.

Table 11 Comparison of results with respect to state-of-the-art PSO-based approaches. (|) indicates that the results for this function were not available.

Comparison with state-of-the-art PSO-based approaches.						
Problem & best-known solution		Toscano & Coello [42]	Li, Tian & Kong [20]	Lu & Chen [25]	Cagnina et al. [2]	IPSO
g01	Best	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**
-15.000	Mean	**-15.000**	**-15.000**	-14.418	**-15.000**	**-15.000**
	Worst	**-15.000**	**-15.000**	-12.453	-134.219	**-15.000**
g02	Best	**-0.803432**	|	-0.664	-0.801	-0.802629
-0.803619	Mean	**-0.790406**	|	-0.413	0.765	-0.713879
	Worst	**-0.750393**	|	-0.259	0.091	-0.600415
g03	Best	-1.004	|	-1.005	**-1.000**	-0.641
-1.000	Mean	-1.003	|	-1.002	**-1.000**	-0.154
	Worst	-1.002	|	-0.934	**-1.000**	-3.747E-03
g04	Best	-30665.500	-30665.600	**-30665.539**	**-30665.659**	**-30665.539**
-30665.539	Mean	-30665.500	-30665.594	**-30665.539**	-30665.656	**-30665.539**
	Worst	-30665.500	-30665.500	**-30665.539**	-25555.626	**-30665.539**
g05	Best	5126.640	5126.495	**5126.484**	5126.497	5126.498
5126.498	Mean	5461.081	**5129.298**	5241.054	5327.956	5135.521
	Worst	6104.750	5178.696	5708.225	2300.5443	**5169.191**
g06	Best	-6961.810	**-6961.837**	-6961.813	-6961.825	-6961.814
-6961.814	Mean	-6961.810	**-6961.814**	-6961.813	-6859.075	**-6961.814**
	Worst	-6961.810	-6961.644	-6961.813	64827.544	**-6961.814**
g07	Best	24.351	|	**24.306**	24.400	24.366
24.306	Mean	25.355	|	**24.317**	31.485	24.691
	Worst	27.316	|	**24.385**	4063.525	25.15
g08	Best	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**
-0.095825	Mean	**-0.095825**	**-0.095825**	**-0.095825**	-0.095800	**-0.095825**
	Worst	**-0.095825**	**-0.095825**	**-0.095825**	-0.000600	**-0.095825**
g09	Best	680.638	**680.630**	**680.630**	680.636	680.638
680.630	Mean	680.852	680.654	**680.630**	682.397	680.674
	Worst	681.553	680.908	**680.630**	18484.759	680.782
g10	Best	7057.900	|	**7049.248**	7052.852	7053.963
7049.248	Mean	7560.047	|	**7049.271**	8533.699	7306.466
	Worst	8104.310	|	**7049.596**	13123.465	7825.478
g11	Best	**0.749**	**0.749**	**0.749**	**0.749**	**0.749**
0.749	Mean	0.750	**0.749**	**0.749**	0.750	0.753
	Worst	0.752	**0.749**	**0.749**	0.446	0.776
g12	Best	**-1.000**	|	**-1.000**	**-1.000**	**-1.000**
-1.000	Mean	**-1.000**	|	**-1.000**	**-1.000**	**-1.000**
	Worst	**-1.000**	|	**-1.000**	9386	**-1.000**
g13	Best	0.068	|	**0.053**	0.054	0.066
0.053949	Mean	1.716	|	0.681	0.967	**0.430**
	Worst	13.669	|	2.042	1.413	**0.948**

The statistical results show that IPSO provides the most consistent results in problems g05 (with a combination of equality and inequality constraints), g06 and g13. IPSO also has a similar performance with respect to the PSOs compared in problems g01, g04, g08, g11 and g12. Moreover, IPSO obtains the second best performance in problems g02, g07, g09 and g10. In problem g03 IPSO is not competitive at all. Regarding the computational cost of the compared approaches, Toscano and Coello [42] and Cagnina et al. [2] use 340,000 evaluations, Li et al. [20] do not report the number of evaluations required and Lu and Chen [25] report 50,000 evaluations. However, Lu and Chens' approach requires the definition of an extra parameter called ω and the original problem is also modified. IPSO requires 160,000 evaluations and does not add any extra operator or complex mechanism to the original PSO, keeping its simplicity.

9 Conclusions and Future Work

This chapter presented a novel PSO-based approach to solve CNOPs. Unlike traditional design steps to generate an algorithm to deal with constrained search spaces, which in fact may produce a more complex technique, in this work a preliminary analysis of the behavior of the most known PSO variants was performed as to get an adequate search engine. Furthermore, empirical evidence about the convenience of using PSO local best variants in constrained search spaces was found (constriction factor was better than inertia weight). From this first experiment, the local best PSO with constriction factor was the most competitive variant and two simple modifications were added to it: (1) a dynamic adaptation mechanism to control k and c_2 parameters, these to be used for a dynamically adapted percentage of particles in the swarm and (2) the use of a dominance criterion to compare infeasible solutions in such a way that new solutions are accepted only if both, the sums of inequality and equality constraint violations (handled separately) are decreased. This Improved PSO (IPSO) was compared against original PSO variants based on their final results and also based on their on-line behavior. IPSO's final results were significantly improved with respect to the original variants. On the other hand, IPSO was not the fastest to reach the feasible region and it did not improve considerably the ability to move inside the feasible region. In other words, the way the original PSO works in constrained search spaces was modified in such a way that a slower approach to the feasible region allowed IPSO to enter it from a more promising area. However, this issue requires a more in-depth analysis. The convergence behavior shown by IPSO suggest that their mechanisms promote a better exploration of the search space to avoid local optimum solutions in most of the test problems. Finally IPSO, which does not add further complexity to PSO, provided competitive and even better results, with a moderate computational cost, when compared with four state-of-the-art PSO-based approaches. A final conclusion of this work is that, regarding PSO to solve CNOPs, the previous knowledge about the heuristic used as a search engine led to a less complex but competitive approach. Part of the future work is to improve the dynamic adaptation proposed in this chapter i.e. adaptive mechanism, testing more recent PSO variants such as the Fully Informed PSO [26], to test PSO variants with other constraint-handling mechanisms such as adaptive penalty functions [41] and to use IPSO in real-world problems.

Acknowledgements. The first author acknowledges support from CONACyT through project number 79809-Y. The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at LANIA A.C.

References

1. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York (1996)
2. Cagnina, L.C., Esquivel, S.C., Coello, C.A.C.: A Particle Swarm Optimizer for Constrained Numerical Optimization. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 910–919. Springer, Heidelberg (2006)
3. Clerc, M., Kennedy, J.: The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
4. Coello, C.A.C.: Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. Computer Methods in Applied Mechanics and Engineering 191(11-12), 1245–1287 (2002)
5. Deb, K.: An Efficient Constraint Handling Method for Genetic Algorithms. Computer Methods in Applied Mechanics and Engineering 186(2/4), 311–338 (2000)
6. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions of Systems, Man and Cybernetics-Part B 26(1), 29–41 (1996)
7. Eiben, A., Smith, J.E.: Introduction to Evolutionary Computing. Natural Computing Series. Springer, Heidelberg (2003)
8. Eiben, G., Schut, M.C.: New Ways to Calibrate Evolutionary Algorithms. In: Siarry, P., Michalewicz, Z. (eds.) Advances in Metaheuristics for Hard Optimization. Natural Computing Series, pp. 153–177. Springer, Heidelberg (2008)
9. Engelbrecht, A.: Fundamentals of Computational Swarm Intelligence. John Wiley & Sons, Chichester (2005)
10. Fogel, L.: Autonomous Automata. Industrial Research 4(12), 14–19 (1962)
11. Glover, F., Laguna, F.: Tabu Search. Kluwer Academic Publishers, Norwell (1997)
12. Glover, F., Laguna, M., Ri, M.: Scatter Search. In: Advances in Evolutionary Computing: Theory and Applications, pp. 519–537. Springer, New York (2003)
13. He, S., Prempain, E., Wu, Q.H.: An Improved Particle Swarm Optimizer for Mechanical Design Optimization Problems. Engineering Optimization 36(5), 585–605 (2004)
14. Holland, J.H.: Concerning Efficient Adaptive Systems. In: Yovits, M.C., Jacobi, G.T., Goldstein, G.D. (eds.) Self-Organizing Systems, pp. 215–230. Spartan Books, Washington (1962)
15. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann, UK (2001)
16. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220, 671–680 (1983)

17. Koza, J.R.: Genetic Programming. On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
18. Krohling, R.A., dos Santos Coelho, L.: Coevolutionary Particle Swarm Optimization Using Gaussian Distribution for Solving Constrained Optimization Problems. IEEE Transactions on Systems, Man and Cybernetics Part B 36(6), 1407–1416 (2006)
19. Lampinen, J.: A Constraint Handling Approach for the Diifferential Evolution Algorithm. In: Proceedings of the Congress on Evolutionary Computation (CEC 2002), vol. 2, pp. 1468–1473. IEEE, Piscataway (2002)
20. Li, H., Jiao, Y.C., Wang, Y.: Integrating the Simplified Interpolation into the Genetic Algorithm for Constrained Optimization problems. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS, vol. 3801, pp. 247–254. Springer, Heidelberg (2005)
21. Li, X., Tian, P., Kong, M.: Novel Particle Swarm Optimization for Constrained Optimization Problems. In: Zhang, S., Jarvis, R. (eds.) AI 2005. LNCS, vol. 3809, pp. 1305–1310. Springer, Heidelberg (2005)
22. Li, X., Tian, P., Min, X.: A Hierarchical Particle Swarm Optimization for Solving Bilevel Programming Problems. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS, vol. 4029, pp. 1169–1178. Springer, Heidelberg (2006)
23. Liang, J.J., Suganthan, P.N.: Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constrain-Handling Mechanism. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2006), pp. 316–323. IEEE, Vancouver (2006)
24. Liang, J.J., Runarsson, T., Mezura-Montes, E., Clerc, M., Suganthan, P.N., Coello, C.C., Deb, K.: Problem Definitions and Evaluation Criteria for the CEC 2006, Special Session on Constrained Real-Parameter Optimization. Tech. rep. (2006), `http://www3.ntu.edu.sg/home/EPNSugan/`
25. Lu, H., Chen, W.: Dynamic-Objective Particle Swarm Optimization for Constrained Optimization Problems. Journal of Combinatorial Optimization 12(4), 409–419 (2006)
26. Mendes, R., Kennedy, J., Neves, J.: The Fully Informed Particle Swarm: Simpler, Maybe Better. IEEE Transactions on Evolutionary Computation 8(3), 204–210 (2004)
27. Mezura-Montes, E., Coello, C.A.C.: Identifying On-line Behavior and Some Sources of Difficulty in Two Competitive Approaches for Constrained Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005), vol. 2, pp. 1477–1484. IEEE, Edinburgh (2005)
28. Mezura-Montes, E., Coello-Coello, C.A.: Constrained Optimization via Multiobjective Evolutionary Algorithms. In: Multiobjective Problems Solving from Nature: From Concepts to Applications. Natural Computing Series, pp. 53–76. Springer, Heidelberg (2008)
29. Mezura-Montes, E., López-Ramírez, B.C.: Comparing Bio-Inspired Algorithms in Constrained Optimization problems. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007), pp. 662–669. IEEE, Singapore (2007)
30. Michalewicz, Z., Schoenauer, M.: Evolutionary Algorithms for Constrained Parameter Optimization Problems. Evolutionary Computation 4(1), 1–32 (1996)
31. Paquet, U., Engelbrecht, A.P.: A New Particle Swarm Optimiser for Linearly Constrained Optimization. In: Proceedings of the Congress on Evolutionary Computation (CEC 2003), vol. 1, pp. 227–233. IEEE, Canberra (2003)
32. Parsopoulos, K., Vrahatis, M.: Unified Particle Swarm Optimization for Solving Constrained Engineering Optimization Problems. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3612, pp. 582–591. Springer, Heidelberg (2005)

33. Powell, D., Skolnick, M.M.: Using Genetic Algorithms in Engineering Design Optimization with Non-Linear Constraints. In: Forrest, S. (ed.) Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA 1993), University of Illinois at Urbana-Champaign, pp. 424–431. Morgan Kaufmann Publishers, San Mateo (1993)
34. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution. In: A practical Approach to Global Optimization. Springer, Heidelberg (2005)
35. Runarsson, T.P., Yao, X.: Stochastic Ranking for Constrained Evolutionary Optimization. IEEE Transactions on Evolutionary Computation 4(3), 284–294 (2000)
36. Schoenauer, M., Xanthakis, S.: Constrained GA Optimization. In: Forrest, S. (ed.) Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA 1993), University of Illinois at Urbana-Champaign, pp. 573–580. Morgan Kauffman Publishers, San Mateo (1993)
37. Schwefel, H.P. (ed.): Evolution and Optimization Seeking. Wiley, New York (1995)
38. Shi, Y., Eberhart, R.: A Modified Particle Swarm Optimizer. In: Proceedings of the IEEE World Congress on Computational Intelligence, pp. 69–73 (1998)
39. Shi, Y., Eberhart, R.C.: Parameter Selection in Particle Swarm Optimization (1998), `http://www.engriupuiedu/ shi/PSO/Paper/EP98/psof6/ep98_psohtml`
40. Takahama, T., Sakai, S., Iwane, N.: Solving Nonlinear Constrained Optimization Problems by the ε Constrained Differential Evolution. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Taipei, Taiwan, pp. 2322–2327 (2006)
41. Tessema, B., Yen, G.G.: A Self Adaptative Penalty Function Based Algorithm for Constrained Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2006), pp. 950–957. IEEE, Vancouver (2006)
42. Toscano-Pulido, G., Coello Coello, C.A.: A Constraint-Handling Mechanism for Particle Swarm Optimization. In: Proceedings of the Congress on Evolutionary Computation (CEC 2004), vol. 2, pp. 1396–1403. IEEE, Portland (2004)
43. Wei, J., Wang, Y.: A novel multi-objective PSO algorithm for constrained optimization problems. In: Wang, T.-D., Li, X.-D., Chen, S.-H., Wang, X., Abbass, H.A., Iba, H., Chen, G.-L., Yao, X. (eds.) SEAL 2006. LNCS, vol. 4247, pp. 174–180. Springer, Heidelberg (2006)

Applying River Formation Dynamics to Solve NP-Complete Problems

Pablo Rabanal, Ismael Rodríguez, and Fernando Rubio

Abstract. For obvious practical reasons, NP-complete problems are typically solved by applying heuristic methods. In this regard, nature has inspired many heuristic algorithms to obtain *reasonable* solutions to complex problems. One of these algorithms is *River Formation Dynamics* (RFD). This heuristic optimization method is based on imitating how water forms rivers by eroding the ground and depositing sediments. After *drops* transform the landscape by increasing/decreasing the altitude of places, solutions are given in the form of paths of *decreasing* altitudes. Decreasing gradients are constructed, and these gradients are followed by subsequent drops to compose new gradients and reinforce the best ones. In this chapter, we apply RFD to solve three NP-complete problems, and we compare our results with those obtained by using *Ant Colony Optimization* (ACO).

1 Introduction

NP-complete problems are (strongly) supposed to require exponential time in the worst case to be solved. Fortunately, heuristic methods can be used to obtain sub-optimal solutions in reasonable time. Nature has been a source of inspiration for obtaining many interesting and useful heuristic algorithms (see e.g. [12, 13, 4, 11, 5, 10]). Among them, we would like to highlight *Ant Colony Optimization* (ACO) [7, 5, 6]. This method provides algorithms based on how (natural) ants find the shortest path from the colony to the food source. This efficient method is well-known: (1) Ants release pheromones as they move; (2) ants tend to follow pheromone trails; and thus, (3) paths are reinforced. In the long term, the reinforcement of paths is stronger in short paths than in long paths because ants can traverse the former paths more often per unit of time. Eventually, only a good short path

Pablo Rabanal · Ismael Rodríguez · Fernando Rubio
Dept. Sistemas Informáticos y Computación, Facultad de Informática,
Universidad Complutense de Madrid, 28040 Madrid, Spain
e-mail: `prabanal@fdi.ucm.es`, `{isrodrig, fernando}@sip.ucm.es`

prevails and the rest of paths vanish. A pheromone value is attached to each edge, and ants probabilistically tend to choose those edges where the ratio *'pheromone trail at destination'/'edge cost'* is the highest.

Alternatively, let us suppose that ants' decisions were based on the *gradient* of trail values instead of the trail values themselves. In particular, let us suppose that ants probabilistically tend to choose the movement providing the highest ratio *'difference of trails between the new place and the current place'/'edge cost'* (for instance, higher the decrease, higher the probability). Leaving aside for now the important question of how ants could iteratively create paths of *decreasing* pheromone trails (which will be addressed later), what are the differences between this gradient approach and the standard approach? First, in the standard approach ants can be led by pheromone trails in such a way that, after some movements, it is impossible not to repeat a node, i.e. a local cycle is followed. Let us note that following a cyclic pheromone trail does not imply that any previous ant actually followed this cycle; in particular, each part of the cycle could have been reinforced by an ant following a different path. When an ant finds that it cannot avoid to repeat a node, it is either *killed* or reinserted at the origin node. In both cases, the computational effort required to move it was useless. However, following a cycle is impossible in the gradient approach because it would require an *ever decreasing* cycle, which is contradictory. Second, let us note that in the standard approach, when an ant finds a shorter path, it needs a lot of movement to *convince* other ants following older well-reinforced paths to join the new path. Technically speaking, reinforcing the new path until pheromone trails are higher than in older paths requires a lot of subsequent steps. On the other hand, if the difference of trails is considered, then when a shorter path is discovered, from this precise moment onwards its edges are preferable (on average) to the edges of older paths. This is because the difference of pheromone trails between the final destination and the origin is the same in these paths (the origin and the destination are the same indeed), but the cost is lower in the shorter path. So, the ratio *'total difference of trails'/'total cost'* is higher in the shorter path. On the contrary, when a shorter path is found in the standard approach, the edges of this path are not preferable yet (not even when considered as a whole) because the amount of pheromones in its edges is still negligible.

This alternative approach provides some advantages, but the following question arises: How can we get pheromone trails to decrease along the steps of each path? We can find an answer to this question by giving the ant metaphor up and getting some inspiration from another nature-based phenomenon: The *river formation dynamics*. Let us consider that a water mass is unleashed at some high point. Gravity will make it follow a path down until it cannot go down anymore. In geological terms, when it rains in a mountain, water tries to find its own way down to the sea. Along the way, water erodes the ground and transforms the landscape, which eventually creates a riverbed. When a strong downward slope is traversed by water, it extracts soil from the ground on the way. This soil is deposited later when the slope is lower. Rivers affect the environment by reducing (i.e. eroding) or increasing (i.e. depositing) the altitude of the ground. Let us note that if water is unleashed at all points of the landscape (e.g., it rains) then the river form tends to optimize the task of collecting *all* the water and taking it to the sea, which does not imply taking the shortest path from a *given* origin point to the sea. Let us remark that there are *a lot of* origin points to consider (one for each point where a drop falls). In fact, a kind of *combined* grouped shortest path is created in this case. The formation of tributaries and meanders is a consequence of this. However, if water flows from a *single* point and no other water source is considered, then the water tends to form an efficient way to reduce the altitude (i.e., it tends to form a short path between the origin and the destination).

An algorithm based on these ideas called *River Formation Dynamics* (RFD) was presented in [17]. In order to apply the previous scheme, ants are substituted by *drops* and pheromone trails are replaced by *altitudes*. Drops tend to flow down the slope and they modify altitudes in the process. A classical benchmark NP-complete problem, the *Traveling Salesman Problem* (TSP) [9, 1], was considered, which required to adapt the general scheme to this particular problem (for instance, since the general framework implicitly instructs that drops avoid cycles, a change was introduced to allow cycles involving *all* nodes). The applicability of RFD to other NP-complete problems was studied in [18]. Given a cost-evaluated graph, let us consider the problems of (a) finding the minimum spanning tree, and (b) finding the minimum *distances* tree (that is, a tree such that the addition of distances from each node to a given *exit node* is minimal). The standard forms of both problems do not require using heuristic methods because they can be polynomially solved (e.g., by using Kruskal and Dijkstra algorithms, respectively). However, some generalizations of both problems are NP-complete indeed. In particular, let us consider that the cost of taking an edge e depends on the path followed so far. That is, if we traverse e *after* following a path σ then the cost of adding e to the path is $c_{e,\sigma}$; in general, we have $c_{e,\sigma} \neq c_{e,\sigma'}$ for any other path σ'. We denote these graphs as *variable-cost graphs*. The problems of finding a minimum spanning tree or a minimum distances tree for a variable-cost graph (denoted by MSV and MDV, respectively) were defined in [18], where the capability of RFD to solve them was considered. As we will discuss later, both generalizations of the standard problems are NP-complete, and they are applicable to some IT domains (e.g. *formal testing methods* and *routing*). However, to the best of our knowledge, they have not been considered in the literature before.

Since TSP, MSV, and MDV consist in finding some kinds of short paths, the characteristics of RFD commented before (that is, the avoidance of local cycles and the fast reinforcement of shorter paths) make it a suitable choice. Moreover, the geological metaphor provides another characteristic that is important in this regard. Let us note that the *erosion* process provides a method to *punish* inefficient paths as well as to avoid blocked paths: If a path leads to a node that is lower than any adjacent node (i.e., it is a blind alley) then the drop will deposit its sediment, which will increase the altitude of the node. Eventually, the altitude of this node will match the altitude of its neighbors, which will avoid other drops falling on this node. If the ground reaches this level, other drops will be allowed to cross this node from one adjacent node to another. Thus, paths will not be interrupted at this point.

In [17] and [18], the performance of RFD to solve `TSP` and `MSV`, respectively, was analyzed by comparing the results of RFD with those given by an ACO implementation for the same problem instances. It was observed that the time required by RFD to find good solutions is in general longer than the time required by ACO to find equivalent solutions, though solutions provided by RFD outperformed those given by ACO after some additional time passes. As we will explain later in higher detail, the reasons for these differences lie in the fact that RFD develops a *deeper* exploration of the graph. In this chapter we summarize our previous work on RFD in an integrated fashion. We sketch the main ideas of the algorithm and we recall previous experiments. In addition, we report and analyze other experiments where RFD and ACO are compared for the same instances of the `MDV` problem. Moreover, we conduct new experiments to study the capability of RFD and ACO to deal with *dynamic* graphs, i.e. graphs where nodes and edges can appear/disappear along time, in the three problems. This will allow us to study the capability of drops and ants to dynamically *adapt* paths found so far to environmental changes. Finally, in order to demonstrate the difficulty of `MSV` and `MDV`, we prove the NP-completeness of both problems. In particular, we polynomially reduce `3-SAT` to each of them.

The rest of the chapter is structured as follows. Next we describe the main ideas of the RFD algorithm. In Section 3, we formally define the problems we have considered in this chapter to analyze the performance of RFD. Next, in Section 4 we apply RFD and ACO to solve `TSP`, `MSV`, and `MDV` in the case where graphs are static (i.e., they do not change along time), and we report some results. These three problems are revisited in Section 5, where we repeat these experiments in a context where graphs are dynamic. We present our conclusions and lines of future work in Section 6. We prove the NP-completeness of `MSV` and `MDV` in the appendix of this chapter.

2 Main Ideas of RFD

In this section we introduce the basic structure of our method based on river formation dynamics. The method works as follows. Instead of associating pheromone values to edges, we associate *altitude* values to nodes. *Drops* erode the ground (they reduce the altitude of nodes) or deposit the sediment (increase it) as they move. The probability of the drop to take a given edge instead of others is proportional to the gradient of the downward slope in the edge, which in turn depends on the difference of altitudes between both nodes and the edge distance (i.e. the *cost* of the edge). At the beginning, a flat environment is provided, that is, all nodes have the same altitude. The exception is the destination node, which is a *hole*. Drops are unleashed at the origin node and spread around the flat environment until some of them fall in the destination node. This erodes adjacent nodes, which creates new downward slopes, and in this way the erosion process is propagated. New drops are inserted in the origin node to transform paths and reinforce the erosion of promising paths. After some steps, good paths from the origin to the destination are found. These paths are given in the form of sequences of decreasing edges from the origin to the destination.

This method provides some advantages over ACO that were briefly outlined before in the introduction. On one hand, local cycles are not created and reinforced because they would imply an *ever decreasing cycle*, which is contradictory. Though ants usually take into account their past path to avoid repeating nodes, they cannot avoid being led by pheromone trails through some edges in such a way that a node must be repeated in the next step. However, *altitudes* cannot lead drops to these situations. Moreover, since drops do not have to worry about following cycles, in general drops do not need to be endowed with *memory* of previous movements, which releases some computational memory and reduces some execution time.[1] On the other hand, when a shorter path is found in RFD, the subsequent reinforcement of the path is fast: Since the same origin and destination are concerned in both the old and the new path, the difference of altitude is the same but the distance is different. So, the (global) gradient of the shorter path makes it preferable to any other path connecting the same nodes. This does not necessarily imply that all *individual* edges in this path are immediately preferred to their respective competitors; this only means that these edges are preferred on an *average*. In particular, bad-gradient steps in the shorter path must be compensated by other good gradients in the path; otherwise, the alternative path cannot be shorter. The erosion/sedimentation process tends to equalize gradients in paths, so bad gradients in the shorter path tend to gain the surplus from other good gradients in the path. In this way, all edges of this path easily tend to be preferable to other edges. On the contrary, when a shorter path is found in ACO, this path is not immediately preferable (not even if considered as a whole) because pheromone trails are still negligible on this path. In particular, the pheromone trail at each individual edge is also negligible, so *all* edges of this path are still far away from being preferable compared to their respective competitors. Thus, the reinforcement of shorter paths is faster in RFD than in ACO. Finally, the erosion process provides a method to avoid inefficient solutions because sediments tend to be cumulated in blind alleys (in our case, in *valleys*). These nodes are filled until eventually their altitude matches those of adjacent nodes, i.e., the valley disappears. This differs from typical methods to reduce pheromone trails in ACO: Usually, the trails of *all* edges are periodically reduced at the same rate. On the contrary, RFD intrinsically provides a *focused* punishment of bad paths where, in particular, those nodes blocking alternative paths are modified.

Let us consider the applicability of RFD to `MSV` and `MDV` (its applicability to `TSP` will be considered below). These problems consist in finding a kind of *combination* of short paths, in particular a tree. After executing RFD for some time, for each node we take the edge with the highest gradient, and we discard the rest of edges. This guarantees that selected edges form a tree: If the formed subgraph includes two or more paths to go from A to B then, for at least one node, there are two

[1] In fact, each drop still needs memory to know the amount of sediments it is carrying. This is a single value so, for each drop, the size of the required memory is in $\mathcal{O}(1)$. On the contrary, if drops were required to record the previously traversed path, then the required memory would be in $\mathcal{O}(n)$, where n is the number of nodes of the graph.

or more outgoing edges (which contradicts the fact that, for each node, only the edge providing the highest gradient is taken). As discussed before, natural rivers do not tend to form solutions where each drop goes to the sea through its shortest path, but they tend to form grouped solutions. This allows RFD to implicitly deal with *path conflicts*, i.e. situations where, at a given node, two drops coming from different origins have different preferences regarding which edge should be taken next (because costs are different for each of them; recall that, in `MSV` and `MDV`, we are considering that costs depend on previously followed paths). In these situations, the tendency of RFD to form grouped solutions implicitly leads to forming paths with a suitable cost tradeoff between available choices: After some steps, the erosion will reinforce more strongly the slopes providing the lowest overall cost. In addition, in the *minimum spanning tree* problem (`MSV`), the tendency of drops to join each other is very appropriate: If drops tend to join the main *flow*, instead of following their respective individual shortest paths, then less edges are added to the tree and the tree cost is reduced.

Let us note that the tendency of ACO methods to form grouped solutions is well known, so similar arguments can be given in the case of ACO. In particular, ACO allows to form short paths from a single node to a single destination. However, combining some short paths departing from different points in such a way that a *tree* is formed is not a natural task for ACO. Let us suppose that two paths coming from different origins join at a given node and then continue together.[2] Ants coming from a departure node can be confused by pheromone trails and go on to the *other* departure node, instead of following to the destination node. Solving this problem requires to use some artificial methods (e.g., using different types of pheromones, using *directed* pheromones, associating ants to specific areas, etc). On the contrary, edge gradients formed by RFD are *intrinsically* directed, and their direction naturally leads to the destination node. This eases the task of constructing trees in RFD. Interestingly, we can adapt RFD to the *minimum distances tree* problem (`MDV`) just by changing a parameter: If we reduce the erosion caused by *high flows*, then the incentive of drops to join each other is partially reduced, and thus each drop tends to follow its own shortest path. For instance, we can achieve this effect by changing the erosion rules in such a way that, if n drops traverse an edge, then they make the effect of e.g. a single drop. In this case, grouped paths are promoted by the method only when they are required to solve path conflicts. Moreover, by considering *intermediate* erosion effects, we can construct trees partially fitting into the objectives of both problems (i.e., a combination of minimum spanning tree and minimum distances tree). This may be a suitable choice for several optimization problems.[3]

Regarding the third NP-complete problem considered in this chapter, `TSP`, let us note that some of the previous considerations apply to this case as well. In particular, the mechanism of focalized punishment of inefficient paths, the avoidance of local

[2] That is, the convergence area reminds the form of a 'Y' letter.

[3] For instance, we design a subway network to carry citizens from different areas to downtown in such a way that (a) the time spent by citizens to arrive to downtown is minimized (i.e., we need a minimum distances tree), and (b) the expenses required to build tunnels are minimized (i.e., we need a minimum spanning tree).

cycles, and the fast reinforcement of shorter paths are also good for solving `TSP`. However, facing this problem requires slightly modifying the general RFD scheme. In particular, let us note that `TSP` requires finding a *cycle* in the graph, but RFD implicitly avoids cycles. The adaptation of RFD to solve `TSP` will be addressed later in Section 2.2.

2.1 Basic Algorithm

The basic scheme of the RFD algorithm follows:

```
initializeDrops()
initializeNodes()
while (not allDropsFollowTheSamePath()) and
      (not otherEndingCondition())
   moveDrops()
   erodePaths()
   depositSediments()
   analyzePaths()
end while
```

This scheme shows the main ideas of the proposed algorithm. We comment on the behavior of each step. First, drops are initialized (`initializeDrops()`), i.e., all drops are put in the initial node(s). Next, all nodes of the graph are initialized (`initializeNodes()`). This consists of two operations. On one hand, the altitude of the destination node is fixed to 0. In terms of the river formation dynamics analogy, this node represents the *sea*, that is, the final goal of all drops. On the other hand, the altitude of the remaining nodes is set to some equal value.

The `while` loop of the algorithm is executed until either all drops find the same solution (`allDropsFollowTheSamePath()`), that is, all drops departing from the same initial nodes traverse the same sequences of nodes, or another alternative finishing condition is satisfied (`otherEndingCondition()`). This condition may be used, for example, for limiting the number of iterations or the execution time. Another choice is to finish the loop if the best solution found so far is not surpassed during the last *n* iterations.

The first step of the loop body consists of moving the drops across the nodes of the graph (`moveDrops()`) in a partially random way. The following *transition rule* defines the probability that a drop *k* at a node *i* chooses the node *j* to move next:

$$P_k(i,j) = \begin{cases} \frac{decreasingGradient(i,j)}{\Sigma_{l\in V_k(i)}\, decreasingGradient(i,l)} & \texttt{if } \ j \in V_k(i) \\ 0 & \texttt{if } \ j \notin V_k(i) \end{cases} \tag{1}$$

where $V_k(i)$ is the set of nodes that are *neighbors* of node *i* that can be visited by the drop *k* and have a negative value of *decreasingGradient(i,j)*, which represents the gradient between nodes *i* and *j* and is defined as follows:

$$decreasingGradient(i,j) = \frac{altitude(j) - altitude(i)}{distance(i,j)} \tag{2}$$

where *altitude(x)* is the altitude of the node x and *distance(i,j)* is the length of the edge connecting node i and node j. Let us note that, at the beginning of the algorithm, the altitude of all nodes is the same, so $\sum_{l \in V_k(i)} decreasingGradient(i,l)$ is 0. In order to give a special treatment to flat gradients, we modify this scheme as follows: We consider that the probability of a drop moving through an edge with 0 gradient is set to some (non null) value. This enables drops to spread around a flat environment, which is required, in particular, at the beginning of the algorithm.

In fact, going one step further, we also introduce this improvement: We let drops climb *increasing* slopes with a low probability. This probability will be inversely proportional to the increasing gradient, and it will be reduced during the execution of the algorithm by using a method similar to the one followed by *Simulated Annealing* (see [12, 8]). This new feature improves the search of good paths. Let us note that solutions found during the first steps tend to bias the exploration of the graph afterwards. This is because previously formed paths tend to be followed by subsequent drops. Enabling drops to climb increasing slopes with some low probability allows us to find alternative choices and enables the exploration of other paths in the graph. This partially decouples the method from its behavior in the first steps. Actually, the probability of climbing increasing slopes encapsulates most of the dependency of the method on previous solutions in a single value. As usual in heuristic search algorithms, this dependency must provide a suitable tradeoff between past solutions and alternative choices. Let us note that allowing climbing up gradients does not invalidate the argument that local cycles are avoided in practice in our method: After following a sequence of downward gradients, completing a cycle requires to climb up all the altitude lost so far, and the probability of climbing up a high gradient is negligible.

The climbing up mechanism works as follows. Given a drop d located at node k, we randomly decide whether d can climb upward gradients according to the following probability:

$$P(d) = \frac{1}{notClimbingFactor} \tag{3}$$

where *notClimbingFactor* is a variable that is initially set to 1 and is slightly increased after each loop iteration. Every N iterations, this variable is not increased, but *decreased* (this is done to emulate the *tempering* process used in Simulated Annealing). When the drop is *allowed* to climb up and it decides its next move, it can also choose moving upwards – as well as moving downwards or going through a null gradient, as the rest of drops can. In particular, the transition rule of a climbing drop is defined as follows:

$$P_k(i,j) = \begin{cases} \frac{decreasingGradient(i,j)}{total} & \texttt{if} \;\; j \in V_k(i) \\ \frac{\omega/|decreasingGradient(i,j)|}{total} & \texttt{if} \;\; j \in U_k(i) \\ \frac{\delta}{total} & \texttt{if} \;\; j \in F_k(i) \end{cases} \tag{4}$$

where

$$total = \left(\sum_{l \in V_k(i)} decreasingGradient(i,l) \right) + \sum_{l \in U_k(i)} \left(\frac{\omega}{|decreasingGradient(i,l)|} \right) + \sum_{l \in F_k(i)} \delta$$

and $V_k(i)$, $U_k(i)$ and $F_k(i)$ are the sets of nodes that are *neighbors* of node i that can be visited by the drop k and are connected to k through a down, up, and flat gradient, respectively. We consider that δ and ω are parameters of the algorithm. On the other hand, if a drop fails to be considered as a climbing drop then it only considers taking down or flat gradients.

In the next phase (`erodePaths()`) paths are eroded according to the movements of drops in the previous phase. In particular, if a drop moves from node A to node B then we erode A. The reduction of the altitude of this node depends on the current gradient between A and B. In particular, the erosion is higher if the downward gradient between A and B is high. The altitude of the eroded node A is modified as follows:

$$altitude(A) := altitude(A) - erosion(A,B)$$

$$erosion(A,B) = \frac{paramErosion}{(numNodes - 1) \cdot numDrops} \cdot decreasingGradient(A,B)$$

where *paramErosion* is a parameter of the erosion process, *numNodes* is the number of nodes of the graph, and *numDrops* is the number of drops used in the algorithm. On the contrary, if the edge is flat or increasing then a small erosion is performed. However, the altitude of the final node (i.e., the *sea*) is never modified and it remains equal to 0 during the execution.

Once the erosion process finishes, the altitude of all nodes of the graph is slightly increased (`depositSediments()`). The objective is to avoid, after some iterations, the erosion process leading to a situation where all altitudes are close to 0, which would make gradients negligible and would ruin all formed paths. In particular, the altitude of a node N is increased according to the following expression:

$$altitude(N) := altitude(N) + (erosionProduced/(numNodes - 1))$$

where *erosionProduced* is the sum of erosions introduced in all graph nodes in the previous phase, and *numNodes* is the number of nodes of the graph.

We also enable individual drops to *deposit* sediment on nodes. This happens when all movements available for a drop imply climbing an increasing slope and the drop fails to climb any edge (according to the probability assigned to it). In this case, the drop is blocked and it deposits the sediments it is transporting. This increases the altitude of the current node in proportion to the cumulated sediment carried by the drop, which in turn is proportional to the erosions produced by the drop in previous movements. If a drop gets blocked at node N, then the altitude of N is increased as follows:

$$altitude(N) := altitude(N) + paramBlockedDrop \cdot cumulatedSediment$$

where *paramBlockedDrop* is a parameter and *cumulatedSediment* is the amount of sediment carried by the drop.

Finally, the last step (`analyzePaths()`) studies all solutions found by drops and stores the best solution found so far.

Some additional improvements can be introduced to this basic scheme (see [17] for details). For instance, we can gather drops to reduce the number of individual movements, we can consider the formation of *lakes* in blind alleys, etc.

2.2 Adapting RFD to TSP

Though the basic RFD scheme is suitable for solving `MSV` and `MDV` as it is, some adaptations are required to apply RFD to `TSP`. We present them in this section (further details can be found in [17]). We can define `TSP` as follows: Given a set of cities and the costs of traveling from any city to any other city, compute the cheapest round-trip route that visits each city exactly once and then returns to the starting city. Note that, since we are looking for a cyclic tour, it is irrelevant what concrete node is the origin of the tour: The cycle A-B-C-D-A has the same length as B-C-D-A-B or C-D-A-B-C.

The adaptation of our method to `TSP` has several similarities with the way ACO is applied to this problem. In ACO, endowing ants with the capability of *remembering* all nodes traversed so far is necessary to avoid repeating nodes. Let us recall that the use of *gradients* strongly minimizes these situations in RFD. In particular, only sequences of low-probability climbing movements can lead drops to follow local cycles.[4] Thus, in RFD memories are not as useful as in ACO, and not using them is preferable in the general case because less operations are performed (e.g., this is the case in `MSV` and `MDV`). However, in `TSP` memories are actually required to identify round-trips: Any path not including all nodes is *not* a round-trip, and so it should not be reinforced. Thus, drops must be endowed with memory indeed.

Another difference with the general scheme of RFD is that altitudes are not eroded after each individual drop movement. On the contrary, altitudes of all traversed nodes are eroded all together when the drop actually completes a round-trip (as ants actually increase pheromone trails). This is another reason to keep track of the sequence of traversed nodes. As in ACO, when a drop finds that all adjacent nodes have already been visited, the drop disappears (again, this is less probable in RFD than in ACO). In RFD, there is an additional reason why a drop may disappear: When all adjacent nodes are *higher* than the actual node (i.e., the node is *valley*) and the drop fails to climb any up gradient, the drop is removed and it deposits the sediment it carries at the current node. Let us note that in this case the computations performed to move the drop should not be considered as *lost* despite the fact that it did not find a solution. This is because the cumulation of sediments increases the node altitude, and thus gradients coming from adjacent nodes are flattened. This eventually avoids that other drops fall in the same *blind alley*.

[4] This may happen e.g. at the earliest steps of the algorithm, where the environment is still almost flat.

Other peculiarities of the adaptation of RFD to TSP are specific to RFD and do not appear in ACO. Our method provides an intrinsic method to avoid drops traversing cycles, but the goal of TSP consists in finding a *cycle* indeed. In order to allow drops to follow a cycle involving *all* nodes, the origin node (which, as we said before, can be *any* fixed node) is cloned as well as all edges involving it. The original node plays the role of *origin* node and the cloned node plays the role of *destination* node. In this way, drops can form decreasing gradients from the origin node to the destination node (for us, from a node to *"itself"*).

Finally, the adaptation of RFD to TSP requires introducing other additional nodes for different technical reasons. Let us suppose that a solution A-B-C-D-E-A' is found (A' being the clone of A playing the role of *destination* node). Drops create a decreasing gradient along this path. In particular, the altitude of B is higher than the altitude of C, C is higher than D, and D is higher than E. Let us also suppose that there exists an edge from B to E. Since the difference of altitude between B and E is the addition of the differences between B and C, C and D, and D and E, the decreasing gradient from B to E could be so big that drops *prefer* to go directly from B to E. However, in that case drops will fail in finding a solution: C, D, and E would not be included in this path, but solutions must traverse all nodes. In order to avoid the altitude of adjacent nodes wrongly deviating paths, an auxiliary node will be created at each *edge* of the graph. These new nodes, called *barrier* nodes, are introduced as follows: If there is an edge connecting (standard) nodes X and Y, this edge is replaced by an edge connecting X and a new *barrier* node xy, as well as another edge connecting xy and Y. If a drop traverses all standard nodes (i.e., it finds a solution) then barrier nodes traversed in the path are eroded exactly as standard nodes are. For instance, let us consider the solution A-B-C-D-E-A' given in the previous example. This solution actually traverses nodes A-ab-B-bc-C-cd-D-de-E-ea'-A', ab being the barrier node appearing between A and B, and so on. These barrier nodes will be eroded when finding this solution. However, the barrier node between B and E, be, is not eroded by drops following this path. In fact, if a drop moves directly from B to E and next moves to A', then it will not find a solution, so node be will not be eroded by this alternative path. Thus, the altitude of node be will remain high and drops at B will not prefer moving to be. That is, be imposes a *barrier* between B and E.

Let us note that barrier nodes must be taken into account in the initialization and sedimentation phases of the algorithm. In the initialization phase, we must set the height of barrier nodes. This height will be the same as the height of the rest of nodes of the graph. Regarding the sedimentation phase, it will be necessary to increase their heights in the same way as any other node.

3 Formal Definition of MSV and MDV

Though the TSP problem is well-known, a precise definition of the other two problems considered in this chapter, MSV and MDV, is required. In this section we formally define them, and we briefly consider their applicability to some computational problems. As we said in the introduction, finding a minimum spanning tree (i.e. a tree embracing all nodes where the addition of costs of all edges in the tree is minimal) or a minimum distances tree (i.e. a tree where the addition of distances from each node to the exit node through edges in the tree is minimal) are polynomial time problems, and thus heuristic methods are not necessary in this case. However, if we assume that the cost of including an edge in the tree depends on the *rest* of edges included in the tree, then these problems are not that simple. In particular, let us consider that the cost of an edge depends on the path we have to traverse in the tree before taking it. More technically, we assume that the cost of a path of edges $e_1, \ldots, e_n$ from a given origin node o to a given destination node d depends on the evolution of a *variable* through the path. Initially, a value v_o is assigned to this variable at node o. Then, the cost added to the path due to the inclusion of edge e_1 is an amount depending on v_o. After traversing e_1, the value of the variable is updated to a new value v_1. Next, the cost of adding e_2 to the path depends on v_1. After taking e_2, the value of the variable is updated again, and the process continues so on until we obtain the whole cost of the path $e_1, \ldots, e_n$.

Following this idea, we can define a variable-cost graph by attaching some information to a standard graph. Let us consider a set of *origin* nodes (in particular, this set could include *all* nodes of the graph). Then, (1) we assign an *initial value* to each origin node; (2) we assign a *cost function* to each edge. Depending on the value of the variable just before traversing the edge, taking the edge adds a different cost; and (3) we assign a *transformation function* to each edge. Given the value of the variable before traversing the edge, it returns the new value after taking it.

Let us suppose that a variable-cost graph defined in these terms is given. On one hand, a *minimum distances tree* is a tree connecting each origin node with the destination node in such a way that the addition of costs of all *paths* from each origin node to the destination is minimal. Since the returned solution is a tree, paths departing from different origin nodes could share some edges (in particular, different sequences of edges could share some suffixes). Let us note that, in general, the cost of a shared edge is different for each path because the value of the variable when the edge is reached may be different for each path. On the other hand, a *minimum spanning tree* is a tree connecting all origin nodes with the destination node in such a way that the addition of costs of all *edges* included in the tree is minimal. In this case, the cost of an edge e in a tree t is computed as follows. Let us consider all the paths of t connecting an origin node with the destination node and including edge e. The cost of e in t is the *average* of the cost of e for all of these paths. Let us note that, in both problems, trees are not required to include all nodes from the original graph, but only those actually used to connect origin nodes to the destination node. In particular, if all nodes are considered origin nodes then the resulting tree must include all nodes indeed.

Definition 1. A *variable-cost graph* is a tuple $G = (N, O, d, V, A, E)$ where:

- N is a finite *set of nodes*,
- $O \subseteq N$ is the set of *origin nodes*,
- d is the *destination node*,
- $V = \{v_1, \ldots, v_n\}$ is a finite set of *values*,

- $A : O \longrightarrow V$ is the *initial value function*, that is, a function assigning an initial value to each origin node.
- E is the *set of edges*. Each edge $e \in E$ is a tuple (n_1, n_2, C, T) where $n_1, n_2 \in N$ are the *origin* and *destination* nodes, respectively, and
  - $C : V \longrightarrow \mathbb{N}$ is the *cost function* of e. Given a value in V denoting the current value of the variable, it returns the cost of traversing e.
  - $T : V \longrightarrow V$ is the *transformation function* of e. Given the current value of the variable, it returns the new value assigned to the variable if e is traversed.

Paths are sequences of edges departing at an origin node and arriving to the destination node. Formally, a path of G is a sequence of edges $\sigma = (e_1, \ldots, e_k)$ with $e_i = (n_i, n'_i, C_i, T_i) \in E$ for all $1 \leq i \leq k$ such that $n_1 \in O$, $n'_k = d$, and for all $1 \leq i \leq k-1$ we have $n'_i = n_{i+1}$. The *cost* of σ, denoted by $c(\sigma)$, is equal to

$$C_1(A(n_1)) + C_2(T_1(A(n_1))) + C_3(T_2(T_1(A(n_1)))) + \ldots + C_k(T_{k-1}(\ldots(T_2(T_1(A(n_1))))\ldots))$$

The term denoting the cost of traversing e_i in the previous expression, that is $C_i(T_{i-1}(\ldots(T_2(T_1(A(n_1))))\ldots))$, will be denoted by $c_{e_i}(\sigma)$. In a notation abuse, we will write $e \in \sigma$ if $e = e_i$ for some $1 \leq i \leq k$.

We say that $G' = (N', O, d, V, A, E')$ with $N' \subseteq N$ and $E' \subseteq E$ is a *tree* of G if for all $o \in O$ there exists a single path $\sigma = (e_1, \ldots, e_k)$ of G' departing from o, that is, such that $e_1 = (o, n, C, T)$ for some n, C, T. For each $o \in O$, we denote by σ_o the unique path of G' departing from o.

The *distances cost* of G', denoted by $dc(G')$, is equal to $\sum_{o \in O} c(\sigma_o)$. The *spanning cost* of G', denoted by $sc(G')$, is equal to $\sum_{e' \in E'} \frac{\sum\{c_{e'}(\sigma_o) | o \in O, e' \in \sigma_o\}}{|\{c_{e'}(\sigma_o) | o \in O, e' \in \sigma_o\}|}$. □

Now we are provided with all the needed machinery to formally define the problems considered in this chapter. As it is usual in Complexity theory, these minimization problems are defined in terms of their equivalent decision problems.

Definition 2. The problem of the *minimum distances tree for a variable-cost graph*, denoted by `MDV`, is stated as follows: Given a variable-cost graph G and a natural number $K \in \mathbb{N}$, is there any tree G' of G such that $dc(G') \leq K$?

The problem of the *minimum spanning tree for a variable-cost graph*, denoted by `MSV`, is stated as follows: Given a variable-cost graph G and a natural number $K \in \mathbb{N}$, is there any tree G' of G such that $sc(G') \leq K$? □

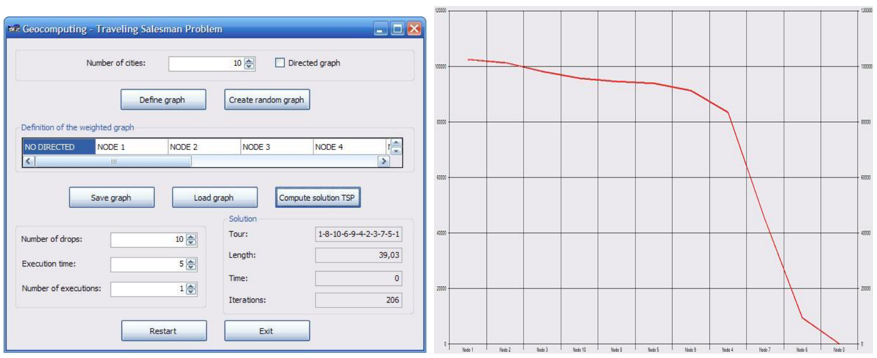
The previous problems generalize the classical *minimum spanning tree* and the *minimum distances tree* problems to the case where the cost of traversing each edge depends on the path traversed before taking the edge. The past path is abstracted by the *value* of the variable, which particularizes the cost of each edge for each path. Let us note that, in formal terms, we do not need to consider *several* variables in the problem definition because the dependence on past paths can be denoted by using a single variable. Though several variants of the minimum spanning tree and the minimum distances tree problems have been studied in the literature, as far as we are concerned the variant problems proposed in this chapter have not been considered. Hence, their properties must be analyzed. A proof of the NP-completeness of both problems is presented in the appendix of this chapter. In fact, variants of both problems where only graphs fulfilling $N = O$ are considered (that is, where *all* nodes are origin nodes) are NP-complete as well. These alternative problems are also considered in the appendix.

As a matter of fact, the generalization introduced in MDV and MSV (that is, the use of variable-cost graphs instead of fix-cost graph) increases their applicability to new interesting scenarios. In fact, we recently came across these problems because we were constructing some *testing derivation algorithms* (for an introduction to *Formal Testing Techniques*, see e.g. [14, 16, 2, 15, 20]). Typically, the goal of a testing methodology is to interact with the analyzed system so that all system states are reached *at least once*. If previous system configurations can be *restored* then we can explore a part of the system, then go back to a previously traversed point, and next go on through a different way. Thus, the problem of reaching all states consists in creating a tree embracing all states. Since the time required to go from state s to state s' depends on the previous activities of the system (available resources, values of variables, etc), composing the optimal tree reaching all states at least once requires taking past activities into account. We can use a variable-cost graph to denote how the execution time of each activity depends on the current values of variables. Thus, if we assume that previous configurations can be *restored* then finding a tree which allows reaching all states in minimum time essentially consists of solving MSV for this graph. There exist other related testing problems whose basic structure fits into this problem scheme as well.

Next we consider an applicability example of MDV. Let us consider that a local area network (LAN) is constructed on top of a given existing networking infrastructure. The transmission cost of a given connection (i.e. *edge*) depends on the kind of information being transmitted (e.g., low connections are unacceptable for a real-time video stream, but may be suitable for low priority packets). Besides, the kind of information being transmitted depends on the kind of sender machine. Thus, a variable-cost graph can be used to define communication costs in the existing infrastructure. Let us suppose that we want to design a networking tree that allows all nodes to communicate with a central dispatcher in such a way that average communication costs are minimized. Finding this tree consists of solving MDV. Other applicability scenarios of MSV and MDV may be considered as well (for instance, this is the case of the *subway design problem* we briefly sketched before).

4 Applying RFD and ACO to MDV, MSV, and TSP

In this section we describe the application of our approach to solve MDV, MSV, and TSP, and we report some experimental results. Only static graphs will be considered in this section. Experiments where nodes and edges appear/disappear along time will be considered in the next section. In both sections, we compare the results obtained by using ACO methods and the solutions obtained by using our method. All experiments were performed in an Intel Core Duo T7250 with a 2.00 GHz processor. The basic aspect of our application interface can be seen in Figure 1 (left);



Geocomputing - Traveling Salesman Problem

Number of cities: 10 ☐ Directed graph

Define graph | Create random graph

Definition of the weighted graph

NO DIRECTED	NODE 1	NODE 2	NODE 3	NODE 4

Save graph | Load graph | Compute solution TSP

Number of drops: 10

Execution time: 5

Number of executions: 1

Solution

Tour: 1-8-10-6-9-4-2-3-7-5-1

Length: 39,03

Time: 0

Iterations: 206

Restart | Exit

Fig. 1 Application interface (left) and altitudes in a TSP solution (right)

the picture on the right shows the altitudes of each of the nodes after solving an instance of TSP.

Next we introduce the values of parameters of RFD and ACO used in these experiments. Parameters depend on the problem we are executing. The number of drops and ants used to solve MDV and MSV was detected not to be very significant; they are set to 10 in both cases. However, we observed that these values did influence results when TSP was solved. In this case, the number of drops is set to 50 and the number of ants is 1000. In RFD, the initial altitude of the nodes is 1000 in the case of MDV and MSV, and 10000 in the case of TSP. Furthermore, parameters *paramErosion* and *paramBlockedDrop* (see Section 2.1) are set to 1 for all problems, and δ and ω are set to 1 and 0.1. The parameter *notClimbingFactor* is increased by 0.01 after every loop iteration, and after every 100 iterations it is decreased by 0.5. For ACO, the initial amount of pheromone in edges is set to 1000 and the amount of pheromone deposited by an ant after a movement is 100. The evaporation rate is set to a standard value, 0.5. The α and β parameters, considered as in [7], are set to 4 and 2, respectively, when MDV and MSV are considered, and they are equal to 1 and 5 when TSP is solved.

4.1 Static MSV

Next we present the results obtained when MSV is solved by both methods. In the case of RFD, we have directly applied the method presented in Section 2, while in the case of ACO we have used an implementation inspired by [3]. Three randomly generated variable-cost graphs with 100, 200, and 300 nodes were considered.[5] In these graphs, each node is connected to approximately 40% of the rest of nodes. Variables can take up to 10 possible values. Cost functions and transformation functions attached to edges are randomly generated. In particular, features such as

[5] All graphs used in experiments in this paper can be downloaded from `http://kimba.mat.ucm.es/~prabanal/`.

monotonicity or injectivity are not required in these functions. Figure 2 shows the results of an experiment where the input of both algorithms was the graph with 100 nodes. The graph shows the cost of the solution found by each algorithm for each execution time (in seconds). Analogously, Figure 3 contains the results obtained by using the graph with 200 nodes as the input of both algorithms, while Figure 4 shows the results for the 300 nodes graph. All figures show the evolution of the algorithms in a *single* execution, but the same basic shape has been obtained for most of the executions. In order to report solutions that are not biased by a single execution, each algorithm was executed thirty times for each of the graphs. Table 1 summarizes the arithmetic mean (Avg), the variance (Var), and the best solution (Best) found by each method among all thirty executions.

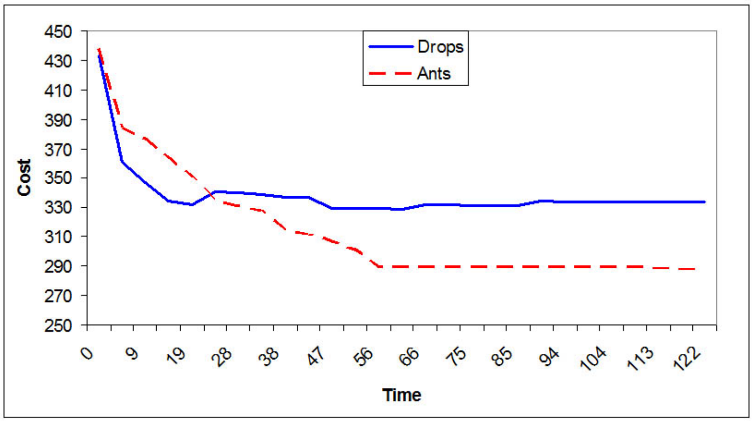


Fig. 2 MSV results for a randomly generated variable-cost graph with 100 nodes

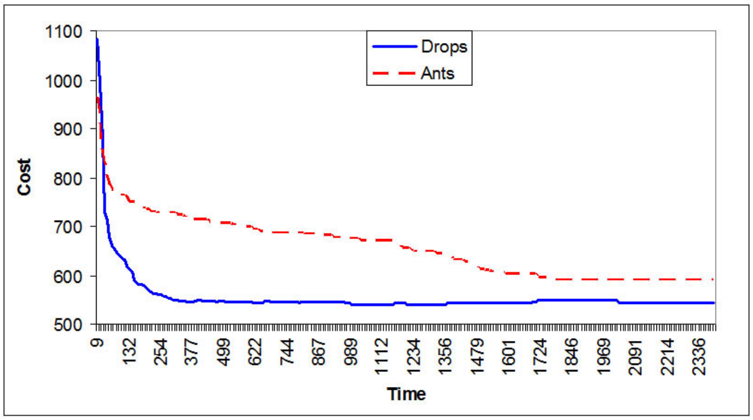


Fig. 3 MSV results for a randomly generated variable-cost graph with 200 nodes

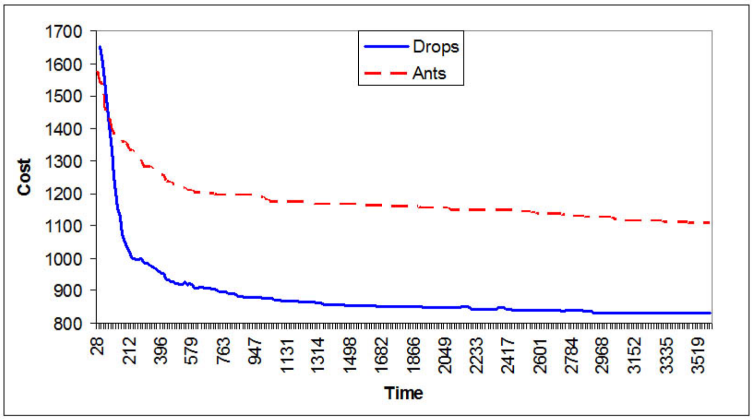


Fig. 4 MSV results for a randomly generated variable-cost graph with 300 nodes

Table 1 Summary of MSV results. Static case.

Graph size	Avg RFD	Avg ACO	Var RFD	Var ACO	Best RFD	Best ACO
100 nodes	300.20	273.30	193.73	17.33	262.75	263.03
200 nodes	538.76	577.54	160.33	365.01	506.36	542.32
300 nodes	829.22	1128.73	128.40	641.79	802.44	1069.99

The results presented in the previous table show that average solutions found by the RFD method are better than the solutions found by ACO. Moreover, the variance is also lower in our algorithm than in the ACO algorithm, with the only exception of the smallest graph.

4.2 Static MDV

Next we study the application of ACO and RFD to solve MDV. Let us recall that changing a single parameter which defines the erosion caused by drops, makes RFD solve either MSV or MDV: If n drops traversing an edge per unit of time erode the ground, as if the erosion effect of a single drop were multiplied by n, then constructed trees will approximately solve MSV. However, if n drops have the effect of a *single* drop, then trees tend to solve MDV.

The comparison results obtained for the MDV problem are similar to the case of MSV. That is, RFD again obtains better solutions (on average), but solutions provided by ACO in short times are better. Times required by RFD to surpass ACO solutions are now a little bit longer. Figures 5, 6, and 7 show the results of experiments performed with 100, 200, and 300 nodes graphs. Again, in all these cases the figures show the evolution of the algorithms in a *single* execution. Following the same methodology as in the previous example, each algorithm was executed thirty

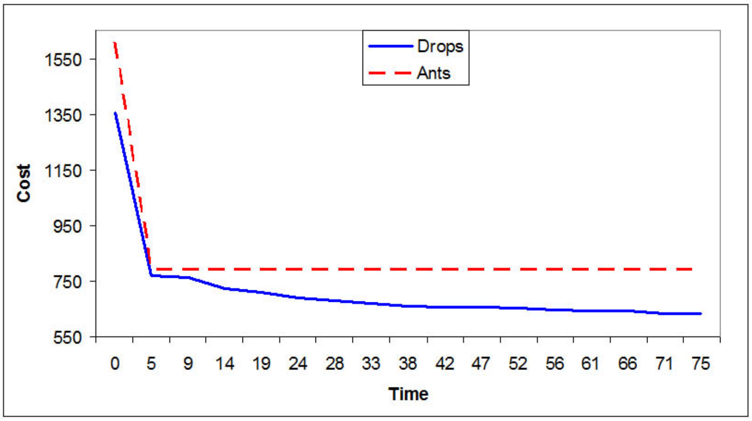


Fig. 5 MDV results for a randomly generated variable-cost graph with 100 nodes

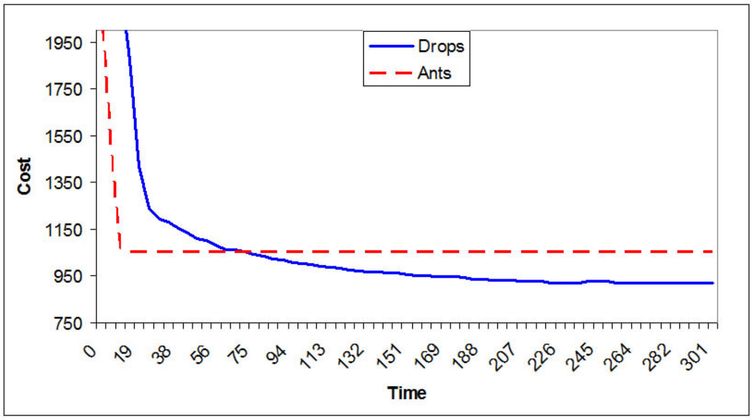


Fig. 6 MDV results for a randomly generated variable-cost graph with 200 nodes

Table 2 Summary of MDV results. Static case.

Graph size	Avg RFD	Avg ACO	Var RFD	Var ACO	Best RFD	Best ACO
100 nodes	613.81	711.90	60.92	4026.26	600.14	587.52
200 nodes	884.05	933.20	111.41	5275.66	854.78	771.64
300 nodes	1362.05	1414.31	244.50	8804.20	1330	1239.73

times for each of the graphs. Table 2 summarizes the same parameters as those considered before for MSV.

As in the case of MSV, the results presented in the previous table show that average solutions found by the RFD method are again slightly better than those found by ACO. Moreover, the variance is lower again. However, now RFD needs more time

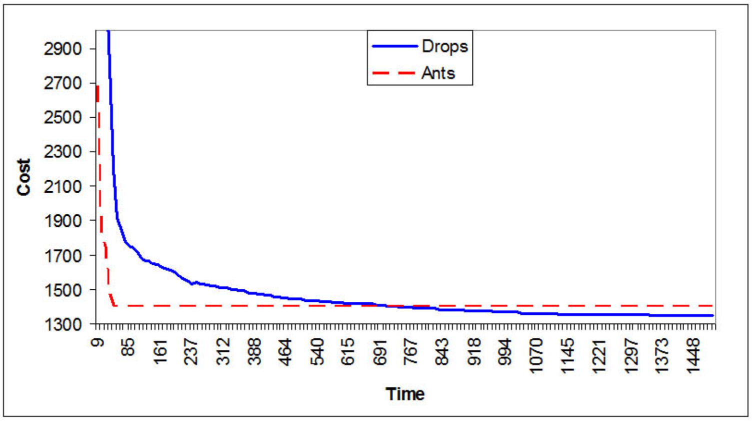


Fig. 7 MDV results for a randomly generated variable-cost graph with 300 nodes

to surpass the quality of the solutions found by ACO. In fact, despite the fact that the mean of solutions is better in RFD than in ACO, the best solution found in all the thirty executions is usually found by ACO. Thus, although the results obtained by RFD are good, the advantage over ACO is not as significant as in the case of MSV. The main reason is that RFD intrinsically promotes the formation of *grouped* paths instead of individual paths, which reduces the size of constructed trees (as required by MSV). In particular, let us note that the incentive of drops to join the main flow is *structurally* provided by the use of *gradients* in RFD: If a drop d moving at a high altitude falls into a strongly eroded flow then, due to the fast reinforcement of shorter paths in RFD, subsequent drops traversing the same area as d will quickly tend to join this flow as well. This feature helps RFD to properly solve MSV (but it is not that useful for solving MDV).

4.3 Static TSP

We apply RFD and ACO to solve some instances of TSP, and we report the collected results. Let us recall that handling TSP by means of RFD requires taking into account the adaptations previously sketched in Section 2.2.

Figures 8 and 9 show the results of two experiments. The first one shows the results for a graph taken from the TSPLIB library [19]. TSPLIB is a library of sample instances for the TSP that has become a standard benchmark for this problem. Nodes are defined by means of points in a 2D plain, and there is an edge between all pairs of nodes. Thus, fully connected graphs are represented. The distance of each edge is the euclidean distance between both points of the plain.

Figure 9 contains a larger case example where we consider a graph of 100 nodes. This graph has been randomly generated assuming that each node is connected with only a few other nodes (between 10 and 20 connections per node) as it is the most common case in complex networks. The basic shape shown in the figures is also obtained with other benchmark examples.

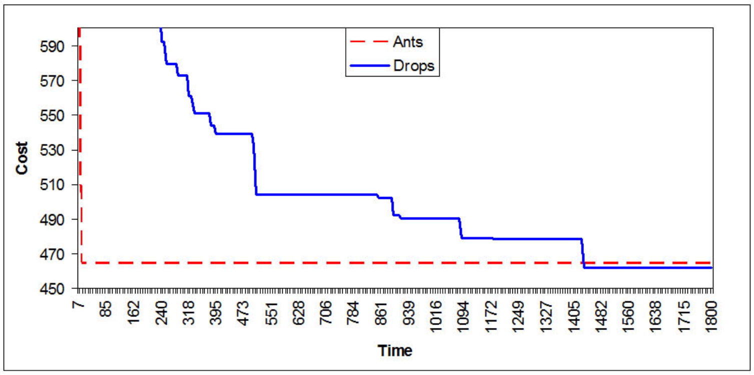


Fig. 8 `TSP` results for the TSPLIB `eil51` graph

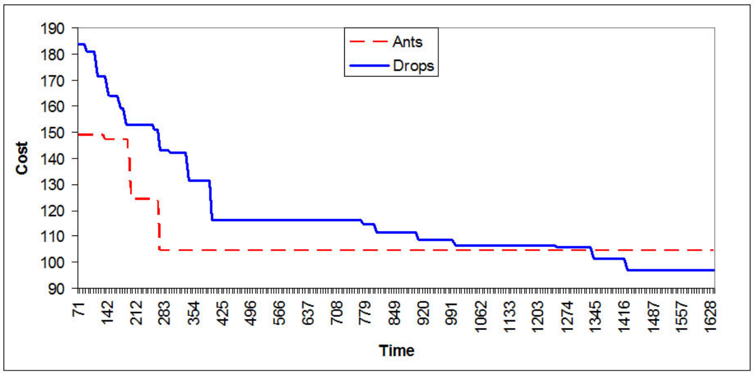


Fig. 9 `TSP` results for a 100 nodes graph

Table 3 Summary of `TSP` results. Static case.

Graph size	Avg RFD	Avg ACO	Var RFD	Var ACO	Best RFD	Best ACO
eil51	458.08	457.97	51.19	3.05	441.9	454.42
100 nodes	101.36	102.61	197	12.06	84.15	96.55

Note that the shapes shown in these figures are analogous to the case of `MSV` and `MDV`: ACO finds solutions faster, while RFD finds better solutions after some time. However, in this case it seems harder for RFD to surpass ACO solutions. The main reason is that drops need memory to register the traversed path in TSP, so one of the advantages of RFD with respect to ACO (that is, the absence of memories) is lost.

Following the same methodology as in the previous example, each algorithm was executed thirty times for each of the graphs. Table 3 summarizes the same parameters as in previous tables.

4.4 Summarizing Results

We extract the following conclusions from the experimental results obtained for the three considered problems. In all problems, we observe that ACO usually provides good solutions earlier than RFD. However, after some additional time passes, the solutions provided by RFD usually surpass the quality of those given by ACO. These features are a consequence of the fact that the exploration of the graph is *deeper* in RFD than in ACO, which in turn is due to the differences between both methods. First, let us note that the erosion-sedimentation mechanism of RFD provides a dynamic dual mechanism to promote/punish good/bad paths or parts of paths. This process allows not only to locally reinforce good parts of paths, but also to locally punish bad sequences. This is an active *try and fail* mechanism which enables a more exhaustive exploration of the graph. Besides, as we said before, the construction of paths of decreasing altitudes implicitly avoids the formation of local cycles, which in turn avoids inefficient movements of drops. Moreover, this forces drops not to be concentrated in local areas but to spread around the graph, which allows a wider and more homogeneous search. In addition, let us recall that the use of gradients makes *shorter* paths preferable overall from the time they are discovered, which accelerates the process of reinforcing them. This increases the competitiveness of shorter paths: At a given time, *several* new shorter paths may have attractive gradients, even if they are still young. This helps new shorter paths strongly compete with each other, which again enables a deeper exploration of the graph.

The previous considerations are common to all three problems, though the differences of RFD and ACO is affected by an additional factor in the cases of `MSV` and `MDV`. As we said before, when RFD solves `TSP`, drops keep a memory of previously traversed nodes, though no memory is used in RFD when `MSV` or `MDV` are solved. This contrasts with ACO, where memories are used in the three problems. The implicit avoidance of cycles of RFD allows us not to provide drops with any memory in `MSV` and `MDV`, but ants still need memories in both problems because they can fall in cycles indeed. Though cycles are avoided by formed gradients in RFD, gradients are still weak during the first steps of the algorithm. Thus, drops *do* follow some cycles during these early steps. In the long term, when gradients are stronger, drops avoid cycles without maintaining any memory structure, which boosts their performance with respect to ACO. Thus, the absence of memory in `MSV` and `MDV` also promotes the general characteristic we already pointed out before: Solutions given by ACO are better in the short term, but RFD surpasses solutions given by ACO after some time.

5 Applying RFD and ACO to MDV, MSV, and TSP in Dynamic Graphs

In this section we reconsider the previous problems in a scenario where graphs are *dynamic*, that is, nodes and edges can change along time. In particular, we are interested in studying the capability of each method to *adapt* previous solutions to new configurations after each change is introduced. In these experiments, we follow the following procedure for each of the considered problems. First, we calculate the solution of an instance of the problem by using either ACO or RFD. Next, we introduce one or more of the following changes in the graph: (a) We delete an edge that is common to the solutions found by both methods; (b) we delete a node of the graph; (c) we add a new node. Once the change is introduced, we calculate a solution for this instance of the problem. Let us remark that we do not restart the algorithms, but we keep the state of the methods (that is, the amount of pheromone at each edge and the altitudes of the nodes, respectively) just before the change. In most cases we observe that, after some time, the quality of the solutions of our method surpasses the quality of the solutions provided by ACO. In fact, the time needed to surpass ACO solutions is shorter than in the static case. Moreover, it is specially remarkable that for some graphs ACO does not find a solution at all after changing the graph (but, in all of these cases, RFD does).

5.1 Dynamic MSV

Let us start by considering the `MSV` problem. As we did in the static case, we will compare the performance of RFD and ACO by using the same input graphs. In particular, we consider the same graphs introduced in the static case, and in each experiment we introduce several changes at the same time. In the case of the 100 nodes graph, we remove two nodes, we remove two edges (one of them belongs to the solution found by ACO and another one to the solution found by RFD), and we add other two new nodes. The results can be seen in Figure 10. In the case of the 200 nodes graph (see Figure 11) we proceed in an analogous way: We remove four nodes as well as four edges (where two edges belong to the solution given by ACO and the other two edges belong to the solution provided by RFD), and we add

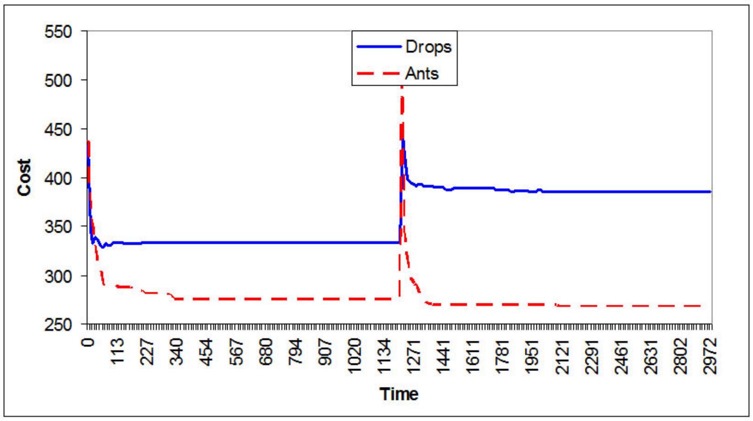


Fig. 10 `MSV` results for a dynamic randomly generated variable-cost graph with 100 nodes

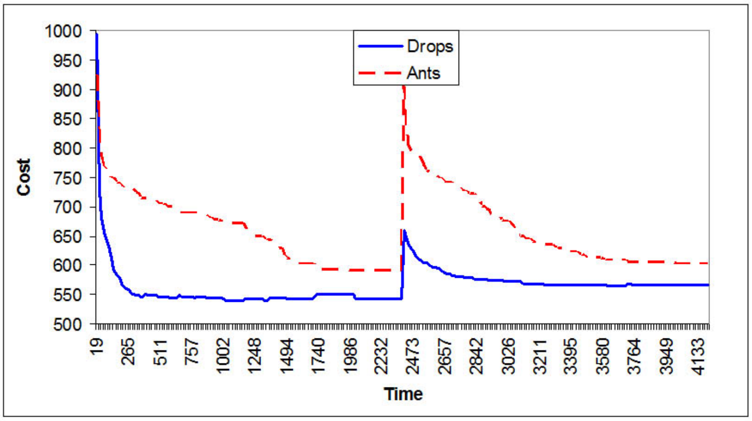


Fig. 11 MSV results for a dynamic randomly generated variable-cost graph with 200 nodes

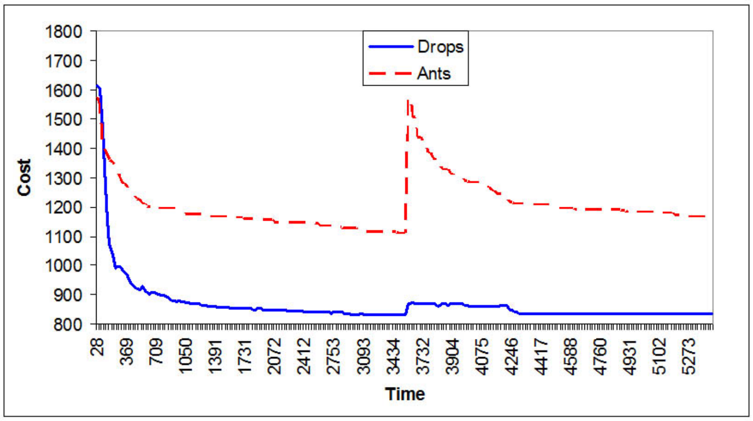


Fig. 12 MSV results for a dynamic randomly generated variable-cost graph with 300 nodes

two new nodes. We proceed analogously in the 300 nodes graph (see Figure 12): six nodes and six edges are removed, and next two new nodes are added.

In all cases we see that the time needed to react to the modifications is similar both in ACO and RFD. However, the results obtained by using RFD are better than those given by ACO in all graphs but the smallest one. Thus, the advantages of RFD in the static case are still valid in the dynamic case, while the main disadvantage of RFD in the static case is minimized in dynamic frameworks.

Table 4 summarizes observed results after executing each algorithm thirty times for each graph.

Let us remark that the results are somehow similar to those obtained in the static case. That is, ACO finds better solutions in the smallest graph, while RFD obtains better results in the larger graphs. Moreover, the time needed to react to the changes in both cases is very similar.

Table 4 Summary of MSV results. Dynamic case.

Graph size	Avg RFD	Avg ACO	Var RFD	Var ACO	Best RFD	Best ACO
100 nodes	383.53	265.08	11.72	14.23	376.45	258.68
200 nodes	559.15	602.90	3.87	172.93	556.94	578.41
300 nodes	835.10	1121.66	1.38	200.71	834.56	1058.86

5.2 *Dynamic MDV*

Next we analyze the performance of RFD and ACO when MDV is solved in environments where the graph changes along time. Again, we consider the same graphs introduced in the static case and, for each experiment, we introduce several changes at the same time. The changes introduced are the same as in the MSV case.

The results can be seen in Figures 13, 14, and 15, where experiments with the graphs of 100, 200, and 300 nodes, respectively, are considered. The situation now is similar to the MSV case: (i) the time needed to react to the modifications is similar both in ACO and in RFD; (ii) the results obtained using RFD are better than in ACO. In fact, we can see that RFD has a bit more advantage over ACO in the dynamic case than the static case.

Following the same methodology as in the previous case, each algorithm was executed thirty times for each of the graphs. Table 5 summarizes the same parameters as in previous tables.

Notice that in the dynamic case the advantage of RFD over ACO has increased with respect to the static case: The average results and the variance are better, and the best results are usually found by RFD. However, the main advantage with respect to the static case is that the time needed by RFD to surpass the quality of the solutions found by ACO is much less than in the static case. In fact, both RFD and ACO require similar times to react to changes.

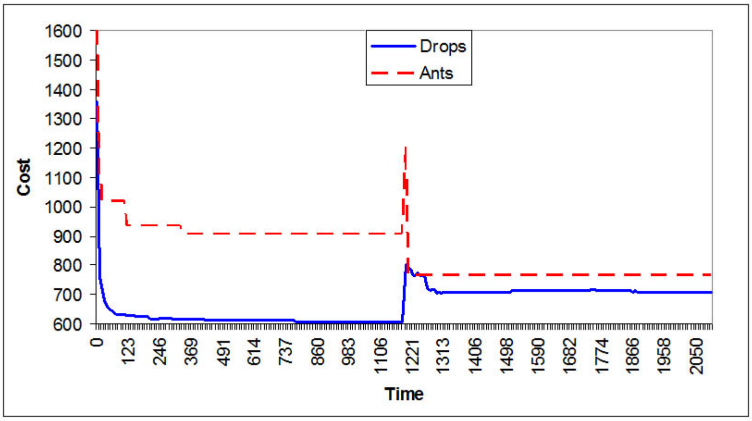


Fig. 13 MDV results for a dynamic randomly generated variable-cost graph with 100 nodes

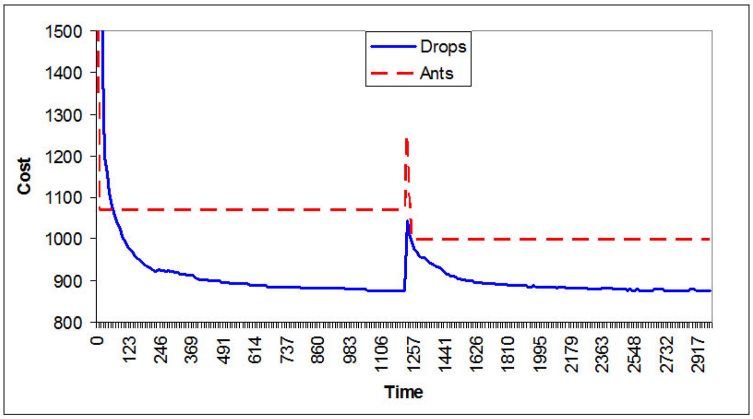


Fig. 14 MDV results for a dynamic randomly generated variable-cost graph with 200 nodes

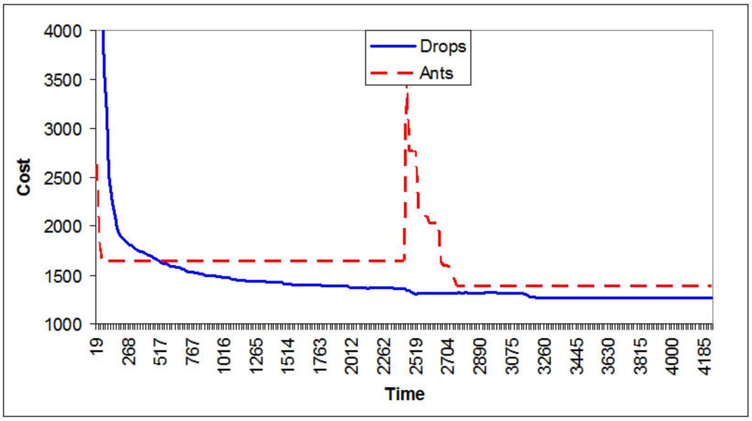


Fig. 15 MDV results for a dynamic randomly generated variable-cost graph with 300 nodes

Table 5 Summary of MDV results. Dynamic case.

Graph size	Avg RFD	Avg ACO	Var RFD	Var ACO	Best RFD	Best ACO
100 nodes	699.49	753.35	65.53	5216.30	681.69	618.25
200 nodes	878.02	1241.58	10.47	18653.47	873.37	1012
300 nodes	1264.53	1473.34	1.18	16106.36	1263	1275

5.3 Dynamic TSP

Next we use ACO and RFD to solve TSP in the case where changes are introduced in the graph. Figure 16 shows the results after removing one edge that was common to solutions found by both ACO and RFD, considering the 100 nodes graph.

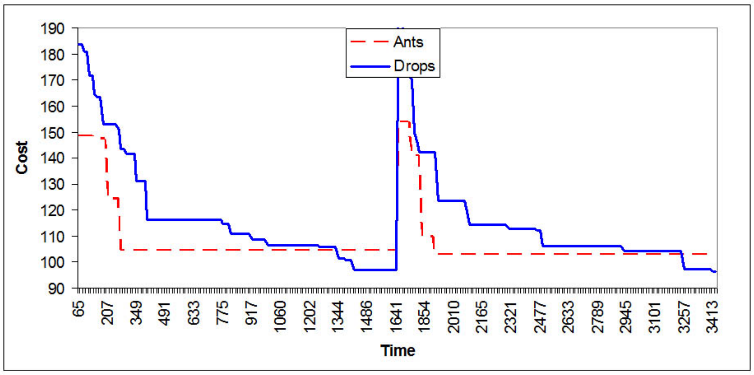


Fig. 16 `TSP` results for the 100 nodes graph, after removing an edge

Next, Figure 17 shows the results after removing one node in the same graph, and Figure 18 considers the case where several changes (in particular, deleting two common edges, adding two new edges, adding two new nodes, and deleting other two nodes) are introduced at the same time in the same graph. Next, Figure 19 shows the results after introducing a new node in the TSPLIB `eil51` graph. Finally, Figure 20 shows the results after introducing several changes (in particular, the same as in the case of the 100-nodes graph) at the same time in `eil51`.

Let us remark that, in one of these experiments, ACO did not find any solution to the problem after several modifications were introduced. That is, ACO did not find a way to either integrate the modifications within its previous solution or compose a completely new solution where the modifications were considered. However, RFD obtains a solution. Hence, in this case the advantage of RFD over ACO is quite relevant. Obviously, we could completely restart the computation of ACO from the beginning. This is a good choice indeed, because the time needed by ACO to provide reasonable solutions when executed from scratch is actually shorter. However, this

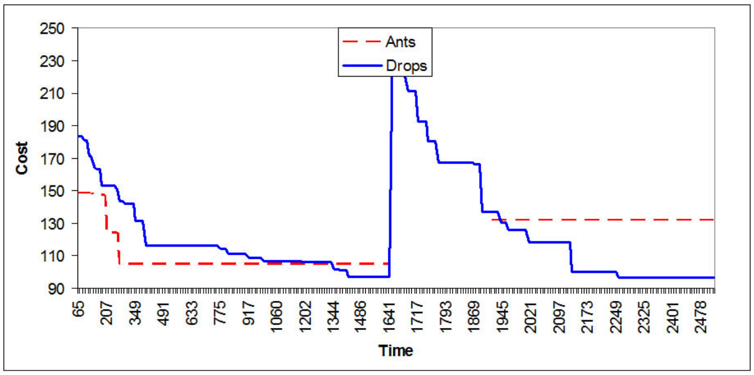


Fig. 17 `TSP` results for the 100 nodes graph, after removing a node

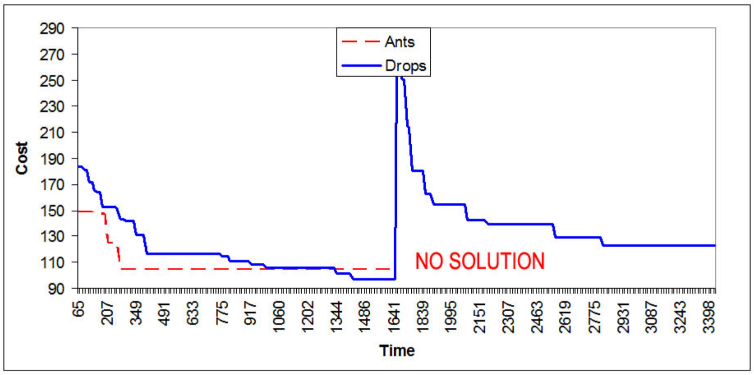


Fig. 18 `TSP` results for the 100 nodes graph, after performing several modifications together

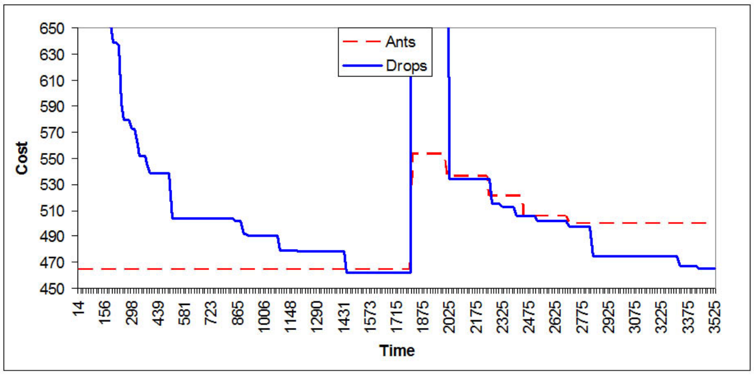


Fig. 19 `TSP` results for the TSPLIB `eil51` graph, after adding a node

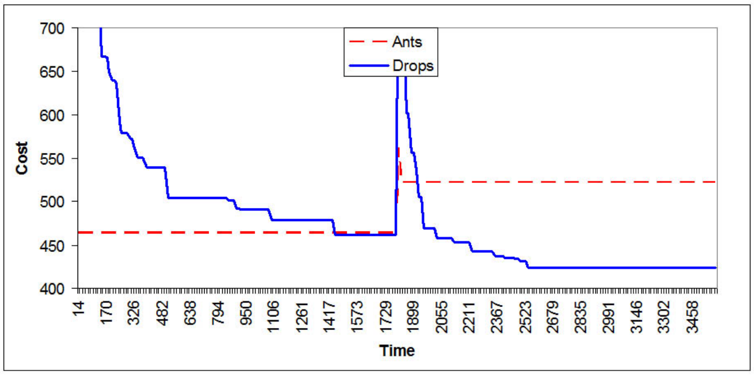


Fig. 20 `TSP` results for the TSPLIB `eil51` graph, after performing several modifications together

shows that the adaptability of ACO to changing landscapes (which is the point of considering the dynamic case) is worse than the adaptability of RFD.

Regarding the cases where ACO finds a solution after the modifications, we observe that ACO finds them faster than RFD, but RFD finds better solutions after some time. Moreover, it is worth pointing out that, in general, the time needed by RFD to surpass ACO is smaller after a modification is introduced than in the case where we compute the initial solution from scratch. However, the advantage of RFD over ACO in terms of quality of results was clearer in `MSV` and `MDV`. The reason in that `TSP` is harder for RFD to solve than `MSV` and `MDV`, as now drops need to keep memories of the path traversed so far. Anyway, it is remarkable that RFD can always react to changes in the graph, while that is not the case for ACO.

Next we study in higher detail the case where several changes are introduced at the same time, considering both the randomly generated graph of 100 nodes and the `eil51` graph from TSPLIB. Following the same methodology as in previous examples, each algorithm was executed thirty times for each graph. Table 6 summarizes the same parameters as in previous tables. The '–' symbol denotes that no solution was found in this case. Interestingly, ACO did not find any solution for the 100 nodes graph in all the thirty executions.

Table 6 Summary of `TSP` results. Dynamic case.

Graph size	Avg RFD	Avg ACO	Var RFD	Var ACO	Best RFD	Best ACO
eil51	436.40	441.76	57.39	28.63	427.68	436.35
100 nodes	149.86	-	199.49	-	123.11	-

5.4 Summarizing Results

We extract the following general conclusions from experiments where RFD and ACO were applied to solve `MSV`, `MDV` and `TSP` in dynamic graphs:

(1) In both static and dynamic graphs, RFD usually obtains better solutions in the long term;
(2) In dynamic graphs, ACO and RFD require similar time to react to changes, with the only exception of `TSP`, where ACO is faster than RFD.
(3) Once RFD has found a good solution in the static case and when several changes were introduced later in the graph, RFD works faster than in the static case because it can exploit the slopes created before; and
(4) RFD always obtains a solution after a modification is introduced, while sometimes ACO cannot adapt the solution constructed before the graph was modified to the new scenario (in particular `TSP`).

These features are again a consequence of the fact that the exploration of the graph is *deeper* in RFD than in ACO, which in turn is due to the differences between both methods.

The fact that sometimes ACO does not find a new solution for `TSP` after a change is introduced deserves additional comments. First, let us note that finding *any* solution is harder for `TSP` than for `MSV` or `MDV` (in particular, given a graph, there are *less* round-trips than spanning trees). When ACO converges to a solution, pheromone trails not involved in the solution sequence are negligible, i.e. only the edges included in the solution sequence have significant pheromone trails. Hence, no information about other alternatives is available when the graph changes. In particular, resetting missing pheromone trails to some default values does not recover this information. On the other hand, when RFD converges to a solution, all nodes are provided with some altitude, which gives them a relative position in the graph. That is, providing a solution does not require to completely resetting part of the graph information. Thus, this information helps RFD react to subsequent graph changes.

6 Conclusions and Future Work

In this chapter we have studied the application of the River Formation Dynamics approach to three NP-complete problems. One of them, `TSP`, is well-known and has been extensively studied in the literature. The other problems, `MSV` and `MDV`, are novel generalizations of other known tree-construction problems. Let us note that RFD is conceptually related to both ACO methods and other gradient-oriented Evolutionary Computation (EC) approaches. On one hand RFD is, in a rough sense, a gradient-driven variant of ACO. On the other hand, the gradient orientation of RFD reminds of methods like Hill Climbing (HC) or Genetic Algorithms (GA) which traverse a space of solutions by seeking solutions with higher fitness. However, there is a big difference between RFD and these methods. RFD modifies the points of a given structure (a *graph*) by iteratively traversing and transforming these points. In this way, the structure is iteratively transformed as well, and finally the formed structure constitutes the returned solution. In HC and GA, the traversed structure is the *space of solutions* itself, which is of exponential size in general. Hence, only a small proportion of these points can be traversed. In these cases, aiming at iteratively *modifying* this space is unfeasible.

Other features of RFD make it different from other EC approaches, specifically ACO. The main ones are the mechanism of focalized punishment of inefficient paths, the avoidance of local cycles, and the fast reinforcement of shorter paths. These characteristics are a consequence of the natural tendency of RFD to form paths that are intrinsically *directed* towards a given final destination. As commented in previous sections, these features are specially suitable for dealing with `MDV` and `MSV` problems. Interestingly, a simple parameter change allows RFD to solve either of these problems. RFD's intrinsic features also make it an interesting choice for solving `TSP`, though in this case one of the advantages of RFD with respect to ACO is lost (in particular, the possibility of *not* endowing drops with memory to register the path traversed so far).

Since RFD is a young method, there is still enough room for introducing both general improvements and purpose-specific variants. Out of a long list of choices, we are specially interested in simulating the *speed* of the drops. Intuitively, when a drop falls through a strong downward slope its speed increases. This gives the drop some *kinetic energy* allowing it to climb up slopes later. Thus, the speed can be thought as a kind of drop *credit*. In this way, a second derivative mechanism would be introduced in RFD. We are also interested in developing a *hybrid* RFD-ACO system. Altitudes and pheromone trails would be simultaneously represented and considered by *drop-ant* hybrids. The rate of influence of each approach could change along time depending on the suitability of each approach for each execution stage. In this way, we could take the best characteristics of both approaches.

Acknowledgements. The authors would like to thank the anonymous reviewers for valuable suggestions on improving this chapter.

Research partially supported by the MCYT project TIN2006-15578-C02-01, the Junta de Castilla-La Mancha project PAC06-0008-6995, and the Marie Curie project MRTN-CT-2003-505121/TAROT.

Appendix: Proving MDV, MSV $\in$ NP-complete

In this appendix we prove that the novel problems considered in this chapter, `MDV` and `MSV`, belong to the `NP-complete` class. This implies that exponential times are (very probably) required to optimally solve them. Thus, sub-optimally solving them by means of heuristic algorithms like those considered in this chapter is an appropriate choice. The proof is structured as follows. First, we prove that both problems belong to the `NP` class. Next, we prove that a well-known NP-complete problem, `3-SAT`, can be polynomially reduced to each considered problem, which implies that they belong to the `NP-complete` class. At the end of this appendix, we study variants of `MDV` and `MSV` where only graphs fulfilling $N = O$ (that is, graphs where *all* nodes are origin nodes) are considered, showing that these variants are also NP-complete.

Lemma 1. `MDV` $\in$ `NP` and `MSV` $\in$ `NP`.

Proof. We consider `MDV` $\in$ `NP`; proving `MSV` $\in$ `NP` is similar. We prove that `MDV` can be solved in polynomial time by a non-deterministic algorithm. Given a variable-cost graph G and a natural number $K \in \mathbb{N}$, this algorithm non-deterministically constructs a subgraph G' of G and next deterministically checks whether (a) G' is a tree of G, and (b) we have $dc(G') \leq K$. Both operations are performed in polynomial time with respect to the size of G and the size of K (measured in bits). Given a subgraph G' of G, checking whether G' is a tree of G requires polynomial time. Next, if G' is a tree of G, calculating $dc(G')$ requires traversing all paths connecting each origin node to the destination node and adding the costs of all of these paths. The length of each of these paths is polynomial, so calculating the cost of a path requires polynomial time. Since G' is a tree, for each origin node there exists a single path connecting it to the destination node. Thus, the number of paths to be considered is polynomial. Hence, we can check whether the property $dc(G') \leq K$ holds or not in polynomial time. $\square$

In order to prove the NP-completeness of MDV and MSV, we construct a polynomial reduction of a known NP-complete problem to each of these problems. In particular, we consider the well-known 3-SAT problem. Next we introduce some notions related to this problem as well as the problem itself.

Definition 3. The 3-SAT problem is stated as follows: Given a propositional logic formula φ expressed in conjunctive normal form where each disjunctive clause has at most 3 literals, is there any valuation ν satisfying φ?

Let $\varphi \equiv (l_{11} \vee l_{12} \vee l_{13}) \wedge \ldots \wedge (l_{k1} \vee l_{k2} \vee l_{k3})$ be an input for 3-SAT. We denote by $\mathtt{props}(\varphi) = \{p_1, \ldots, p_n\}$ the set of propositional symbols appearing in φ. We denote the i-th disjunctive clause of φ by c_i, that is, $c_i \equiv l_{i1} \vee l_{i2} \vee l_{i3}$.

We say that c_i *holds when* p_j *is equal to* $x \in \{\top, \bot\}$, formally denoted by $\mathtt{h}(p_j, x, c_i)$, if for all valuation ν fulfilling $\nu(p_j) = x$ we have that c_i evaluates to $\top$. That is, $\mathtt{h}(p_j, \top, c_i)$ iff $l_{im} \equiv p_j$ for some $1 \leq m \leq 3$, and $\mathtt{h}(p_j, \bot, c_i)$ iff $l_{im} \equiv \neg p_j$ for some $1 \leq m \leq 3$. □

Theorem 1. 3-SAT $\in$ NP-complete. □

We prove MDV, MSV $\in$ NP-complete as follows (next we consider MDV; the same arguments are given in the case of MSV). Given an input φ of 3-SAT, we show that we can construct an input (G, K) of MDV from φ in polynomial time in such a way that the solution of 3-SAT for φ is *yes* iff the solution of MDV for the variable-cost graph G and the natural number K is *yes*. By the definition of the NP-complete class, this implies MDV $\in$ NP-complete. In particular, if we were able to solve MDV in polynomial time then we could solve the NP-complete problem 3-SAT in polynomial time as well: We could just transform φ into (G, K), next call the algorithm that solves MDV in polynomial time, and finally return the answer given by it.

Before formally presenting the construction of (G, K) from φ, let us informally introduce it. Each origin node of the constructed graph G represents a *disjunctive clause* of φ. From each of these origin nodes, edges iteratively lead through some nodes representing each *proposition symbol* appearing in φ. Each of these proposition nodes is connected to the next proposition node through two edges. One of them represents valuating the corresponding proposition symbol to $\top$, while the other edge represents giving it the $\bot$ value. Depending on the origin node where we come from (that is, depending on the disjunctive clause we are considering), taking the edge that evaluates the proposition symbol to true or to false adds a different cost to the path. This cost is 1 *unless* the proposition valuation represented by the edge allows to make true the disjunctive clause *for the first time* in the path. In this case, the edge adds 0 to the overall path cost. In order to keep track of this information, the value of the *variable* of the variable-cost graph G codifies the considered clause, as well as whether this clause necessarily holds (according to the valuation represented by the path traversed so far). In particular, variable values follow the form $v_{j,w}$ where j is an index denoting a clause and $w \in \{already\top, notyet\top\}$. A value $v_{j,w}$ denotes that the current path departed at an origin node denoting the j-th clause of φ, and $w = already\top$ denotes that the j-th clause must be true regardless of the

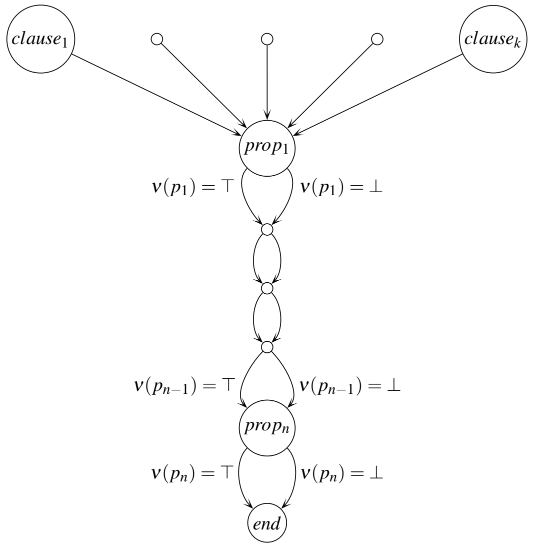


Fig. 21 Structure of the variable-cost graph G

valuation of the remaining proposition symbols (because the valuation implicitly defined by the path traversed so far necessarily makes it true). Otherwise, we consider $w = notyet\top$. After the last proposition node is traversed, the destination node of G is reached. The structure of G is depicted in Figure 21.

Recall that MDV seeks a tree where the addition of costs from each origin node to the destination node is minimal. On the other hand, MSV seeks a tree where the addition of average edge costs is minimal. Let us note that, given the variable-cost graph G, a tree of G can include only *one* of the edges that connect each proposition node to the next proposition node (otherwise, it would not be a tree). Hence, given G, trees computed by both problems represent valuations of proposition symbols. Since MDV searches the cheapest tree connecting all origin nodes to the destination node, MDV actually seeks a tree that allows making as many clauses true as possible. In particular, we will prove that the cost of the cheapest tree found by MDV is under a given threshold if and only if *all* clauses are true under the constructed valuation, that is, iff φ holds. Moreover, due to the specific form of G, finding a tree where the addition of costs from each origin to the destination is minimal is equivalent to finding a tree where the addition of *average* costs of edges is minimal: Due to the definition of G, for both problems the tree cost is minimized if the valuation represented by the tree makes true as many clauses as possible. Hence, we can also define a threshold such that the cost of the minimum tree for MSV is under it iff φ is satisfiable.

Theorem 2. MDV $\in$ NP-complete and MSV $\in$ NP-complete.

Proof. First, we prove MDV $\in$ NP-complete. Due to Lemma 1, if we polynomially reduce 3-SAT to MDV then MDV $\in$ NP-complete. Let φ denote a conjunctive normal form $\varphi \equiv c_1 \wedge \ldots \wedge c_k$ where $\texttt{props}(\varphi) = \{p_1, \ldots, p_n\}$. We construct a variable-cost graph $G = (N, O, d, V, A, E)$ as follows:

- $N = \{clause_1, \ldots, clause_k, prop_1, \ldots, prop_n, end\}$,
- $O = \{clause_1, \ldots, clause_k\}$,
- $d = end$,
- $V = \{v_{j,w} | 1 \leq j \leq k \wedge w \in \{already\top, notyet\top\}\}$
- For all $clause_i \in O$ we have $A(clause_i) = v_{i,notyet\top}$.
- $E = \{(clause_i, prop_1, C, T) | 1 \leq i \leq k \wedge \forall\, v \in V : (C(v) = 0 \wedge T(v) = v)\}$

$$\bigcup$$

$$\left\{ \begin{pmatrix} prop_i, \\ prop_{i+1}, \\ C_i^x, \\ T_i^x \end{pmatrix} \middle| \begin{array}{l} 1 \leq i \leq k-1 \wedge x \in \{\top, \bot\} \wedge \\ C_i^x(v_{j,notyet\top}) = \left\{ \begin{array}{l} 0 \texttt{ if } \mathrm{h}(p_i, x, c_j) \\ 1 \texttt{ otherwise} \end{array} \right\} \wedge \\ C_i^x(v_{j,already\top}) = 1 \wedge \\ T_i^x(v_{j,notyet\top}) = \left\{ \begin{array}{ll} v_{j,already\top} & \texttt{if } \mathrm{h}(p_i, x, c_j) \\ v_{j,notyet\top} & \texttt{otherwise} \end{array} \right\} \wedge \\ T_i^x(v_{j,already\top}) = v_{j,already\top} \end{array} \right\}$$

$$\bigcup$$

$$\left\{ \begin{pmatrix} prop_n, \\ end, \\ C_n^x, \\ T_n^x \end{pmatrix} \middle| \begin{array}{l} x \in \{\top, \bot\} \wedge \\ C_n^x(v_{j,notyet\top}) = \left\{ \begin{array}{l} 0 \texttt{ if } \mathrm{h}(p_n, x, c_j) \\ 1 \texttt{ otherwise} \end{array} \right\} \wedge \\ C_n^x(v_{j,already\top}) = 1 \wedge \\ T_n^x(v_{j,notyet\top}) = \left\{ \begin{array}{ll} v_{j,already\top} & \texttt{if } \mathrm{h}(p_n, x, c_j) \\ v_{j,notyet\top} & \texttt{otherwise} \end{array} \right\} \wedge \\ T_n^x(v_{j,already\top}) = v_{j,already\top} \end{array} \right\}$$

We show that constructing G from φ requires polynomial time. This property is a consequence of the following conditions:

(a) $|N|$ is equal to the number of clauses of φ plus the number of proposition symbols of φ plus 1 (the *end* node), which is polynomial with respect to the size of φ.

(b) $|V|$ is equal to the number of disjunctive clauses of φ multiplied by 2. Thus, for each edge in E, defining functions C and T by means of extensional arrays (relating each input value with its output value) requires polynomial size and time.

(c) $|E|$ is equal to the number of clauses plus the number of propositions multiplied by 2, which is polynomial with respect to the size of φ.

Finally, we prove that the *answer* of MDV for G and a given threshold is *yes* iff φ is satisfiable. In particular, we prove that φ is satisfiable iff there exists a tree G' of G such that $dc(G') \leq k * (n-1)$. We consider each implication of this statement:

$\Rightarrow$: Let us note that a tree G' of G must include all edges connecting each node $clause_i$ with $prop_1$. All of these edges have 0 cost. Besides, for each pair of edges connecting each node $prop_i$ with node $prop_{i+1}$, the tree G' must include exactly one of these edges. Let us consider a valuation ν such that for all $1 \leq i \leq n$ we have $\nu(p_i) = \top$ if G' includes the edge $(prop_i, prop_{i+1}, C_i^\top, T_i^\top)$ and $p_i = \bot$

if G' includes $(prop_i, prop_{i+1}, C_i^{\perp}, T_i^{\perp})$. For all $clause_i \in O$, the cost of the path from $clause_i$ to end in G' is $n-1$ if v makes c_i true, and n otherwise. This is because if v makes c_i true then all edges in the path but *one* add 1 cost to this path. The exception is the edge that makes c_i true for the first time, which adds 0 cost. If φ is satisfiable then there exists a valuation v' making *all* clauses c_i true. Thus, there exists a way to choose the edges connecting each $prop_i$ with $prop_{i+1}$ in such a way that, for all $clause_i$, the unique path from $clause_i$ to end has $n-1$ cost. In this case, $dc(G') = k*(n-1)$.

$\Leftarrow$:Let us consider a valuation v defined as in the previous case. If the cost of G' is $k*(n-1)$ then the cost from each $clause_i$ to end must be $n-1$. This implies that, for each $1 \leq i \leq n$, v makes the clause c_i true. Hence, φ is satisfiable.

We prove $\texttt{MSV} \in \texttt{NP-complete}$ by following very similar arguments. In particular, we can construct a polynomial reduction of `3-SAT` to `MSV` by using the same variable-cost graph G defined before. In this case, we argue that φ is satisfiable iff there exists a tree G' of G such that $sc(G') \leq n-1$. Let us recall that, in `MSV`, the cost of each edge e of G' is the average cost of e for all paths traversing e. Due to the structure of G, it is easy to check that $sc(G') = \frac{dc(G')}{k}$. Thus, φ is satisfiable iff $sc(G') = \frac{k*(n-1)}{k} = n-1$. □

It is worth pointing out that the goal of the previous construction is proving the NP-completeness of `MDV` and `MSV`, not providing a suitable graph construction to solve `3-SAT` by means of RFD or ACO. In particular, if the variable-cost graph G were used to find solutions to `3-SAT` by means of RFD, then we would need to introduce a *barrier node* at each edge connecting a node $prop_i$ with $prop_{i+1}$ (see details about barrier nodes in [17]).

Finally, we study the relation of `MSV` with other NP-complete problems. Let us note that `MSV` is a generalization of the *Minimum Steiner Tree* problem. This NP-complete problem is stated as follows: Given a (non-variable) cost-evaluated graph G and a subset S of its nodes, find the minimum spanning tree including (at least) all nodes in S. `MSV` generalizes this problem by considering *variable-cost* graphs instead of fixed-cost graphs (in particular, the set of origin nodes O in `MSV` corresponds to the set S given in the previous problem definition). Thus, we may argue that the NP-completeness of `MSV` is a consequence of the NP-completeness of the Minimum Steiner Tree problem. However, the NP-completeness of `MSV` (as well as the NP-completeness of `MDV`) does not lie *only* in the fact that a given *subset* of nodes is required to be included in the tree. In fact, even if only trees including *all* nodes are considered, the NP-completeness of `MDV` and `MSV` is met – though, in this case, the Minimum Steiner Tree problem would *not* be NP-complete, because it would be equivalent to the (standard) Minimum Spanning Tree problem (which can be polynomially solved).

Let $\texttt{MSV}'$ and $\texttt{MDV}'$ be problems defined as `MDV` and `MSV`, respectively, but with the following difference: Only graphs fulfilling $N = O$ (that is, graphs where all nodes are origin nodes) are considered. We prove $\texttt{MSV}', \texttt{MDV}' \in \texttt{NP-complete}$ by using very similar arguments as before. In particular, let us consider the same construction as in the proof of Theorem 2, but now nodes $prop_1,\ldots,prop_n,end$ are also included in the set of origin nodes O. A new variable value *nullCost* $\in V$ is assigned, as initial value, to all of these new nodes, that is, $A(a) = nullCost$ for all $a \in \{prop_1,\ldots,prop_n,end\}$. If *nullCost* is the current variable value, then taking the next edge is costless and the value *nullCost* remains after taking any edge. Formally, for all transition $(n_1,n_2,C,T) \in E$ we have $C(nullCost) = 0$ and $T(nullCost) = nullCost$. For the rest of variable values, the behavior of edges remains exactly as defined in the proof of Theorem 2. By using the same arguments as those given in that proof, we infer $\mathtt{MSV}', \mathtt{MDV}' \in \mathtt{NP\text{-}complete}$. Thus, the presence of *variable-cost* edges is a sufficient condition for the NP-completeness of both problems.

References

1. Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton (2006)
2. Brinksma, E., Tretmans, J.: Testing transition systems: An annotated bibliography. In: Cassez, F., Jard, C., Rozoy, B., Dermot, M. (eds.) MOVEP 2000. LNCS, vol. 2067, pp. 187–195. Springer, Heidelberg (2001)
3. Bui, T., Zrncic, C.: An ant-based algorithm for finding degree-constrained minimum spanning tree. In: GECCO, pp. 11–18. ACM Press, New York (2006)
4. Davis, L., et al.: Handbook of genetic algorithms. Van Nostrand Reinhold New York (1991)
5. Dorigo, M.: Ant Colony Optimization. MIT Press, Cambridge (2004)
6. Dorigo, M., Gambardella, L.: Ant colonies for the traveling salesman problem. BioSystems 43(2), 73–81 (1997)
7. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man and Cybernetics, Part B 26(1), 29–41 (1996)
8. Fleischer, M.: Simulated annealing: past, present, and future. In: Proceedings of the 27th conference on Winter simulation, pp. 155–161 (1995)
9. Gutin, G., Punnen, A.: The Traveling Salesman Problem and Its Variations. Kluwer, Dordrecht (2002)
10. Jong, K.: Evolutionary computation: a unified approach. MIT Press, Cambridge (2006)
11. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942–1948 (1995)
12. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220, 671–680 (1983)
13. Langton, C.: Studying artificial life with cellular automata. Physica D 22, 120–149 (1986)
14. Lee, D., Yannakakis, M.: Principles and methods of testing finite state machines: A survey. Proceedings of the IEEE 84(8), 1090–1123 (1996)
15. López, N., Núñez, M., Rodríguez, I.: Specification, testing and implementation relations for symbolic-probabilistic systems. Theoretical Computer Science 353(1–3), 228–248 (2006)
16. Petrenko, A.: Fault model-driven test derivation from finite state models: Annotated bibliography. In: Cassez, F., Jard, C., Rozoy, B., Dermot, M. (eds.) MOVEP 2000. LNCS, vol. 2067, pp. 196–205. Springer, Heidelberg (2001)

17. Rabanal, P., Rodríguez, I., Rubio, F.: Using river formation dynamics to design heuristic algorithms. In: Akl, S.G., Calude, C.S., Dinneen, M.J., Rozenberg, G., Wareham, H.T. (eds.) UC 2007. LNCS, vol. 4618, pp. 163–177. Springer, Heidelberg (2007)
18. Rabanal, P., Rodríguez, I., Rubio, F.: Finding minimum spanning/distances trees by using river formation dynamics. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.) ANTS 2008. LNCS, vol. 5217, pp. 60–71. Springer, Heidelberg (2008)
19. Reinelt, G.: TSPLIB 95. Research Report, Institut für Angewandte Mathematik, Universität Heidelberg, Heidelberg, Germany. Tech. rep. (1995), `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`
20. Rodríguez, I., Merayo, M., Núñez, M.: HOTL: Hypotheses and observations testing logic. Journal of Logic and Algebraic Programming 74, 57–93 (2008)

Algorithms Inspired in Social Phenomena

Antonio Neme and Sergio Hernández

Abstract. Natural computing finds its source of inspiration in diverse biological phenomena and social behaviors from mainly insects and birds. In this chapter, we instead propose human social phenomena. The presented algorithms have been applied in optimization endeavours with success or are promising tools in the design of optimization techniques.

1 Introduction

Computer Science has turned its attention to a wide variety of natural phenomena with the aim of abstracting new optimization algorithms. Such phenomena can either be physical, biological or even social processes, like simulated annealing, genetic algorithms and interactions between large communities of insects respectively.

Social phenomena have been extensively studied as a basis for several algorithms, many of them in the area of optimization. The great majority of these social phenomena come from the social behavior shown by ants, termites and some other insects, as well as from bird flocks and fish schools [9]. Most of the social phenomena that have been used as inspiration for optimization algorithms are mainly cases of collaboration between organisms, and are referred to as swarm intelligence [10].

The relevance, complexity and computational power of optimization algorithms taken from social phenomena in species other than humans (swarm intelligence) is enormous and can be traced elsewhere, as, for example, in [15, 19, 64]. In this

Antonio Neme
Non-lineal Dynamics and Complex Systems Group,
Universidad Autónoma de la Ciudad de México
e-mail: antonio.neme@uacm.edu.mx

Sergio Hernández
Posgraduate Program in Non-linear Dynamics and Complex Systems,
Universidad Autónoma de la Ciudad de México
e-mail: cheko@ciencias.unam.mx

chapter we discuss social phenomena in human societies that have been or may be the foundation for optimization algorithms.

2 Social Phenomena

Since their early stages, human societies have proven to be very successful in several, if not all, aspects of technological development [6]. People have been optimizing every aspect of daily life, from water supply for agriculture to machine design and transportation, to name just a few. Many of these processes are the result of a group of people explicitly working on them. In this sense, a group of say, engineers working toward optimizing a mechanical engine is a social phenomenon. On the other hand, some social phenomena not explicitly intended to reach a certain goal could also be the source of inspiration for optimizing algorithms. We are interested in those social phenomena that are the result of a non-conscious procedure, that is, social phenomena in which there is not an explicit agreement between individuals to perform a given optimizing algorithm.

The study of social phenomena has been dictated by the cultural influence of their time. Human behavior and social phenomena were analyzed at the light of pendulums, steam engines, computers and so on [6]. The complex systems theory establishes that the behavior of a system is not accessible from the study of its separated components [11]. To fully understand the richness of complex systems, it is mandatory to explicitly define the relations that are observed between the components. Almost all these relations are non-linear and in many cases conflicts among components are present.

Modern social sciences have recently incorporated the multiagent-based modeling, which has shed new light on some unclear aspects in social behavior [11]. In multiagent models, actors of the studied process are abstracted into a group of agents whose behavior is guided by a set of rules that represents the relationship observed among the actors. There is a two-way communication bridge between social and computer sciences, as social phenomena have been the basis for several algorithms [15, 19].

The main idea behind Algorithms Inspired in Social Phenomena is that the computational power of the inspired algorithms is correlated to the richness and complexity of the social behavior. Social phenomena arise as the result of interaction among individuals. These interactions may be non-linear and the number of interacting agents is required to surpass a given threshold. In general, the agents may not be aware of the state of all other agents.

By optimization we refer to finding a minimum or a maximum of a given function. It is defined over a triplet (S,Ω,f) in which S is the search space defined by a set of variables, Ω is the set of constraints over the variables and f is the objective function to be optimized.

The objective function to minimize (maximize) is

$$f(x) \tag{1}$$

subject to constraints $\Omega = g_i \cup h_j$, of which there are k inequalities:

$$g_i(x) \geq 0, i = 1, ..., k \tag{2}$$

and r equalities:

$$h_j(x) = 0, j = 1, ..., r \tag{3}$$

In this chapter we present social phenomena that are present mainly in human societies and that have motivated several optimization algorithms. We also refer to some social processes whose metaphor may lead to new algorithms. The hypothesis is that some of these phenomena, the ones with high complexity, have more computational power than other, less complex phenomena.

Human interactions cannot be explained only by means of biological information, as a great variety of social phenomena are present in societies [35]. Human social phenomena tend to show a high level of diversity. Even if there is not an optimizing principle behind social interactions they lead to robust social structures that may even present a high level of stability. The former features make social phenomena a valuable source of inspiration for algorithms.

In the following sections, the human social phenomena we present are leadership and influence from prominent counterparts, alliance formation, neighborhood segregation, and social labeling of individuals.

3 Leadership

Leaders have been important for societies since the dawn of humankind. In the early ages, wizards and magicians, followed by feudal lords and religious officers, and now presidents or football stars, determine many of the individual behaviors and, in some sense, are the idols to whom many people are attracted [34]. Leaders have been identified as outstanding members of a society and they tend to do better than the rest. Leaders influence their counterparts who are drawn toward the former's position, circumstance that may be seen as a searching strategy.

3.1 Society Civilization Algorithm

Several algorithms have been inspired by the leadership phenomena. For example in [43], an optimization algorithm is defined in terms of leader guidance, the so called *society civilization algorithm* (SCA). In this model search space is explored by candidate solutions which at the beginning are distributed in clusters that resemble societies. The best solutions in each society bias the search toward them, that is, the leaders influence their counterparts to follow them. Some leaders may migrate to other regions and thus, their counterparts in former societies migrate with them. The set of all societies is defined as the civilization, in which all individuals (solutions) may interact by means of their leaders. The general algorithm of these schemes is summarized as:

```
1. t ← 0.

2. Generate the civilization $C(t)$ of $N$ individuals: $C(t) = I_1,...,I_N$ uniformly distributed in the parameter space.

3. Evaluate individuals by computing the objective function as well as constraints.

4. Construct $S(t)$ societies from $C(t)$ as clusters. Clusters may be formed by any cluster analysis technique.

5. Identify leaders in each society.

6. Migrate individuals in each society toward the location of its nearest leaders.

7. Identify leaders in the civilization $C(t)$ from the leaders of the societies $S(t)$.

8. Migrate society leaders toward the location of civilization leaders.

9. t ← t + 1.

10. If the stop condition is not met, go to step 3.
```

The social phenomena that inspired SCA are i) migration and ii) leadership, but also iii) cooperation: leaders share their knowledge, that is their position in search space, to the rest of individuals. From these features, SCA algorithms have been applied in several areas. For example, in [47], a variant of SCA is applied to optimize the economic dispatch with multiple minima, a well-known problem in electric power systems operation, and results are promising as performance is comparable to those from mathematical programming, with less computational effort. In [58], a variation of the particle swarm optimization model that incorporates the concept of leadership by allowing particles to move toward the location of the best evaluated particle lead to good results in several benchmarks, while reducing the number of collisions between particles.

In the context of data analysis, several techniques may be classified as leader-guided, as that of self-organizing maps [31], in which the so-called best matching unit attracts toward them the weight vectors of their neighbor units. An explicit leader-guided algorithm has been proposed in [56]. The proposed algorithm obtains a hierarchical structure from data in which each leader is the centroid of a cluster and there are one or more subleaders within that cluster.

3.2 Cultural Algorithms

A unique aspect that characterizes all human societies is the concept of culture. Culture is based on learning from experienced individuals and in that sense, there is also a guidance (leadership) from them to the rest of the individuals. Reynolds [44] introduced a type of algorithm based on how cultures gather information to solve the problems presented to them; these are called cultural algorithms (CA). Such algorithms are a vehicle for modeling social evolution and learning. The intention is to abstract the necessary knowledge from experiences needed to solve some specific problem into, as Reynolds calls it, a space belief which can vary among normative, spatial, temporal domain and exemplar knowledge.

The main idea behind CA is to divide the process of learning and information retrieving into three phases. First, a coarse-grained phase is established which is expected to grasp a general idea of the problem in order to identify regions to explore. Then, a fine grained phase and finally a phase that comes into action when the search process gets stagnated.

The approach to problem optimization inspired by cultural algorithms is an abstraction of how cultures learn to solve their problems by means of a space belief and the proper use of this space belief in order to retrieve useful facts related to the problem at hand. The intention is to abstract a methodology used by humans to optimize their solutions and this can only be done when the variables and the problem domain are fully understood.

Cultural algorithms are inspired by human behavior. They were proposed to incorporate some statements about cultural change. These determine a belief space, which, together with the population space, constrains the search space. The cultural algorithm is also guided by an influence scheme that prevents some individuals from modifying the belief space, while allowing others to do so. In this sense there is a desired, although unknown, behavior that is represented as a location that should be reached by individuals.

Cultural algorithms have been applied in multiobjective optimization, as for example, that of [16, 55], in which a CA is combined with evolutionary programming (CAEP) that achieves outstanding performance. The CAEP is sketched as:

```
1. Generate $k$ individuals that are the initial population.
2. Evaluate the initial population.
3. Initialize the belief space.
4. While stop conditions are not met:
5.     Apply mutation to generate $p$ offspring.
6.     Evaluate each offspring.
```

7. Obtain the relative performance of each solution by means of random mutations.
8. Select the q individuals with the largest number of victories to produce the new generation.
9. Add the non-dominated individuals to an external memory.
10. Modify the belief space with individuals in external memory.

It is important to realize that the cited algorithms seek to improve the performance of individuals in optimization problems. This is achieved by means of learning from the ”good” individuals. These algorithms complement some other convergence-oriented models that try to avoid unfeasible regions of the search space. In that sense, there is also a guide from the good individuals or leaders to the rest of the population.

In leader-based algorithms, such as SCA and CA, an important assumption is made about individuals: their inability to consider self experiences, that is, they are guided toward the leaders’ location without relying on their own previous experiences (locations). Apart from leadership, migration toward leaders and cooperation from leaders to the rest of individuals, free-will has also been included in some algorithms, as that in [17], in which there is also a liberty parameter that allows or stops the individuals migrating to the region occupied by the leader. From this liberty of choosing the desired region to move to, a different set of algorithms have been proposed: the leaderless algorithms.

3.3 Leaderless Algorithms

In Leaderless Algorithms (LA) there are not privileged individuals that bias the direction of searching nor attract others toward them. In LA all individuals have a lack of global information and tend to present the same capabilities while communication is a necessary condition for the algorithm to be successful. In [18], a parallel searching algorithm is presented based on homogeneous agents that are able to construct a map of undirected graphs without a supervisor. The algorithm is a distributed version of the standard depth-first algorithm, but as there are several anonymous agents, they are not able to distinguish between nodes visited by themselves and nodes visited by others. Each agent generates a partial map, T_i, and the complete map of the graph is obtained by a spanning algorithm executed by the first agent that finishes its exploration.

In [48], a group of moving agents reaches a stable velocity as a consensus among all agents. No agent’s preferences are identified as the desired option and in a scenario of unbounded time, consensus is always reached. Agent’s individual performance cost is minimized and constrained to partial information and by doing so, the overall cost of reaching a stable and safe velocity is minimized.

LA has a strong inspiration in the family of algorithms known as swarm intelligence [10], but also in some human behaviors such as those observed in certain business and corporative firms in which duties and privileges are equally shared, as described in [40].

Leaders tend to group individuals toward them. Grouping is also a complex behavior in human societies and has been a source of inspiration for several algorithms. A leader may form a group, but leadership is not the only factor that causes group formation, as will be shown in the next section.

4 Alliance Formation Basis

An alliance, also known as a coalition, is relatively stable group of agents. Agents may be people, political parties, software programs, robots or firms that work toward a common goal that in general cannot be achieved by the agents on their own.

Humans and groups of humans tend to form alliances in several contexts. For example, in democratic regimes with multi-parties such as in Europe and Latin America, alliance formation is a very common practice. An alliance, or coalition, is a set of agents (humans or groups) that may have a similar set of features and a common goal, that in general contrast with goals and features of a rival alliance [21]. Agents may seek their own good and form alliance only by their selfish interests but alliance formation is only profitable when all agents gain more by being part of the group than interacting on their own. An agent might join one alliance not only because it is wealthy but also because by making the alliance wealthier, it increases the chances to defeat the other one in which rival agents may be clustered.

4.1 Basic Definitions

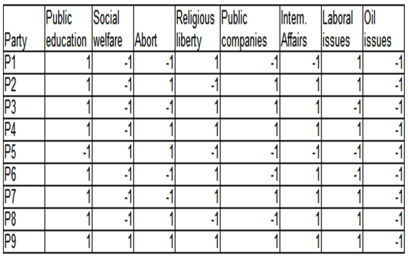
Coalition formation has been studied extensively in game theory mainly with the unfeasible condition that all participants possess full information about the features and preferences of each agent. From the multi-agent community, several proposals have been made with more feasible constraints, such as that of a lack of complete information about the rest of participants, although not all algorithms guarantee stable coalition formation. In [3], an algorithm that leads to stable alliances even with incomplete information is proposed, and since then, several algorithms with more constraints have been studied.

An alliance is goal-directed and in general short-lived [27]. Human alliances may also be very stable at short times, but very unstable after a critical time which translates into rifts and reconfigurations [22]. Alliance or coalition formation has been extensively studied in Computer Science, as there are many applications in which several agents work toward a common goal [42].

In general, the problem of alliance formation can be stated as finding the best partition such that all agents belong to one and only one coalition and at the same time a set of constraints is satisfied. The constraints are derived from the preferences of agents to interact with specific agents, or with agents that present specific features, while avoiding being in the same partition with agents that show some non-desirable features. Agents have features that describe them and allow the interaction with others. In the political scene, an agent, i.e. a party, may have a certain position about public education, public welfare, and science and research policies that determine those parties to whom it may ally with.

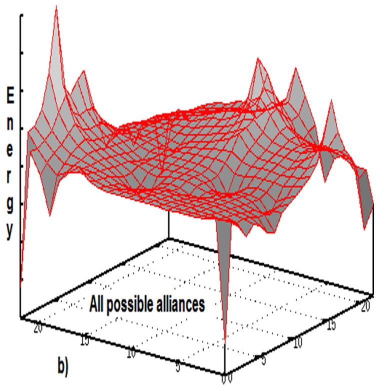
In the traditional alliance formation problem, there are only two alliances, which lead to a total of 2 possible alliance ensembles. Each point in the alliance space has a given energy, or feasibility. Some ensembles may not be very probable in the sense that some of their members may have very different characteristics. For example, a possible party alliance may include the left-most and the right-most parties, but as they show very different positions on the same issues, this alliance is very energetic, i.e., not feasible.

The space of possible ensembles was studied by Axelrod and Bennett under the name of landscape theory [4]. The landscape defined by all the possible ensembles may have several local minima according to the energy, but also may have a global minimum and thus, the ensemble be optimal. The quantity that is optimized here is the energy. Fig. 1 shows a landscape for a fictitious group of 9 political parties and their positions about certain social aspects. There are 256 possible alliances and in fig. 1-b, the energy for each one of them is shown.



Party	Public education	Social welfare	Abort	Religious liberty	Public companies	Intern. Affairs	Laboral issues	Oil issues
P1	1	-1	-1	1	-1	-1	1	-1
P2	1	-1	1	-1	1	1	1	-1
P3	1	-1	-1	1	1	1	-1	-1
P4	1	-1	1	1	1	1	1	-1
P5	-1	1	1	-1	-1	-1	-1	-1
P6	1	-1	-1	1	-1	1	-1	-1
P7	1	-1	-1	1	1	1	1	-1
P8	1	-1	1	-1	-1	1	1	-1
P9	1	1	1	1	1	1	1	-1

a)



b)

Fig. 1 a) Nine political parties and positions about eight social aspects. b)Landscape for all possible alliances

In mathematical terms, the alliance ensemble problem is stated as follows. Let A be the group of N agents and A_i the description of agent i in terms of its features (vector) and C the optimal alliance ensemble, which consists of at least two disjoint alliances. The ensemble contains alliances C_j, $j > 1$, that is, by definition there is more than one alliance and each agent A_i belongs to one and only one of them.

The optimal alliance ensemble is defined as:

$$C = minarg \sum_{i,k} \delta(A_i, Ak) \tag{4}$$

where $\delta(A_i, A_k)$ are the differences between agent A_i and agent A_j over all the feature vectors, measured as the distance between vectors A_i and A_j.

Each agent is defined through a set of variables and each agent may change one or more of these variables over time. Agents try to interact with each other in order to devise better states, that is, better alliances. From the statistical mechanics point of view, the landscape defined by all the possible ensembles may have several local minima, but also may have a global minimum and thus, the ensemble be optimal.

Axelrod proposed this model, based on spin glass theory [4], as an alternative explanation to alliance formation in European countries in the Second World War. There was a global optimum consisting of two alliances which were very similar to the ones observed during the conflict. Although the landscape model is very interesting from the sociophysics point of view [20, 57], it is not feasible in most human alliances as it requires absolute information access for all agents, being humans, parties, companies or any other social structure. In the great majority of cases, agents are not allowed to be aware of the state and features that describe other agents.

4.2 Algorithms

Several alliance formation algorithms have been proposed in the context of multi-agent systems [49, 29, 50]. The main idea is to maximize the sum of the payoffs to all the alliances by identifying the optimal combination of alliances and the division of agents into these alliances. Each agent has a set of tasks that may or may not be similar to the tasks assigned to other agents. The general alliance formation algorithm can be summarized as:

```
1. Construct a list of possible alliances, $S_i(q)$ of up to $q$ agents.
2. While $S_i(q)$ is not empty, do:
3.     Contact agents $A_j \in S_i(q)$.
4.     Evaluate the benefit of joining $A_j$ in an alliance, subject to
        preferences and constraints.
5.     Extract $q$ from $S_i(q)$ and share subtasks.
6.     If contacted by agent $A_k$, substract $q$ as well as the
        common tasks.
```

Alliance formation algorithms may be classified according to the general assumptions of communication and information access for individual agents. The first class is that of complete information assumptions, which includes dynamic programming algorithms that guarantee to find the optimal alliance, but whose complexity is prohibitive for real applications [62, 42, 30]. In this scheme, the messages that are sent between agents in order to share their preferences could be exponential, although some alternatives have been proposed to reduce complexity. The second class is that of heuristic-based algorithms that do not guarantee finding the optimal, but reach solutions very fast [45, 61].

Coalition formation in human societies does not follow the dynamics specified by many of the known alliance formation algorithms, though alliances still tend to be robust (at least for a short period of time). In human societies, agents are unreliable, do not completely share information, and may even lie in order to obtain better profits [22]. Alliance formation is thus based on a weak communication scheme, i.e., agents do not completely share their information.

As a consequence of the lack of pervasive information, each agent is forced to explore its local environment. Agents, therefore, have to join an alliance based solely on local information and a limited knowledge of the features of other agents in the alliance.

Human alliances are guided by heuristics that do not always lead to robust ensembles, i.e., the alliances may not correspond to a global optimum. However, some human alliances tend to be robust and long lasting [21]. It is of special interest to study the general behavior of agents in this context as a source of inspiration for new algorithms that do not need all the available information.

From Political and Management Sciences as well as from Sociology, there are several successful examples of human alliance formation, at least for the allies' immediate purposes. However, the grounds for those alliances are not always obvious from the analysis of the agents, and, in some cases, the alliance might even appear contradictory in itself. In general, social agents are not homogeneous. They are often expressed with weighted influence of some over others or with non-linear relations among them. For example, agent i may be in the same alliance as agent j, as long as agent k is also part of it. However, agent k may be inclined to form an alliance with agent j as long as agent i is not part of it [21].

In human alliance formation, dynamics are mainly governed by two types of coordination. The first one is a centralized organization where agents communicate with agents from other groups via their group as a whole [42]. The second coordination scheme is decentralized, i.e., there is no group of agents that is pervasive in the environment and therefore, the coordination must be achieved through individual interactions.

Based on the above-mentioned constraints and features of human alliances, several coalition algorithms have been proposed. In [37], the agents do not have knowledge of the complete feature vector of all agents, leading to restricted alliance formation. A set of agents is a restricted alliance when all agents agree to share information within the set, but not with the rest of agents. [38, 41] describe a multiagent system model of humanitarian assistance services in conflict areas, in which agents represent non-government organizations that, in general, have a common objective. The available information is not always reliable and the agents are aware of this fact when they decide whether or not to group with another agent.

The variety of reasons why an individual may have to become part of an alliance have been included in some schemes in order to obtain stable alliances when complete information is not available. For example, in [12] each agent i is provided with a variable that quantifies its strength of character determining its predisposition to form complete new alliances, C_i. A second variable G_i defines its attraction to obtain a profit within that alliance. Finally, a third variable R_i is defined that states its reluctance to abandon its current alliance to join another one.

Agents with a high C_i will send invitations to form new coalitions more often that those with lower predisposition to form new aliances. When an agent i that is part of a coalition A_{act} is invited to join another alliance A_n, its decision is based not only on the benefit x_i^n it will receive but also on the individual parameters that define its *personality*. The gains from both the new and actual coalition, $S(A_n)$ and $S(A_{act})$, are determined by the agent through the benefits it would achieve and from individual preferences:

$$S(A_{act}) = G_i \times \frac{x_i^{act}}{x_i^{act} + x_i^n} + R_i \times (1 - sb_{act}) \tag{5}$$

$$S(A_n) = G_i \times \frac{x_i^n}{x_i^{act} + x_i^n} + R_i \times (1 - sb_n) \tag{6}$$

where sb_{act} is a parameter that summarizes the stability of the coalition that contains agent i, and sb_n is the stability of the new coalition. So, if $S(A_{act}) \leq S(A_n)$, the agent decides to abandon alliance A_{act} and be part of the alliance A_n. When agents are homogeneous ($G_i = R_i = C_i$, $\forall i$), the simulations converge to a unique stable structure, but when the personality is heterogeneous, several coalitions may result. As each agent's decision is influenced not only by the profit it will obtain by joining an alliance but also by its own personality, non-optimal alliances are avoided.

5 Optimization through Social Labeling

Classification of individuals in a society is a common phenomenon. The label or tag assigned to an individual may be the result of prejudices, ignorance or true facts. An individual's tag may influence the way other individuals interact with him/her, by inducing a positive or negative reaction/feeling [26]. Some algorithms have been inspired by such social tags.

Achieving cooperation in Peer to Peer networks (P2P) is not a trivial task. Some peers may decide to stop cooperating once they have obtained the resources they were looking for, as often happens in file-sharing schemes. A P2P network in which the number of attempts made to obtain a given resource for all peers is minimized is highly desirable. In [25], a method is presented that improves cooperation in P2P networks based on the evaluation among peers leading to each individual being assigned a tag. Peers that tend to avoid cooperation are tagged as non-cooperative whereas peers that tend to cooperate are tagged positively. The algorithm is:

```
1. While the number of generations is not met:
2.     for each agent $i$ in the population:
3.         Select a game partner agent $j$ with a similar tag (whenever possible).
4.         Peers $i$ and $j$ interact through their strategies and get payoff.
5.     Reproduce agents proportionally to their payoff.
6.     Mutate tags and strategies of each reproduced agent.
```

Each peer is assigned a strategy that states its behavior for interacting with other agents. This strategy is based on the Prisoner's Dilemma (PD), as proposed in [4]. Two agents (peers) are involved in a situation where they have the option to cooperate (C) with each other or to defect (D). Each agent obtains a payoff as a function of its action and the action of the other peer. The payoff proposed in the PD is as follows:

$$T > R > P > S \tag{7}$$

$$2R > T + S \tag{8}$$

where T is the payoff an agent receives if it defects and the other cooperates, R is the payoff when both agents cooperate, P is the payoff when both of them defect, and S is the payoff an agent receives if it cooperates and the other defects. The second condition is to prevent an agent from alternating between cooperation and defection.

As it is known in game theory, if there is only one encounter between two agents, the best strategy is to defect (D). In the proposed model there is no possibility of confronting the same peer twice, which makes the result surprising: the best strategy, in terms of a cooperative network, is that in which cooperative peers (C) are dominant. In other words, the overall number of attempts for each peer to obtain a given resource is minimized through cooperation.

Optimization is achieved by choosing a more convenient counterpart (step 3 in the previous algorithm) in order to have a significant level of cooperation [25]. When tags are removed, the achieved network is highly inefficient, as the peers do not obtain the desired resource because of the lack of cooperative agents. An alternative is to find an outstanding member within the society that performs better than the rest as in [43].

Another algorithm based on social tags is that presented in [2]. Here, a mining newsgroup algorithm leads to the classification of people in two opposite camps over a discussion issue. The algorithm is based on the assumption that people respond more frequently to messages that contain ideas they do not agree with. Through their responses, people get tagged and this tag is taken into account to define the topology of a bipartite network in which each vertex represents a participiant and edges (E) represent responses between participants.

The algorithm seeks a partition of the vertices into two sets: F and A, one representing participants in favor and the other representing users against the discussion issue. The central hypothesis is: if most edges in the graph associated to the newsgroup represent disagreement then the optimum choice of F and A maximizes $f(F,A)$, which is the cut function: $f(F,A) = |E \cap (F \times A)|$). It is only possible to seek this bipartition when the opinions are tagged (favor, against). The mining algorithm, through tags, obtains better results than those obtained by statistical analysis of texts, which is the most used algorithm.

Tags may lead to a separation of cooperative and non-cooperative peers as a side effect in P2P networks and discrimination against participants in debates. However, the richness of segregation has been explicitly studied in several algorithms that are presented in the next section.

6 Neighborhood Delimitation and Segregation

Although house formation and house-keeping processes exist in termites, ants, birds and other species [14], the complexity and richness of human neighborhood formation and segregation is immense. The genesis, evolution and structure of cities are the result of several factors, some external and some endogenous. Economic rules, political constraints and psychological factors, among many other factors determine the overall distribution of neighbors over a city [32, 7].

Residential segregation and neighborhood delimitation has been widely studied from many perspectives within the social sciences [33, 28, 39]. There are at least two approaches to this process. The first is that of phenomenological analysis, as in the previous cited works, and the second is a constructive approach based on multiagent models. One of the early examples of the later approach is that of [46]. In this model, the consequences of many individual decisions and its counterintuitive foundations in urbanism were presented and the dynamics underlying the city genesis and evolution were subject of formal analysis.

The segregation behavior states that all householders try to live near people very similar to them and avoid householders radically different from them. Each householder is described by a feature vector. In mathematical terms, each householder H_i has a propensity to stay comfortably with his/her neighbors defined by the total differences between H_i and H_k, where k are the neighbors of H_i.

In the Schelling model (SM), each agent occupies a cell in a rectangular lattice. The segregation of dissimilar agents is guided by a simple set of rules. At the beginning, agents are randomly distributed in the lattice and after a few decisions by each agent, a structured (segregated) distribution appears [53]. This process is an example of self-organization, a common feature present in several processes, among them social phenomena. Each agent A_i is subject to the following rules:

1. Compute the fraction of neighbors that are of the same color, that is, that have a similar feature vector, F_i.

2. If the agent is satisfied with its neighbors, then it stays in its actual location: $F : i > T_i$, where T_i is the satisfaction threshold. If this condition is satisfied, then the agent ends.

3. The agent looks for the nearest available location that satisfies its requirements and moves there.

The Schelling model has been widely studied as a theoretical tool to explain segregation phenomena not only in the cities, but also in international conflicts [60]. Besides these applications in social sciences, SM has also been studied in other contexts. It has inspired several algorithms in network routing. For example, in [52] a Schelling-resembling algorithm dynamically modifies the topology of a network of hubs, and in [51], it manages to improve bandwidth in P2P networks.

P2P networks are distributed and thus, present a lack of any central control. All nodes present the same functionality and peers can cooperate and communicate with each other based only on a virtual communication network with the constraint that peers are aware only of their local topology.

In a P2P network the topology is dynamic, i.e., peers in contact may decide to delete the link or peers who were previously not in contact may create a new link. In this kind of network, nodes have a maximum number of connections and when the network grows, some algorithms tend to form hubs, i.e., nodes with a higher number of connections than the rest of the nodes. This may lead to several communication problems. The Schelling abstract algorithm (SAA) is a topology modification scheme that not only prevents the emergence of hubs, but also leads to topologies with high cohesion [54]. Each peer has a desired percentage of neighbors with similar properties, $PNSP_{des}$, that it tries to preserve. The general algorithms is:

```
1. set the $PNSP_{des}$.
2. while true
3.     obtain $PNSP_{act}$
4.     if $PNSP_{act} < PNSP_{des}$
```

```
5.        drop a neighbor with a different property.
6.        search for another peer with more similarity.
```

Peers are described in terms of their number of connections, their bandwidth and other features. When the actual percentage of peers with similar properties with whom peer *i* is linked to (its neighborhood), $PNSP_{act}$, is lower than $PNSP_{des}$, peer *i* deletes one link. The link to one of the most different peers is the one chosen for deletion. Then, peer *i* starts its search for another peer with some resemblance to it.

The SAA segregates peers based on their descriptions leading to well structured networks in which the achieved topologies are connected, which is a very desirable feature in P2P networks [63]. As $PNSP_{des}$ increases, the algorithm leads to networks with a higher number of segregating clusters and at a critical point, the network may become unconnected.

7 Further Horizons and Conclusions

In this section, as a part of a wider panorama into algorithms inspired by social phenomena, we propose some ideas that may contribute to enrich the existing options in the Computer Science community.

Leadership is presented as an influence model in which the best individuals are followed by others. However, there are plenty of social phenomena [14] that are decentralized and distributed, i.e., there is not a leader at all. Human behavior such as that of opinion dynamics has attracted attention from the computer science community [5, 20] as it is stated that opinions may not be influenced (at least directly) by leaders opinion.

The assumption that leaders are the best solutions is not entirely true in more realistic approaches because there is a wide range of individuals with some responsibilities that may be interesting to study. Among them is the negotiator, which is responsible for communicating with other leaders and transmitting to them the desires of his/her own leader. A negotiator is a character that could make the work of leaders a lot easier.

In leader-based algorithms, in which migration is important, human patterns of mobility at local [23] or global scales [59] have not been widely considered. A common strategy of exploring unknown spaces in mammals, including humans, is that of Lévy flight [13], which has been incorporated into some optimizing algorithms, such as those of [24], which tend to copy the outstanding foraging strategy of many species.

The self-organizing process of neighborhood segregation is very simple and very elegant. However, there are other aspects of neighborhood delimitation and segregation that are being considered in more detailed models, for example, a wider neighborhood of influence or a more detailed preference for each agent [7, 8]. Algorithms inspired by these behaviors may be applied in, for example, cluster formation.

Most of the algorithms whose origin is a metaphor of social phenomena obtain results equivalent to algorithms inspired by other metaphors. Although the results are encouraging, a lot of ideas and metaphors from human social phenomena are still waiting for further exploration.

References

1. Larisa, A.: Como sobreviven los marginados. Siglo XXI editores, México (1975)
2. Agrawal, R., Rajagopalan, S., Srikant, R., Xu, Y.: Mining newsgroups using networks arising from social behavior. In: Proceedings of the 12th International Conference on WWW, pp. 529–535 (2003), doi:10.1145/775152.775227
3. Aknine, S.: A reliable algorithm for multi-agent coalition formation. In: Proceedings of the IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics, Cambridge, MA, USA, pp. 290–295 (1999), doi:10.1109/ISIC.1999.796670
4. Axelrod, R.: The evolution of cooperation. Basic Books (1984)
5. Bagnoli, F., Franci, F., Rechtman, R.: Opinion formation and phase transitions in a probabilistic cellular automaton with two absorbing states (2005) (accessed July 25, 2008), `http://arxiv.org/abs/nlin/0511001`
6. Ball, P.: Critical mass. Farrar, Straus and Giroux, London (2006)
7. Batty, M.: Cities and complexity. MIT Press, Cambridge (2007)
8. Benenson, I.: Multiagent simulations of residential dynamics in the city. Computers, Environment and Urban Systems 22(1), 24–42 (1998)
9. Bonabeau, E., Dorigo, M., Theraulaz, G.: Inspiration for optimization from social insect behaviour. Nature 406, 39–42 (2000)
10. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: From natural to artificial systems. Oxford University Press, Oxford (1999)
11. Bonabeau, E.: Agent-based modeling: Methods and techniques for simulating human systems. PNAS 99, 7280–7287 (2002)
12. Bonnevay, S., Kabachi, N., Lamure, M.: Agent-based simulation of coalition formation in cooperative games. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Rechnology, pp. 136–139 (2005), doi:10.1109/IAT.2005.33
13. Boyer, D., Miramontes, O., Ramos-Ferández, G., Mateos, J., Cocho, G.: Modeling the searching behavior of social monkeys. Physica A 342, 329–335 (2004)
14. Camazine, S., Deneubourg, J., Franks, N., Sneyd, J., Theraulaz, G., Bonabueau, E.: Self-organization in biological systems. Princeton University Press, Princeton (2003)
15. de Castro, L.: Fundamentals of natural computing. Chapman & Hall, Boca Raton (2007)
16. Coello, C.A.C., Landa, R.: Evolutionary multiobjective optimization using a cultural algorithm. In: Proceedings of the IEEE Swarm Intelligence Symposium, pp. 6–13 (2003), doi:10.1109/SIS.2003.1202240
17. Daneshyari, M., Yen, G.: Talent based social algorithm for optimization. In: Proceedings of the Congress on Evolutionary Computation, pp. 786–791 (2004), doi:10.1109/CEC.2004.1330939
18. Das, S., Flocchini, P., Kutten, S., Nayak, A., Santoro, N.: Map construction of unknown graphs by multiple agents. Theoretical Computer Science 385, 34–48 (2007)

19. Flake, G.: The computational beauty of nature. MIT Press, Cambridge (1998)
20. Florian, R., Galam, S.: Optimizing conflicts in the formation of strategic alliances (2004) (accessed Febraury 20, 2008), `http://arxiv.org/abs/cond-mat/0004216v1`
21. Gil, J., Schmidt, J.: La red de poder en México. UNAM-IIMAS, México (1999)
22. Gil, J., Schmidt, J., Castro, J., Ruiz, A.: A dynamic analysis of the Mexican network of power. Connections 20, 34–55 (1997)
23. González, M., Hidalgo, C., Barabasi, A.: Understanding individual human mobility patterns. Nature 453, 779–782 (2008)
24. Gutowski, M.: Lévy flights as an underlying mechanism for global optimization algorithms (2001) (accessed June 10, 2008), `http://arxiv.org/abs/math-ph/0106003v1`
25. Hales, D., Edmonds, B.: Applying a socially inspired technique (tags) to improve cooperation in P2P networks. IEEE Transactions on Systems, Man and Cybernetics Part A 35(3), 385–395 (2005)
26. Holland, J.: The effect of label (tags) on social interactions. SFI Working Paper 93-10-064 (1993) (accessed January 19, 2008), `http://www.santafe.edu/sfi/publications/wplist/1993`
27. Horling, B., Lesser, V.: A survey of multi-agent organizational paradigms. The Knowledge Engineering Review 19(4), 281–316 (2005)
28. Jargowsky, P.: Take the money and run: Economic segregation in US metropolitan areas. American Sociological Review 61(6), 984–998 (1996)
29. Kijima, K.: Agent-based simulation of alliance formation and its stability analysis: Application to aviation industry (2003) (accessed May 18, 2008), `http://aisel.aisnet.org/pacis2003/81`
30. Klush, G.: Dynamic coalition formation among rational agents. IEEE Intelligent Systems 17, 42–47 (2005)
31. Kohonen, T.: Self-organizing maps. Springer, Heidelberg (2000)
32. Kotkin, J.: The city, a global history. Weidenfeld & Niolson (2005)
33. van der Laan, W.: Involuntary isolation: Ethnic preferences and residential segregation. Journal of Urban Affairs 29(3), 289–309 (2007)
34. Langaney, A., Clottes, J., Guilaine, J., Simmonet, D.: La plus belle histoire de l'homme. Editions du Seuil, Paris (1998)
35. Lewontin, R.: It ain't necessarily so: The dream of the human genome and other illusions. New York Review of Books (2000)
36. MacKay, D.: Information theory, inference and learning algorithms. Cambridge University, Cambridge (2003)
37. Mashkov, V.: Restricted alliance and coalitions formation. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 329–332 (2004)
38. Mashkov, V., Marik, V.: Alliance formation process and communication traffic (2002) (accessed February 10, 2008), `http://www.actapress.com/PaperInfo.aspx?PaperID=25670&reason=500`
39. Massey, D., Gross, A., Shibuya, K.: Migration, segregation and the geographic concentration of poverty. American Sociological Review 59(3), 425–445 (1994)
40. Nielsen, J.: The myth of leadership: Creating leaderless organizations. Davies-Black Publishing (2004)

41. Pechoucek, M., Marik, V., Barta, J.: A knowledge-based approach to coalition formation. IEEE Intelligent Systems 17(3), 17–25 (2002)
42. Rahwan, T., Jennings, N.: An improved dynamic programming algorithm for coalition structure generation. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1417–1420 (2008)
43. Ray, T., Liew, M.: Society and civilization: An optimization algorithm based on the simulation of social behavior. IEEE Transactions on Evolutionary Computation 7(4), 386–396 (2004)
44. Reynolds, R.: An introduction to cultural algorithms. In: Sebald, A.V., Fogel, L.J. (eds.) Proceedings of the 3rd Annual Conference on Evolutionary Programming, pp. 131–139. World Scientific Press, Singapore (1994)
45. Sen, S., Dutta, P.: Searching for optimal coalition structures. In: Proceedings of the 4th International Conference on Multiagent Systems, pp. 286–292 (2000)
46. Schelling, T.: Micromotives and Macrobehavior. Ed Norton (2006)
47. Selvakumar, A., Thanushkodi, K.: Optimizing using civilized swarm: Solution to economic dispatch with multiple minima. Electric Power Systems Research 79(1), 8–16 (2009)
48. Semsar-Kazerooni, E., Khorasani, K.: Optimal consensus algorithms for cooperative team of agents subject to partial information. Automatica 44(11), 2766–2777 (2008)
49. Shehory, O., Kraus, S.: Task allocation via coalition formation among autonomous agents. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp. 655–661 (1995)
50. Shehory, O., Kraus, S.: Formation of overlapping coalitions for precedence-ordered task-execution among autonomous agents. In: Proceedings of the 2nd International Conference on Multi-Agent Systems, pp. 330–337. AAAI Press/MIT Press (1996)
51. Singh, A.: Topology adaptation in p2p networks using Schelling model. The University of Dublin Trinity College (2004) (accessed March 21, 2008), `https://www.cs.tcd.ie/publications/tech-reports/reports.04/TCD-CS-2004-41.pdf`
52. Singh, A., Haar, M.: Creating an adaptive network of hubs using Schelling model. Communications of the ACM 49(3), 69–73 (2006)
53. Singh, A., Vainchtein, D., Weiss, H.: Schelling's segregation model: Parameters, scaling, and aggregation (2007) (accessed July 20, 2008), `http://arxiv.org/abs/0711.2212v1`
54. Singh, A., Haahr, M.: Decentralized clustering in pure P2P overlay networks using Schelling's Model. In: Proceedings of the IEEE International Conference on Communications, pp. 1860–1866 (2007), doi:10.1109/ICC.2007.310
55. Ursem, R.: Multinational evolutionary algorithms. In: Proceedings of the Congress on Evolutionary Computation, Washington, DC, USA, pp. 1633–1640 (1999)
56. Vijaya, P., Murty, N., Subramanian, D.: Leaders-subleaders: An efficient hierarchical clustering algorithm for large data sets. Pattern Recognition Letters 25, 505–513 (2004)
57. Wan, Y., Wu, H.: Convergence and optimization of agent-based coalition formation. Physica A: Statistical Mechanics and its Applications 348, 641–658 (2004)
58. Wang, J., Wang, D.: Particle swarm optimization with a leader and followers. Progress in Natural Science 18(11), 1437–1443 (2008)
59. Weaver, T., Roseman, C.: New developments in the genetic evidence for modern human origins. Evolutionary Anthropology 17, 69–80 (2008)

60. Yamamoto, K.: International relations and agent-based modeling. Paper presented at the Annual Meeting of the International Studies Association 48th Annual Convention, Chicago, IL, USA (10-09-2008) (2007), `http://www.allacademic.com/meta/p180204_index.html`
61. Yang, S., Su, C.: Co-evolutionary learning with strategic coalition for multiagents. Applied Soft Computing 5, 193–203 (2005)
62. Yeh, D.: A dynamic programming approach to the complete set partitioning problem. Numerical Mathematics 26(4), 467–474 (1986)
63. Zhang, Y., Li, Y., Yue, Y., Zhu, L.: An ecology-based model for efficient distributed search in P2P networks. In: Proceedings of the 7th World Congress on Intelligent Control and Automation, Chongqing, China, pp. 4485–4488 (2008), doi:10.1109/WCICA.2008.4593645
64. Zomaya, A.: Handbook of nature-inspired and innovative algorithms. Springer, Heidelberg (2006)

Artificial Immune Systems for Optimization

Heder S. Bernardino and Helio J.C. Barbosa

Abstract. Artificial Immune Systems (AISs) are computational methods inspired by the biological immune system and thus classified as a nature-inspired metaheuristic along with genetic algorithms, ant colony optimization, particle swarm optimization, and others. This chapter is focused on the application of AISs to solve optimization problems. Optimization is a mathematical principle largely applied to design and operational problems in all types of engineering, as well as a tool for formulating and solving inverse problems such as parameter identification in scientific and engineering situations. This chapter provides a survey of the applications of AISs in optimization. The main contributions are studied and contrasted with respect to the different concepts of the biological immune system, such as clonal selection, hypermutation, immune network, affinity, etc. that have inspired such techniques. The main types of optimization problems are considered, namely, (i) unconstrained problems, (ii) constrained problems, (iii) multimodal problems where several optima are to be obtained in a single run, and, finally (iv) multi-objective problems, where an approximation to the Pareto set is to be obtained as the result of a single run of the algorithm. Although AISs are good for solving optimization problems, useful features from other techniques have been combined with a "pure" AIS in order to generate hybrid AIS methods with improved performance.

1 Introduction

Computer Science has, for some time now, found inspiration in nature in order to find new ways of solving computational problems. Artificial Immune Systems

Heder S. Bernardino
Laboratório Nacional de Computação Científica - LNCC, Brasil
e-mail: `hedersb@gmail.com`

Helio J.C. Barbosa
Laboratório Nacional de Computação Científica - LNCC, Brasil
e-mail: `hcbm@lncc.br`

(AISs) are composed of intelligent methodologies, inspired by biological immune systems, to solve real world problems [34].

In nature, when an animal is exposed to antigens, an efficient immune response is developed in order to defend the organism. To do so, specific antibodies are produced to combat the antigens. The best antibodies are cloned, hypermutated, and selected, while random antibodies (produced by the bone marrow) are also generated to improve the diversity of the population. Also, if the organism is again attacked by that antigen a quicker immune response is developed. This skill of adaptation is known as clonal selection and affinity maturation by hypermutation or, more simply, clonal selection [39].

The natural immunity comprises innate and adaptive immunities [64]. The innate immune system is composed of cells and mechanisms that defend the host from attacks by other organisms, in a non-specific manner. The adaptive immune system comprises highly specialized cells and processes that defend the organism from antigens. The main adaptive immunity feature is to distinguish between proteins produced by cells of the body (self) and the ones produced by intruders or by cells under virus control (non-self). The clonal selection mechanism belongs to the adaptive immune system.

It is natural to think of the application of an AIS to security problems since the immune system provides protection against foreign invaders in the biological world. However, they have been exploited for other classes of problems as well, such as classification and data analysis, pattern recognition, image processing, machine learning, robotics, control, scheduling, and optimization. Good surveys of the AIS field can be found in [34, 14, 72, 47, 75].

For optimization problems, our main focus in this chapter, different techniques based on AISs have been developed. AISs are relatively new to solving optimization problems, when compared with other nature-inspired meta-heuristics such as genetic algorithms (GAs) and evolution strategies (ESs), both inspired by evolution. It is then only natural to compare the AIS with other bio-inspired algorithms [47], identifying similarities and differences. In the context of optimization, GAs and ESs are, perhaps, most commonly used for such comparisons. The majority of AIS algorithms applied to optimization are based on immune network theory [13, 76] and the clonal selection principle [17, 74]. Those algorithms, and other immune inspired ones, are studied in this chapter with emphasis on their application to optimization problems.

The clonal selection and immune network algorithms are similar to other stochastic search methods and, as such, can be hybridized with other methods. The AIS technique has a good balance between global and local exploration and is simple to implement [80].

This chapter is organized as follows. Section 2 briefly discusses AISs and identifies immune mechanisms which have been used in computational algorithms. Section 3 shows the main artificial immune algorithms used to solve optimization problems. A review of several immune techniques for continuous or discrete, constrained or unconstrained, multimodal, and multi-objective, optimization problems is presented in Section 4. Finally, Section 5 concludes this chapter.

2 Artificial Immune Systems

AISs are computational techniques, inspired by the biological immune system, which can be used to solve complex real world problems. The AIS technique evolves improved solutions of the problem by means of clonal selection, immune network theory, vaccination, or other immune system concepts.

Although there is no consensus on what should be the canonical immune algorithm for, say, solving a standard optimization problem, there is general agreement on what a clonal selection or immune network algorithm looks like [75].

In general, an immune optimization algorithm will have a population of individuals (candidate solutions), called antibodies, and other individuals (or objectives) that the antibodies attempt to reach or match. Those are called antigens. The main differences among the AIS techniques applied to optimization problems reside in how their antibodies evolve.

The CLONALG algorithm [17], or CSA (as it was called in [15]), evolves the antibodies inspired by the concept of clonal selection. The clonal selection evolution is based on the principle that each individual is cloned, hypermutated, and those with higher affinity are selected. The mutation rate is, normally, inversely proportional to the affinity of the antibody with respect to the antigens. AISs usually do not use recombination operators (such as crossover in GAs).

Based on the immune network theory, the opt-aiNet algorithm [13] is another well known immune-inspired technique. The main idea of this method is to construct a network of the best non-similar antibodies, i.e., maintain in the immune system the cells of the body (self) that have less affinity among themselves and more affinity with the ones produced by intruders or by cells under virus control (non-self).

Keko et al[52] produced good solutions to combinatorial optimization problems by including a vaccination process in their AIS algorithm, which can be seen as a way of introducing domain knowledge into the algorithm.

Although there are many different algorithms which are referred to as "artificial immune systems" it is interesting to identify some mechanisms of the natural immune system that have inspired specific computational procedures which can be found in many different implementations of AISs. Somatic hypermutation, clonal selection, immune network, and vaccination are briefly discussed in the following sections.

2.1 Somatic Hypermutation

This process, also known as affinity maturation, is directly responsible for the evolution of the algorithm. Unlike standard Evolutionary Algorithms (EAs), it is a mutation of the individuals applied with a high rate, which is inversely proportional to the fitness of the antibody (affinity antibody-antigen). In this way, worst individuals are subject to more modification than the best ones, which need a finer tuning. Also, it is easy to see that this procedure is a random search method if applied alone. A selection method is then necessary to keep the good solutions, eliminate the worst ones, and maintain diversity.

2.2 Clonal Selection

According to clonal selection theory, there is a selection mechanism which leads to the evolution of the immune system repertoire during the lifetime of the individual [12]. By this theory, on binding with a suitable antigen, activation of lymphocytes occurs. Once activated, clones of the lymphocyte are produced expressing receptors identical to the original lymphocyte that encountered the antigen. This process is known as clonal expansion. Any lymphocyte that has receptors specific to molecules of its own body must be deleted (these lymphocytes will not be cloned). This ensures that only an antigen may cause a clonal expansion. In this way, the immune system can be viewed as a classifier of cells into either self or non-self cells. The non-self cells are called antigens, and are assumed as being from a pathogen and thus need to be removed from the body. The affinity maturation is the increase in the average of the affinity between the antibodies and antigens due to the somatic hypermutation and selection mechanisms of clonal expansion. It is responsible for the fact that upon a subsequent exposure to the antigen, a stronger immune response is produced [1].

2.3 Immune Network

In [51] Jerne proposed the immune network theory which helps to explain some properties of the natural immune system. By this theory, any lymphocyte receptor within an organism can be recognised by a subset of the total receptor repertoire. The receptors of this recognising set have their own recognising set and so on, forming an immune network of interactions, often referred to as idiotypic networks [1]. While the body is not attacked by an antigen, the immune system may interact with itself. From this behavior, the immune system acquires tolerance and memory [51]. Computationally speaking, this process keeps the diversity of the population, covers the search space, and eliminates similar individuals. Immune network inspired algorithms are discussed in Section 3.2.

2.4 Vaccination

In the XIth century, smallpox attacked the world population. At that time, it was observed that those who managed to recover from a first contamination became resistant to a subsequent contact with the disease. From this knowledge, children (who were the main victims) were infected with extracts of pustules of smallpox. The idea was that if they manage to survive, these children would be immunized against any further contagion. After some time it was discovered that the use of milder variations of smallpox pustules also made individuals resistant to the disease. At the end of the XVIIIth century, the English physician Edward Jenner observed that workers that milked the cattle and had picked up the smallpox from cows, a mild variation, were protected against the human variation of the disease. Jenner used cattle pustules to cure infected people. This substrate was known as vaccínia or cowpox (vacca, Latin for cow) and so the procedure was known as vaccination. After Jenner, in the XIXth century, Robert Koch proved that the infectious diseases were caused by different microorganisms. By 1900, Louis Pasteur, in his studies on vaccination, concluded that exposure to a non-virulent strain of an agent causing the disease can protect one against future infection of other similar agent [5].

The vaccination process, when used in AISs, usually partially modifies the candidate solution according to *a-priory* knowledge of the problem that is being solved. The method produces good solutions (higher fitness or affinity) with greater probability than if the individual is mutated randomly [61]. For the TSP, for instance, it seems clear that the shortest distances between pairs of cities have a good probability of generating a good tour and such domain knowledge can then be regarded as a kind of vaccine. When inserting the vaccine, there is a higher probability of lowering the total distance of the tour than that of increasing the total distance. However, this procedure requires *a-priory* knowledge of the problem which, many times, is not available or is too expensive to extract.

3 Immune Optimization Algorithms

Due to the more widespread use of EAs in optimization on one hand, and the fact that some AISs (like clonal selection algorithms) can be seen as a subset of EAs on the other, it is natural to look for similarities and contrasts between AISs and EAs.

Wierzchoń [81] makes a distinction between an immune optimization algorithm and GA-based methods with the same objective. The main differences are the crossover and mutation (at a low rate) operators. If the comparison is made between AIS and ESs, besides the obvious differences, Garrett [40] cites the control of mutation (the mutation rate is altered at each generation, multiplying or dividing the current value by a constant value, often 1.3 [69]) and the real-valued encoded individuals (this difference considered the first immune optimization algorithms; today real encoded immune inspired methods are available [39]). Also, both GA and ES have the mutation rate fixed when applied in any given generation. The AIS tunes the mutation rate according to the fitness of the solutions [40] (lower mutation rates are applied to better solutions). The dialect is also presented as a difference between these techniques [81].

Although there are many AIS algorithms to solve optimization problems, they are often classified into one of the following classes [1]: (i) Clonal Selection or (ii) Immune Network Algorithms, which are detailed in the following sub-sections.

3.1 Clonal Selection Algorithms

Based on the clonal selection theory, de Castro and Von Zuben [15, 17] proposed an AIS that performs computational optimization and pattern recognition tasks. The clonal selection evolution is inspired by the principle that each individual is cloned, hypermutated, and those with higher affinity are selected. Sometimes, the AIS algorithms use the idea of memory cells. This mechanism retains solutions that can be used later and it is often applied in pattern recognition problems.

In [15] the CSA (as the algorithm was called) was proposed as "a powerful computational implementation of the clonal selection principle" and applied to three problems: binary character recognition, multimodal optimization, and a 30-city instance of the Travelling Salesman Problem (TSP). Of course only the last two ones are true optimization problems and they will be discussed in Section 4. CSA was shown to be a good meta-heuristic to solve multimodal and combinatorial optimization problems. In fact, the clonal selection principle can be interpreted as a remarkable microcosm of Darwinian evolution [33] and can be considered an evolutionary algorithm.

The improved CSA is known as CLONALG and was proposed in [17]. In that work, a sensitivity analysis with respect to the user-defined parameters was presented, and benchmark problems were considered in order to evaluate the performance of the algorithm (see Section 4).

A pseudo-code for the CLONALG algorithm is given in Algorithm 1, but optimized to reduce the computational complexity (as seen in [39]).

Algorithm 1. A CLONALG pseudo-code for optimization problems.

```
Data: $antibodies, nGenerations, \beta, nSelection, nReplaces$
Result: $bestAffinity, antibodies$
1  begin
2      $affinities \longleftarrow$ calcAffinities($antibodies$);
3      for $generation = 0; generation < (nGenerations - 1)$ do
4          $selectedAntibodies \longleftarrow$ select($antibodies, affinities, nSelection$);
5          $clones \longleftarrow$ clone($selectedAntibodies, affinities, \beta$);
6          $clones \longleftarrow$ hypermutate($clones, affinities$);
7          $cloneAffinities \longleftarrow$ calcAffinities($clones$);
8          update($antibodies, affinities, clones, cloneAffinities, nReplaces$);
9      $bestAffinity \longleftarrow$ getBestAffinity($antibodies$);
10 end
```

In Algorithm 1, *antibodies* is a population of individuals (candidate solutions), *nGenerations* is the number of generations that the algorithm will perform, β defines the number of clones each antibody will generate, *nSelection* is the number of antibodies selected to be cloned, *nReplaces* is the number of lower affinity antibodies which will be replaced by the new ones, and *bestAffinity* is the best value found by the algorithm. Also, in the same algorithm, the following functions can be found: calcAffinities, calculates the affinities between each antibody and all antigens (in optimization problems this value often corresponds to the value of the objective function); select, selects the *nSelection* best individuals to be cloned; clone, clones the selected antibodies; hypermutate, applies the somatic hypermutation in generated clones; update, replaces the *nReplaces* worst antibodies by other ones from *clones* and update the *affinities*; and getBestAffinity, returns the best found affinity. The number of clones N_c is given by

$$N_c(antibody_i) = round\left(\frac{\beta.|antibodies|}{i}\right),$$

where *round* is the operator that rounds its argument towards the closest integer.

If evolutionary algorithms are to be applied to situations where the user requires several different solutions to be obtained simultaneously in the final population, these algorithms must be extended with niching techniques [3, 63]. Fitness sharing [42] and crowding are niching techniques well-known in the literature. CLONALG does not require such extensions in order to maintain multiple solutions. Instead, two parameters may assume default values:

- $nSelection = |antibodies|$, i.e., all antibodies from the population will be selected to be cloned;
- N_c may be the same for all antibodies, i.e., $N_c(antibody_i) = round(\beta.|antibodies|)$, $\forall i$.

In CLONALG it is not necessary to define a crossover operator probability (parameter always used in GAs, for example). However, it is easy to see that there are some other user-defined parameters in this algorithm. They are: *nSelection* (number of antibodies to be selected for cloning), β (parameter used to define the number of clones for each antibody), and *nReplaces* (number of low-affinity antibodies to be replaced [17] (for the sensitivity analysis see [17]).

There is source code available for many immune inspired algorithms on the Web [1] while the CLONALG source code can be found in [18]. The Optimization Algorithm Toolkit [11] is an open source project that contains implementations for some bio-inspired algorithms; among them: Adaptive Clonal Selection (ACS), Optimization Immune Algorithm (opt-IMMALG), Optimized Artificial Immune Network (opt-aiNET), Optimization Immune Algorithm (opt-IA), Clonal Selection Algorithm (CLONALG), B-Cell Algorithm (BCA), Cloning, Information Gain, Aging (CLIGA), and Immunological Algorithm (IA).

Other immune algorithms, inspired by clonal selection, to solve optimization problems are presented in Section 4.

3.2 Immune Network Algorithms

Inspired by the immune network theory, de Castro [16] proposed an algorithm for data analysis which was called aiNet. An adaptation of this model to solve optimization problems, called opt-aiNet, was proposed later in [13].

The opt-aiNet is an extension of CLONALG (see Section 3.1) [13], the difference between the algorithms being the inclusion of steps involving the iteration among the network cells. In opt-aiNet, the population size is free, since suppression eliminates similar antibodies from the network and keeps the better ones. A pseudo-code for the opt-aiNet, which is similar to the one presented in [13] and revised in [74], is shown in Algorithm 2.

Algorithm 2. A opt-aiNet pseudo-code.

```
Data: $antibodies$, $\beta$, $\alpha$ $\sigma$, $pRandom$
Result: $bestFitness$, $antibodies$
1  begin
2      $fitness \longleftarrow$ calcFitness($antibodies$);
3      while $stopping\ criteria\ is\ not\ met$ do
4          $normalizedFitness \longleftarrow$ normalize($fitness$);
5          $clones \longleftarrow$ clone($antibodies$, $normalizedFitness$, $\beta$);
6          $clones \longleftarrow$ hypermutate($clones$, $normalizedFitness$, $\alpha$);
7          $cloneFitness \longleftarrow$ calcFitness($clones$);
8          $clones \longleftarrow$ getBetterAntibodies($clones$);
9          $clones \longleftarrow$ suppression($clones$, $cloneFitness$, $\sigma$);
10         update($antibodies$, $normalizedFitness$, $clones$, $cloneFitness$,
           $pRandom$);
11         $antibodies \longleftarrow$ suppression($antibodies$, $fitness$, $\sigma$);
12     $bestFitness \longleftarrow$ getBestFitness($antibodies$);
13 end
```

In Algorithm 2, fitness is the value of the objective function to be optimized and the affinities are the euclidean distances between the network cells [13]. Also, σ is the suppression threshold, α is a parameter used to compute the proportional mutation, *normalizedFitness* contains the fitness of the antibodies normalized in the interval $[0,1]$, *pRandom* is the percentage of randomly generated cells that will be introduced in the network, and the function "suppression" eliminates similar individuals (i.e. those with affinity less than σ).

The mutation is performed as

$$antibody_c = antibody + \frac{1}{\alpha} e^{-f_n} N(0,1),$$

where $antibody_c$ is the result of mutating the cell *antibody*, $N(0,1)$ is a Gaussian random variable of zero mean and unit standard deviation, and f_n is the fitness of the *antibody* (normalized in the interval $[0,1]$).

It is easy to see that the population size is continuously increasing. According to [13], that is an indication that the problem has many local optima and that the algorithm is capable of finding all of them.

The source code for opt-aiNet is available on the Web [18].

Other immune algorithms inspired by the immune network theory to solve optimization problems are presented in Section 4.

3.3 Hybrid Immune Algorithms

Hybrid algorithms are designed by combining existing methods in order to obtain performance higher than that of each method working alone. Yen et al. [83] describe four forms of hybrids in an EA context: (i) pipelining hybrids, (ii) asynchronous hybrids, (iii) hierarchical hybrids, and (iv) the use of additional operators. In pipelining hybrids, the techniques are applied sequentially: one generating data to be used by the other, while an asynchronous hybrid maintains a population shared by both techniques which work cooperatively and asynchronously. A hierarchical hybrid uses different search procedures at different levels of the problem (for instance, topology of a neural net and weights of the connections). Finally, a hybrid algorithm can also be constructed by introducing a given search procedure as an additional move operator.

A hybrid immune algorithm uses the immunological principles combined with other methods to generate improved algorithms.

In this way, Hajela and co-workers [43, 44, 45, 84], used a GA in order to increase the similarity (or reduce the distance) between infeasible elements (antibodies) and feasible ones (antigens), and embedded it into the original one. The inner GA uses as fitness function the genotypical distance between the antigens and antibodies in order to evolve better (hopefully feasible) individuals. As a result, there is no need for additional expensive evaluations of the original fitness function of the problem since the internal loop uses a relatively inexpensive fitness function based on Hamming distance calculations. Coello and Cruz-Cortés[22] proposed an extension of Hajela's algorithm, together with a parallel version, and tested them in a larger problem set. In that work, the embedded algorithm was called an AIS, although the evolution of the candidate solutions was actually done by standard evolutionary operators (selection, crossover and mutation).

Bernardino et al. [6, 7, 8, 5] proposed hybrid algorithms with new inner loops and selection procedures. The inner GA was replaced by CLONALG to minimize the distance between feasible and infeasible individuals through genotype changes. Although the AIS changes the genotype of the individuals, the fitness values are kept unchanged.

In [5, 9] the authors proposed a new hybrid immune algorithm which uses a GA in parallel with an AIS. In this work, the AIS' objective is to minimize the constraint violations, while the GA is used to optimize the objective function. All hybrid algorithms presented above are used to solve constrained optimization problems and are better detailed in the Section 4.2.

Wang et al. [79] proposed a hybrid optimization method based on the Ant Colony Optimization (ACO) [37, 24] and clonal selection algorithm. The idea is to introduce the cloning and mutation operations in the ant colony algorithm in order to enhance its search capability.

Along the same lines, Yuan et al. [85] proposed an evolutionary algorithm based on Ant Colony Optimization algorithm and the AIS model to solve the Geometric Constraints Problem (a non-convex constrained optimization problem). In this algorithm, the affinity calculation process and pheromone trail deposition maintain diversity and carry out the global and the local searches.

4 Solving Optimization Problems

Optimization problems cover a large number of real-world applications. The AIS algorithm, as mentioned before, was initially proposed to solve other kinds of problems. Nevertheless, there are many papers in the literature that present the efficiency of the AISs in solving optimization problems.

Although a Markov-chain analysis proves weak convergence of the AIS algorithms (see, for example, [20, 77, 32]), good results can actually be found for many classes of optimization problems. Also, Cutello et al. [32] have shown the convergence of a general immune algorithm to a global optimum under certain assumptions (any cloned antibody can be mutated with a probability greater than zero and the best individual survives in each generation with probability equal to 1).

This section shows some of these algorithms and the problems they were applied to. The optimization problems considered can be classified as: unconstrained, constrained, combinatorial, and multi-objective. A new paradigm, immune programming, is also commented upon.

4.1 Unconstrained Optimization Problems

An unconstrained minimization problem can be stated as finding x^*, such that $f(x^*) \leq f(x)$ for all x in the domain of the problem. Although bound constraints are often present, they need not be considered here since such bounds can be easily enforced in the AIS implementation. The unconstrained optimization problems can be static or dynamic. In dynamic problems, the objective function changes over time and the algorithm has to track the moving optimum in the search space.

It is easy to see that it should be natural for AIS algorithms to solve dynamic optimization problems, since the problem faced by the natural immune system is a dynamic one: the immune system must be adaptive in order to face different antigens over time.

In this way, Gaspar and Collard [41] proposed an AIS algorithm to solve Time Dependent Optimization (TDO). This algorithm, called Sais (Simple Artificial Immune System), is based on the model of the idiotypic network and was compared with other evolutionary algorithms to illustrate its efficiency. According to [41], Sais opened a new way in the study of the adaptive dynamics through Time Dependent Optimization Tasks.

CLONALG, perhaps the most popular immune inspired algorithm to solve optimization problems, was tested with functions having many local optima in [17, 15] where it was able to find all peaks including the global optimum. CLONALG does not require a niching method, as required by any standard evolutionary algorithm which tends to converge the whole population of individuals towards a single solution.

The clonal selection algorithm was also applied to the travelling salesman problem (TSP). The TSP is a well-known combinatorial optimization problem which arises in many applications [15]. The travelling salesman must visit all cities in a list, only once, and return to the starting one. The objective function is the sum of the costs of traveling between each two cities in a given tour. For the instance considered, the CSA found the global optimum [65].

In [17], CLONALG was tested on four optimization tasks: three multimodal optimization problems and a 30-city instance of the TSP. One multimodal problem and the TSP were solved in [15]. One of the multimodal functions was used in [3, 42] to evaluate niching methods for GAs. Also, this work presents many comparisons with evolutionary algorithms, and the empirical comparisons with a GA with fitness sharing [3, 63] demonstrated that CLONALG is capable of locating a larger number of local optima.

A version of CLONALG was applied to dynamic optimization problems in [78] where the algorithm was compared with an evolution strategy. That study concluded that, usually, an evolution strategy can optimise more quickly than a clonal selection algorithm. According to [78, 40], clonal selection tends to perform better than ESs at low dimensionalities while the ESs perform better when higher dimensional spaces are considered.

To solve dynamic optimization problems, Kelsey et al. adopted, in [53], the B-Cell algorithm. The B-Cell algorithm (BCA) is an immune clonal selection based algorithm that was proposed in [54]. The important features of the BCA are the use of a unique mutation operator, called continuous somatic hypermutation [54], and the size of the population, which evolves together with the candidate solutions. The BCA was found to be significantly more effective than a hybrid GA when tracking the fixed points of the two mappings considered in [54].

In [74], Timmis et al. presented an empirical study of opt-aiNet, BCA, and an HGA (Hybrid Genetic Algorithm) for function optimization. The authors discussed if AISs have something to offer to the world of optimization. In the tests, the BCA and HGA were found more effective. The conclusion was that the BCA performs well in terms of quality of the solutions found and number of the objective function evaluations. According to the authors, the possible reason for this is the mutation of the algorithm which endows the BCA with the ability to escape from local optima more quickly.

Cutello et al. investigated the search capability of hypermutation operators [26] by applying them to trap functions and to the 2D HP model [36] for the protein structure prediction problem. Four mutation operators were considered: static hypermutation, proportional hypermutation, inversely proportional hypermutation, and hypermacromutation. In the last one, two integers, i and j, are chosen at random such that $i+1 \leq j \leq l$, where l is the chromosome length, and most values in the range $[i, j]$ are changed. The tests indicated good performance with the hypermacromutation operator when coupled with static or inversely proportional hypermutation.

As it can be seen in Section 3.1, the clonal selection algorithm requires several parameters. Also, it was proposed in [17] that using a binary representation can limit the accuracy of the results [39] in some cases. To avoid those shortcomings, Garrett proposed in [39] a real-valued, parameter-free clonal selection algorithm called Adaptive Clonal Selection (ACS). The modifications proposed were: (i) to change from a binary to a real-valued representation, (ii) to remove the parameter used to control the amount of mutation, (iii) to remove the parameter defining the size of the sub-population that will be cloned, and (iv) to remove the parameter used to calculate the number of clones of each antibody. Modification (ii) is based on the principle of adaptation used by ESs: the parameter is either multiplied or divided by 1.3 at each generation [2]. Note that this adaptation is different from the ES because it is adapting the parameter that controls the amount of mutation (in AIS, the mutation is usually inversely proportional to antibody affinity). The sub-population size and the number of clones (iii and iv) are adapted considering the ratio between fitness of the fittest members of two contiguous generations:

$$\Delta = \frac{max(f_{\gamma-1})}{max(f_{\gamma})}. \tag{1}$$

where a minimization problem is assumed. To verify the performance of the algorithm, uni- and multimodal functions were considered. The results demonstrate that ACS is a good algorithm, capable of reaching accurate approximations of the global optimum. Also, this algorithm can be applied without prior tuning of its parameters. For the test-functions considered, Garrett verified that the ACS algorithm is more effective than a pre-parametrized ES.

Cutello et al. [27] reviewed and compared CLONALG and opt-IA algorithms in optimization and pattern recognition problems over a range of user defined parameters and concluded that the opt-IA algorithm performed better than CLONALG. Later [28] the opt-IA algorithm was compared with evolutionary algorithms and was shown to be an effective numerical optimization algorithm in terms of solution quality.

Another clonal selection inspired algorithm, called opt-IMMALG, which is an improved version of opt-IA, was proposed in [31] to solve continuous global optimization problems with real-coded representation. Opt-IMMALG produced better results than opt-IA, was competitive with respect to particle swarm optimization, but was outperformed by the differential evolution algorithm.

In the same symposium, Cutello et al. [29] proposed a new evolutionary algorithm with two mutation operators: one for local search and another (hypermacromutation) for global search. The Evolutionary Algorithm with Local Search (EALS) was then applied to trap functions and the results were found to outperform those obtained by opt-IA.

A parallel immune algorithm, par-IA, based on opt-IA and using a master-slave approach, was proposed in [30]. The master process delegates the tasks to slaves and communicates with them when the termination condition is reached. The slaves perform the cloning, hypermutation, and evaluation of the antibodies sent by the master. Experiments indicated that par-IA obtains high speedup values when applied to problems with large search spaces which is reasonable, considering that the function evaluation consumes, on most real-world situations, more computation resources than other immune algorithm steps.

Wang et al. [79] proposed a hybrid immune ant colony algorithm to solve multimodal static and dynamic optimization problems where a clonal selection algorithm was combined with an ACO to enhance its search capability. Six static and one dynamic problems were used to evaluate the performance of the algorithm. The tests demonstrated the remarkable advantages of this approach in diverse optimal solutions, closely tracking the moving optimum, and with improved convergence speed [79].

Two algorithms, suppression control algorithm (SCA) and its parallel version (PSCA), were proposed by Lau and Tsang [60]. In SCA, the mutation rate depends on the inverse of the affinities and the class of the individuals (the population is partitioned in three subsets according to affinity levels). The antibodies from the low affinity class are replaced by randomly generated ones. A method called general suppression control framework (GSCF) [56, 57] is used to regulate the dynamics of the population, where antibodies with high affinity to self cells are eliminated. The PSCA is organized as an island model where the overall population is distributed among the processes. Each island is responsible for the evolution of its sub-population and the communication among them corresponds to the migration, at the end of each iteration, of a high affinity subset of antibodies to another island. The algorithms were tested on eleven numerical benchmark functions, and PSCA produced in most cases better results than the other immune inspired techniques, but was outperformed by the differential evolution algorithm.

4.2 Constrained Optimization Problems

A standard constrained optimization problem in R^n corresponds to finding the vector of design/decision variables $x \in R^n$ which minimizes a given objective function $f(x)$ subject to inequality constraints $g_p(x) \geq 0, p = 1,2,\ldots,\bar{p}$ as well as equality constraints $h_q(x) = 0, q = 1,2,\ldots,\bar{q}$. Additionally, the variables are usually subject to bounds, which do not require special treatment here.

Coello and Cruz-Cortés[22] proposed an extension of Hajela's algorithm (see [43, 44, 45, 84]), together with a parallel version, and tested them on a larger problem set comprising a well-known benchmark set of mathematical functions and some optimization problems from the mechanical engineering field. Comparisons were made with several penalty techniques in order to check the performance of the proposed approach.

Cruz-Cortez et al. [25] proposed a CLONALG algorithm to solve constrained optimization problems. To evolve the immune system, any feasible antibody (a candidate solution that satisfies all constraints) is better than any infeasible individual. Binary as well as real representations were implemented. As the results for the real-coded version of CLONALG were disappointing, the authors proposed an alternative mutation operator in order to improve the performance of the AIS with real representation. In that approach, the mutation operator depends on (i) the affinity of the antibodies, (ii) the range of each decision variable, and (iii) the antibody population size. The proposed AIS showed good results when compared with other well-known algorithms. However, it was unable to outperform the results obtained using the stochastic ranking technique [70].

In [62], an AIS was proposed for the fuzzy c-means clustering algorithm. The algorithm is called AINFCM (Artificial Immune Network for Fuzzy C-Means). Fuzzy c-means (FCM) is a clustering method which allows one piece of data to belong to two or more clusters. A clustering problem can be viewed as an optimization problem that locates the optimal centroids of the clusters directly under the membership function constraints, rather than finding an optimal partition. The problem can be seen as the minimization of an objective function given by

$$J(U,c_1,\ldots,c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c}\sum_{j}^{n} u_{ij}^m d_{ij}^2,$$

under the constraints:

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1,\ldots,n$$

where $u_{ij} \in [0,1]$ indicates the membership of data vector x_j assigned to i-th cluster, c_i is the i-th centroid, and d_{ij} is the Euclidean distance between data vector x_j and centroid c_i. The fuzzifier $m \in [1,\infty)$, is a cluster sensitivity parameter. More information about the problem can be found in [55, 62]. For the AINFCM algorithm, the antigens represent the data set for data clustering and the antibodies are the possible centroid clusters [62]. Some sensitivity analyses of the user-defined parameters were performed. To evaluate the performance of the algorithm, it was compared with other fuzzy clustering methods (such as K-means, K-medoid, and FCM algorithm). The Partition Coefficient Index (PC), which measures the amount of "overlapping" between clusters, and the Separation Index (S), which is the ratio of the sum of compactness and separation of the clusters, were the compared values. AINFCM showed better fuzzy clustering behavior according to both PC and S indexes [62].

Another approach used to solve constrained optimization problems is to hybridize an AIS with a GA. The GA-AIS [6, 7, 8] hybrid follows the idea proposed in [43, 44, 45, 84] and extended in [22]. The idea of the method is that an AIS (CLONALG) is called to help the GA in increasing the number of feasible individuals in the population. The hybrid GA-AIS consists of an outer (GA) search loop where the current population is checked for constraint violation and then segregated into feasible (antigens) and infeasible (antibodies) individuals . If there are no feasible individuals, the two (or some other user defined quantity) "least infeasible" ones (those with the lowest constraint violation) are moved to the antigen population. The selection operation is then performed in order to apply recombination and mutation operators to the selected parents producing a new population and finishing the external (GA) loop. The AIS is introduced as an inner loop where antibodies are first cloned and then mutated. Next, the distances (affinities) between antibodies and antigens are computed. Those with higher affinity (smaller sum of distances) are selected, thus defining the new antibodies (closer to the feasible region). This AIS cycle is repeated a number of times. The resulting antibody population is then passed to the GA. It is important to notice that each cloned hypermutated individual inherits its parent's fitness, so that no fitness function evaluation is required at this point. In [6], the GA-AIS was tested and compared with other algorithms well-known in the literature using a set of benchmark functions and presented good results. In [7], the procedure of changing the population for a new one was modified. In this new approach an individual is replaced by the best between an antibody and its parent. Many tests were done comparing different distance definitions. A niching method, a modified version of Petrowski's clearing procedure [68], was introduced later in [8]. Mechanical as well as structural engineering problems were considered and the GA-AIS found good solutions when compared with other GAs with binary encoding (as the GA-AIS was proposed with binary encoding). A new hybrid algorithm was proposed in [9]. This last algorithm divides the population into feasible and infeasible individuals. A GA is used to optimize the feasible population, as if the problem were unconstrained, and the infeasible individuals are evolved by an AIS to minimize their constraint violations, instead of their distance to feasible individuals. This algorithm has shown good results when applied to a simple structural engineering optimization problem.

Wu [82] combined two techniques in his approach: clonal selection and idiotypic network theory, the later being used to control the number of good solutions. The clonal selection operator explores the search space looking for good solutions and maintaining the diversity of the antibodies' population. The performance of that algorithm was evaluated in constrained optimization problems with continuous variables.

Based on the work by Hajela and Yoo [45], Rajasekaran and Lavanya [71] proposed an immune network for constraint handling in GA. The technique was applied to obtain optimal sectional areas for minimum weight of tri-dimensional truss structures subject to static loading and earthquake ground motion. Test problems were also conducted for the design of the optimal mix of high-performance concrete.

4.3 Combinatorial Optimization Problems

Besides TSP, scheduling problems are very important combinatorial optimization problems. Optimal solutions to such problems are often fragile [50]: if the original problem changes slightly, the existing optimal solution cannot be easily modified, and a new one must be produced. As such changes happen frequently, there is a great interest in the scheduling community in generating robust, good-enough schedules rather than optimal ones. Hart et al.[48, 46] described some results applying an AIS to job-shop scheduling problems exploiting the properties of the immune system to produce robust rather than just optimal schedules. Furthermore, they have shown that scheduling is much more robust using clonal selection than using a simple GA, which may be an interesting line of research of AIS [47].

Coello [23] showed that an AIS algorithm using gene-libraries and a clonal selection mechanism was highly competitive with respect to GRASP [10] for solving a series of 31 benchmark job-shop scheduling problems taken from the OR-library [4]. Their algorithm showed improvements over GRASP on some problems and was less computationally expensive.

Another clonal-selection based algorithm known as ClonaFlex was proposed by Ong et all [67] and applied to a benchmark set of 12 flexible job-shop problems, comparing results against GENACE, a cultural EA [49]. Only one objective function "makespan" is considered and comparable results were found, although practical difficulties in optimising the algorithm parameters for individual problem instances have been observed.

4.4 Immune Programming

This technique [66], inspired by the Genetic Programming (GP) [58] paradigm is also applied to evolve programs in a given language. A candidate solution is often represented as a tree structure and can, in principle, be decoded into a computer program, a numerical function in symbolic form, and also a candidate design, such as an analog circuit. An objective function which translates a conveniently defined performance index of the candidate program, function, or design is then maximized or minimized. The immune programming (IP) technique was applied to symbolic regression (to derive the analytical expression of a function that best approximates a given set of data points) where it performed better than GP in the tests conducted.

In recent work by Cicazzo et al. [19] "elitist Immune Programming", or simply eIP, was proposed. That method is inspired by clonal selection and uses the genetic programming theory to generate the candidate solutions. The elitism is due to the fact that the best individual of the population and its hypermuted clone are always kept. The proposed technique was applied to optimize a circuit design and the results obtained were compared to those given by the standard IP [66], and Koza's GP [59] showing that the eIP algorithm outperforms both in that test problem.

It is felt then that the application of AIS ideas to more complex search spaces also seems to be an important and promising area of research where more research work is needed.

4.5 Multi-Objective Optimization Problems

In real-world problems one is often faced with more than one objective functions that should be maximized or minimized, $f_1(x),\dots,f_m(x)$ where $x \in \Re^n$, f_i is the i-th objective, and m is the number of objectives of the problem. As the objective functions are often non-comensurable and conflicting, it is not usually possible to define a unique optimal solution. By introducing the concept of dominance, the set of non-dominated solutions, often called the Pareto set, becomes the target to be pursued.

Population based algorithms seem to be well suited to the task of approximating the Pareto set, which can then be presented to the decision maker to make the final decision based on his or her set of preferences. A multi-objective optimization problem can be proposed to search a unique solution or a set of solutions (Pareto optimum).

After developing multi-objective clonal selection based algorithms [23], Villalobos-Arias et al. [77] provide a complete proof for their multi-objective immune inspired algorithm. They discuss the B-cell algorithm [73] and how it may be possible to use this algorithm to solve complex optimisation problems, and also analyse the dynamic behavior of the algorithm using dynamical systems theory.

Clark et al. [20] have produced a theoretical analysis of the B-Cell algorithm [73] providing a complete model of the B-cell algorithm with a proof of convergence. In addition, from their model, it seems possible to locate the optimum mutation rate for a given function [47].

Freschi and Repetto [38] observed that opt-IA performs comparably to the fast evolutionary programming (FEP) technique on 23 numeric optimization problems. They also find that their multi-objective immune algorithm, VAIS, is comparable to NSGA2, an evolutionary state-of-the-art multi-objective optimization solver, although their comparative study involved only three problems.

The omni-aiNet [21] is an algorithm proposed to solve single- and multi-objective optimization problems. It is based on opt-iaNet (see Section 3.2) and its search engine adapts itself to the exploration of the search space according to the intrinsic demands of the optimization problem [21]. As with the opt-aiNet, the population size of this method evolves during its execution. The difference between the omni-aiNet and opt-aiNet is that the first one added a variation mechanism called Gene Duplication. This mechanism selects randomly a decision variable i (the individuals are real-coded) of an antibody and replaces its value x_i in all other individuals of the immune system. The method was compared with the original version of the omni-optimizer algorithm [35] when applied to multi-objective optimization problems and the results showed that the omni-aiNet was even capable of outperforming the original algorithm (omni-optimizer) for two of the problems treated.

An immune inspired method was proposed by Zhang [86] for solving constrained nonlinear multi-objective optimization problems. This algorithm was compared with other well-known methods from the literature and performed well when compared with the Strength Pareto Evolutionary Algorithm (SPEA), which is a good multi-objective EA.

5 Conclusions

This chapter presented several computational techniques inspired by the natural immune system and their applications in solving optimization problems. The two most popular immune methodologies, clonal selection and immune networks, have been given special attention.

It seems that AIS algorithms do not outperform other bio-inspired meta-heuristics when solving numerical optimization problems in general, but they can be applied with advantage in certain classes of problems, such as dynamic optimization, or situations where multiple solutions are required. Also, immune programming algorithms, a paradigm that uses complex structures to encode the candidate solutions, have shown good results and seem to be an interesting research field. It is also noted that much more work is required in the three promising application areas mentioned above.

Of course, AISs are also open to profitable hybridization with other available optimization techniques, nature-inspired or classical, in order to improve the overall performance.

One question raised in the literature [47, 74] is whether immune algorithms have added value to the field of optimisation. The authors think AISs have added and will continue to add value to the field, and hope that this chapter will help readers form their own assessment of the subject.

Finally, the AIS area has shown continuous evolution of methods to solve optimization problems, as well as to tackle other complex computational problems. It is expected that there will always be inspiration for conceptualizing and implementing artificial systems derived by new discoveries and theories developed from the highly complex natural immune system.

Acknowledgements. The authors thank the support from CNPq (140551/2008-5 and 311651/2006-2) as well as the corrections and suggestions from the reviewers.

References

1. AISWeb The online home of artificial immune systems (2008) (accessed November 09, 2008), `http://www.artificial-immune-systems.org`
2. Bäck, T., Hoffmeister, F., Schwefel, H.: A survey of evolution strategies. In: Proceedings of the International Conference on Genetic Algorithms, San Diego, CA, pp. 2–9 (1991)
3. Bäck, T., Fogel, D., Michalewicz, Z.: Evolutionary Computation 2: Advanced Algorithms and Operations. Taylor & Francis, Abington (2000)
4. Beasley, J.E.: Or-library: distributing test problems by electronic mail. Journal of the Operational Research Society 41(11), 1069–1072 (1990)
5. Bernardino, H.S.: Hibridização de algoritmos genéticos e sistemas imunológicos artificiais para problemas de otimização com restrições em engenharia. Master's thesis, Universidade Federal de Juiz de Fora - UFJF (2008)
6. Bernardino, H.S., Barbosa, H.J.C., Lemonge, A.C.C.: Constraint handling in genetic algorithms via artificial immune systems. In: Grahl. J. (ed) Late breaking paper at Genetic and Evolutionary Computation Conference (GECCO 2006), Seattle, WA, USA (2006)
7. Bernardino, H.S., Barbosa, H.J.C., Lemonge, A.C.C.: Um algoritmo genético híbrido para problemas de otimização com restrições. In: Proceedings of the XXVII Iberian Latin American Congress on Computational Methods in Engineering - CILAMCE 2006, Belém, Pará, Brazil (2006)
8. Bernardino, H.S., Barbosa, H.J.C., Lemonge, A.C.C.: A hybrid genetic algorithm for constrained optimization problems in mechanical engineering. In: Proceedings of the IEEE Congress on Evolutionary Computation - CEC 2007, pp. 646–653. IEEE Press, Singapore (2007)

9. Bernardino, H.S., Barbosa, H.J.C., Lemonge, A.C.C., Fonseca, L.G.: A new hybrid AIS-GA for constrained optimization problems in mechanical engineering. In: Proceedings of the IEEE Congress on Evolutionary Computation - CEC 2008, pp. 1455–1462. IEEE Press, Hong Kong (2008)
10. Binato, S., Hery, W., Loewenstern, D., Resende, M.: A grasp for job shop scheduling. In: Hansen, P., Ribeiro, C.C. (eds.) Essays and surveys on metaheuristics, pp. 59–79. Kluwer Academic Publishers, Dordrecht (2001)
11. Brownlee, J.: Optimization algorithm toolkit (2008) (accessed November 09, 2008), http://optalgtoolkit.sourceforge.net/
12. Burnet, F.M.: The Clonal Selection Theory of Acquired Immunity. Cambridge University Press, Cambridge (1959)
13. de Castro, L.N., Timmis, J.: An artificial immune network for multimodal function optimization. In: Proceedings of the 2002 IEEE World Congress on Computational Intelligence, Honolulu, Hawaii, USA, vol. I, pp. 669–674 (2002)
14. de Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer, Heidelberg (2002)
15. de Castro, L.N., Zuben, F.J.V.: The clonal selection algorithm with engineering applications. In: Workshop Proceedings of the Genetic and Evolutionary Computation Conference - GECCO 2000, Las Vegas, Nevada, USA, pp. 36–37 (2000)
16. de Castro, L.N., Zuben, F.J.V.: aiNet: An Artificial Immune Network for Data Analysis. Idea Group Publishing, USA (2001)
17. de Castro, L.N., Zuben, F.J.V.: Learning and optimization using the clonal selection principle. IEEE Transactions on Evolutionary Computation 6(3), 239–251 (2002)
18. de Castro, L.N., Zuben, F.J.V.: CLONALG source code (2008) (accessed November 09, 2008), http://www.dca.fee.unicamp.br/~lnunes/manual.html
19. Ciccazzo, A., Conca, P., Nicosia, G., Stracquadanio, G.: An advanced clonal selection algorithm with ad-hoc network-based hypermutation operators for synthesis of topology and sizing of analog electrical circuits. In: Bentley, P.J., Lee, D., Jung, S. (eds.) ICARIS 2008. LNCS, vol. 5132, pp. 60–70. Springer, Heidelberg (2008)
20. Clark, E., Hone, A., Timmis, J.: A markov chain model of the b-cell algorithm. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 318–330. Springer, Heidelberg (2005)
21. Coelho, G.P., Zuben, F.J.V.: omni-aiNet: An immune-inspired approach for omni optimization. In: Proceedings of the International Conference on Artificial Immune Systems, pp. 294–308 (2006)
22. Coello, C.A.C., Cortés, N.C.: Hybridizing a genetic algorithm with an artificial immune system for global optimization. Engineering Optimization 36(5), 607–634 (2004)
23. Coello, C.A.C., Rivera, D.C., Cortés, N.C.: Use of an artificial immune system for job shop scheduling. In: Timmis, J., Bentley, P.J., Hart, E. (eds.) ICARIS 2003. LNCS, vol. 2787, pp. 1–10. Springer, Heidelberg (2003)
24. Colorni, A., Dorigo, M., Maniezzo, V.: Distributed optimization by ant colonies. In: Proceedings of the European Conference on Artificial Life, pp. 134–142. Elsevier, Paris (1991)
25. Cruz-Cortés, N., Trejo-Pérez, D., Coello, C.A.C.: Handling constraints in global optimization using an artificial immune system. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 234–247. Springer, Heidelberg (2005)

26. Cutello, V., Nicosia, G., Pavone, M.: Exploring the capability of immune algorithms: A characterization of hypermutation operators. In: Nicosia, G., Cutello, V., Bentley, P.J., Timmis, J. (eds.) ICARIS 2004. LNCS, vol. 3239, pp. 263–276. Springer, Heidelberg (2004)
27. Cutello, V., Narzisi, G., Nicosia, G., Pavone, M.: Clonal selection algorithms: A comparative case study using effective mutation potentials. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 13–28. Springer, Heidelberg (2005)
28. Cutello, V., Narzisi, G., Nicosia, G., Pavone, M.: An immunological algorithm for global numerical optimization. In: Talbi, E.-G., Liardet, P., Collet, P., Lutton, E., Schoenauer, M. (eds.) EA 2005. LNCS, vol. 3871, pp. 284–295. Springer, Heidelberg (2006)
29. Cutello, V., Nicosia, G., Oliveto, P.S.: Analysis of an evolutionary algorithm with hypermacromutation and stop at first constructive mutation heuristic for solving trap functions. In: Proceedings of the ACM Symposium on Applied computing - SAC 2006, pp. 945–949. ACM, New York (2006)
30. Cutello, V., Nicosia, G., Pavia, E.: A parallel immune algorithm for global optimization. In: Proceedings of the International Conference on Intelligent Information Processing and Web Mining - IIPWM 2006, Ustrón, Poland, pp. 467–475. Springer, Heidelberg (2006)
31. Cutello, V., Nicosia, G., Pavone, M.: Real coded clonal selection algorithm for unconstrained global optimization using a hybrid inversely proportional hypermutation operator. In: Proceedings of the ACM symposium on Applied computing - SAC 2006, pp. 950–954. ACM, New York (2006)
32. Cutello, V., Nicosia, G., Romeo, M., Oliveto, P.: On the convergence of immune algorithms. In: Proceedings of the IEEE Symposium on Foundations of Computational Intelligence - FOCI 2007, Honolulu, Hawaii, USA, pp. 409–415 (2007)
33. Cziko, G.: The Immune System: Selection by the Enemy. In: Without Miracles. The MIT Press, Cambridge (1995)
34. Dasgupta, D.: Artificial Immune Systems and Their Applications, 1st edn. Springer, Heidelberg (1998)
35. Deb, K., Tiwari, S.: Omni-optimizer: A Procedure for Single and Multi-objective Optimization. In: Evolutionary Multi-Criterion Optimization, pp. 47–61 (2005)
36. Dill, K.A.: Theory for the folding and stability of globular proteins. Biochemistry 24(6), 1501–1509 (1985)
37. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: An autocatalytic optimizing process. Tech. Rep. 91-016 Revised, Milano, Italy (1991)
38. Freschi, F., Repetto, M.: Multiobjective optimization by a modified artificial immune system algorithm. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 248–261. Springer, Heidelberg (2005)
39. Garrett, S.M.: Parameter-free, adaptive clonal selection. In: Proceedings of the Congress on Evolutionary Computation - CEC 2004, San Diego, CA, USA, pp. 1052–1058 (2004)
40. Garrett, S.M.: How do we evaluate artificial immune systems? Evolutionary Computation 13(2), 145–177 (2005)
41. Gaspar, A., Collard, P.: From GAs to artificial immune systems: Improving adaptation in time-dependent optimization. In: Proceedings of the Congress on Evolutionary Computation - CEC 1999, pp. 1859–1866. IEEE Press, Los Alamitos (1999)
42. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the Conference on Genetic Algorithms on Genetic algorithms and their application, pp. 41–49. Lawrence Erlbaum Associates, Inc., Mahwah (1987)

43. Hajela, P., Lee, J.: Constrained genetic search via schema adaptation. An immune network solution. In: 1st World Congress of Stuctural and Multidisciplinary Optimization, pp. 915–920. Pergamon Press, Goslar (1995)
44. Hajela, P., Lee, J.: Constrained genetic search via schema adaptation. An immune network solution. Structural Optimization 12, 11–15 (1996)
45. Hajela, P., Yoo, J.S.: Immune network modelling in design optimization. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 167–183. McGraw-Hill, New York (1999)
46. Hart, E., Ross, P.: An immune system approach to scheduling in changing environments. In: Proceedings of the Genetic And Evolutionary Computation Conference - GECCO 1999, pp. 1559–1566 (1999)
47. Hart, E., Timmis, J.: Application areas of ais: The past, the present and the future. Applied Soft Computing 8(1), 191–201 (2008)
48. Hart, E., Ross, P., Nelson, J.: Producing robust schedules via an artificial immune system. In: Proceedings of the Congress on Evolutionary Computation - CEC 1998, pp. 464–469 (1998)
49. Ho, N., Tay, J.: GENACE: an efficient cultural algorithm for solving the flexible job-shop problem. In: Proceedings of the Congress on Evolutionary Computation - CEC 2004, vol. 2, pp. 1759–1766 (2004)
50. Jensen, M.: Generating robust and flexible job shop schedules using genetic algorithms. IEEE Transactions on Evolutionary Computation 1(3), 275–288 (2003)
51. Jerne, N.K.: Towards a network theory of the immune system. Ann. Immunol (Paris) 125C(1-2), 373–389 (1974)
52. Keko, H., Skok, M., Skrlec, D.: Artificial immune systems in solving routing problems. In: The International Conference on Computer as a Tool - EUROCON, pp. 62–66 (2003)
53. Kelsey, J., Timmis, J.: Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 207–218. Springer, Heidelberg (2003)
54. Kelsey, J., Timmis, J., Hone, A.: Chasing chaos. In: Proceedings of the Congress on Evolutionary Computation - CEC 2003, pp. 413–419. IEEE Press, Canberra (2003)
55. Klawonn, F., Höppner, F.: What is fuzzy about fuzzy clustering? understanding and improving the concept of the fuzzifier. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 254–264. Springer, Heidelberg (2003)
56. Ko, A., Lau, H., Lau, T.: An immuno control framework for decentralized mechatronic control. In: Nicosia, G., Cutello, V., Bentley, P.J., Timmis, J. (eds.) ICARIS 2004. LNCS, vol. 3239, pp. 91–105. Springer, Heidelberg (2004)
57. Ko, A., Lau, H., Lau, T.: General suppression control framework: Application in self-balancing robots. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 375–388. Springer, Heidelberg (2005)
58. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems). MIT Press, Cambridge (1992)
59. Koza, J.R., Bennett III, F.H., Andre, D., Keane, M.A.: Synthesis of topology and sizing of analog electrical circuits by means of genetic programming. Computer Methods in Applied Mechanics and Engineering 186(2-4), 459–482 (2000)
60. Lau, H.Y.K., Tsang, W.W.P.: A parallel immune optimization algorithm for numeric function optimization. Evolutionary Intelligence 1(3), 171–185 (2008)
61. Lei, W., Licheng, J.: The immune evolutionary algorithm. In: Proceedings of the third International Conference on Knowledge-Based Intelligent Information Engineering Systems, pp. 99–102 (1999)

62. Liu, L., Xu, W.: An immune-inspired evolutionary fuzzy clustering algorithm based on constrained optimization. In: Proceedings of the International Conference on Intelligent Systems Design and Applications - ISDA 2006, vol. 1, pp. 966–970. IEEE Press, Los Alamitos (2006)
63. Mahfoud, S.W.: Niching methods for genetic algorithms. PhD thesis, Champaign, IL, USA (1995)
64. Middlemiss, M., Whigham, P.A.: Innate and adaptive principles for an artificial immune system. In: Wang, T.-D., Li, X.-D., Chen, S.-H., Wang, X., Abbass, H.A., Iba, H., Chen, G.-L., Yao, X. (eds.) SEAL 2006. LNCS, vol. 4247, pp. 88–95. Springer, Heidelberg (2006)
65. Moscato, P., Fontanari, J.F.: Stochastic versus deterministic update in simulated annealing. Physics Letters A 146, 204–208 (1990)
66. Musilek, P., Lau, A., Reformat, M., Wyard-Scott, L.: Immune programming. Information Sciences 176(8), 972–1002 (2006)
67. Ong, Z.X., Tay, J.C., Kwoh, C.K.: Applying the clonal selection principle to find flexible job-shop schedules. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 442–455. Springer, Heidelberg (2005)
68. Petrowski, A.: A clearing procedure as a niching method for genetic algorithms. In: Proceedings of the third IEEE International Conference on Evolutionary Computation, pp. 798–803 (1996)
69. Rechenberg, I.: Evolution strategy. In: Zurada, J.M., Marks II, R.J., Robinson, C.J. (eds.) Computational Intelligence: Imitating Life, pp. 147–159. IEEE Press, Los Alamitos (1994)
70. Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Transactions on Evolutionary Computation 4(3), 284–294 (2000)
71. Rajasekaran, S., Lavanya, S.: Hybridization of genetic algorithm with immune system for optimization problems in structural engineering. Structural and Multidisciplinary Optimizationn 34(5), 415–429 (2007)
72. Timmis, J.: Artificial immune systems: today and tomorrow. Natural Computing 6(1), 1–18 (2007)
73. Timmis, J., Edmonds, C.: A comment on opt-ainet: An immune network algorithm for optimisation. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 308–317. Springer, Heidelberg (2004)
74. Timmis, J., Edmonds, C., Kelsey, J.: Assessing the performance of two immune inspired algorithms and a hybrid genetic algorithm for function optimisation. In: Proceedings of the Congress of Evolutionary Computation - CEC 2004, pp. 1044–1051 (2004)
75. Timmis, J., Andrews, P., Owens, N., Clark, E.: An interdisciplinary perspective on artificial immune systems. Evolutionary Intelligence 1(1), 5–26 (2008)
76. Toma, N., Endo, S., Yamada, K., Miyagi, H.: Evolutionary optimization algorithm using mhc and immune network. In: Proceedings of the International Conference of the IEEE Industrial Electronics Society - IECON 2000, pp. 2849–2854 (2000)
77. Villalobos-Arias, M., Coello Coello, C.A., Hernández-Lerma, O.: Convergence analysis of a multiobjective artificial immune system algorithm. In: Nicosia, G., Cutello, V., Bentley, P.J., Timmis, J. (eds.) ICARIS 2004. LNCS, vol. 3239, pp. 226–235. Springer, Heidelberg (2004)
78. Walker, J., Garrett, S.: Dyanmic function optimisation: Comparing the performance of clonalg and evolution strategies. In: Timmis, J., Bentley, P.J., Hart, E. (eds.) ICARIS 2003. LNCS, vol. 2787, pp. 273–285. Springer, Heidelberg (2003)

79. Wang, X., Gao, X., Ovaska, S.: An immune-based ant colony algorithm for static and dynamic optimization. IEEE International Conference on Systems, Man and Cybernetics - ISIC 2007, pp. 1249–1255 (2007)
80. Watanabe, K., Campelo, F., Igarashi, H.: Topology optimization based on immune algorithm and multigrid method. IEEE Trans. on Magnetics 43(4), 1637–1640 (2007)
81. Wierzchoń, S.T.: Function optimization by the immune metaphor. Task Quarterly 6(3), 1–16 (2002)
82. Wu, J.Y.: Artificial immune system for solving constrained global optimization problems. In: Artificial Life 2007, ALIFE 2007, Honolulu, HI, pp. 92–99 (2007)
83. Yen, J., Liao, J., Lee, B., Randolph, D.: A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics 28(2), 173–191 (1998)
84. Yoo, J.S., Hajela, P.: Immune network simulations in multicriterion design. Structural Optimization 18, 85–94 (1999)
85. Yuan, H., Li, Y., Li, W., Zhao, K., Wang, D., Yi, R.: Combining immune with ant colony algorithm for geometric constraint solving. In: Proceedings of the International Workshop of Knowledge Discovery and Data Mining - WKDD 2008, pp. 524–527 (2008)
86. Zhang, Z.: Immune optimization algorithm for constrained nonlinear multiobjective optimization problems. Applied Soft Computing 7(3), 840–857 (2007)

Ranking Methods in Many-Objective Evolutionary Algorithms

Antonio López Jaimes, Luis Vicente Santana Quintero, and Carlos A. Coello Coello

Abstract. This chapter presents a comparative study of different ranking methods on many-objective problems. The aim of this work is to investigate the effectiveness of different approaches in order to determine any possible limitations and/or advantages of each of the ranking methods studied and, in general, their performance. Thus, the results may help practitioners to select a suitable ranking method for a problem at hand, and can serve researchers as a guideline to develop new ranking schemes or further extensions of the Pareto optimality relation.

1 Introduction

In many disciplines, optimization problems have two or more objectives, which are normally in conflict with one another, and that we wish to optimize simultaneously. These are called *multi-objective optimization problems* (MOPs), and their solution involves the design of algorithms different from those adopted for dealing with single-objective optimization problems. In single-objective optimization, the determination of the optimum among a set of given solutions is clear. However, in the absence of preference information, in multi-objective optimization there does not exist a unique or straightforward way to determine if a solution is better than other. The notion of optimality most commonly adopted is the one called *Pareto optimality* [27] which leads to trade-offs among the objectives. Thus, by using this relation, it is not possible to obtain a single solution, but instead, we produce a set of them called the *Pareto optimal set*.

Nowadays, Multi-objective Evolutionary Algorithms (MOEAs) have shown an acceptable performance in many real-world problems with their origins in engineering,

Antonio López Jaimes · Luis Vicente Santana Quintero · Carlos A. Coello Coello
CINVESTAV-IPN (Evolutionary Computation Group), Departamento de Computación,
Av. IPN No. 2508, Col. San Pedro Zacatenco, México, D.F. 07360, Mexico
e-mail: `tonio.jaimes@gmail.com`, `lvspenny@hotmail.com`, `ccoello@cs.cinvestav.mx`

scientific and industrial areas [4]. Nonetheless, most of the publications in this area consider problems with only two or three objectives, in spite of the fact that many real-world problems involve a larger number of objectives (4 or more).[1] The MOEAs that are based on traditional Pareto dominance and that are the most representative and most cited in the current literature are: PAES [20], NSGA-II [7], SPEA2 [39] and micro-GA [5]. Besides the difficulty to analyze the Pareto front when there are more than three objectives, recent studies [17, 18, 28, 36] have shown that MOEAs based on Pareto optimality have difficulties to find a good Pareto front approximation in problems with a large number of objectives, which are called *many-objective problems*.[2] One of the reasons for this limitation is that the proportion of nondominated solutions (*i.e.*, equally good solutions regarding Pareto optimality) in a population increases rapidly with the number of objectives. In [14] it is shown that this number goes to infinity when the number of objectives approaches infinity. This implies that in the presence of a large number of objectives the selection of new solutions is carried out almost at random since a large number of the solutions are equally good.

In the current literature we can identify two approaches commonly adopted to cope with many-objective problems, namely: *i*) to propose relaxed forms of Pareto optimality as in [2, 11, 14, 33], and *ii*) to reduce the number of objectives of the problem to ease the decision making or the search processes [3, 8, 21].

Since relaxed forms of Pareto optimality are the most common approach found in literature, in this chapter we present a comparative study that tries to reveal the advantages and disadvantages of some ranking methods used as optimality relations in many-objective problems. That is, methods that induce a partial order in a set of vectors but with a finer grain resolution than the one induced by traditional Pareto optimality. The assessment to the different ranking methods is based on the distribution of the ranks (*i.e.,* the number of ranks and the number of solutions in each rank) and on the plots of the solutions in decision space versus their ranks. The ranking methods considered in this study include redefinitions of the Pareto optimality relation or methods that complement it by inducing a finer ordering on the nondominated solutions found. In Section 5, we will briefly describe the ranking methods considered for this study. These methods were adopted because they follow considerably different approaches, do not require extra parameters and have shown promising results.

The remainder of this chapter is organized as follows. In Section 2, we provide some basic concepts related to multi-objective optimization. We mention the scalability problems of the Pareto optimality relation in Section 3 and the optimality relations are discussed in Section 4. The ranking methods are all discussed in Sections 5 and 6. Section 7 includes the comparison and analysis of the ranking methods.

[1] See for example [23, 29] in which the use of MOEAs in circuit optimization is discussed.

[2] Although this term is commonly used in the specialized literature, there is no consensus about how many objectives are considered ‘many’. However, judging by the scalability difficulties shown by Pareto-based MOEAs, we consider that we deal with a many-objective problem if it has more than 4 objectives.

Finally, in Section 8, we provide our final remarks with respect to the comparative study performed.

2 Background Concepts

2.1 Multi-Objective Optimization Problem (MOP)

The Multi-objective optimization problem can be formally defined as the problem of finding:

$\mathbf{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ which satisfies the m inequality constraints:

$$g_i(\mathbf{x}) \leq 0; i = 1, \ldots, m$$

the p equality constraints:

$$h_i(\mathbf{x}) = 0; i = 1, \ldots, p$$

and optimizes the vector function:

$$\mathbf{f}(\mathbf{x}) = f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})$$

In other words, we aim to determine from among the set $\mathscr{F}$ of all vectors (points) which satisfy the constraints those that yield the optimum values for all the k objective functions simultaneously. The constraints define the feasible region $\mathscr{F}$ and any point $\mathbf{x}$ in the feasible region is called a feasible point.

2.2 Pareto Dominance

Pareto Dominance is formally defined as follows:

A vector $\mathbf{u} = (u_1, \ldots, u_k)$ *is said to dominate* $\mathbf{v} = (v_1, \ldots, v_k)$ *if and only if* $\mathbf{u}$ *is partially less than* $\mathbf{v}$*, i.e.,* $\forall i \in (1, \ldots, k), u_i \leq v_i \wedge \exists i \in (1, \ldots, k) : u_i < v_i$ (assuming minimization).

In order to say that a solution dominates another one, this one needs to be strictly better in at least one objective, and not worse in any of them. So when we are comparing two different solutions A and B, there are 3 possibilities:

- A dominates B
- A is dominated by B
- A and B are nondominated

2.3 Pareto Optimality

The formal definition of *Pareto optimality* is provided next:

A solution $\mathbf{x_u} \in \mathscr{F}$ *(where* $\mathscr{F}$ *is the feasible region) is said to be Pareto optimal if and only if there is no* $\mathbf{x_v} \in \mathscr{F}$ *for which* $\mathbf{v} = f(\mathbf{x_v}) = (v_1, \ldots, v_k)$ *dominates* $\mathbf{u} = f(\mathbf{x_u}) = (u_1, \ldots, u_k)$, where k is the number of objectives.

In other words, this definition says that $\mathbf{x}_u$ is Pareto optimal if there exists no feasible vector $\mathbf{x_v}$ which would decrease some objective without causing a simultaneous increase in at least one other objective (assuming minimization).

This definition does not provide us a single solution (in decision variable space), but a set of solutions which form the so-called *Pareto Optimal Set* (P^*). The vectors that correspond to the solutions included in the Pareto optimal set are *nondominated*.

2.4 Pareto Front

When all the nondominated solutions of a MOP are plotted in objective function space, their nondominated vectors are collectively known as the *Pareto Front* (PF^*). Formally:

$$PF^* := \{\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})) | \mathbf{x} \in P^*\}$$

It is, in general, impossible to find an analytical expression that defines the Pareto front of a problem, so the most common way to obtain the Pareto front is to compute a sufficient number of points in the feasible region, and then filter out the nondominated vectors from them.

The previous definitions are graphically depicted in Figure 1, showing the *Pareto front*, the *Pareto Optimal Set* and the *dominance* relations among solutions. Please refer to [4] for more in-depth information about multi-objective optimization.

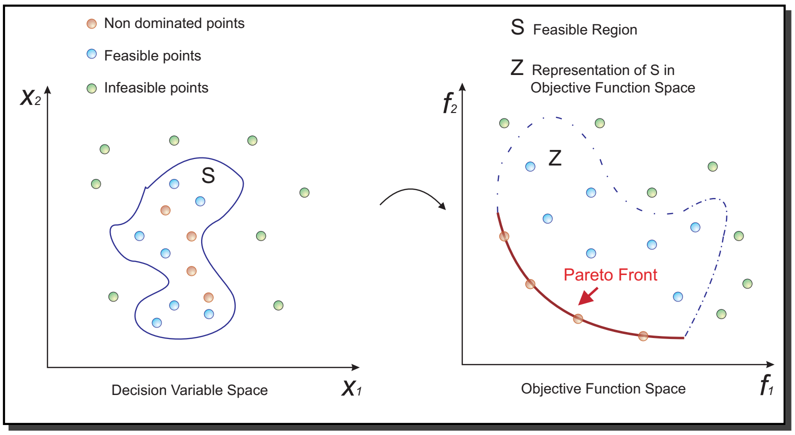


Fig. 1 Mapping of the Pareto Optimal Solutions to the Objective Function Space

3 Scalability Problems When Dealing with Many Objectives

Since the implementation of the first MOEA in the mid-1980s [31], a wide variety of new MOEAs have been proposed, gradually improving in effectiveness and efficiency for solving MOPs. However, the typical validation of such MOEAs is done by adopting test problems with only two or three objectives, and soon researchers realized that the traditional Pareto ranking schemes (in spread use today) scale poorly when the number of objectives increases. It is therefore, a natural step to start designing MOEAs that can deal with problems having a large number of objectives, and therefore the importance of studies such as the one presented in this chapter.

Recent experimental [17, 18, 28, 36] and analytical [6, 19, 34] studies have shown that MOEAs based on Pareto optimality scale poorly in MOPs with a high number of objectives (4 or more). Although this limitation seems to affect only the Pareto-based MOEAs, optimization problems with a large number of objectives (also known as many-objective problems) introduce some difficulties common to any other multi-objective optimizer. Three of the most serious difficulties due to high dimensionality are the following:

1. *Deterioration of the Search Ability*. One of the reasons for this problem is that the proportion of nondominated solutions (*i.e.*, equally good solutions) in a population increases rapidly with the number of objectives [14]. According to Bentley *et al.* [1] the number of nondominated k-dimensional vectors on a set of size n is $O(\ln^{k-1} n)$. This implies that in problems with a large number objectives, the selection of solutions is carried out almost at random or guided by diversity criteria. In fact, Mostaghim and Schmeck [25] have shown that a random search optimizer achieves better results than NSGA-II [7] in a problem with 10 objectives.
2. *Dimensionality of the Pareto front*. Due to the 'curse of dimensionality' the number of points required to represent accurately a Pareto front increases exponentially with the number of objectives. The number of points necessary to represent a k-dimensional Pareto front with resolution r is given by $O(kr^{k-1})$ (*e.g.*, see [32]). This poses a challenge both to the data structures to efficiently manage that number of points and to the density estimators to achieve an even distribution of the solutions along the Pareto front.
3. *Visualization of the Pareto front*. Clearly, with more than three objectives is not possible to plot the Pareto front as usual. This is a serious problem since visualization plays a key role for a proper decision making. Parallel coordinates [38] and self-organizing maps [26] are some of the methods proposed to ease the decision making in many-objective problems. However, more research in this area is required.

Currently, there are mainly two approaches to solve many-objective problems, namely:

1. Adopt or propose an optimality relation that yields a solution ordering finer than that yielded by Pareto optimality. Among these alternative relations we can find average ranking [2], k-optimality [14], preference order ranking [11], favour relation [33], and a method that controls the dominance area [30]. Besides providing

a richer ordering of the solutions, these relations obtain an optimal set that usually is a subset of the Pareto optimal set. Therefore, these techniques can be used as a remedy for the first and second issues of the previous enumeration.

2. Reduce the number of objectives of the problem during the search process or, *a posteriori*, during the decision making process [3, 8, 21]. The main goal of this kind of reduction techniques is to identify the redundant objectives (or redundant to some degree) in order to discard them. A redundant objective is one that can be removed without changing the dominance relation[3] induced by the original objective set.

4 Optimality Relations to Discriminate Solutions

Optimization techniques search a problem domain for the most efficient solution. Some techniques focus on one solution at a time and some other techniques can process a set of solutions (called "population"). The optimizer locates a single point in the objective space, tests it, compares its fitness to the previous best result, and determines the next point to test. Population-based optimizers are more complex, because they need to identify a whole set of points in the objective space, test all the points, rank their fitnesses, and determine the next set of solutions to test.

An important aspect of population-based algorithms is their need to rank the solutions before they are processed by the optimizer. The ranking procedure takes place during the evaluation process, after the objective functions have been evaluated and an array filled with result values has been created. Ranking means to transform the resultant array that is produced by the multi-objective evaluation into a resultant vector. Ranking is required because a set of solutions in the problem domain is being tested and the results have to be presented to the optimization paradigm in a uniform structure. Ranking is fundamental to those methods because it guides the search: the best solutions in a set are given the top ranking. The optimizer relies on the top ranks to identify high-potential regions of the search space to be explored. Therefore, any ranking method that is used needs to be efficient. The ranking method has to be robust enough to handle multiple objectives (i.e., it must be scalable). If the ranking method adopted is not robust, then many different ranking methods will have to be used as the number of objectives increases. To develop a generalized ranking method the following question has to be answered: Can a single ranking method be developed that remains consistent across a range of many objectives?

The taxonomy of approaches that we will cover in this comparative study is shown in Figure 2. These techniques were selected because they were proposed specially for many-objective optimization problems and they represent a substantial difference with respect to the Pareto optimality relation. In the literature, we can find other optimality relations. However, they are minor variations, whether it be of Pareto optimality or some other optimality relation studied here. For instance, the fuzzy optimality relation presented in [37] is a variant of that defined by Farina and

[3] The dominance relation induced by a given set $\mathscr{F}$ of objectives is defined by $\preceq_{\mathscr{F}} = \{(\mathbf{x},\mathbf{y}) | \forall f_i \in \mathscr{F} : f_i(\mathbf{x}) \leq f_i(\mathbf{y})\}$.

Amato [15] and the winning score relation [22] is equivalent to the average ranking method.

In this proposed taxonomy, we divided the ranking techniques in two groups: (1) those that do not need extra parameters to rank all the solutions, and (2) those in which at least one parameter is required to rank the solutions properly. Each of these two groups are discussed in this chapter.

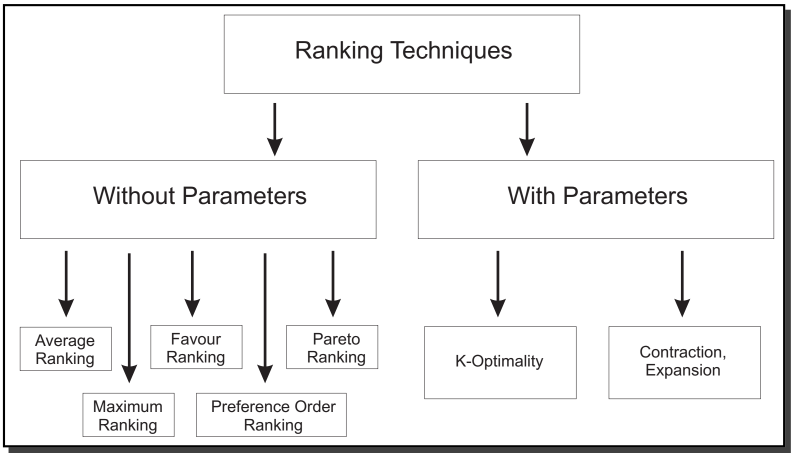


Fig. 2 Mapping of the Pareto Optimal Solutions to the Objective Function Space

5 Ranking Methods without Parameters

5.1 Average and Maximum Ranking Methods

Although without a specific interest in many-objective problems, Bentley and Wakefield [2] proposed three alternative ranking methods to Pareto optimality, namely: average ranking (AR), sum of ratios (SR) and maximum ranking (MR). The AR method computes for each solution a different rank considering each objective independently. The final rank of a solution is obtained by summing all their ranks on each objective. Table 1 illustrates the AR method with a small example with six 3-objective solutions.

In this study, an equivalent method is used to compute the *AR* for each solution. First, the following matrix is defined for solutions $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$a_{ijk} = \begin{cases} 1 \text{ if } f_k(\mathbf{x}_i) < f_k(\mathbf{x}_j) \\ 0 \text{ if } f_k(\mathbf{x}_i) = f_k(\mathbf{x}_j) \\ -1 \text{ if } f_k(\mathbf{x}_i) > f_k(\mathbf{x}_j) \end{cases}$$

From these values, the rank *AR* for each solution $\mathbf{x}_i$ is computed by:

Table 1 An example of the Average, Maximum, Favour and Preference Order ranking methods

				a)	b)	c)	d)
Solution	rank 1	rank 2	rank 3	Average	Maximum	Favour	Preference Order
(4, 3, 5)	3	3	4	10	3	3	3
(1, 4, 7)	1	4	6	11	1	3	2
(6, 2, 2)	4	2	1	7	1	1	1
(7, 7, 6)	5	5	5	15	5	4	5
(8, 1, 3)	6	1	2	9	1	2	3
(3, 8, 4)	2	6	3	11	2	3	4

$$AR(\mathbf{x}_i) = KN - \sum_{k=1}^{K} \sum_{j \neq i}^{N} a_{ijk},$$

where K is the number of objectives, N the number of solutions.

The maximum ranking takes the best rank as the global rank for each solution. Clearly this method favors extreme solutions, i.e., solutions with high performance in some of the objectives, although with poor overall performance. Table 1 shows an example of the use of this method.

5.2 Favour Ranking

This ranking method was proposed by Drechsler in [13] and consists of a new relation called *favour*. This technique requires no user interaction and can handle infeasible solutions.

$$x <_f y \Leftrightarrow |i : f_i(x) < f_i(y), 1 \leq i \leq n| > |j : f_j(y) < f_j(x), 1 \leq j \leq n| \qquad (1)$$

This means that x is *favoured* to y ($x <_f y$) iff i components of x are better than the corresponding components of y and only j components of y are better than the corresponding components of x. For example: $f_{x_1} = (1,1,2)$ and $f_{x_2} = (5,3,1)$, then we have that: $f_{x_1} <_f f_{x_2}$.

Also, in this model, the authors proposed that the solutions are divided into so-called *Satisfiability Classes* (SCs) depending on their quality. Solutions of the same quality belong to the same SC. This property helps the mechanism in using a graph representation to describe properly the relation *favour* ($<_f$) , in which each element is a node and preferences are given by edges. The relation $<_f$ is not transitive, thus the relation $<_f$ can generate “cycles” in the graph, causing elements that describe a cycle to be denoted as not comparable and they are included in the same SC given the same rank to each of the elements in that cycle. The graph that contains all the populations (G_Z) needs to be a directed graph without internal cycles, so all the cycles have to be identified and replaced by a single node representing the single cycle.

Once we have the final directed graph (G_Z), we know that (G_Z) is acyclic, and it is possible to determine a *level sorting* of the nodes. For each node in G_Z we define a SC. Level sorting of the nodes in G_Z determines the ranking of the SCs; each level contains at least one node of G_Z. Then each level corresponds to exactly one SC. Using the level sorting, it is possible to group nodes (sets of solutions) that are not connected by an edge in the same SC. These solutions are not comparable with respect to relation $<_f$ and thus they should be ranked in the same level of quality.

For the case n = 2 it holds that the $<_f$ and $<_d$ are equal, where $<_d$ is the Pareto dominance relation. So, when n = 2, the favour relation is exactly the same as the Pareto dominance relation. Relation $<_f$ can handle infeasible solutions. When an element is infeasible, the element is considered as the worst possible value. The computation time for the SC classification is $O(|P|^2 \cdot n)$, where P is the set of solutions.

5.3 Preference Order Ranking

This is a ranking procedure that exploits the definition of preference order (PO) proposed by di Pierro in [12]. The preference order definition is:

A point $\mathbf{x}^* \in \omega$ *is considered efficient in order k if* $f(\mathbf{x}^*)$ *is not dominated by any member of P for any of the k-element subsets of the objectives. In other words, a point is efficient of order k if it is a Pareto optimal in all the* $\binom{m}{k}$ *subspaces of F obtained considering only k objectives at a time.*

It is clear that the efficiency of order m for an MOP with exactly m objectives simply enforces Pareto optimality.

The condition of efficiency of order can be used to help reduce the number of points in a set by retaining only those that are regarded as "best compromises". In fact, it is intuitive that the less extreme components a point has, the more likely it is to be efficient of order. When the number of points selected is still considerably large, a more stringent criterion is required to sort out better solutions.

Definition (Efficiency of order k with degree z): A point $\mathbf{x}^*$ is said to be efficient of order k with degree z if it is not dominated by any member of P for exactly z out of the possible $\binom{m}{k}$ – element subsets.

At every generation t from a population P:

1. Identify the Pareto nondominated solutions of P and group them into the subset $R^{(1)}$, which is given rank 1.
2. Assign to the individuals of $R^{(1)}$ a rank according to a strategy based on Preference Order with Degree z and the worst given rank is w.
3. Identify the Pareto nondominated individuals of $P \setminus R^{(1)}$ and group them into the subset $R^{(2)}$, which will be given rank $w+s$.
4. Iterate (Step 2) and (Step 3) until $P \setminus R^{(s)} = \emptyset$, where $R^{(s)}$ is the subset that contains the worst individuals.

6 Ranking Methods with Parameters

6.1 K-Optimality

Farina and Amato [15] proposed an alternative relation which takes into account the number of improved objectives between two solutions. This relation employs three quantities, $n_b(\mathbf{x}_1,\mathbf{x}_2), n_e(\mathbf{x}_1,\mathbf{x}_2)$ and $n_w(\mathbf{x}_1,\mathbf{x}_2)$, which denote the objectives where $\mathbf{x}_1$ is better, equal or worse than $\mathbf{x}_2$, respectively. Using these, the concepts of $(1-k)$-*Dominance* and *k-Optimality* are defined. A solution $\mathbf{x}_1$ $(1-k)$-dominates $\mathbf{x}_2$ if and only if

$$\begin{cases} n_e(\mathbf{x}_1,\mathbf{x}_2) < M \\ n_b(\mathbf{x}_1,\mathbf{x}_2) \geq \frac{M-n_e}{k+1} \end{cases}$$

In a similar way to Pareto optimality, a solution $\mathbf{x}^*$ is *k-optimum* if and only if there is no $\mathbf{x}$ in the decision variable space such that $\mathbf{x}$ k-dominates $\mathbf{x}^*$.

Then, this definition is fuzzificated by introducing fuzzy numbers to define $n_b(\mathbf{x}_1,\mathbf{x}_2), n_e(\mathbf{x}_1,\mathbf{x}_2)$ and $n_w(\mathbf{x}_1,\mathbf{x}_2)$. Finally, they propose a further extension that introduces a fuzzy definition for the Pareto dominance relation itself.

6.2 Contraction – Expansion

Sato, Aguirre and Tanaka [30] proposed a method to control the dominance area of solutions. This method can control the degree of expansion or contraction of the dominance area adopting a user-defined parameter S. To contract and expand the dominance area of solutions, the authors modify the fitness value for each objective function by changing the user defined parameter S_i in the following equation:

$$f_i'(\mathbf{x}) = \frac{r \cdot \sin(\omega_i + S_i \cdot \pi)}{\sin(S_i \cdot \pi)} \ \forall\, i = 1,2,\ldots,m$$

where r is the norm of $f(\mathbf{x})$, $f_i(\mathbf{x})$ is the fitness value in the $i-th$ objective, and ω_i is the declination angle between $f(\mathbf{x})$ and $f_i(\mathbf{x})$.

If the user adopts a value of $S_i < 0.5$, the dominance area is expanded and produces a more fine grained ranking of solutions and would strengthen selection. On the other hand, if the user sets $S_i > 0.5$, the dominance area is contracted from the original one and produces a coarser ranking of solutions, weakening the selection procedure.

7 Analysis of Parameterless Ranking Methods

7.1 Ranking Distributions

One criterion to estimate the quality of a ranking method is to analyze the distribution of the ranks assigned to a set of solutions. A ranking method will favor the selection process if it is able to generate a richer range of ranks. To measure the scalability of a method with respect to the number of objectives we can determine if the shape and range of the distribution is maintained when the number of objectives is incremented. Along with the four ranking methods described in Section 5, a Pareto-based ranking method is included as a reference to compare the other ranking methods. The Pareto-based ranking method used is Fonseca and Fleming's method [16] which ranks solutions based on the Pareto dominance relation. The ranking distributions presented in this section belong to the ranking of 10000 random solutions for the problems DTLZ2 and DTLZ7 described in Table 2. For each problem we used 3, 4, 5, 8, 10, 15 and 20 objectives.

With regard to problems DTLZ2 and DTLZ7, for every number of objectives considered, the Pareto-based ranking method concentrates most of the solutions under rank 1 and the frequency for the worst ranks quickly approaches zero (see Figures 3 and 4). This behavior provides few different ranks to the selection process. In DTLZ7, for 3 objectives, about 60% of the solutions have rank 1 (see Figure 4). By observing the distributions for more objectives we can appreciate a phenomenon previously reported in the specialized literature [17, 18, 19, 36]. That is, the number of nondominated solutions (rank 1) increases quickly with the number of objectives. For example, for 8 objectives, around 80% of the solutions are nondominated in both MOPs, while for 15 and 20 objectives, all the solutions are nondominated in DTLZ7.

The distribution of the maximum ranking method (MR) for 3 and 4 objectives in DTLZ7 is similar to that of the Pareto-based ranking method. However only about 32% of the solutions have the best rank and, consequently, there is a larger range of ranks available. This behavior changes for more than 5 objectives. For that number of objectives, most of the solutions are assigned medium ranks, some solutions have the worst rank, and just a few solutions have rank 1 (only 2%). When the number of objectives is increased, the number of solutions with the best rank decreases while the number of solutions with the worst rank increases. For 10 objectives, for instance, 80% of the solutions have the worst rank and only 1% have the best rank. We believe that a distribution where the worst solutions represent the majority of the population may hinder the progress towards the Pareto front. For instance, if we carried out a tournament selection, most of the tournaments would include bad solutions. Good solutions would have small chances of survival. With respect to DTLZ2, MR produces a distribution with two tails until 10 objectives, since when the number of objectives is increased the range of ranking values is reduced (see Figure 3). For 10 or more objectives, approximately 80% of the solutions have the best rank. This means that the maximum ranking method, although scales better than the Pareto-based method, has poor scalability with respect to the other methods studied.

In both problems, the preference order ranking (POR) method also presents a skewed right distribution where the frequency of the ranks decreases slowly. Nonentheless, in contrast with the previous ranking methods, the POR method's distribution is conserved for all the objectives considered, although the range of the distribution is reduced as the number of objectives is incremented. This distribution suggests that the POR method scales well with the number of objectives.

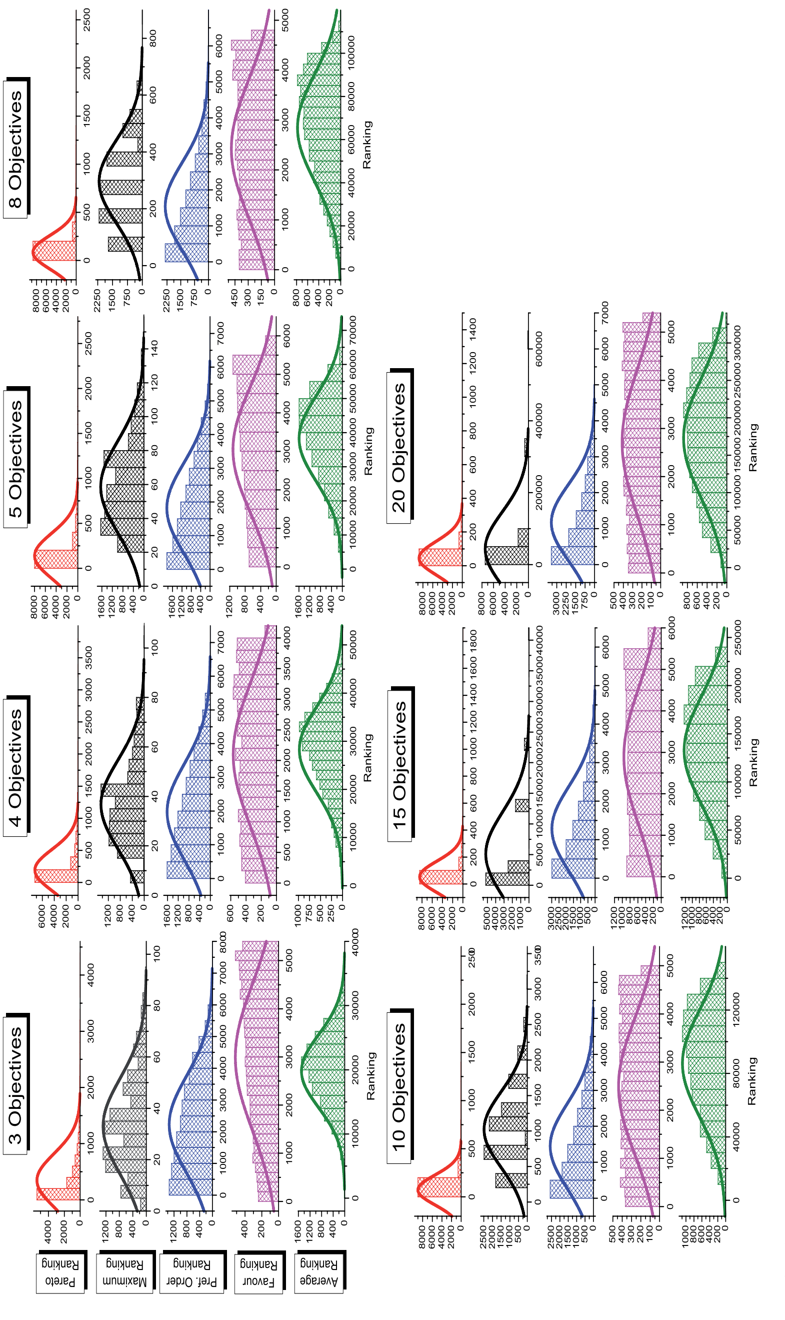


Fig. 3 A histogram that shows the density of the rankings per each different ranking method for the DTLZ2 test problem

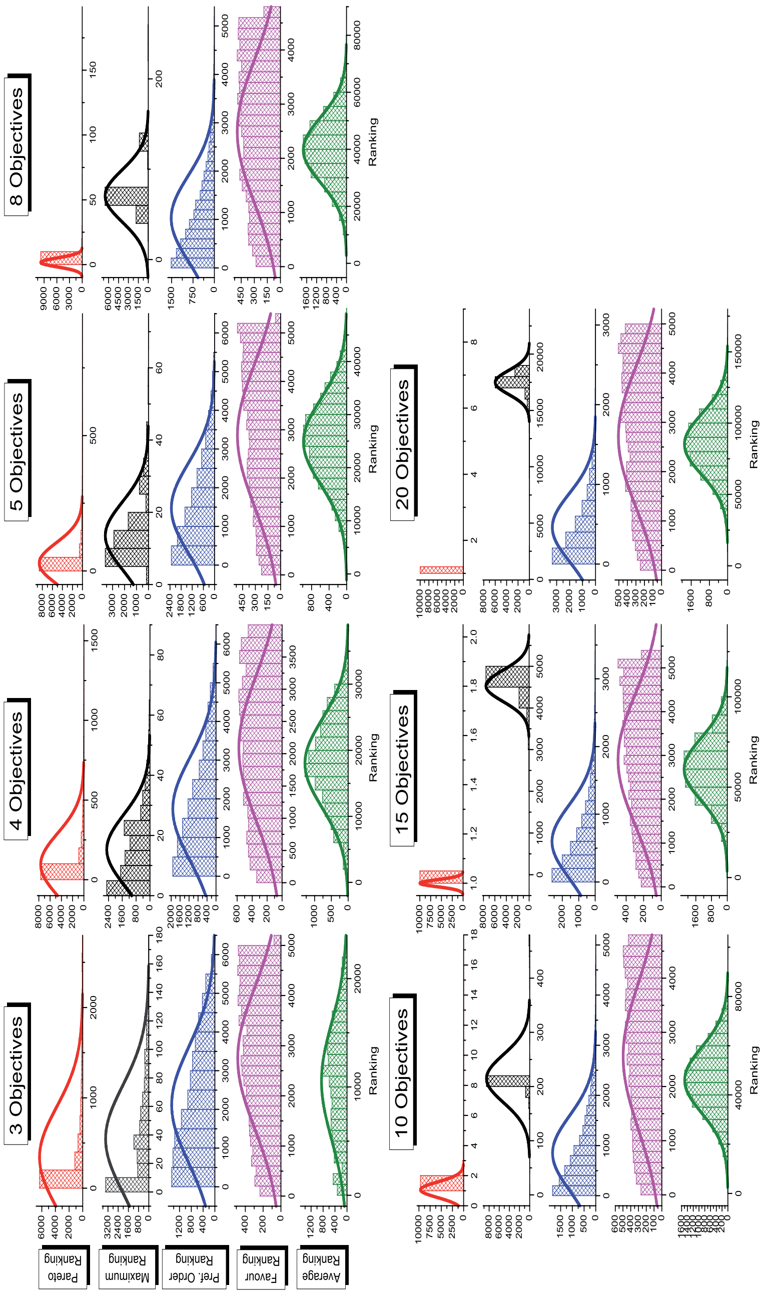


Fig. 4 A histogram that shows the density of the rankings per each different ranking method for the DTLZ7 test problem

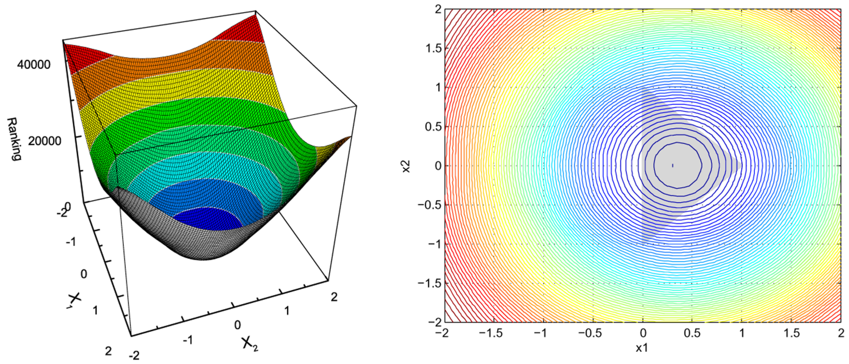


Fig. 5 Average Ranking's ranking landscape and contour - Viennet

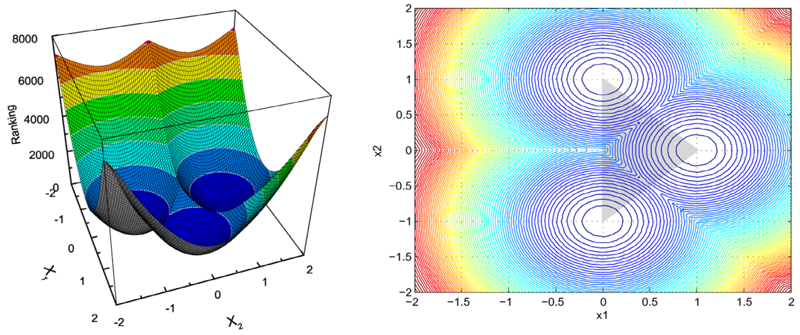


Fig. 6 Maximum Ranking - Viennet

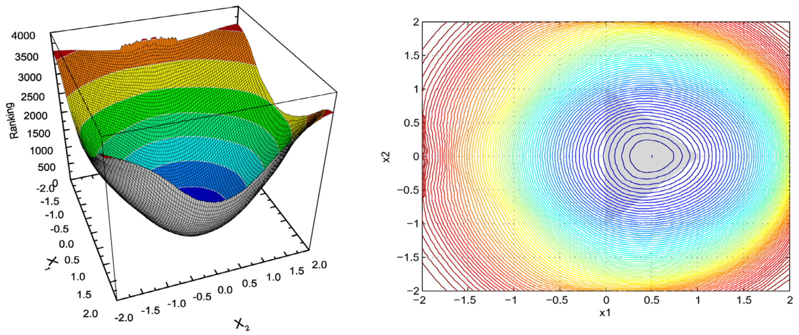


Fig. 7 Favour Ranking - Viennet

The favour ranking method maintains well the shape and range of its rank distribution through all the objectives considered and for the two problems. However, it is the only ranking method that shows for all objectives a slightly skewed left distribution where all the ranks have a similar frequency distribution.

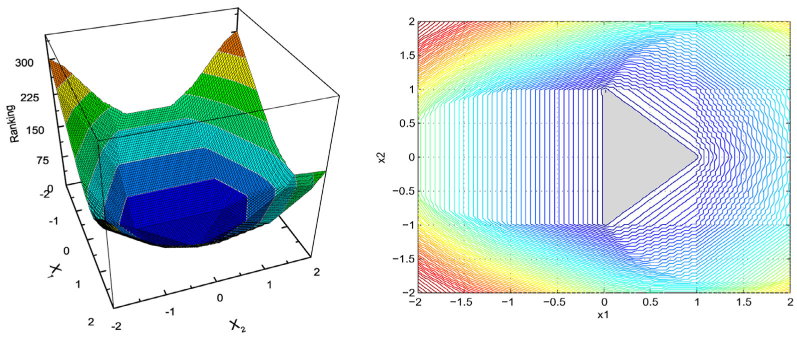


Fig. 8 Preference Order Ranking - Viennet

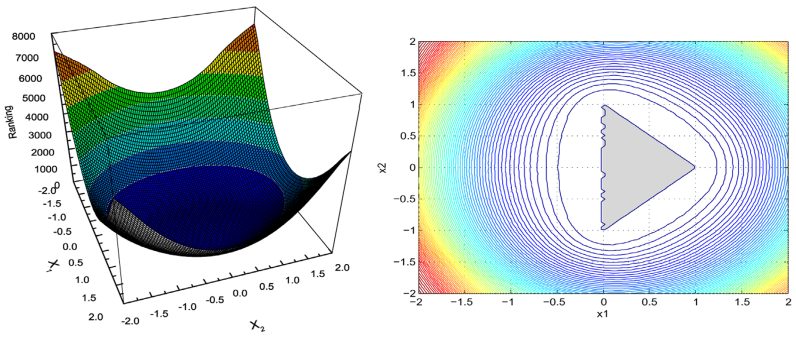


Fig. 9 Pareto Ranking - Viennet

In DTLZ7, the AR method presents a bell-shaped distribution which is more defined as the number of objectives is increased. In contrast to the other ranking methods, the range of the AR's distribution is increased with the number of objectives. With respect to DTLZ2, for a high number of objectives this method produces a slightly skewed distribution.

7.2 Ranking Landscapes

Similar to the fitness landscapes used in single-objective optimization, it is possible to visualize the behavior of a ranking method by plotting the variables against the ranking values assigned to each point of the decision space. To make this visualization possible the MOP should have at most two variables. In this study we adopted two multiobjective problems with only two variables. A 3-objective problem defined by Viennet [35] and a 5-objective problem proposed by [24]. Figures 5–9 and 10–14 show the ranking landscape generated by each ranking method in the Viennet's and Miettinen's problems respectively. Each ranking landscape is accompanied by its isocontour plot where the Pareto optimal set is shown with a shaded region.

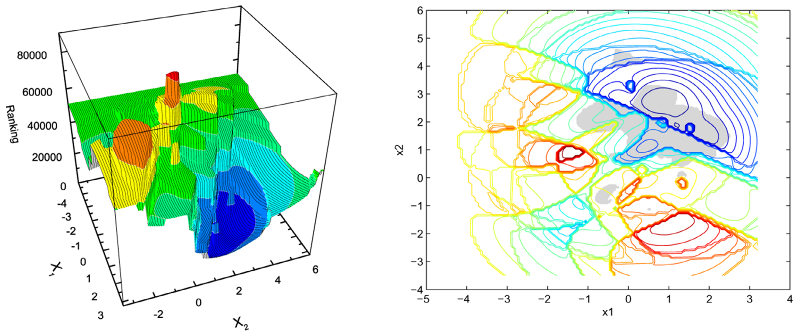


Fig. 10 Average Ranking - Miettinen

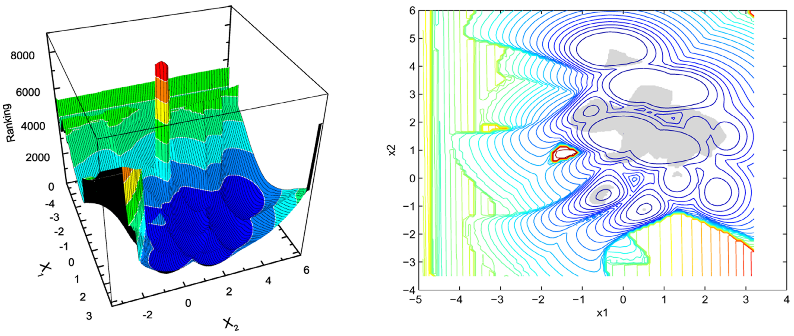


Fig. 11 Maximum Ranking - Miettinen

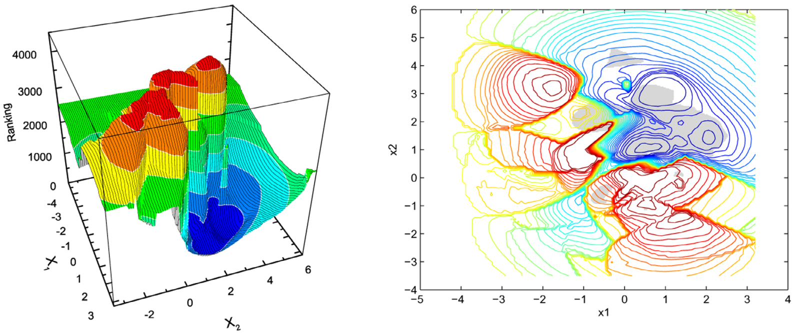


Fig. 12 Favour Ranking - Miettinen

The ranking landscape for Viennet's MOP presents a smooth and unimodal surface for all the ranking methods except for the maximum ranking method (see Figure 6). It is interesting to note that, since this method favors extreme solutions, the surface for this method has three local optima, one for each objective of the

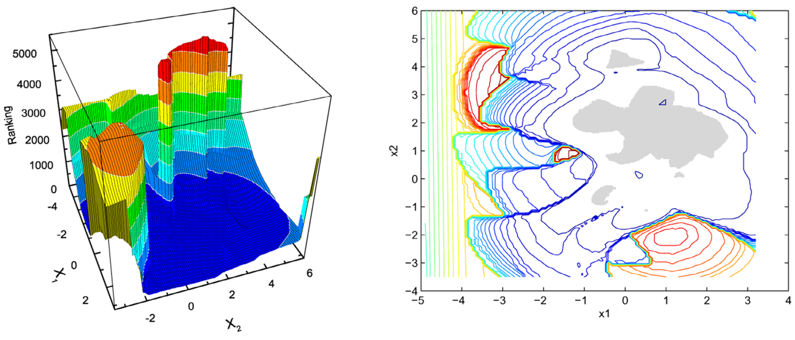


Fig. 13 Preference Order Ranking - Miettinen

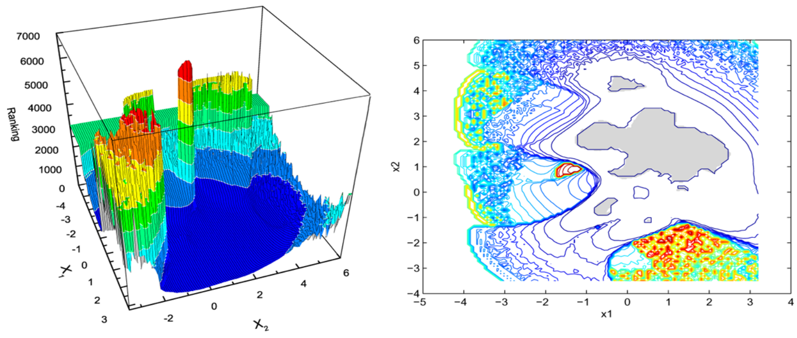


Fig. 14 Pareto Ranking - Miettinen

problem. Even so, the surface generated for all the methods would allow an optimizer to converge easily towards the optimal solutions.

With respect to Miettinen's MOP (see Figures 10–14), the ranking lanscape generated by all the ranking methods presents a multi-modal surface. All the surfaces have peaks and plateaus which hinder the convergence towards the optimal solutions. There are some interesting observations about these ranking landscapes. First, the isocontour plots show clearly that the ranking methods converge only to some regions of the Pareto optimal set and, consequently, some regions of the corresponding Pareto front. For example, the average and favour ranking methods only cover the upper part of the Pareto optimal set. The preference order ranking method is the only one that converges to a region more similar to the Pareto optimal set. Secondly, if we see the peaks near the optimal regions we can realize that some solutions (those in the top of the peaks) may receive worst ranks than those solutions behind the peaks but farther from the optimal region. That is, the peaks act as a barrier that keeps them from reaching the optimal solutions. It is interesting to note that the maximum ranking method generates a smoother surface with only one peak before the optimal region. This fact suggests that the maximum ranking method is useful for approaching quickly the Pareto optimal solution, although without covering the whole set.

Table 2 MOP Test Functions

Function	Definition
DTLZ2 [9, 10]	$F = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x}))$, where: $$\begin{aligned} f_1(\mathbf{x}) &= (1+g(\mathbf{x_M}))\cos(\frac{\pi}{2}x_1)\cos(\frac{\pi}{2}x_2)\ldots\cos(\frac{\pi}{2}x_{M-2})\cos(\frac{\pi}{2}x_{M-1}) \\ f_2(\mathbf{x}) &= (1+g(\mathbf{x_M}))\cos(\frac{\pi}{2}x_1)\cos(\frac{\pi}{2}x_2)\ldots\cos(\frac{\pi}{2}x_{M-2})\sin(\frac{\pi}{2}x_{M-1}) \\ f_3(\mathbf{x}) &= (1+g(\mathbf{x_M}))\cos(\frac{\pi}{2}x_1)\cos(\frac{\pi}{2}x_2)\ldots\cos(\frac{\pi}{2}x_{M-2}) \\ &\vdots \quad \vdots \\ f_{M-1}(\mathbf{x}) &= (1+g(\mathbf{x_M}))\cos(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2) \\ f_M(\mathbf{x}) &= (1+g(\mathbf{x_M}))\sin(\frac{\pi}{2}x_1) \\ &\text{where:} \\ g(\mathbf{x_M}) &= \sum_{x_i \in x_M}(x_i-0.5)^2 \\ x_i \in [0,1] &\;\forall\; i=1,2,\ldots,n \\ n=M+k-1 &\;,\; k=10 \end{aligned}$$
DTLZ7 [9, 10]	$F = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x}))$, where: $$\begin{aligned} f_1(\mathbf{x}) &= x_1 \\ f_2(\mathbf{x}) &= x_2 \\ f_3(\mathbf{x}) &= x_3 \\ &\vdots \quad \vdots \\ f_{M-1}(\mathbf{x}) &= x_{M-1} \\ f_M(\mathbf{x}) &= (1+g(\mathbf{x_M}))h(f_1,f_2,\ldots,f_{M-1},g(\mathbf{x})) \\ &\text{where:} \\ g(\mathbf{x}) &= 1+\frac{9}{\lvert\mathbf{x}_m\rvert}\sum_{x_1 \in x_M} x_i \\ h(f_1,f_2,\ldots,f_{M-1},g(\mathbf{x})) &= M-\sum_{i=1}^{M-1}\left(\frac{f_i}{1+g(\mathbf{x})}(1+\sin(3\pi f_i))\right) \\ x_i \in [0,1] &\;\forall\; i=1,2,\ldots,n \\ n=M+k-1 &\;,\; k=10 \end{aligned}$$
Viennet [35]	$F = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}))$, where: $$\begin{aligned} f_1(x,y) &= 0.5*(x^2+y^2)+sin(x^2+y^2), \\ f_2(x,y) &= \frac{(3x-2y+4)^2}{8}+\frac{(x-y+1)^2}{27}+15, \\ f_3(x,y) &= \frac{1}{(x^2+y^2+1)}-1.1e^{(-x^2-y^2)} \\ &x \in [-3,3] \\ &y \in [-3,3] \end{aligned}$$

Table 2 (*continued*)

Function	Definition
Miettinen [24]	$F = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}), f_4(\mathbf{x}), f_5(\mathbf{x}))$, where: $\begin{aligned} f_1(x_1,x_2) &= f(x_1,x_2), \\ f_2(x_1,x_2) &= f(x_1-1.2, x_2-1.5), \\ f_3(x_1,x_2) &= f(x_1+0.3, x_2-3.0), \\ f_4(x_1,x_2) &= f(x_1-1.0, x_2+0.5), \\ f_5(x_1,x_2) &= f(x_1-0.5, x_2-1.7), \\ \text{where:} & \\ f(x_1,x_2) &= -u_1(x_1,x_2) - u_2(x_1,x_2) - u_3(x_1,x_2) + 10 \\ u_1(x_1,x_2) &= 3(1-x_1)^2 \exp(-x_1^2 - (x_2+1)^2) \\ u_2(x_1,x_2) &= -10\left((1/4)x_1 - x_1^3 - x_2^5\right)\exp(-x_1^2 - x_2^2) \\ u_3(x_1,x_2) &= (1/3)\exp\left(-(x_1+1)^2 - x_2^2\right) \\ & x_1 \in [-4.9, 3.2] \\ & x_2 \in [-3.5, 6] \end{aligned}$

8 Conclusion and Future Work

In this study we have compared some ranking methods with respect to their rank distribution and their ranking landscape which is the surface generated by the ranks assigned to solutions. The inspection of the rank distribution provided a guide to determine the scalability of the ranking methods and their possible disadvantages in the search process. The ranking landscapes allowed us to observe easily how the ranking method could assist or hinder the progress towards the Pareto optimal set.

One of our findings is that the preference order ranking is the method with the best scalability among all the methods included in this study. Also it shows a distribution similar to the one produced by Pareto-based ranking methods with two or three objectives. This behavior suggests that the introduction of preference order ranking would perform effectively if incorporated into a MOEA. Another finding is that although maximum ranking does not induce a promising ranking distribution, its ranking landscape suggests that it can be used to reach quickly some regions of the Pareto optimal set. In addition, the ranking landscapes allow us to see that each ranking method converges to a different subset of the Pareto optimal set. That is, some methods cover the Pareto front better than others. This means that if we want to find the whole Pareto front using some of these ranking methods we have to use an additional technique or to modify the method to achieve this.

As part of our future work we want to incorporate the ranking methods included in this chapter into a MOEA in order to correlate some features observed here with convergence capabilities.

Acknowledgements. The authors thank the anonymous reviewers for their comments, which greatly helped to improve the contents of this chapter.

The first two authors acknowledge support from CONACyT through a scholarship to pursue graduate studies at the Computer Science Department of CINVESTAV-IPN. The third author acknowledges partial support from CONACyT project number 45683-Y and from the "Laboratorio Franco-Mexicano de Informática y Automática Aplicada UMI-LAFMIA".

References

1. Bentley, J.L., Kung, H.T., Schkolnick, M., Thompson, C.D.: On the Average Number of Maxima in a set of Vectors and Applications. Journal of the ACM 25(4), 536–543 (1978)
2. Bentley, P.J., Wakefield, J.P.: Finding Acceptable Solutions in the Pareto-Optimal Range using Multiobjective Genetic Algorithms. In: Chawdhry, P.K., Roy, R., Pant, R.K. (eds.) Soft Computing in Engineering Design and Manufacturing, Part 5, pp. 231–240. Springer, London (1997)
3. Brockhoff, D., Zitzler, E.: Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 533–542. Springer, Heidelberg (2006)
4. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007)
5. Coello Coello, C.A., Toscano Pulido, G.: A Micro-Genetic Algorithm for Multiobjective Optimization. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 126–140. Springer, Heidelberg (2001)
6. Corne, D., Knowles, J.: Techniques for Highly Multiobjective Optimisation: Some Non-dominated Points are Better than Others. In: Thierens, D. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007), vol. 1, pp. 773–780. ACM Press, New York (2007)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
8. Deb, K., Saxena, D.K.: Searching for Pareto-optimal Solutions through Dimensionality Reduction for certain Large-dimensional Multi-objective Optimization Problems. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2006), pp. 3353–3360. IEEE, Los Alamitos (2006)
9. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Multi-Objective Optimization Test Problems. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), vol. 1, pp. 825–830. IEEE, Los Alamitos (2002)
10. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In: Abraham, A., Jain, L., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization. Theoretical Advances and Applications, pp. 105–145. Springer, USA (2005)
11. di Pierro, F.: Many-Objective Evolutionary Algorithms and Applications to Water Resources Engineering. Ph.D. thesis, School of Engineering, Computer Science and Mathematics, UK (2006)
12. di Pierro, F., Khu, S.T., Savić, D.A.: An Investigation on Preference Order Ranking Scheme for Multiobjective Evolutionary Optimization. IEEE Transactions on Evolutionary Computation 11(1), 17–45 (2007)
13. Drechsler, N., Drechsler, R., Becker, B.: Multi-Objected Optimization in Evolutionary Algorithms Using Satisfyability Classes. In: Reusch, B. (ed.) Fuzzy Days 1999. LNCS, vol. 1625, pp. 108–117. Springer, Heidelberg (1999)

14. Farina, M., Amato, P.: On the Optimal Solution Definition for Many-criteria Optimization Problems. In: Proceedings of the NAFIPS-FLINT International Conference 2002, pp. 233–238. IEEE, Piscataway (2002)
15. Farina, M., Amato, P.: A Fuzzy Definition of "Optimality" for Many-criteria Optimization Problems. IEEE Transactions on Systems, Man, and Cybernetics Part A—Systems and Humans 34(3), 315–326 (2004)
16. Fonseca, C.M., Fleming, P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: Forrest, S. (ed.) Proceedings of the 5th International Conference on Genetic Algorithms, University of Illinois at Urbana-Champaign, pp. 416–423. Morgan Kauffman Publishers, San Mateo (1993)
17. Hughes, E.J.: Evolutionary Many-Objective Optimisation: Many Once or One Many? In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005), vol. 1, pp. 222–227. IEEE, Edinburgh (2005)
18. Khare, V., Yao, X., Deb, K.: Performance scaling of multi-objective evolutionary algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 376–390. Springer, Heidelberg (2003)
19. Knowles, J., Corne, D.: Quantifying the Effects of Objective Space Dimension in Evolutionary Multiobjective Optimization. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 757–771. Springer, Heidelberg (2007)
20. Knowles, J.D., Corne, D.W.: Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. Evolutionary Computation 8(2), 149–172 (2000)
21. López Jaimes, A., Coello Coello, C.A., Chakraborty, D.: Objective Reduction Using a Feature Selection Technique. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008), pp. 674–680. ACM Press, Atlanta (2008)
22. Maneeratana, K., Boonlong, K., Chaiyaratana, N.: Compressed-Objective Genetic Algorithm. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 473–482. Springer, Heidelberg (2006)
23. McConaghy, T., Palmers, P., Gielen, G., Steyaert, M.: Simultaneous Multi-topology Multi-objective Sizing across Thousands of Analog Circuit Topologies. In: Proceedings of the 44th annual conference on Design Automation (DAC 2007), pp. 944–947. ACM Press, San Diego (2007)
24. Miettinen, K., Lotov, A., Kamenev, G., Berezkin, V.: Integration of Two Multiobjective Optimization Methods for Nonlinear Problems. Optimization Methods and Software 18(1), 63–80 (2003)
25. Mostaghim, S., Schmeck, H.: Distance based Ranking in Many-objective Particle Swarm Optimization. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 753–762. Springer, Heidelberg (2008)
26. Obayashi, S., Sasaki, D.: Visualization and Data Mining of Pareto Solutions Using Self-Organizing Map. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 796–809. Springer, Heidelberg (2003)
27. Pareto, V.: Cours D'Economie Politique. F. Rouge (1896)
28. Praditwong, K., Yao, X.: How Well Do Multi-objective Evolutionary Algorithms Scale to Large Problems. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007), pp. 3959–3966. IEEE, Singapore (2007)
29. Rutenbar, R.A., Gielen, G.G., Roychowdhury, J.: Hierarchical Modeling, Optimization, and Synthesis for System-Level Analog and RF Designs. Proceedings of the IEEE 95(3), 640–669 (2007)

30. Sato, H., Aguirre, H.E., Tanaka, K.: Controlling Dominance Area of Solutions and Its Impact on the Performance of MOEAs. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 5–20. Springer, Heidelberg (2007)
31. Schaffer, J.D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, pp. 93–100. Lawrence Erlbaum, Mahwah (1985)
32. Sen, P., Yang, J.: Multiple Criteria Decision Support in Engineering Design. Springer, London (1998)
33. Sülflow, A., Drechsler, N., Drechsler, R.: Robust multi-objective optimization in high dimensional spaces. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 715–726. Springer, Heidelberg (2007)
34. Teytaud, O.: How Entropy-Theorems can show that Approximating High-dim Pareto-fronts is too Hard. In: Bridging the Gap between Theory and Practice - Workshop PPSN-BTP, International Conference on Parallel Problem Solving from Nature PPSN IX, Reykjavik, Iceland (2006)
35. Viennet, R., Fontiex, C., Marc, I.: Multicriteria Optimization Using a Genetic Algorithm for Determining a Pareto Set. International Journal of Systems Science 27(2), 255–260 (1996)
36. Wagner, T., Beume, N., Naujoks, B.: Pareto-, Aggregation-, and Indicator-Based Methods in Many-Objective Optimization. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 742–756. Springer, Heidelberg (2007)
37. Wang, G., Wu, J.: A New Fuzzy Dominance GA Applied to Solve Many-objective Optimization Problem. In: Proceedings of the 2nd International Conference on Innovative Computing, Information and Control (ICICIC 2007), p. 617. IEEE Computer Society, Washington (2007)
38. Wegman, E.J.: Hyperdimensional Data Analysis using Parallel Coordinates. Journal of the American Statistical Association 85, 664–675 (1990)
39. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In: Giannakoglou, K., Tsahalis, D., Periaux, J., Papailou, P., Fogarty, T. (eds.) EUROGEN 2001 - Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Athens, Greece, pp. 95–100 (2001)

On the Effect of Applying a Steady-State Selection Scheme in the Multi-Objective Genetic Algorithm NSGA-II

Antonio J. Nebro and Juan J. Durillo

Abstract. Genetic Algorithms (GAs) are among the most popular techniques to solve multi-objective optimization problems, with NSGA-II being the most well-known algorithm in the field. Although most of multi-objective GAs (MOGAs) use a generational scheme, in the last few years some proposals using a steady-state scheme have been developed. However, studies about the influence of using those selection strategies in MOGAs are scarce. In this chapter we implement a steady-state version of NSGA-II, which is a generational MOGA, and we compare the two versions with a set of four state-of-the-art multi-objective metaheuristics (SPEA2, OMOPSO, AbYSS, and MOCell) attending to two criteria: the quality of the resulting approximation sets to the Pareto front and the convergence speed of the algorithms. The obtained results show that search capabilities of the steady-state version of NSGA-II significantly improves the original version, providing very competitive results in terms of the quality of the obtained Pareto front approximations and the convergence speed.

1 Introduction

Genetic Algorithms (GAs) have been widely applied for solving optimization problems in many areas. Since the appearance of the first multi-objective genetic algorithm (MOGA), the *Multiple Objective Optimization with Vector Evaluated Genetic Algorithm* (VEGA) [21], there has been a growing interest in these kinds of algorithms for problems with two or more objectives. GAs are very popular in multi-objective optimization in part because they can obtain a front of solutions in one single run. Thus, the most well-known algorithms in this field are GAs: NSGA-II [3] and SPEA2 [26]. GAs belong to a family of nature-inspired techniques, the

Antonio J. Nebro · Juan J. Durillo
Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, University of Málaga, Campus de Teatinos, 29071 Málaga, Spain
e-mail: `antonio@lcc.uma.es`, `durillo@lcc.uma.es`

evolutionary algorithms, which form part of a broader set of approximation techniques known as *metaheuristics* [11]. Other metaheuristic techniques include particle swarm optimization, ant colony optimization, scatter search, etc.

Based on their selection scheme, there exist two main models of GAs: generational and steady-state. In the generational model, the algorithm creates a population of individuals from an old population using the typical genetic operators (selection, crossover, and mutation); this new population becomes the population of the next generation. On the other hand, a steady-state GA creates typically only one new member which is tested for insertion into the population at each step of the algorithm.

In this chapter our purpose is, taking as starting point a steady-state version of NSGA-II, to study the search enhancements of that scheme over the generational approach of NSGA-II in the context of a comparison against four state-of-the-art multi-objective metaheuristics, namely, SPEA2 [26] (GA), AbYSS [18] (scatter search), MOCell [16] (cellular GA), and OMOPSO [19] (particle swarm optimization). For a broader comparison of these algorithms, we have evaluated them by using test functions from three different benchmarks (ZDT [25], DTLZ [6], and WFG [12]) and we have considered two different criteria. First, we have assessed the quality of the Pareto fronts obtained by those algorithms by applying the additive unary epsilon ($I^1_\varepsilon +$) [14], spread (Δ) [3], and hypervolume (HV) [24] quality indicators. Second, we have studied their convergence speed, i.e., the number of function evaluations required by the algorithms to converge towards the optimal Pareto front.

The remainder of this chapter is structured as follows. The next section provides background information about multi-objective optimization. In Section 3 we review previous works in the literature. Section 4 describes the NSGA-II algorithm and its steady-state version. The algorithms used in the comparative study are described in Section 5. The next two sections are devoted to the experimentation and analysis of the obtained results. Finally, Section 8 draws some conclusions and lines of future work.

2 Multi-Objective Optimization Background

In this section, we provide some background on multiobjective optimization. First, we define basic concepts such as Pareto optimality, Pareto dominance, Pareto optimal set, and Pareto front. In these definitions we assume, without loss of generality, the minimization of all the objectives.

A general multiobjective optimization problem (MOP) can be formally defined as follows:

Definition 1 (MOP). Find a vector $\mathbf{x}^* = [x^*_1, x^*_2, \ldots, x^*_n]$ which satisfies the m inequality constraints $g_i(\mathbf{x}) \geq 0, i = 1, 2, \ldots, m$, the p equality constraints $h_i(\mathbf{x}) = 0, i = 1, 2, \ldots, p$, and minimizes the vector function $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})]^T$, where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables.

The set of all the values satisfying the constraints defines the *feasible region* Ω and any point $\mathbf{x} \in \Omega$ is a *feasible solution*. As mentioned before, we seek for the *Pareto optima*. Its formal definition is provided next:

Definition 2 (Pareto Optimality). A point $\mathbf{x}^* \in \Omega$ is Pareto Optimal if for every $\mathbf{x} \in \Omega$ and $I = \{1,2,\ldots,k\}$ either $\forall_{i \in I} (f_i(\mathbf{x}) = f_i(\mathbf{x}^*))$ or there is at least one $i \in I$ such that $f_i(\mathbf{x}) > f_i(\mathbf{x}^*)$.

This definition states that $\mathbf{x}^*$ is Pareto optimal if no feasible vector $\mathbf{x}$ exists which would improve some criteria without causing a simultaneous worsening in at least one other criterion. Other important definitions associated with Pareto optimality are the following:

Definition 3 (Pareto Dominance). A vector $\mathbf{u} = (u_1,\ldots,u_k)$ is said to dominate $\mathbf{v} = (v_1,\ldots,v_k)$ (denoted by $\mathbf{u} \preccurlyeq \mathbf{v}$) if and only if $\mathbf{u}$ is partially less than $\mathbf{v}$, i.e., $\forall i \in \{1,\ldots,k\}, u_i \leq v_i \wedge \exists i \in \{1,\ldots,k\} : u_i < v_i$.

Definition 4 (Pareto Optimal Set). For a given MOP $\mathbf{f}(\mathbf{x})$, the Pareto optimal set is defined as $\mathscr{P}^* = \{\mathbf{x} \in \Omega | \neg \exists \mathbf{x}' \in \Omega, \mathbf{f}(\mathbf{x}') \preccurlyeq \mathbf{f}(\mathbf{x})\}$.

Definition 5 (Pareto Front). For a given MOP $\mathbf{f}(\mathbf{x})$ and its Pareto optimal set $\mathscr{P}^*$, the Pareto front is defined as $\mathscr{P}\mathscr{F}^* = \{\mathbf{f}(\mathbf{x}) | \mathbf{x} \in \mathscr{P}^*\}$.

Pareto dominance relates two solutions and it can be used as a binary operator. Thus, the application of this operator to two solutions in the objective space returns either one solution that *dominates* another or that the solutions do not dominate each other (i.e., they are *non-dominated* solutions). The Pareto optimal set is composed of all those solutions which are non-dominated, and the Pareto front is the correspondence of the Pareto optimal set in the objective space.

Obtaining the Pareto front of a MOP is the main goal of multiobjective optimization. When stochastic techniques such as metaheuristics are applied, the goal is to obtain a finite set of solutions having two properties: *convergence* to the true Pareto front and homogeneous *diversity*. The first property ensures that we are dealing with optimal solutions, while the second one, which refers to obtaining a uniform-spaced set of solutions, indicates that we have carried out an adequate exploration of the search space, so we are not losing valuable information.

3 Related Work

In this section we analyze previous works in the literature which make use of a steady-state scheme in multi-objective GAs. Many MOGAs using such scheme have been proposed in the last few years; here, we focus on the most salient proposals.

One of the first steady-state multi-objective algorithms described in the literature was the *Pareto Converging Genetic Algorithm* (PCGA) [15]. PCGA used a $(\mu + 2)$ scheme and a novel mechanism based on histograms of ranks for assessing convergence to the Pareto front. It was found to produce diverse sampling of the Pareto front without niching and with significantly less computing effort than NSGA, the previous version of NSGA-II. Nevertheless, the algorithms were evaluated using only three test problems and no comparisons with PCGA using a generational scheme were reported.

The *Simple Evolutionary Algorithm for Multi-Objective Optimization* (SEAMO) was proposed in [23]. It was a simple steady-state approach following a $(\mu + 1)$ scheme that used only one population and depended entirely on the replacement policy used: no rankings, subpopulations, niches or auxiliary approach were required. Due to the fact that a generational version of SEAMO may not make sense, it was only compared with NSGA-II and SPEA2 using as benchmark the multiple knapsack problem.

Deb et *al.* proposed in [4] a ε-Domination Based Steady State MOEA, which was also evaluated in [5]. This algorithm used a $(\mu + 1)$ scheme and was composed of a population and an archive, which used a ε-Domination mechanism. In each generation, one parent from the population and one from the archive were selected to create new offsprings, which were then tested for insertion in both the population and the archive using different strategies. It was compared with several state-of-the-art MOEAS using both bi-objective and three-objective optimization problems. No comparisons with the same algorithm using a generational scheme were reported.

Two multi-objective steady-state algorithms were presented in [1]: the *Objective Exchanging Genetic Algorithm for Design Optimization* (OEGADO), and the *Objective Switching Genetic Algorithm for Design Optimization* (OSGADO). The former proposal consists of several steady-state single-objective optimization GAs which periodically exchange information about the objectives; the second algorithm is also composed of multiple single-objective optimization algorithms, but in this case these algorithms periodically switch the objective they optimize. Both algorithms were compared with NSGA-II using four benchmark academic problems and two engineering problems. In this work, neither OSGADO and OEGADO were evaluated using generational single-objective GAs.

Emmerich et *al.* presented in [10] the so-called *S metric selection EMOA* (SMS-EMOA), which is a hypervolume based steady-state GA. It has a $(\mu + 1)$ scheme as well. The paper included a theoretical analysis in which the advantages of using a steady-state scheme in terms of the complexity of this kind of algorithms were proofed. The algorithm was evaluated using the ZDT benchmark, and it was compared with NSGA-II, SPEA2, and the above described ε-MOEA. No comparisons using a generational scheme were reported.

Srinivasan et *al.* proposed in [22] a new version of the NSGA-II algorithm. This algorithm uses a $(\mu + \lambda)$ scheme, like the original NSGA-II. The main difference was that once all the individuals have been generated, they are considered to update the population in a steady-state model. The new proposal was evaluated using nine benchmark problems and compared with the original NSGA-II algorithm.

Igel et *al.* studied in [13] the effect of two different steady-state schemes, a $(\mu + 1)$ and a $(\mu_{<} + 1)$, for the *Multi-objective covariance matrix adaption evolution strategy* (MO-CMA-ES). The latter steady-state scheme did not consider all the population for selecting the parents. These different approaches were compared with a

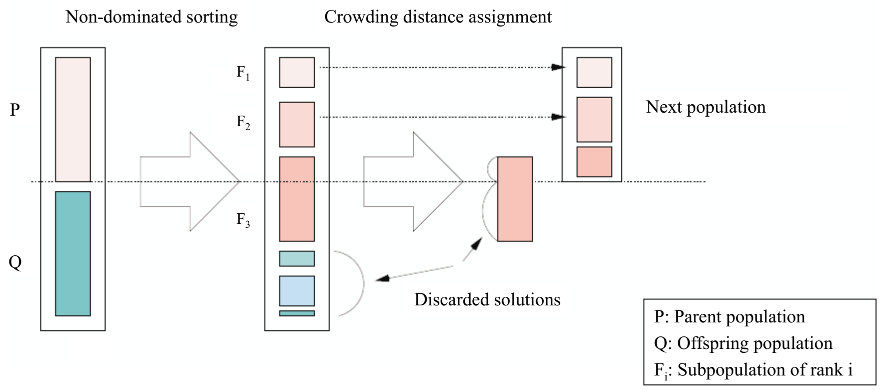


Fig. 1 The NSGA-II procedure, which follows a generational selection scheme

generational scheme, NSGA-II and SPEA2 using a benchmark composed of constrained and unconstrained test functions.

In [9] Durillo et *al.* proposed a steady-state version of the NSGA-II algorithm with a $(\mu + 1)$ scheme and they compared it to the original one. Results showed that by using such a scheme, the algorithm was able to outperform the original one in terms of convergence towards the optimal Pareto front and spread of the resulting fronts of solutions.

Summarizing this section, many of the works in the literature only present new steady-state algorithms and compare them against the state-of-the-art MOGAs; comparisons with the same algorithm using a generational scheme are scarce. Furthermore, many of these proposals are only evaluated using a benchmark composed of small number of problems, and they take into account only the quality of the final front obtained without paying attention to other issues such as the convergence speed of the algorithms.

4 Steady-State NSGA-II

In this section we present the steady-state version of NSGA-II. First, we describe the original (generational) algorithm, and then we go into details related to the steady-state proposal.

The NSGA-II algorithm was proposed by Deb *et al.* [3]. It is based on a ranking procedure, consisting of extracting the non-dominated solutions from a population and assigning them a rank of 1; these solutions are removed and the next group of non-dominated solutions have a rank of 2, and so on. NSGA-II is a generational MOGA, in which a current population is used to create an auxiliary one, the offspring population (see Fig. 1); after that, both populations are combined to obtaining the new current population. The procedure is as follows: the two populations are sorted according to their rank, and the best solutions are chosen to create the new population; in the case of selection between some individuals with the same rank,

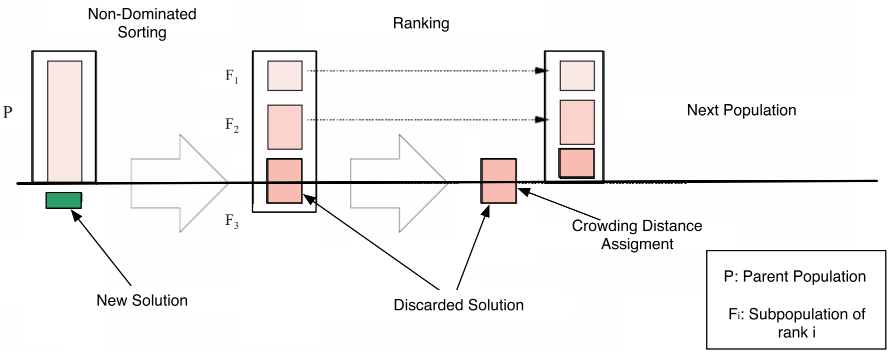


Fig. 2 The NSGA-II$_{ss}$ algorithm. Only one offspring is generated and tested to be inserted at each step.

a density estimation based on the crowding distance to the surrounding individuals of the same rank is used to get the most promising solutions. Typically, both the current and the auxiliary populations have the same size.

A steady-state version of NSGA-II can be easily implemented by using an offspring population of size 1. In this way, the newly generated individual is immediately incorporated into the evolutionary cycle. However, this also means that the ranking and crowding procedures have to be applied each time a new individual is created, so the time required by the algorithm increases notably. The procedure of this version is shown in Fig. 2. In the rest of this work we will refer to the steady-state version as NSGA-II$_{ss}$, and to the original one as NSGA-II$_{gen}$.

5 Description of the Evaluated Algorithms

In this section we briefly describe the four algorithms that we have considered for comparison with the two versions of NSGA-II. We have included SPEA2 because it is, along with NSGA-II, the most popular MOGAs. The other three techniques, OMOPSO, MOCell, and AbYSS are more recent algorithms, and they have been proven to be more effective than NSGA-II and SPEA2 in previous works [17][16][18][20].

The main features of these techniques are described next:

- SPEA2 was proposed by Zitler *et al.* in [26]. In this algorithm, each individual has a fitness value that is the sum of its strength raw fitness plus a density estimation. The algorithm applies the selection, crossover, and mutation operators to fill an archive of individuals; then, the non-dominated individuals of both the original population and the archive are copied into a new population. If the number of non-dominated individuals is greater than the population size, a truncation operator based on the distances to the k-th nearest neighbor is used. This way, the individuals having the minimum distance to any other individual are chosen to be discarded.

- OMOPSO (Optimized MOPSO, Coello *et al.* [19]) is a multi-objective particle swarm optimization algorithm. Its main features include the use of an external archive based on the crowding distance of NSGA-II to filter out leader solutions and the use of mutation operators to accelerate the convergence of the swarm. OMOPSO also has an archive to store the best solutions found during the search. This archive makes use of the concept of ε-dominance to limit the number of solutions stored. Here, we consider the population containing the leaders as the final approximation set.
- AbYSS is an adaptation of the *scatter search* metaheuristic to the multi-objective domain proposed by Nebro *et al.* in [18]. This algorithm uses an external archive similar to the one employed by OMOPSO. AbYSS incorporates operators of the evolutionary algorithms domain, including polynomial mutation and simulated binary (SBX) crossover in the improvement and solution combination methods, respectively.
- MOCell (Nebro *et al.* [16]) is a cellular GA. As OMOPSO and AbYSS, it includes an external archive to store the non-dominated solutions found so far. This archive makes use of the crowding distance of NSGA-II to maintain diversity. MOCell incorporates a feedback procedure: after each interation some random solutions in the current population are replaced by solutions contained in the archive. Here, we have used an asynchronous version of MOCell, called aMOCell4 in [16]. Furthermore, in this version the feedback procedure takes place through the selection operator: one parent is selected from the neighborhood of the current solution, and the other parent is selected randomly from the archive.

We have used the implementation of these algorithms provided by jMetal [8], a Java-based framework aimed at multi-objective optimization problem solving.

6 Experimentation

In this section we explain the benchmark problems used to evaluate the algorithms, the quality indicators used to assess their performance, the criterion used to measure the convergence speed, the parameter settings used, the followed methodology, and the statistical tests carried out.

6.1 Benchmark Problems

Here, we describe the different sets of problems addressed in this work. These problems are well-known, and they have been used in many studies in this area.

The problems families are the following:

- **Zitzler-Deb-Thiele (ZDT):** This benchmark is composed of five bi-objective problems [25]: ZDT1 (convex), ZDT2 (nonconvex), ZDT3 (nonconvex, disconnected), ZDT4 (convex, multimodal), and ZDT6 (nonconvex, nonuniformly spaced). These problems are scalable according to the number of decision variables.

- **Deb-Thiele-Laumanns-Zitzler (DTLZ):** The problems of this family are scalable both in the number of variables and objectives [6]. It is composed of the following seven problems: DTLZ1 (linear), DTLZ2-4 (nonconvex), DTLZ5-6 (degenerate), and DTLZ7 (disconnected).
- **Walking-Fish-Group (WFG):** This set is composed of nine problems, WFG1 - WFG9, that have been constructed using the WFG toolkit [12]. The properties of these problems are detailed in Table 1. They all are scalable both in the number of variables and the number of objectives.

Table 1 Properties of the MOPs created using the WFG toolkit

Problem	Separability	Modality	Bias	Geometry
WFG1	separable	uni	polynomial, flat	convex, mixed
WFG2	non-separable	f_1 uni, f_2 multi	no bias	convex, disconnected
WFG3	non-separable	uni	no bias	linear, degenerate
WFG4	non-separable	multi	no bias	concave
WFG5	separable	deceptive	no bias	concave
WFG6	non-separable	uni	no bias	concave
WFG7	separable	uni	parameter dependent	concave
WFG8	non-separable	uni	parameter dependent	concave
WFG9	non-separable	multi, deceptive	parameter dependent	concave

In this work we have used the bi-objective formulation of the DTLZ and WFG problem families. A total of 21 MOPs are used to evaluate the six metaheuristics.

6.2 *Quality Indicators*

To assess the search capabilities of algorithms on the test problems, two different issues are normally taken into account: the distance between the generated solution set by the proposed algorithm to the optimal Pareto front should be minimized (convergence) and the spread of found solutions should be maximized in order to obtain as smooth and uniform a distribution of solutions as possible (diversity). To measure these two criteria it is necessary to know the exact location of the optimal Pareto front; the benchmark problems used in this work have known Pareto fronts.

The quality indicators can be classified into three categories depending on whether they evaluate the closeness to the Pareto front, the diversity in the solutions obtained, or both [2]. We have adopted one indicator of each type.

- **Unary Epsilon Indicator ($I^1_{\varepsilon+}$).** This indicator was proposed by Zitzler et al. [27] and makes direct use of the principle of Pareto-dominance. Given an approximation set of a problem, A, the $I^1_{\varepsilon+}$ indicator is a measure of the smallest distance one would need to translate every point in A so that it dominates the

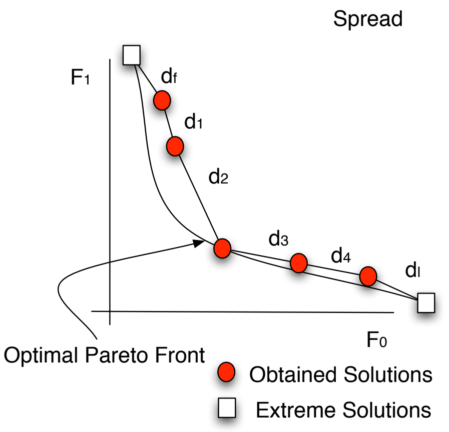


Fig. 3 Calculating the Spread quality indicator

optimal Pareto front of the problem. More formally, given $\mathbf{z^1} = (z_1^1, \ldots, z_n^1)$ and $\mathbf{z^2} = (z_1^2, \ldots, z_n^2)$, where n is the number of objectives:

$$I^1_{\varepsilon+}(A) = inf_{\varepsilon \in \mathbb{R}} \left\{ \forall \mathbf{z^2} \in Pareto\ Optimal\ Front\ \exists \mathbf{z^1} \in A : \mathbf{z^1} \prec_\varepsilon \mathbf{z^2} \right\} \quad (1)$$

where, $\mathbf{z^1} \prec_\varepsilon \mathbf{z^2}$ if and only if $\forall 1 \le i \le n : z_i^1 < \varepsilon + z_i^2$.

- **Spread (Δ).** The diversity *Spread* indicator [3] measures the extent of spread achieved among the obtained solutions. This indicator (illustrated in Fig. 3) is defined as:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1)\bar{d}} \ , \quad (2)$$

where d_i is the Euclidean distance between consecutive solutions, $\bar{d}$ is the mean of these distances, and d_f and d_l are the Euclidean distances to the *extreme* (bounding) solutions of the optimal Pareto front in the objective space (see [3] for the details). Δ takes a value of zero for an ideal distribution, pointing out a perfect spread out of the solutions in the Pareto front. We apply this indicator after a normalization of the objective function values.

- **Hypervolume (HV).** The *HV* indicator calculates the volume, in the objective space, covered by members of a non-dominated set of solutions Q for problems where all objectives are to be minimized [24]. In the example depicted in Fig. 4, the *HV* is the region enclosed within the discontinuous line, where $Q = \{A, B, C\}$ (in the figure, the grey area represents the objective space that has been explored). Mathematically, for each solution $i \in Q$, a hypercube v_i is constructed with a reference point W and the solution i as the diagonal corners of the hypercube. The reference point can be found simply by constructing a vector of worst objective function values. Thereafter, a union of all hypercubes is found and its hypervolume (HV) is calculated:

$$HV = volume \left(\bigcup_{i=1}^{|Q|} v_i \right). \tag{3}$$

Algorithms with larger *HV* values are desirable. Since this indicator is not free from arbitrary scaling of objectives, we have evaluated the metric by using normalized objective function values.

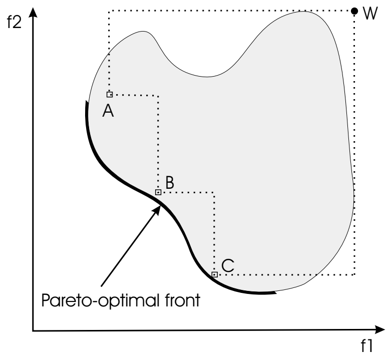


Fig. 4 The hypervolume enclosed by the non-dominated solutions

6.3 Convergence Speed Criterion

Since one of our main interests is to analyze the convergence speed of the analyzed algorithms, it is important to define, first, what we mean by convergence, and to ensure that such definition allows us to measure it in a quantitative and meaningful way. We have studied and defined in [17], a stopping condition based on the *high*

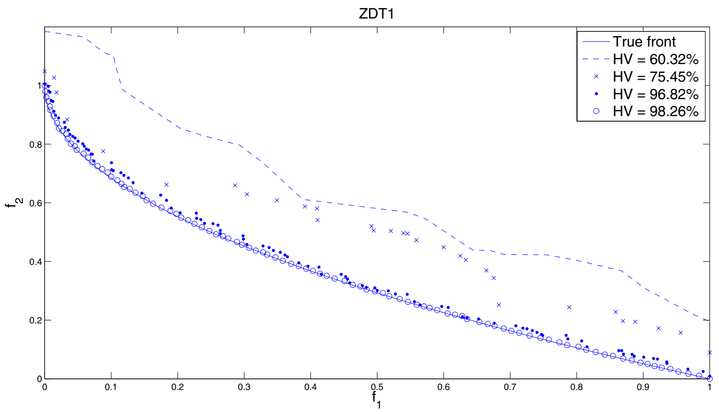


Fig. 5 Fronts with different *HV* values obtained for problem ZDT1

Table 2 Parameterization (L = individual length)

	Parameterization used in NSGA-II [3]
Population Size	100 individuals
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
	Parameterization used in SPEA2 [26]
Population Size	100 individuals
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
	Parameterization used in AbYSS [18]
Population Size	20 individuals
Reference Set Size	10 + 10
Recombination	simulated binary, $p_c = 1.0$
Mutation (local search)	polynomial, $p_m = 1.0/L$
Archive Size	100 individuals
	Parameterization used in MOCell [16]
Population Size	100 individuals (10×10)
Neighborhood	1-hop neighbours (8 surrounding solutions)
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
Archive Size	100 individuals
	Parameterization used in OMOPSO [19]
Swarm size	100 particles
Mutation	uniform + non-uniform
Leaders Size	100

quality of the approximation of the Pareto front found. We have used the HV quality indicator for that purpose.

In Fig. 5 we show different approximations to the Pareto front for the problem ZDT1 with different percentages of HV. We can observe that a front with a hypervolume of 98.26% represents a reasonable approximation to the optimal Pareto front in terms of convergence and diversity of solutions. So, we have taken 98% of the hypervolume of the optimal Pareto front as a criterion to decide that a problem has been successfully solved. In this way, we mean by convergence speed the number of function evaluations required to achieve this termination condition. Those algorithms requiring fewer function evaluations can be considered to be more efficient or faster.

6.4 Parameter Settings

We have chosen a set of parameter settings to guarantee a fair comparison among the algorithms. All GAs (NSGA-II, SPEA2, and MOCell) use an internal population of size equal to 100; the size of the archive is also 100 in SPEA2, OMOPSO, AbYSS, and MOCell. OMOPSO has been configured with 100 particles. For AbYSS, both the population and the reference set have a size of 20 solutions. The two versions of NSGA-II share the same parameterization.

In the GAs we have used SBX and polynomial mutation [2] as operators for crossover and mutation operators, respectively. The distribution indices for both operators are $\eta_c = 20$ and $\eta_m = 20$, respectively. The crossover probability is $p_c = 0.9$ and the mutation probability is $p_m = 1/L$, where L is the number of decision variables. AbYSS uses polynomial mutation in the improvement method and SBX in the solution combination method. OMOPSO applies a combination of uniform and non-uniform mutation. A summary of the parameter settings is included in Table 2.

6.5 Methodology

The stopping criterion is to reach $25,000$ function evaluations in the experiments performed for assessing the quality of the obtained solution sets. The quality indicators are computed after the algorithms have finished their executions.

In the experiments carried out to study the convergence speed, the stopping criterion is to reach either $1,000,000$ function evaluations or a front with 98% HV of the optimal Pareto front. If an algorithm stops according to the first condition, we consider that it has failed to solve the problem. In these experiments, the HV is measured at every 100 function evaluations.

6.6 Statistical Tests

Since we are dealing with stochastic algorithms we have made 100 independent runs of each experiment, and we show the median, $\tilde{x}$, and interquartile range, IQR, as measures of location (or central tendency) and statistical dispersion, respectively. The following statistical analysis has been performed throughout this work [7]. Firstly, a Kolmogorov-Smirnov test was performed in order to check whether the values of the results follow a normal (gaussian) distribution or not. If the distribution is normal, the Levene test checks for the homogeneity of the variances. If samples have equal variance (positive Levene test), an ANOVA test is done; otherwise a Welch test is performed. For non-gaussian distributions, the non-parametric Kruskal-Wallis test is used to compare the medians of the algorithms. Fig. 6 summarizes the statistical analysis.

We always consider in this work a confidence level of 95% (i.e., significance level of 5% or p-value under 0.05) in the statistical tests, which means that the differences are unlikely to have occurred by chance with a probability of 95%. Successful tests are marked with '+' symbols in the last column in all the tables containing the results; conversely, '-' means that no statistical confidence was found (p-value $>$ 0.05). For the sake of better understanding, the best result for each problem has a gray colored background and the second best one has a clearer gray background.

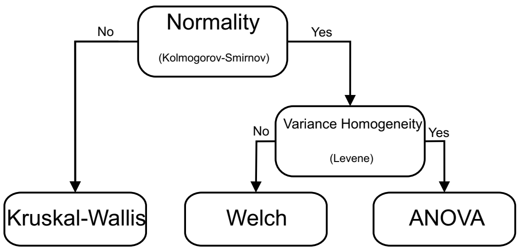


Fig. 6 Statistical analysis performed in this work

7 Results

This section is devoted to presenting and discussing the results of the experiments. We start with the analysis of the obtained values of the $I^1_{\varepsilon+}$, Δ, and HV quality indicators; after that, we focus on the convergence speed.

7.1 Quality Assessment

The results after applying the $I^1_{\varepsilon+}$ indicator are provided in Table 3. Our main interest is to focus on the two versions of NSGA-II, not to determine the best algorithm in the comparisons. We can observe that NSGA-II$_{gen}$ only achieves the best (lower) values in two out of the 21 problems composing the whole benchmark, while NSGA-II$_{ss}$ gets four best and eleven second best results. With the exceptions of problems WFG1 and WFG8, it is clear that the steady-state scheme in NSGA-II allows to improve the convergence of the obtained fronts.

If we make a ranking of the algorithms according to the convergence indicator, considering the best and second best values, it would be headed by MOCell followed by NSGA-II$_{ss}$; after them, OMOPSO, NSGA-II$_{gen}$, AbYSS, and SPEA2. In Fig. 7

Table 3 Median and interquartile range of the $I^1_{\varepsilon+}$ quality indicator

Problem	NSGA-II$_{gen}$ $\bar{x}_{IQR}$	NSGA-II$_{ss}$ $\bar{x}_{IQR}$	SPEA2 $\bar{x}_{IQR}$	AbYSS $\bar{x}_{IQR}$	MOCell $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	
ZDT1	$1.37e-02_{3.0e-03}$	$5.81e-03_{6.3e-04}$	$8.69e-03_{1.1e-03}$	$7.72e-03_{1.8e-03}$	$6.23e-03_{4.1e-04}$	$5.77e-03_{3.8e-04}$	+
ZDT2	$1.28e-02_{2.3e-03}$	$5.79e-03_{5.5e-04}$	$8.73e-03_{1.4e-03}$	$7.10e-03_{1.6e-03}$	$5.57e-03_{3.0e-04}$	$5.64e-03_{3.0e-04}$	+
ZDT3	$8.13e-03_{1.9e-03}$	$5.24e-03_{5.4e-04}$	$9.72e-03_{1.9e-03}$	$6.10e-03_{3.1e-01}$	$5.66e-03_{7.5e-04}$	$6.16e-03_{1.2e-03}$	+
ZDT4	$1.49e-02_{3.0e-03}$	$9.78e-03_{2.6e-03}$	$3.42e-02_{7.9e-02}$	$1.14e-02_{4.2e-03}$	$8.17e-03_{2.3e-03}$	$7.40e+00_{4.5e+00}$	+
ZDT6	$1.47e-02_{2.8e-03}$	$7.02e-03_{7.6e-04}$	$2.42e-02_{5.2e-03}$	$5.06e-03_{3.9e-04}$	$6.53e-03_{5.6e-04}$	$4.67e-03_{3.3e-04}$	+
DTLZ1	$7.13e-03_{1.6e-03}$	$4.62e-03_{1.9e-03}$	$5.89e-03_{2.8e-03}$	$5.85e-03_{5.5e-03}$	$4.02e-03_{1.5e-03}$	$1.54e+01_{1.4e+01}$	+
DTLZ2	$1.11e-02_{2.7e-03}$	$5.13e-03_{3.6e-04}$	$7.34e-03_{1.1e-03}$	$5.39e-03_{4.6e-04}$	$5.09e-03_{2.8e-04}$	$5.23e-03_{2.9e-04}$	+
DTLZ3	$1.04e+00_{1.2e+00}$	$9.63e-01_{1.4e+00}$	$2.28e+00_{1.9e+00}$	$1.66e+00_{1.6e+00}$	$7.91e-01_{1.0e+00}$	$7.87e+01_{7.5e+01}$	+
DTLZ4	$1.13e-02_{9.9e-01}$	$5.24e-03_{9.9e-01}$	$7.66e-03_{9.9e-01}$	$5.39e-03_{3.0e-04}$	$5.74e-03_{9.9e-01}$	$5.55e-03_{4.5e-04}$	+
DTLZ5	$1.05e-02_{2.5e-03}$	$5.14e-03_{3.4e-04}$	$7.47e-03_{1.2e-03}$	$5.36e-03_{5.2e-04}$	$5.08e-03_{3.2e-04}$	$5.27e-03_{2.8e-04}$	+
DTLZ6	$4.39e-02_{3.4e-02}$	$3.07e-02_{2.5e-02}$	$3.03e-01_{5.3e-02}$	$9.50e-02_{4.7e-02}$	$4.16e-02_{3.8e-02}$	$5.18e-03_{4.1e-04}$	+
DTLZ7	$1.04e-02_{2.8e-03}$	$5.13e-03_{4.1e-04}$	$9.09e-03_{1.4e-03}$	$5.51e-03_{9.6e-04}$	$5.19e-03_{1.0e-03}$	$5.39e-03_{4.8e-04}$	+
WFG1	$3.52e-01_{4.6e-01}$	$4.98e-01_{5.3e-01}$	$9.92e-01_{2.1e-01}$	$1.05e+00_{5.1e-01}$	$4.49e-01_{5.0e-01}$	$1.13e+00_{2.1e-01}$	+
WFG2	$7.10e-01_{7.0e-01}$	$7.10e-01_{7.0e-01}$	$7.10e-01_{6.9e-01}$	$7.11e-01_{1.6e-03}$	$7.10e-01_{3.7e-04}$	$9.51e-03_{9.0e-04}$	+
WFG3	$2.00e+00_{5.8e-04}$	$2.00e+00_{4.3e-04}$	$2.00e+00_{1.1e-03}$	$2.00e+00_{1.6e-03}$	$2.00e+00_{5.2e-04}$	$2.00e+00_{2.0e-05}$	+
WFG4	$3.26e-02_{6.7e-03}$	$1.52e-02_{1.5e-03}$	$2.52e-02_{4.0e-03}$	$1.49e-02_{7.7e-04}$	$1.51e-02_{7.3e-04}$	$4.33e-02_{5.6e-03}$	+
WFG5	$8.41e-02_{8.3e-03}$	$6.41e-02_{1.5e-03}$	$7.27e-02_{2.9e-03}$	$6.39e-02_{7.5e-04}$	$6.35e-02_{6.5e-04}$	$6.36e-02_{6.6e-04}$	+
WFG6	$4.14e-02_{1.6e-02}$	$2.50e-02_{2.8e-02}$	$3.11e-02_{1.4e-02}$	$7.84e-02_{5.9e-02}$	$3.65e-02_{5.4e-02}$	$1.43e-02_{6.7e-04}$	+
WFG7	$3.47e-02_{8.1e-03}$	$1.51e-02_{1.5e-03}$	$2.54e-02_{3.0e-03}$	$1.55e-02_{1.1e-03}$	$1.49e-02_{7.5e-04}$	$1.52e-02_{7.6e-04}$	+
WFG8	$3.38e-01_{2.3e-01}$	$5.08e-01_{2.2e-01}$	$5.11e-01_{1.9e-01}$	$5.13e-01_{7.4e-02}$	$5.08e-01_{5.3e-02}$	$5.09e-01_{2.0e-03}$	+
WFG9	$3.73e-02_{7.5e-03}$	$1.80e-02_{3.7e-03}$	$2.92e-02_{5.9e-03}$	$2.21e-02_{6.0e-03}$	$1.94e-02_{3.6e-03}$	$2.55e-02_{2.7e-03}$	+

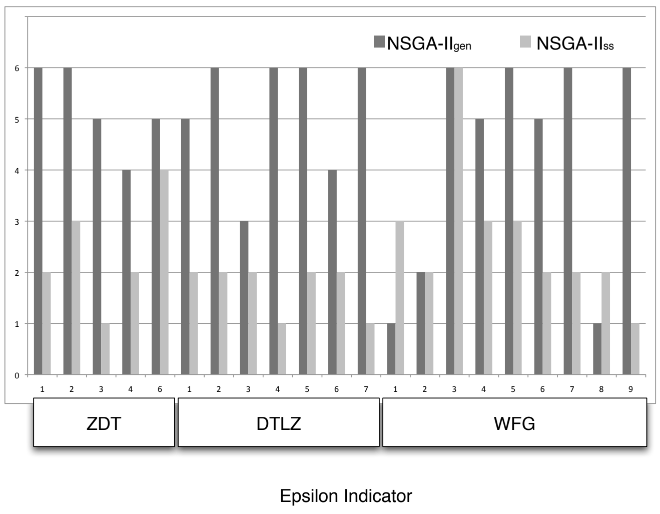


Fig. 7 Positions of NSGA-II$_{gen}$ (left columns) and NSGA-II$_{ss}$ (right columns) in the ranking of the $I^1_{\varepsilon+}$ indicator

we include the positions of each NSGA-II version if we rank the six compared algorithms per individual problem. We can observe clearly that while NSGA-II$_{gen}$ is in the sixth position in many problems, NSGA-II$_{ss}$ is ranked in the first or second positions in all but six instances.

The values of the Δ indicator are included in Table 4. From the table we can see that the steady-state version of NSGA-II yields better (lower) values than its generational counterpart in all the 21 benchmark problems. However, it is unable to

Table 4 Median and interquartile range of the Δ quality indicator

	NSGA-II$_{gen}$	NSGA-II$_{ss}$	SPEA2	AbYSS	MOCell	OMOPSO	
Problem	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	
ZDT1	$3.70e-01_{4.2e-02}$	$7.52e-02_{1.5e-02}$	$1.52e-01_{2.2e-02}$	$1.05e-01_{2.0e-02}$	$7.64e-02_{1.3e-02}$	$7.34e-02_{1.7e-02}$	+
ZDT2	$3.81e-01_{4.7e-02}$	$7.80e-02_{1.3e-02}$	$1.55e-01_{2.7e-02}$	$1.07e-01_{1.8e-02}$	$7.67e-02_{1.4e-02}$	$7.29e-02_{1.5e-02}$	+
ZDT3	$7.47e-01_{1.8e-02}$	$7.03e-01_{3.5e-03}$	$7.10e-01_{7.5e-03}$	$7.09e-01_{9.7e-03}$	$7.04e-01_{6.2e-03}$	$7.08e-01_{6.4e-03}$	+
ZDT4	$4.02e-01_{5.8e-02}$	$1.27e-01_{2.9e-02}$	$2.72e-01_{1.6e-01}$	$1.27e-01_{3.5e-02}$	$1.10e-01_{2.8e-02}$	$8.85e-01_{4.6e-02}$	+
ZDT6	$3.56e-01_{3.6e-02}$	$1.05e-01_{1.5e-02}$	$2.28e-01_{2.5e-02}$	$8.99e-02_{1.4e-02}$	$9.33e-02_{1.3e-02}$	$2.95e-01_{1.1e+00}$	+
DTLZ1	$4.03e-01_{6.1e-02}$	$1.18e-01_{4.0e-02}$	$1.81e-01_{9.8e-02}$	$1.40e-01_{1.7e-01}$	$1.05e-01_{3.6e-02}$	$7.74e-01_{1.3e-01}$	+
DTLZ2	$3.84e-01_{3.8e-02}$	$1.10e-01_{1.6e-02}$	$1.48e-01_{1.6e-02}$	$1.09e-01_{1.9e-02}$	$1.08e-01_{1.7e-02}$	$1.27e-01_{2.0e-02}$	+
DTLZ3	$9.53e-01_{1.6e-01}$	$9.52e-01_{3.4e-01}$	$1.07e+00_{1.6e-01}$	$7.55e-01_{4.5e-01}$	$7.45e-01_{5.5e-01}$	$7.68e-01_{9.3e-02}$	+
DTLZ4	$3.95e-01_{6.4e-01}$	$1.13e-01_{9.0e-01}$	$1.48e-01_{8.6e-01}$	$1.08e-01_{1.8e-02}$	$1.23e-01_{9.0e-01}$	$1.23e-01_{1.9e-02}$	+
DTLZ5	$3.79e-01_{4.0e-02}$	$1.11e-01_{1.6e-02}$	$1.50e-01_{1.9e-02}$	$1.10e-01_{2.0e-02}$	$1.09e-01_{1.7e-02}$	$1.25e-01_{1.9e-02}$	+
DTLZ6	$8.64e-01_{3.0e-01}$	$1.81e-01_{5.3e-02}$	$8.25e-01_{9.3e-02}$	$2.31e-01_{6.3e-02}$	$1.50e-01_{4.3e-02}$	$1.03e-01_{2.1e-02}$	+
DTLZ7	$6.23e-01_{2.5e-02}$	$5.19e-01_{1.9e-03}$	$5.44e-01_{1.3e-02}$	$5.19e-01_{1.3e-03}$	$5.19e-01_{2.9e-02}$	$5.20e-01_{3.7e-03}$	+
WFG1	$7.18e-01_{5.4e-02}$	$5.81e-01_{5.8e-02}$	$6.51e-01_{4.8e-02}$	$6.66e-01_{5.8e-02}$	$5.81e-01_{9.4e-02}$	$1.15e+00_{1.2e-01}$	+
WFG2	$7.93e-01_{1.7e-02}$	$7.47e-01_{1.0e-02}$	$7.53e-01_{1.3e-02}$	$7.46e-01_{4.3e-03}$	$7.47e-01_{2.2e-03}$	$7.60e-01_{2.7e-03}$	+
WFG3	$6.12e-01_{3.6e-02}$	$3.71e-01_{7.2e-03}$	$4.39e-01_{1.2e-02}$	$3.73e-01_{8.7e-03}$	$3.64e-01_{6.3e-03}$	$3.65e-01_{6.9e-03}$	+
WFG4	$3.79e-01_{3.9e-02}$	$1.40e-01_{2.0e-02}$	$2.72e-01_{2.5e-02}$	$1.36e-01_{2.1e-02}$	$1.36e-01_{2.2e-02}$	$3.94e-01_{5.2e-02}$	+
WFG5	$4.13e-01_{5.1e-02}$	$1.38e-01_{1.5e-02}$	$2.79e-01_{2.3e-02}$	$1.31e-01_{2.1e-02}$	$1.32e-01_{2.2e-02}$	$1.36e-01_{2.0e-02}$	+
WFG6	$3.90e-01_{4.2e-02}$	$1.23e-01_{3.2e-02}$	$2.49e-01_{3.1e-02}$	$1.45e-01_{4.3e-02}$	$1.27e-01_{4.0e-02}$	$1.19e-01_{1.9e-02}$	+
WFG7	$3.79e-01_{4.6e-02}$	$1.11e-01_{1.9e-02}$	$2.47e-01_{1.8e-02}$	$1.17e-01_{3.0e-02}$	$1.07e-01_{1.8e-02}$	$1.29e-01_{1.7e-02}$	+
WFG8	$6.45e-01_{5.5e-02}$	$5.63e-01_{5.7e-02}$	$6.17e-01_{8.1e-02}$	$5.86e-01_{7.1e-02}$	$5.57e-01_{4.2e-02}$	$5.42e-01_{3.6e-02}$	+
WFG9	$3.96e-01_{4.1e-02}$	$1.52e-01_{2.1e-02}$	$2.92e-01_{2.0e-02}$	$1.50e-01_{2.2e-02}$	$1.44e-01_{1.7e-02}$	$2.03e-01_{2.0e-02}$	+

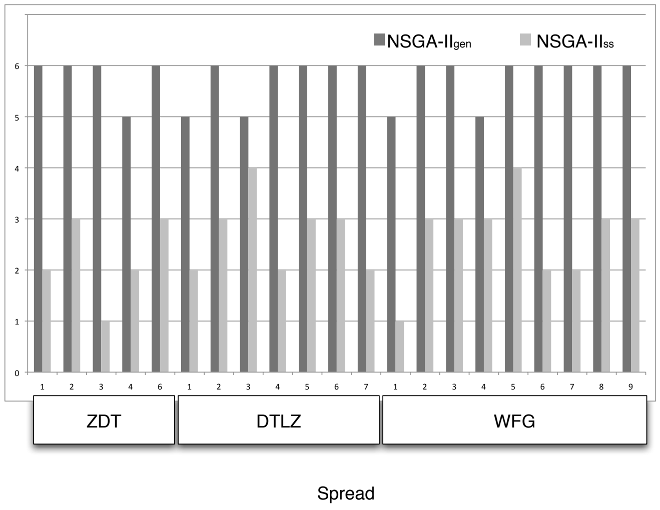


Fig. 8 Positions of NSGA-II$_{gen}$ (left columns) and NSGA-II$_{ss}$ (right columns) in the ranking of the Δ indicator

outperform MOCell and AbYSS, the two best algorithms taking into account this indicator except for the two problems ZDT3 and WFG1.

As before, we include in Fig. 8 the positions of the two versions when ranking the six compared algorithms per problem. The improvements of the steady-state version over the generational one are evident: the latter occupies the last rank position in most of the problems, while the former has two fourth positions as its worst results.

To illustrate the differences between the two versions of NSGA-II, we include in Fig. 9 the approximation sets to the Pareto front obtained by them when solving problem ZDT3. This problem is characterized by having a discontinuos Pareto front. We can observe how the steady-state version produces a front having a uniform spread of the solutions (Fig. 9 - bottom), while the generational one generates a not-so-uniform front.

The results of the *HV* quality indicator are included in Table 5. Those cells containg the symbol "–" mean that the *HV* has a value of 0, meaning that the obtained solution sets are out of the limits of the optimal Pareto front. We observe, first, that NSGA-II$_{ss}$ yield better (higher) values than NSGA-II$_{gen}$ in 18 out the 21 problems. Second, a ranking of the algorithms considering the number of best and second best values would be led by NSGA-II$_{ss}$, because although OMOPSO yields six best values (five in the case of NSGA-II$_{ss}$), it has only a single second best result, while NSGA-II$_{ss}$ has eight. The ranking per problem is included in Fig. 10, which shows that NSGA-II$_{ss}$ has a ranking greater than three in only three problems.

As the *HV* measures both convergence and diversity, and considering the other two indicators, we can conclude that NSGA-II$_{ss}$ not only outperforms NSGA-II$_{gen}$, but it is also a competitive technique based on the convergence of the produced Pareto fronts of all the evaluated algorithms.

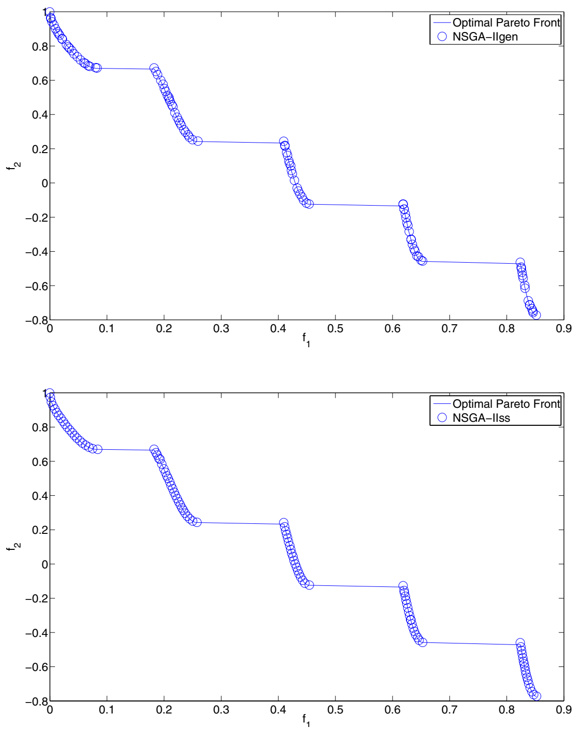


Fig. 9 Pareto fronts obtained by NSGA-II$_{gen}$ (top) and NSGA-II$_{ss}$ (bottom) when solving problem ZDT3

To conclude this section we would like to remark, first, that the obtained results in the experiments carried out have statistical significance, as it can be observed in the '+' symbols in the last column in the three tables. Second, it has to be pointed out that the use of the steady-states scheme has a computational cost that has to be taken into account. Specifically, we have measured the running times of the two versions of NSGA-II when solving all the problems, and NSGA-II$_{ss}$ is about 10 times slower than the generational algorithm: the mean time is about 1.2 seconds per execution, in the case of the original NSGA-II, and roughly 12.5 seconds the steady-state version. We have used the profiling tool provided by the Java IDE Netbeans 6.1 to analyze the execution of the two algorithms when solving the ZDT1 problem. The profiling report has shown that the computing time required to evaluate the problem is less than 1% of the total execution time. So, although the diferences between the two algorithms are important in the context of our study, we must note that in a real scenario the computing time of the algorithms can become negligible. As an example, if evaluating the objective functions of a problem requires 1 second, as we carry out 25,000 evaluations, the total time would be 25,000 seconds (6,94 hours), so it is obvious that the running times of the algorithms would not be relevant in this situation.

Table 5 Median and interquartile range of the *HV* quality indicator

Problem	NSGA-II$_{gen}$ $\bar{x}_{IQR}$	NSGA-II$_{ss}$ $\bar{x}_{IQR}$	SPEA2 $\bar{x}_{IQR}$	AbYSS $\bar{x}_{IQR}$	MOCell $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	
ZDT1	$6.59e-01_{4.4e-04}$	$6.62e-01_{1.4e-04}$	$6.60e-01_{3.9e-04}$	$6.61e-01_{3.2e-04}$	$6.61e-01_{2.5e-04}$	$6.61e-01_{3.2e-04}$	+
ZDT2	$3.26e-01_{4.3e-04}$	$3.28e-01_{1.6e-04}$	$3.26e-01_{8.1e-04}$	$3.28e-01_{2.8e-04}$	$3.28e-01_{4.3e-04}$	$3.28e-01_{2.4e-04}$	+
ZDT3	$5.15e-01_{2.3e-04}$	$5.16e-01_{8.0e-05}$	$5.14e-01_{3.6e-04}$	$5.16e-01_{3.5e-03}$	$5.15e-01_{3.1e-04}$	$5.15e-01_{8.8e-04}$	+
ZDT4	$6.56e-01_{4.5e-03}$	$6.57e-01_{4.0e-03}$	$6.51e-01_{1.2e-02}$	$6.55e-01_{6.0e-03}$	$6.59e-01_{3.0e-03}$	–	+
ZDT6	$3.88e-01_{2.3e-03}$	$3.96e-01_{1.1e-03}$	$3.79e-01_{3.6e-03}$	$4.00e-01_{1.9e-04}$	$3.97e-01_{1.1e-03}$	$4.01e-01_{7.1e-05}$	+
DTLZ1	$4.88e-01_{5.5e-03}$	$4.89e-01_{6.5e-03}$	$4.89e-01_{6.2e-03}$	$4.86e-01_{1.7e-02}$	$4.91e-01_{3.8e-03}$	–	+
DTLZ2	$2.11e-01_{3.1e-04}$	$2.12e-01_{4.3e-05}$	$2.12e-01_{1.7e-04}$	$2.12e-01_{6.5e-05}$	$2.12e-01_{4.5e-05}$	$2.12e-01_{2.8e-04}$	+
DTLZ3	–	–	–	–	–	–	+
DTLZ4	$2.09e-01_{2.1e-01}$	$2.11e-01_{2.1e-01}$	$2.10e-01_{2.1e-01}$	$2.11e-01_{5.9e-05}$	$2.11e-01_{2.1e-01}$	$2.10e-01_{4.0e-04}$	+
DTLZ5	$2.11e-01_{3.5e-04}$	$2.12e-01_{3.7e-05}$	$2.12e-01_{1.7e-04}$	$2.12e-01_{6.8e-05}$	$2.12e-01_{3.1e-05}$	$2.12e-01_{3.0e-04}$	+
DTLZ6	$1.75e-01_{3.6e-02}$	$1.73e-01_{2.8e-02}$	$9.02e-03_{1.4e-02}$	$1.11e-01_{4.1e-02}$	$1.61e-01_{4.2e-02}$	$2.12e-01_{5.0e-05}$	+
DTLZ7	$3.33e-01_{2.1e-04}$	$3.34e-01_{3.9e-05}$	$3.34e-01_{2.2e-04}$	$3.34e-01_{7.8e-05}$	$3.34e-01_{9.5e-05}$	$3.34e-01_{2.2e-04}$	+
WFG1	$5.23e-01_{1.3e-01}$	$4.90e-01_{1.9e-01}$	$3.85e-01_{1.1e-01}$	$2.27e-01_{1.3e-01}$	$4.95e-01_{1.7e-01}$	$1.60e-01_{9.0e-02}$	+
WFG2	$5.61e-01_{2.8e-03}$	$5.62e-01_{2.6e-03}$	$5.62e-01_{2.8e-03}$	$5.61e-01_{1.1e-03}$	$5.62e-01_{2.9e-04}$	$5.64e-01_{6.8e-05}$	+
WFG3	$4.41e-01_{3.2e-04}$	$4.42e-01_{1.8e-04}$	$4.42e-01_{2.0e-04}$	$4.42e-01_{5.9e-04}$	$4.42e-01_{1.9e-04}$	$4.42e-01_{2.2e-05}$	+
WFG4	$2.17e-01_{4.9e-04}$	$2.19e-01_{2.4e-04}$	$2.18e-01_{3.0e-04}$	$2.19e-01_{2.0e-04}$	$2.19e-01_{2.3e-04}$	$2.06e-01_{1.7e-03}$	+
WFG5	$1.95e-01_{3.6e-04}$	$1.96e-01_{6.7e-05}$	$1.96e-01_{1.8e-04}$	$1.96e-01_{6.3e-05}$	$1.96e-01_{6.9e-05}$	$1.96e-01_{6.3e-05}$	+
WFG6	$2.03e-01_{9.0e-03}$	$2.03e-01_{1.9e-02}$	$2.04e-01_{8.6e-03}$	$1.71e-01_{3.3e-02}$	$1.95e-01_{3.4e-02}$	$2.10e-01_{1.1e-04}$	+
WFG7	$2.09e-01_{3.3e-04}$	$2.11e-01_{1.4e-04}$	$2.10e-01_{2.4e-04}$	$2.11e-01_{1.7e-04}$	$2.11e-01_{1.3e-04}$	$2.10e-01_{1.0e-04}$	+
WFG8	$1.47e-01_{2.1e-03}$	$1.48e-01_{1.6e-03}$	$1.47e-01_{2.2e-03}$	$1.44e-01_{3.2e-03}$	$1.47e-01_{2.2e-03}$	$1.46e-01_{1.1e-03}$	+
WFG9	$2.37e-01_{1.7e-03}$	$2.40e-01_{1.9e-03}$	$2.39e-01_{2.3e-03}$	$2.38e-01_{3.6e-03}$	$2.39e-01_{2.6e-03}$	$2.37e-01_{5.8e-04}$	+

7.2 Convergence Speed

In the previous section we have shown that the steady-state version of NSGA-II performed better than the original algorithm in most of the tested problems. In this section we analyze the obtained results when measuring the convergence speed.

Table 6 contains the number of evaluations required by the algorithms to reach a Pareto front with 98% of the *HV* of the optimal Pareto front. There are cases

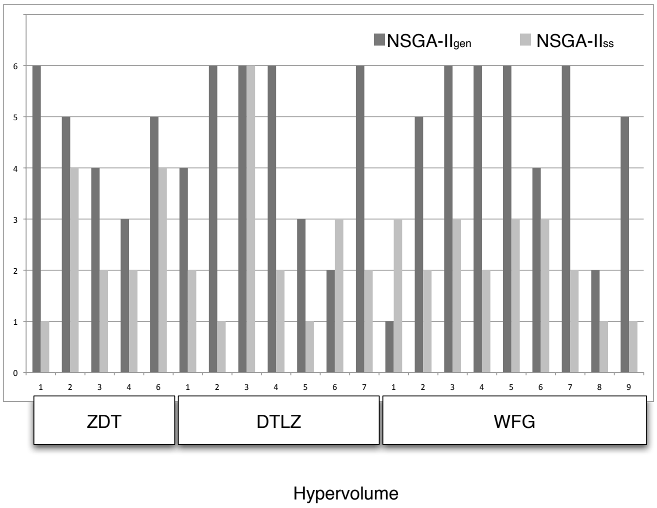


Fig. 10 Positions of NSGA-II$_{gen}$ (left columns) and NSGA-II$_{ss}$ (right columns) in the ranking of the *HV* indicator

Table 6 Median and *IQR* of the number of evaluations computed by the algorithms

Problem	NSGA-II$_{gen}$ $\bar{x}_{IQR}$	NSGA-II$_{ss}$ $\bar{x}_{IQR}$	SPEA2 $\bar{x}_{IQR}$	AbYSS $\bar{x}_{IQR}$	MOCell $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	
ZDT1	1.430e+04 $_{8.0e+02}$	1.160e+04 $_{9.0e+02}$	1.600e+04 $_{1.1e+03}$	1.370e+04 $_{1.6e+03}$	1.300e+04 $_{1.2e+03}$	6.800e+03 $_{2.0e+03}$	+
ZDT2	2.460e+04 $_{1.6e+03}$	1.770e+04 $_{1.3e+03}$	2.480e+04 $_{1.9e+03}$	1.710e+04 $_{2.8e+03}$	1.170e+04 $_{4.0e+03}$	8.900e+03 $_{3.6e+03}$	+
ZDT3	1.280e+04 $_{8.5e+02}$	1.095e+04 $_{1.0e+03}$	1.520e+04 $_{1.0e+03}$	1.270e+04 $_{2.0e+03}$	1.300e+04 $_{1.3e+03}$	9.850e+03 $_{2.7e+03}$	+
ZDT4	2.245e+04 $_{5.9e+03}$	1.985e+04 $_{5.4e+03}$	2.520e+04 $_{6.0e+03}$	2.285e+04 $_{1.1e+04}$	1.635e+04 $_{5.0e+03}$	–	+
ZDT6	2.930e+04 $_{1.4e+03}$	2.280e+04 $_{1.2e+03}$	3.335e+04 $_{1.0e+03}$	1.560e+04 $_{1.2e+03}$	2.090e+04 $_{1.3e+03}$	2.800e+03 $_{1.5e+03}$	+
DTLZ1	2.495e+04 $_{8.4e+03}$	2.225e+04 $_{8.6e+03}$	2.400e+04 $_{7.5e+03}$	2.375e+04 $_{1.2e+04}$	2.015e+04 $_{7.7e+03}$	1.000e+06 $_{4.7e+04}$	+
DTLZ2	8.150e+03 $_{1.2e+03}$	5.300e+03 $_{7.0e+02}$	7.400e+03 $_{8.0e+02}$	4.700e+03 $_{9.0e+02}$	5.600e+03 $_{9.0e+02}$	8.200e+03 $_{3.1e+03}$	+
DTLZ3	1.127e+05 $_{5.3e+04}$	8.270e+04 $_{3.5e+04}$	1.000e+05 $_{3.0e+04}$	1.194e+05 $_{7.5e+04}$	6.735e+04 $_{2.3e+04}$	–	+
DTLZ4	8.650e+03 $_{1.3e+03}$	5.500e+03 $_{7.0e+02}$	7.800e+03 $_{5.0e+05}$	4.800e+03 $_{7.5e+02}$	1.000e+06 $_{9.9e+05}$	1.255e+04 $_{3.8e+03}$	+
DTLZ5	8.300e+03 $_{1.4e+03}$	5.150e+03 $_{6.0e+02}$	7.500e+03 $_{7.0e+02}$	4.650e+03 $_{8.0e+02}$	5.800e+03 $_{8.5e+02}$	8.450e+03 $_{2.9e+03}$	+
DTLZ6	1.000e+06 $_{9.7e+05}$	–	1.000e+06 $_{9.7e+05}$	–	–	4.100e+03 $_{1.5e+03}$	+
DTLZ7	1.360e+04 $_{9.0e+02}$	1.060e+04 $_{9.0e+02}$	1.585e+04 $_{1.1e+03}$	1.060e+04 $_{1.7e+03}$	1.110e+04 $_{1.6e+05}$	6.150e+03 $_{2.6e+03}$	+
WFG1	4.315e+04 $_{5.4e+04}$	3.715e+04 $_{1.5e+04}$	1.096e+05 $_{7.7e+05}$	–	4.160e+04 $_{1.7e+04}$	–	+
WFG2	1.700e+03 $_{4.0e+02}$	1.400e+03 $_{5.0e+02}$	2.000e+03 $_{7.0e+02}$	1.850e+03 $_{2.4e+03}$	1.400e+03 $_{8.0e+02}$	1.800e+03 $_{4.0e+02}$	+
WFG3	–	–	–	–	–	–	-
WFG4	2.050e+04 $_{8.8e+03}$	8.200e+03 $_{2.9e+03}$	1.280e+04 $_{4.6e+03}$	6.750e+03 $_{2.4e+03}$	1.050e+04 $_{3.1e+03}$	2.233e+05 $_{1.3e+05}$	+
WFG5	–	–	–	–	–	–	-
WFG6	–	1.000e+06 $_{9.8e+05}$	1.000e+06 $_{4.8e+04}$	–	1.000e+06 $_{5.5e+05}$	7.300e+03 $_{1.2e+03}$	+
WFG7	1.686e+05 $_{2.5e+05}$	1.035e+04 $_{2.6e+03}$	1.775e+04 $_{5.4e+03}$	9.600e+03 $_{3.4e+03}$	1.215e+04 $_{3.4e+03}$	1.495e+04 $_{2.6e+03}$	+
WFG8	–	–	–	–	–	–	+
WFG9	–	1.000e+06 $_{9.4e+05}$	–	–	–	8.935e+04 $_{4.9e+04}$	+

in which a "–" is reported when the algorithms have computed 1,000,000 function evaluations without producing a solution set with the desired *HV* value.

The results show that NSGA-II$_{ss}$ requires a fewer number of evaluations than NSGA-II$_{gen}$ in all the problems but DTLZ6, which means that the steady-approach makes NSGA-II converge faster. If we consider all the problems, we see that NSGA-II$_{ss}$ is the fastest only in two out of the 21 problems, but it is the second fastest in 10 problems. This can be clearly observed in the ranking per problem in Fig. 11, which

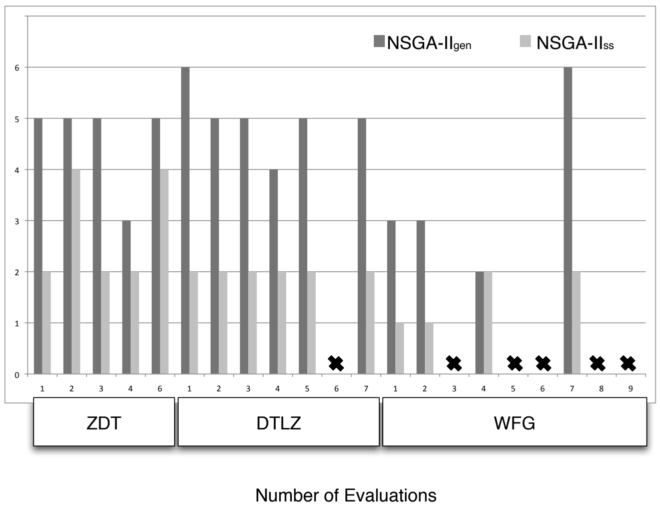
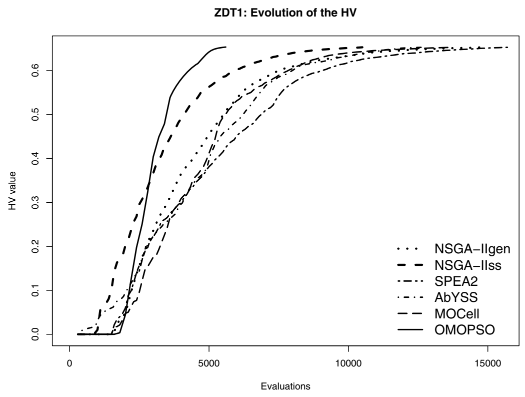


Fig. 11 Positions of NSGA-II$_{gen}$ (left columns) and NSGA-II$_{ss}$ (right columns) in the ranking of the convergence speed



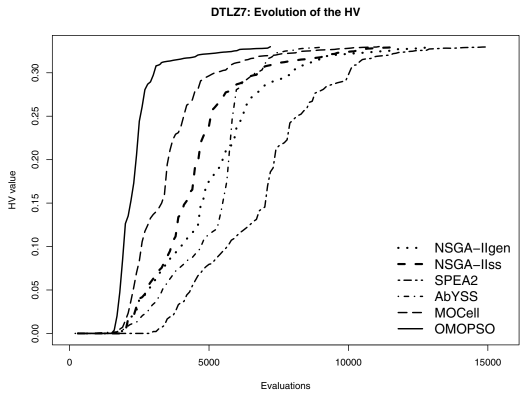


Fig. 12 Evolution of the HV value during the different generations carried out in the problem ZDT1 (top) and DTLZ7 (bottom)

allows us to conclude that the steady-state approach makes NSGA-II improve from being the second slowest algorithm in the comparison (fifth position in most of the problems) to be second one in terms of convergence speed.

We include in Fig. 12 a trace of the evolution of the value of the *HV* when solving problems ZDT1 and DTLZ7 in a single run. The values have been recorded at each 100 function evaluations. Focusing on ZDT1, Fig. 12 - top shows that NSGA-II$_{ss}$ is the second algorithm in achieving the desired *HV* value after OMOPSO, the fastest metaheuristic on this problem. The trace of the DTLZ7 problem (Fig. 12 - bottom) reveals that NSGA-II$_{ss}$ has been the fourth fastest algorithm in the monitored execution. In both cases, we can observe that the NSGA-II$_{ss}$ converges faster than the original algorithm.

8 Conclusions and Future Work

In this chapter we have studied the effect of applying a steady-state selection scheme to NSGA-II, the reference algorithm in multi-objective optimization. Both the original and the steady-state versions have been evaluated using a benchmark composed of 21 bi-objective problems for comparing the performance of the algorithms in terms of the quality of the obtained solutions sets and their converging speed towards the optimal Pareto front. We have compared the two versions with a set of four state-of-the-art multi-objective optimizers (SPEA2, AbYSS, MOCell, and OMOPSO) to have an insight on the search improvements of the steady-state scheme in NSGA-II.

The obtained results have shown that, in the context of the problems, with the quality indicators and the parameter settings considered, the use of a steady-state scheme has improved the results obtained by the generational NSGA-II in most of the problems. Furthermore, it has also shown to be very competitive taking account of the quality of the obtained approximation sets and the convergence speed of the other state-of-the-art algorithms.

Some future research topics along this line are related to the study and application of steady-state scheme to other multi-objective algorithms and to solve benchmarks composed of rotated problems and with more than two objectives.

Acknowledgements. Authors acknowledge funds from the "Consejería de Innovación, Ciencia y Empresa", Junta de Andalucía under contract P07-TIC-03044 DIRICOM project, `http://diricom.lcc.uma.es`.

Juan J. Durillo is supported by grant AP-2006-003349 from the Spanish Ministry of Education and Science.

References

1. Chafekar, D., Xuan, J., Rasheed, K.: Constrained Multi-objective Optimization Using Steady State Genetic Algorithms. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 813–824. Springer, Heidelberg (2003)
2. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
4. Deb, K., Mohan, M., Mishra, S.: Towards a Quick Computation of Well-Spread Pareto-Optimal Solutions. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 222–236. Springer, Heidelberg (2003)
5. Deb, K., Mohan, M., Mishar, S.: Evaluating the ε-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions. Evolutionary Computation 13(4), 501–525 (2005)
6. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In: Abraham, A., Jain, L., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization. Theoretical Advances and Applications, pp. 105–145. Springer, Heidelberg (2005)
7. Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. J. Mach. Learn. Res. 7, 1–30 (2006)

8. Durillo, J.J., Nebro, A.J., Luna, F., Dorronsoro, B., Alba, E.: jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics. Tech. Rep. ITI-2006-10, Dept. de Lenguajes y Ciencias de la Computación, University of Málaga (2006)
9. Durillo, J.J., Nebro, A.J., Luna, F., Alba, E.: A Study of Master-Slave Approaches to Parallelize NSGA-II. In: IEEE Iinternational Symposium on Parallel and Distributed Processing - IPDPS 2008, pp. 1–8 (2008)
10. Emmerich, M., Beume, N., Naujoks, B.: An EMO Algorithm Using the Hypervolume Measure as Selection Criterion. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 62–76. Springer, Heidelberg (2005)
11. Glover, F.W., Kochenberger, G.A.: Handbook of Metaheuristics. Kluwer Academic Publishers, Dordrecht (2003)
12. Huband, S., Hingston, P., Barone, L., While, R.L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans Evolutionary Computation 10(5), 477–506 (2006)
13. Igel, C., Suttorp, T., Hansen, N.: Steady-state Selection and Efficient Covariance Matrix Update in the Multi-objective CMA-ES. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 171–185. Springer, Heidelberg (2007)
14. Knowles, J., Thiele, L., Zitzler, E.: A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Tech. Rep. 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich (2006)
15. Kumar, R., Rockett, P.: Improved Sampling of the Pareto-Front in Multiobjective Genetic Optimizations by Steady-State evolution: A Pareto Converging Genetic Algorithm. Evolutionary Computation 10(3), 283–314 (2002)
16. Nebro, A.J., Durillo, J.J., Luna, F., Dorronsoro, B., Alba, E.: Design issues in a multiobjective cellular genetic algorithm. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 126–140. Springer, Heidelberg (2007)
17. Nebro, A.J., Durillo, J.J., Coello Coello, C.A., Luna, F., Alba, E.: A Study of Convergence Speed in Multi-objective Metaheuristics. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 171–185. Springer, Heidelberg (2008)
18. Nebro, A.J., Luna, F., Alba, E., Dorronsoro, B., Durillo, J.J., Beham, A.: AbYSS: Adapting Scatter Search to Multiobjective Optimization. IEEE Transactions on Evolutionary Computation 12(4), 439–457 (2008)
19. Reyes-Sierra, M., Coello Coello, C.A.: Improving PSO-based multi-objective optimization using crowding, mutation and ε-dominance. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 509–519. Springer, Heidelberg (2005)
20. Reyes-Sierra, M., Coello Coello, C.A.: Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. International Journal of Computational Intelligence Research 2(3), 287–308 (2006)
21. Schaffer, J.D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: Grefenstette, J.J. (ed.) Proceedings of the 1st International Conference on Genetic Algorithms, pp. 93–100 (1985)
22. Srinivasan, D., Rachmawati, L.: An Efficient Multi-objective Evolutionary Algorithm with Steady-State Replacement Model. In: Genetic and Evolutionary Computation - GECCO 2006, pp. 715–722 (2006)

23. Valenzuela, C.L.: A Simple Evolutionary Algorithm for Multi-Objective Optimization (SEAMO). In: IEEE Congress on Evolutionary Computation - CEC 2002, pp. 717–722 (2002)
24. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)
25. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2), 173–195 (2000)
26. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Tech. Rep. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001)
27. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation 7, 117–132 (2003)

Improving the Performance of Multiobjective Evolutionary Optimization Algorithms Using Coevolutionary Learning

Tse Guan Tan and Jason Teo

Abstract. This chapter introduces two algorithms for multiobjective optimization. These algorithms are based on a state-of-the-art Multiobjective Evolutionary Algorithm (MOEA) called Strength Pareto Evolutionary Algorithm 2 (SPEA2). The first proposed algorithm implements a competitive coevolution technique within SPEA2. In contrast, the second algorithm introduces a cooperative coevolution technique to SPEA2. Both novel coevolutionary approaches are then compared to the original SPEA2 in seven scalable DTLZ test problems with 3 to 5 objectives. Overall, the optimization results show that the two proposed approaches are superior to the original SPEA2 with regard to the average distance of the nondominated solutions to the true Pareto front, the diversity of the obtained solutions and also the coverage level. In addition, t-tests have been conducted to validate the significance of the improvements obtained by the augmented algorithms over the original SPEA2. Finally, cooperative coevolution is found to be better than competitive coevolution in terms of enhancing the performance of the original SPEA2.

1 Introduction

According to McFadden and Keeton [15], coevolution is defined as "a process in which two or more different organisms are evolving together, each in response to the other. There is a reciprocal interaction between the two groups where each organism adapts to the selection pressures imposed by the other". Generally, the coevolution model can be either Competitive Coevolution (CE) or Cooperative Coevolution (CC). The competitive coevolution involves individuals that compete against each other for dominance in the population. CE can be classified into three categories [25]: those utilizing only two individuals in the tournament, a predator and a prey; those with a single population tournament, where competition is simply between individuals in a population; and those utilizing more than one

Tan Tse Guan
School of Engineering and Information Technology, Universiti Malaysia Sabah,
Locked Bag No. 2073, 88999 Kota Kinabalu, Sabah, Malaysia
e-mail: `tseguantan@gmail.com`

Jason Teo
School of Engineering and Information Technology, Universiti Malaysia Sabah,
Locked Bag No. 2073, 88999 Kota Kinabalu, Sabah, Malaysia
e-mail: `jtwteo@ums.edu.my`

population for the tournament, where the competition involves a number of populations. On the other hand, the CC concept [17, 18] involves a number of individuals from different subpopulations working together to solve the problem. The coevolution concept has several advantages [23, 26]. One of the benefits is that coevolution does not necessarily need to specify a global fitness function whereby individuals in the population are ranked; rather, only relative fitness is needed. In addition, it is extremely useful when the objective evaluation function is hard to identify or cannot be determined. This concept can be used to decompose the optimization problem into *n* subtasks. Also, it has the potential for open-endedness in terms of the artificial evolution process.

Evolutionary Algorithms (EAs) are one of the most suitable techniques in handling multiobjective optimization problems because of their population-based approach that is able to find a set of trade-off solutions in one single simulation run instead of having to perform a series of separate runs as in the case of traditional optimization techniques. The number of objectives is one of the significant elements that affect the performance of multiobjective optimization algorithms [4]. For instance, the bi-dimensional solution space is plain, but the tri-dimension solution space comprises of a three-dimensional surface and so on, hence greatly increasing the difficulty of finding a globally optimal solution. Additionally, EAs can be used successfully in complex problems, involving features such as discontinuities, multimodality, disjoint feasible spaces and noisy function evaluations [7]. Recently, many Multiobjective Evolutionary Algorithms (MOEAs) have been shown to be very useful techniques in solving real-life multiobjective problems [1]. However, the use of coevolution methods in MOEAs remains largely unexplored. In this study, the coevolution model will be embedded into the SPEA2 in our attempt to enhance SPEA2's optimization performance. Hence, the first proposed algorithm is the integration between SPEA2 and CE using the *K*-Random Opponents (KR) competitive fitness strategy. The resulting algorithm is referred to as SPEA2-CE-KR whereas the second hybrid algorithm is the combination between SPEA2 and the CC mechanism. The proposed algorithms will be compared to the original SPEA2 using seven standard benchmark test problems, namely DTLZ1 to DTLZ7, with the dimensionality of each function ranging from 3 to 5 objectives. The experimental results are evaluated using the generational distance, spacing, and coverage metrics. For the generational distance and spacing, their statistical significance is analyzed using t-tests.

The organization of the rest of the chapter is as follows. Section 2 focuses on the related works. Section 3 summarizes the basic principle of multiobjective optimization. In Section 4, the structure of SPEA2 is reviewed. The proposed algorithms, SPEA2-CE-KR and SPEA2-CC, are introduced in Section 5. Section 6 presents the characteristics of each test problem and the performance metrics. The experimental settings for the algorithms are described in Section 7. The results are analyzed and discussed in Section 8 and Section 9 respectively. Finally, the conclusions and future work are given in Section 10.

2 Related Works

There have been only a very limited number of studies that incorporate competitive coevolution into MOEAs. Lohn *et al.* [12] presented a competitive coevolution genetic algorithm to solve bi-objective problems. In Lohn's approach, the tournament is held between two populations, which are the trial population that consists of candidate solutions, and the target population that encompasses Target Objective Vectors (TOVs). These vectors include a set of targets for the trial population to optimize. In general, the results show that the presented algorithm performed well compared to Random Search (RAND), Fonseca and Fleming's multiobjective GA (FFGA), Niched Pareto GA (NPGA), Hajela and Lin's weighted sum approach (HLGA), Vector Evaluated GA (VEGA), Nondominated Sorting GA (NSGA), a Single-Objective EA using weighted-sum aggregation (SOEA) and Strength Pareto Evolutionary Algorithm (SPEA) at finding optimal solutions on several fitness landscapes.

However, a number of CC algorithms for multiobjective optimization have been proposed recently. Keerativuttitumrong *et al.* [10] proposed the Multiobjective Cooperative Coevolutionary Genetic Algorithm (MOCCGA), which integrates the CC concept with Multiobjective Genetic Algorithm (MOGA). Each species represents a single decision variable. In order to evaluate individuals in any species, collaborators will be selected from the other species to form a complete solution. Then this complete solution is mapped into the objective vector based on the objective function. The evolution of these populations is controlled through MOGA. According to the results, cooperative coevolution improved the MOGA search performance. MOCCGA was found to be superior to MOGA in terms of Pareto front coverage and the closeness of the obtained optimal solutions to the true Pareto front.

Maneeratana *et al.* [13] presented the integration of the cooperation coevolutionary effect using four different evolutionary multiobjective optimization algorithms. The CC methodology employed was similar to Keerativuttitumrong *et al.* [10]. The evolution of these populations is controlled through the MOGA, NPGA, NSGA and Controlled Elitist Nondominated Sorting Genetic Algorithm (CNSGA). The algorithm uses a preservation set to store all unrepeated nondominated solutions and the crowding distance selection method was utilized to normalize the size of the preserved set. The elitist strategy was employed to avoid losing good solutions during the optimization process due to random effects. In general, the integrated algorithms performed well in terms of the solution set coverage, the average distance to the true Pareto front, the distribution of nondominated solutions and the extent of the front described by nondominated solutions, compared to original MOEAs.

Coello Coello and Reyes Sierra [3] introduced a multiobjective evolutionary algorithm that integrates with the cooperative coevolutionary concept, named CO-MOEA. This algorithm separates the search space into several regions and later focuses on the promising regions. The promising regions are measured by an analysis of the current Pareto front. The evolutionary process of this algorithm consists of four distinct stages. The number of generations will determine the changing of stages. Overall, the presented algorithm obtained a good Pareto front regarding the average distance to the true Pareto front and the distribution of the optimal solutions. However in the graphical results, CO-MOEA, NSGA-II and microGA have similar performances in terms of error ratio and coverage metrics.

Tan *et al.* [21] also proposed an algorithm focusing on the CC approach for multiobjective optimization, called CCEA. This algorithm adopted the general framework of the CC effect and implemented a rank assignment strategy. The archive's functions were to store the nondominated solutions and to help evaluate individuals in the subpopulation based on the Pareto-based rank assignment scheme. The diversity of solutions in the archive was maintained using the niching mechanism. The extending operator was used to maintain the even distribution of the archive and to encourage widespread exploration. Overall, CCEA performed well regarding closeness to the true Pareto front and distribution of the nondominated solutions along the true Pareto front.

Iorio and Li [9] describe a CC algorithm which incorporates a novel collaboration formation mechanism. The individuals of each species are rewarded if they are involved in successful collaborations. A nondominated sorting process is used to measure the success of the collaboration. The results of Iorio and Li show that their proposed algorithm performed well compared to NSGA-II.

3 Multiobjective Optimization Problems

The optimization problem consists of three main elements, which are objective function(s), decision variables and constraints. Minimizing or maximizing the objective function is the aim of the problem. On the other hand, the decision variables affect the performance of the objective function(s), while the constraint is the limitation on the values of the decision variables. Usually, optimization problems can be classified into single objective optimization and multiobjective optimization. The single objective optimization is the optimization problem which involves only one objective function to find an optimal solution. However, a multiobjective or vector optimization problem has a number of conflicting objective functions subject to certain constraints which are to be minimized or maximized.

The common explanation of multiobjective optimization problems (MOPs) [1] can be formally defined as:

$$\text{minimize (or maximize) } \boldsymbol{z} = f(\boldsymbol{v}) = [f_1(v_1,\ldots,v_n),\ldots,f_m(v_1,\ldots,v_n)] \quad (1)$$

subject to

$$\text{constraints } p \text{ are } \geq \text{ restriction } a_i(\boldsymbol{v}) \geq 0 \quad (i = 1,2,\ldots,p) \quad (2)$$

$$\text{constraints } q \text{ are } \leq \text{ restriction } b_i(\boldsymbol{v}) \leq 0 \quad (i = 1,2,\ldots,q) \quad (3)$$

$$\text{constraints } r \text{ are equality restriction } c_i(\boldsymbol{v}) = 0 \quad (i = 1,2,\ldots,r) \quad (4)$$

where

$$\boldsymbol{v} = (v_1, v_2, \ldots, v_n) \in V$$
$$\boldsymbol{z} = (z_1, z_2, \ldots, z_m) \in Z$$

$a_i(\boldsymbol{v})$, $b_i(\boldsymbol{v})$ and $c_i(\boldsymbol{v})$ in the formula are the constraint functions. Vector $\boldsymbol{v}$, $(v_1, v_2, \ldots, v_n)$ in the decision space *V* is called the decision vector and *n* is the number of decision variables. Additionally, the vector *z*, $(z_1, z_2, \ldots, z_m)$ in the objective space Z is called the objective vector and m is the number of the objective functions. The entire decision vector under consideration must satisfy a given set of constraints to produce optimum solution values via the objective functions.

The concept of *Pareto dominance* [11] is defined as follows. An objective vector z^1 is said to dominate another objective vector z^2 or z^1 is said to be nondominated by z^2, $(z^1 \succ z^2)$ if z^1 is not worse than z^2 with respect to every objective and z^1 is strictly better than z^2 with respect to at least one objective. An optimal decision vector to a multiobjective problem is denoted as *Pareto optimal* if it is nondominated concerning the entire decision space, and at the same time its image in the objective space is denoted as a Pareto optimal objective vector. The set of all Pareto optimal objective vectors is called the *Pareto front* and the set of all the Pareto optimal decision vectors is called the *Pareto optimal set.*

4 Multiobjective Evolutionary Algorithm: SPEA2

SPEA2, the enhanced version of the original SPEA, is a relatively new approach for finding or approximating the Pareto set in multiobjective optimization problems. SPEA2 was proposed by Zitzler *et al.* [29] as shown in Algorithm 1.

Algorithm 1. SPEA2 Algorithm

```
Input:     M (offspring population size)
           N (archive size)
           T (maximum number of generations)

Output:    A* (nondominated set)

Step1: Initialization: Generate an initial population $P_0$ and create the empty archive (external set) $A_0 = \phi$. Set $t = 0$.

Step2: Fitness_assignment: Calculate fitness values of individuals in $P_t$ and $A_t$.

Step3: Environmental_selection: Copy all nondominated individuals in $P_t$ and $A_t$ to $A_{t+1}$. If size of $A_{t+1}$ exceeds $N$ then reduce $A_{t+1}$ by means of the truncation operator, otherwise if size of $A_{t+1}$ is less than $N$, fill $A_{t+1}$ with dominated individuals in $P_t$ and $A_t$.

Step4: Termination: If $t \geq T$ or another stopping criterion is satisfied then set $A^*$ to the set of decision vectors represented by the nondominated individuals in $A_{t+1}$. Stop.

Step5: Mating_selection: Perform binary tournament selection with replacement on $A_{t+1}$ in order to fill the mating pool.

Step6: Variation: Apply recombination and mutation operators to the mating pool and set $P_{t+1}$ to the resulting population. Increment generation counter ($t = t + 1$) and go to Step 2.
```

This algorithm is an elitist multiobjective evolutionary algorithm which incorporates a fine-grained fitness assignment strategy, a density estimation technique and an improved archive truncation method. Two of the main methods to assign the fitness values in SPEA2 are to use the dominance rank and dominance count, which takes into consideration, for every individual, the number of individuals it dominates and the number of individuals that dominate it. Additionally, SPEA2 uses the *k*-th nearest neighbor technique, which is a density estimation method, to maintain diversity within the obtained Pareto front, which guides the search towards the Pareto set more effectively. Moreover, if the nondominated individuals exceed the archive size, the truncation method is utilized to maintain a constant number of elitists in the archive, which iteratively removes individuals from the archive until the desired archive size is achieved (i.e. the number of nondominated individuals in the archive is equal to archive size).

5 Proposed Algorithms

The research question that motivates the work conducted in this chapter is:

Can coevolutionary methods be used to augment and improve the performance of SPEA2 for multiobjective optimization?

In this research, SPEA2-CE-KR and SPEA2-CC are proposed as different models of SPEA2 by adding two techniques respectively:

i. Competitive coevolution techniques
ii. Cooperative coevolution techniques

Specifically, these augmentations are achieved through the following operations:

i. **Opponents Selection:** Selects the individuals as opponents from the population.
ii. **Reward Assignment:** Computes the reward value for each individual against a set of opponents in each competition.
iii. **Chromosome Selection:** Decomposes the population into various subpopulations, and forms the complete solution from alternative subpopulations.

In the case of cooperative coevolution, the basic idea is that the population of SPEA2 can be dynamically decomposed according to the number of decision variables and solved in parallel. However in competitive coevolution, the opponents or competitors are selected based on *K*-Random opponents' strategies from the population of SPEA2. The fitness values of individuals depend on the competitions.

5.1 SPEA2-CE-KR

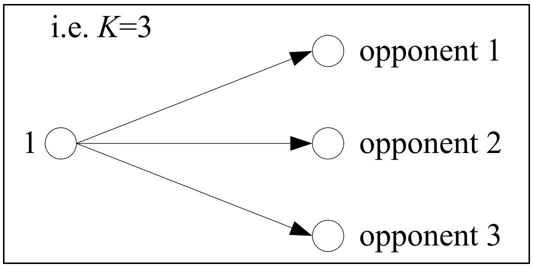
The augmented algorithm SPEA2-CE-KR is presented here, which is the integration of SPEA2 with CE. The CE method will be implemented using the KR strategy. Generally, the framework of this algorithm is similar to the framework of SPEA2 with the exceptions of two additional methods, Opponents_selection and Reward_assignment as shown in Algorithm 2.

Algorithm 2. SPEA2-CE-KR Algorithm

```
BEGIN SPEA2-CE-KR

$gen = 0$
$P_s(gen)$ = randomly initialize population // initial population
Fitness_assignment $P_s(gen)$ // calculate the individual's fitness value
Opponents_selection $P_s(gen)$ // choose the opponents based on the KR
Reward_assignment $P_s(gen)$ // calculate the individual's reward value
Environmental_selection $P_s(gen)$ /* copy nondominated individuals into new archive */

    WHILE Termination = False

        $gen = gen + 1$
        Mating_selection $P_s(gen)$ // perform binary tournament selection
        Variation $P_s(gen)$ // apply crossover and mutation
        Fitness_assignment $P_s(gen)$ // calculate the individual's fitness value
        Opponents_selection $P_s(gen)$ // choose the opponents based on the KR
        Reward_assignment $P_s(gen)$ // calculate the individual's reward value
        Environmental_selection $P_s(gen)$ /* copy nondominated individuals into new archive */

    END

END SPEA2-CE-KR
```

At the initialization stage, SPEA2-CE-KR randomly generates an initial population of individuals. The individuals represent the possible solutions to the problem. Then, the fitness values for each individual in the population are evaluated. Next, the Opponent_selection method selects individuals as the opponents based on the KR strategy. Panait and Luke [16] explained a *K*-Random Opponents strategy as shown in Fig. 1, where each member competes against *K* other opponents. The opponents will be randomly selected from the same population without repetition and rejects self-play. This strategy is the most commonly used, easiest to understand and straightforward to implement. In this investigation, the *K* is tested with the values of 10, 20, 30, 40, 50, 60, 70, 80 and 90. After that, each individual competes against the entire set of opponents. During the tournament, the reward value is calculated for each competition by the reward function. Each reward value is summed up as the fitness score for the individual using the Reward_assignment method. In every generation, the number of competitions is set according to the *K* values. Subsequently, the archive update operation is executed. The archive is updated by copying nondominated individuals into the archive and the archive truncation method is used to maintain boundary solutions. Any dominated solutions are removed from the archive during the update operation. Individuals are then selected, based on their fitness, to form offspring by performing the genetic operations of simulated binary crossover and polynomial mutation.

Fig. 1 *K*-random opponents



Each loop iteration is referred to as a generation. The run of SPEA2-CE-KR terminates when the termination criterion is satisfied. The predefined maximum number of generations serves as the termination criterion of the loop.

The description of the reward function is as (5). *I* represents the participating individual, while *O* represents the opponent. *R* is the raw fitness value, max(*R*) is the maximum raw fitness value and the min(*R*) is the minimum raw fitness value. The range of values in this function is within [-1, 1]. If *Reward*(*I*, *O*) = 0, it corresponds to the competition being a draw.

$$Reward(I,O) = \frac{R(O) - R(I)}{\max(R) - \min(R)} \tag{5}$$

5.2 SPEA2-CC

A combination of SPEA2 with cooperation coevolution is developed in an attempt to create an evolutionary algorithm which further improves the optimization performance of SPEA2. The general framework of SPEA2-CC is shown in Algorithm 3. SPEA2-CC is proposed as a different model to SPEA2 that includes chromosome selection, which is the operation to decompose the population into various subpopulations based on the decision variables, and forming the complete solution from alternative subpopulations, whereas SPEA2 involves only a single population.

With cooperative coevolution, the subpopulations are initialized randomly where each subpopulation represents a decision variable. After that, if the terminating criterion is not met, the CC architecture allows cooperating subpopulations to evolve in parallel and then combine together to produce a complete solution. Each subpopulation is genetically isolated from the others and evolves on its own, similar to the original SPEA2. For evaluation, each subpopulation is considered in turn. Each individual in the population is combined with a representative of the other species so that the combination forms one complete solution. If an obtained solution is neither dominated by any solutions in the archive nor is a duplicate of a solution, this solution is added to the archive and the dominated solution in the archive is discarded. The truncation method is utilized to maintain a constant number of elitists in the archive. The SPEA2-CC is implemented with binary tournament selection, simulated binary crossover and polynomial mutation to produce a set of new individuals.

Generally, a cooperative coevolution model has two methods for selecting collaborators for the purpose of fitness evaluation, known as CC-1 and CC-2 [27]:

Algorithm 3. SPEA2-CC Algorithm

```
BEGIN SPEA2-CC
gen = 0

/* initial fitness of each subpopulation's (species) individual by combining it with
a random individual from other species */

  FOR each species s
    $P_s(gen)$ = randomly initialized population
    Fitness_assignment $P_s(gen)$  // calculate the individual's fitness value
    Environmental_selection $P_s(gen)$ /*copy nondominated individuals into new
                                          archive */
  END

/* compute the fitness of each subpopulation's (species) individual by combining
it with a selected individual from other species */

  WHILE Termination = False
  gen = gen + 1

    FOR each species s
      Mating selection $P_s(gen)$ // perform binary tournament selection
      Variation $P_s(gen)$ // apply crossover and mutation
      Fitness_assignment $P_s(gen)$ // calculate the individual's fitness value
      Environmental_selection $P_s(gen)$ /* copy nondominated individuals into
                                            new archive */
    END

  END

END SPEA2-CC
```

i. **CC-1:** Choose the best individuals from alternative subpopulations, as defined by the fitness obtained from the last evaluation process of that group.
ii. **CC-2:** Select two individuals, the best and a random individual. Evaluate both with the current individual and use the higher objective function value for the current individual's fitness score.

In this proposed algorithm, the first method is used, that is to select the current best individuals from alternative subpopulations for generating a complete chromosome for fitness evaluation.

6 Test Problems and Performance Measures

The algorithms will be benchmarked using seven test problems, DTLZ1 to DTLZ7 for the following reasons [5]:

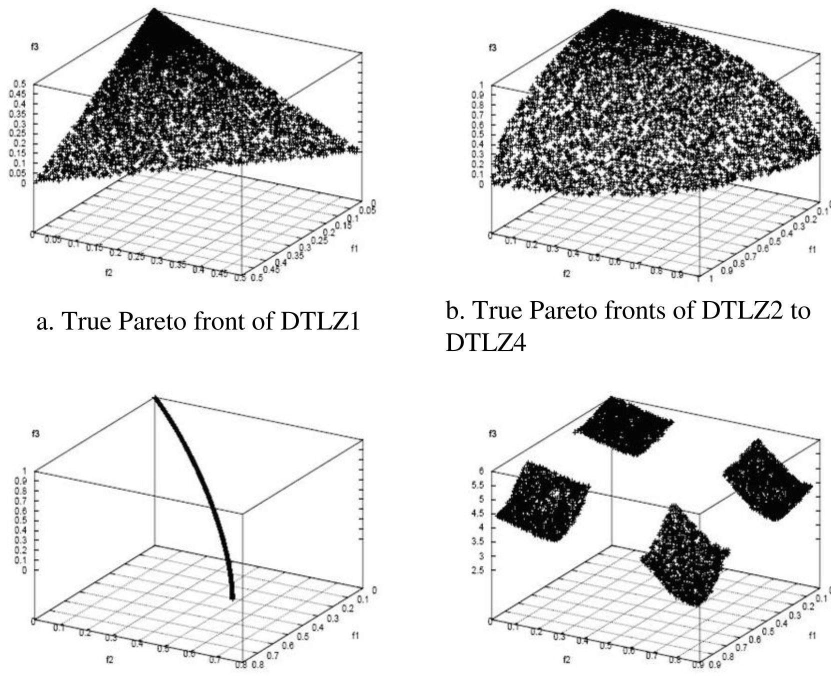
i. It is one of the latest sets of test problems for multiobjective benchmarking but more importantly, these problems can be tested with varying numbers of decision variables and objectives. It is the most-widely used benchmark for MOEAs.

ii. The structure of the test problems is easy to employ and is formed by using a bottom-up technique.
iii. The test problems have exact known shapes as well as the location of the resulting true Pareto front. The corresponding optimal decision variable values are also known.

The views of the true Pareto front are illustrated in Fig. 2. Moreover, Table 1 summarizes the geometrical properties of the test problems.

Table 1 The geometries of DTLZ

Test Problem	Geometry
DTLZ1	Linear
DTLZ2	Concave
DTLZ3	Concave
DTLZ4	Concave
DTLZ5	Curve
DTLZ6	Curve
DTLZ7	Disconnected



a. True Pareto front of DTLZ1

b. True Pareto fronts of DTLZ2 to DTLZ4

c. True Pareto fronts of DTLZ5 and DTLZ6

d. True Pareto front of DTLZ7

Fig. 2 True Pareto fronts of DTLZ test problems

Three performance metrics were used to validate the proposed algorithms, namely generational distance, spacing and coverage metrics. The metrics were chosen because they are the most commonly used evaluation criteria for comparison among MOEAs [2, 14, 19].

Generational Distance (*GD*): The generational distance metric for evaluating the convergence to the obtained Pareto front was proposed by Van Veldhuizen and Lamont [24]. It is used for estimating how far the elements in the obtained Pareto front are from the true Pareto front of the problem. This metric is defined as:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{6}$$

where n is the number of nondominated vectors found by the algorithm being analyzed and d_i is the Euclidean distance (measured in objective space) between each of these vectors and the nearest member of the true Pareto front. A value of $GD = 0$ indicates that all the elements generated are in the true Pareto front of the problem. Therefore, any other value will indicate how "far" the obtained solutions are from the true Pareto front of the problem.

Spacing (*SP*): The spacing metric for evaluating the diversity of the optimal solutions was introduced by Schott [20]. It measures how evenly the points in the approximation set are distributed in the objective space. This metric is defined as:

$$SP = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\bar{d} - d_i)^2} \tag{7}$$

where $d_i = \min_j (\sum_{k=1}^{m} \left| f_m^i - f_m^j \right|)$, $i, j = 1, \ldots, n$. f is the objective function and m is the number of objectives, $\bar{d}$ is the mean of all d_i, and n refers to the number of elements of Pareto optimal set found so far. If $SP = 0$ it means that all the nondominated solutions found are equidistantly spaced.

Coverage (*C*): This metric was proposed by Zitzler *et al.* [28]. By using this metric, two sets of nondominated solutions can be compared to each other. Consider X', X'' as two sets of phenotype decision vectors. C is defined as the mapping of the ordered pair (X', X'') to the interval [0, 1]:

$$C(X', X'') \triangleq \frac{|\{a'' \in X''; \exists a' \in X' : a' \succeq a''\}|}{|X''|} \tag{8}$$

$a' \succeq a''$ if a' dominates a'' or a' equal to a''. The value $C(X', X'') = 1$ means that all the decision vectors in X'' are dominated by X'. The value $C(X', X'') = 0$ represents the situation when none of the points in X'' are dominated by X'. In addition, if $C(X', X'') > C(X'', X')$, then X' is better than X''. Fig. 3 shows

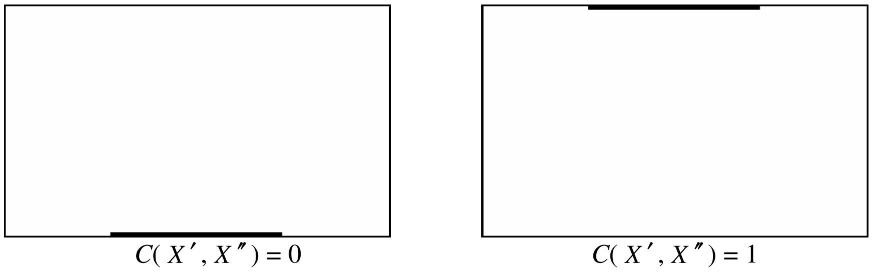


Fig. 3 $C(X', X'') = 0$ and $C(X', X'') = 1$

the graphical presentations for coverage metric. The scale is 0 (no coverage) at the bottom and 1 (total coverage) at the top per rectangle. Thus, a value of 0 means X' is totally dominated by X'' and vice-versa for a value of 1.

7 Parameters Used in the Experiments

In order to have a fair comparison of all algorithms, all runs considered are implemented with the same real-valued representation, simulated binary crossover

Table 2 Evolutionary settings

Representation	: Real-valued vector
Crossover	: Simulated binary crossover
Mutation	: Polynomial mutation
Selection	: Binary tournament selection

Table 3 Parameter settings

Parameter	SPEA2, SPEA2-CE-KR	SPEA2-CC
Population size	100	100
Archive size	100	100
Number of decision variables per generation	12	12
Number of objectives	3 to 5	3 to 5
Number of generations	600	50
Mutation probability	0.08	0.08
Crossover probability	1	1
Polynomial mutation operator	20	20
SBX crossover operator	15	15
Number of repeated runs	30	30
Population size per species	100	100
Total population size per generation	100	1200
Number of species per generation	1	12

(SBX), polynomial mutation and tournament selection. The details of the evolutionary mechanisms are as shown in Table 2 and Table 3, which lists all the parameter settings for each evolutionary multiobjective optimization algorithm.

Each of the DTLZ parameters is represented as a vector of a real number. The population size and the archive size are both set to 100. Parents are selected through binary tournament selection. Offspring are created with the simulated binary crossover operator and the polynomial mutation operator. The distribution parameters associated with the simulated binary crossover and the polynomial mutation operators are $\eta_c = 15$ and $\eta_m = 20$. The crossover probability is 1.0 and the mutation probability is $1/n$, where n is the total number of decision variables, giving 1/12 = 0.08.

In the CC approach, the number of species (subpopulations) per generation is identical to the number of decision variables per generation [9, 10, 13]. In this study, both parameters are fixed to 12. The total population size per generation is determined by using (population size * number of species per generation), thus in the SPEA2-CC, this parameter is equal to 1200. On the other hand, the SPEA2-CE-KR and the original SPEA2 utilize only a single population for each generation, so the total population size per generation is equal to 100.

Furthermore, the total number of evaluations in each run is set to 60,000 and the number of generations is determined by following equation:

$$gen = \frac{e}{p * s} \tag{9}$$

where p is the population size, s is the number of species per generation and e is the number of evaluations. In the experiments, each evolutionary setup was repeated 30 times for each test problem.

8 Optimization Results

The results of applying SPEA2-CC and SPEA2-CE-KR to solve DTLZ1 through DTLZ7 are presented and compared to the original SPEA2. The appendix provides a detailed presentation of the experimental results. In the figures and tables, some symbols are utilized to represent the name of the algorithms. The symbol S2 corresponds to SPEA2 and the symbol CK represents SPEA2-CE-KR and the number refers to the K values. For example, CK10 corresponds to the SPEA2-CE-KR with 10 random opponents. The symbol CO indicates SPEA2-CC. The overall best result for each test problem for the different numbers of objectives is emboldened in the respective tables. Additionally, in the comparison of two means (SPEA2 against the proposed algorithm), t-tests were conducted at 99% confidence interval for the generational distance and spacing metrics. A (+) sign after the mean values indicates that a significantly better result was obtained compared to SPEA2, while a (-) sign indicates a significantly worse result was obtained. From the experimental results, the following can be observed.

Generational distance (*GD*): Fig. 4 to Fig. 6 depict the generational distance results over 3 to 5 objectives in box plot format. The leftmost box plot relates to SPEA2, second leftmost box plot relates to SPEA2-CC, while boxes from the third leftmost onwards relate to SPEA2-CE-KR with a set of *K* values (10, 20, 30, 40, 50, 60, 70, 80 and 90).

SPEA2-CC is observed to be better than SPEA2 and SPEA2-CE-KR in most of the DTLZ test problems. However, this proposed algorithm was unable to locate the true Pareto optimal solutions of the test problem DTLZ4 with 3 objectives, which is a problem with a concave Pareto front. However, the performance of SPEA2-CC increased when the number of objectives increased. It is interesting to note that in all seven problems of this study, SPEA2-CC outperformed SPEA2 for four and five objectives. However, the performance of SPEA2-CC against SPEA2-CE-KR is comparable in 4-objective problems. By contrast, SPEA2-CC has good performance in 5-objective problems compared to SPEA2-CE-KR regarding the average distance from the nondominated solutions to the true Pareto front. These presented results are significant in five out of seven test problems, which are DTLZ1, DTLZ2, DTLZ3, DTLZ5 and DTLZ6.

Additionally, the current study found that SPEA2-CE-KR is better than SPEA2 in almost all of the DTLZ problems except DTLZ5 with 3 objectives. However for 4 and 5 objectives, the results reveal that SPEA2-CE-KR strongly outperformed SPEA2 for the entire set of test problems. Also for several of the test problems, the box plots are close to the bottom of the rectangle, which means that SPEA2-CE-KR was successful as it was able to converge toward the true Pareto front.

Spacing (*SP*): Fig. 4 to Fig. 6 display the box plots of spacing metric, in which the box plot on the left is the SPEA2 algorithm, second left box plot is the SPEA2-CC while boxes from the third left to the last of the box plots are SPEA2-CE-KR algorithm with different *K* values.

In the current study, comparing SPEA2-CC with SPEA2 and SPEA2-CE-KR shows that SPEA2-CC performed well in most of the problems with 3 objectives except DTLZ2 and DTLZ4, which are problems with concave Pareto fronts. Another finding was that SPEA2-CC has the best performance for all the considered test problems with 4 objectives. Similarly, SPEA2-CC had good optimization results in five out of seven problems compared to SPEA2-CE-KR, which are DTLZ1, DTLZ2, DTLZ3, DTLZ5 and DTLZ7. Only in DTLZ7 with 5 objectives, SPEA2 had better diversity than SPEA-CC. Meanwhile, SPEA2-CE-KR produced better spacing results in DTLZ4 and DTLZ7 compared to SPEA2-CC.

Furthermore, the results of this study show that in test problems with 3 objectives, the performances of SPEA2-CE-KR and SPEA2 are similar. Once the number of objectives was increased to four and five, the performance of SPEA2-CE-KR was noticeably better than SPEA2 in distributing nondominated solutions evenly along the Pareto front.

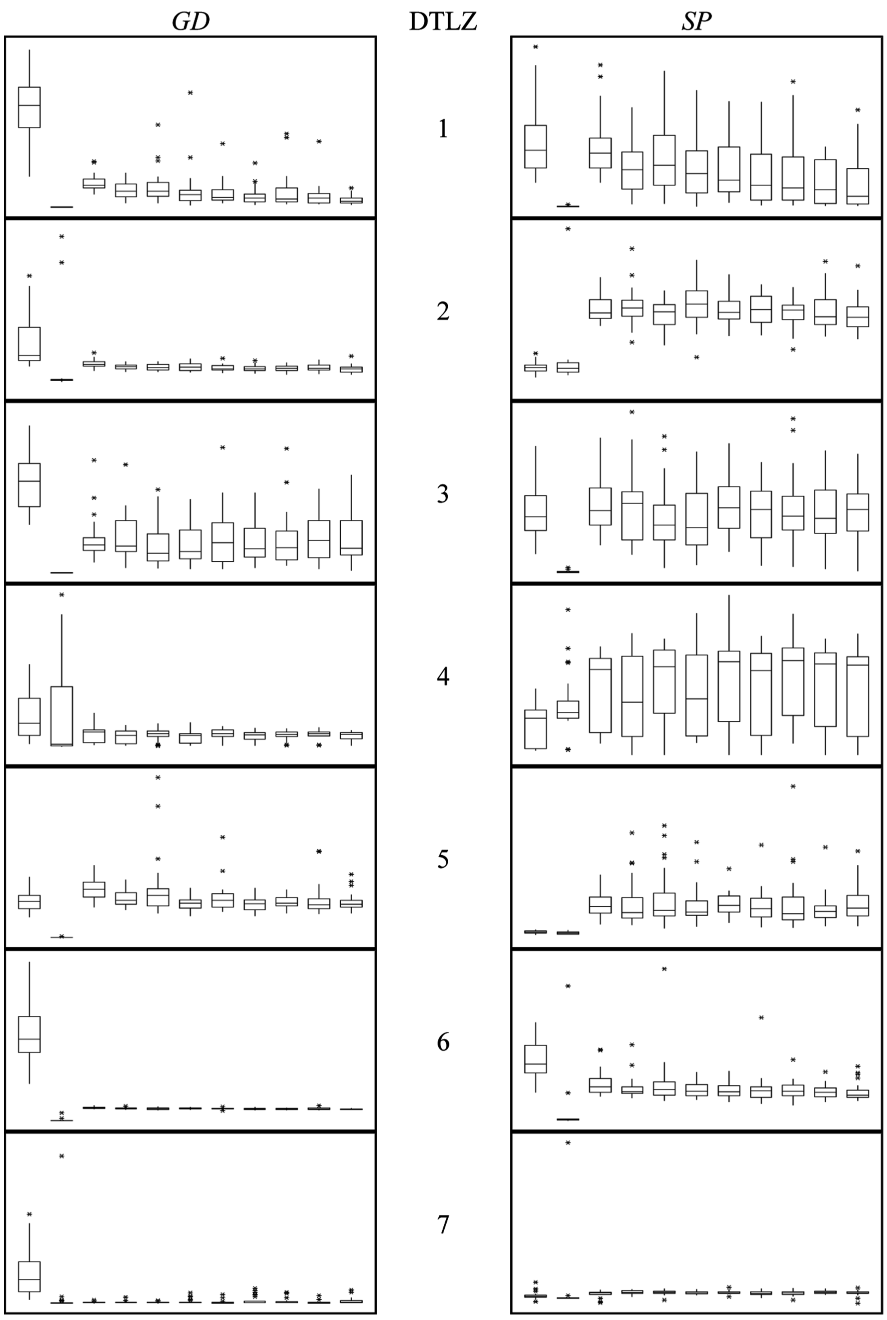


Fig. 4 Box plots of SPEA2, SPEA2-CC and SPEA2-CE-KR for generational distance (*GD*) and spacing (*SP*) with 3 objectives

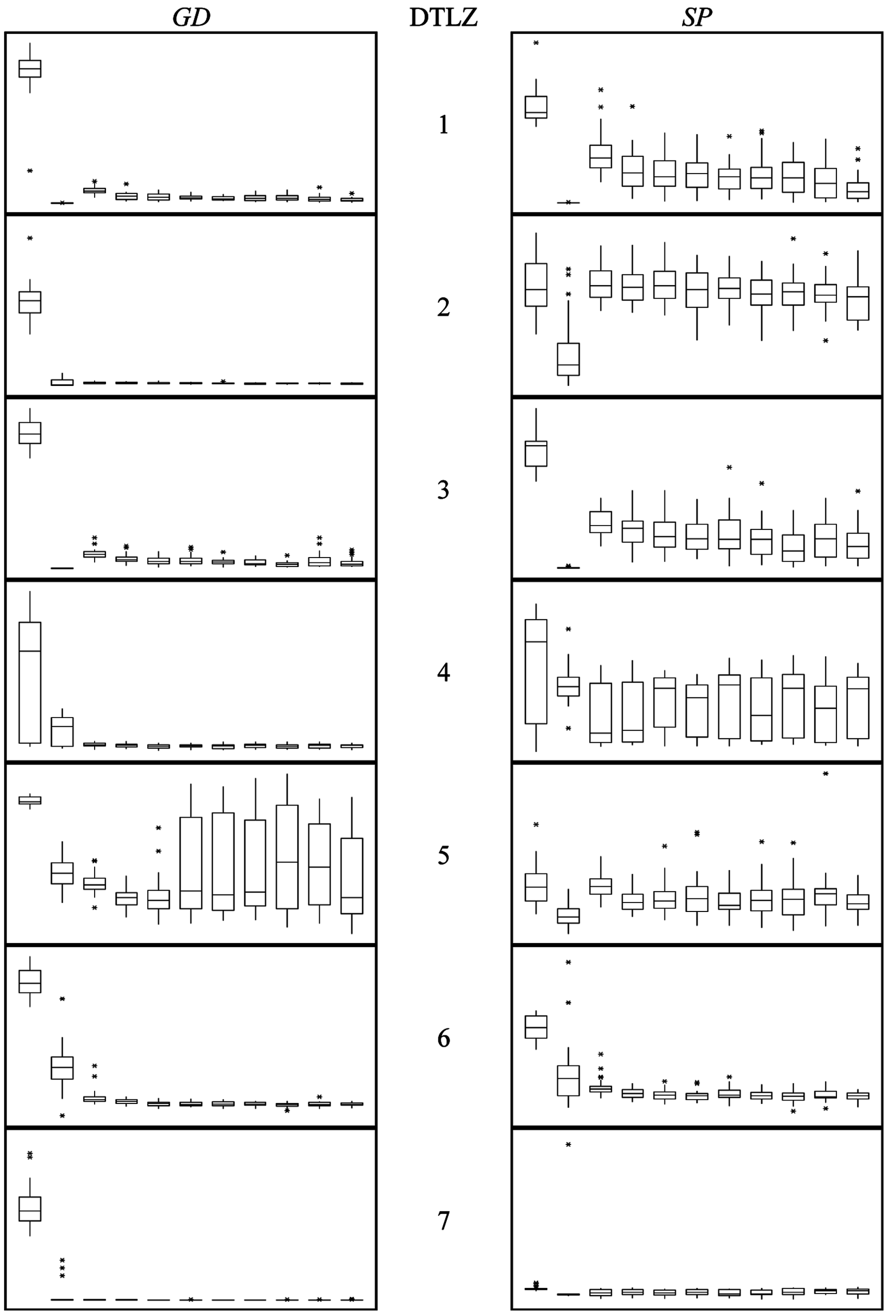


Fig. 5 Box plots of SPEA2, SPEA2-CC and SPEA2-CE-KR for generational distance (*GD*) and spacing (*SP*) with 4 objectives

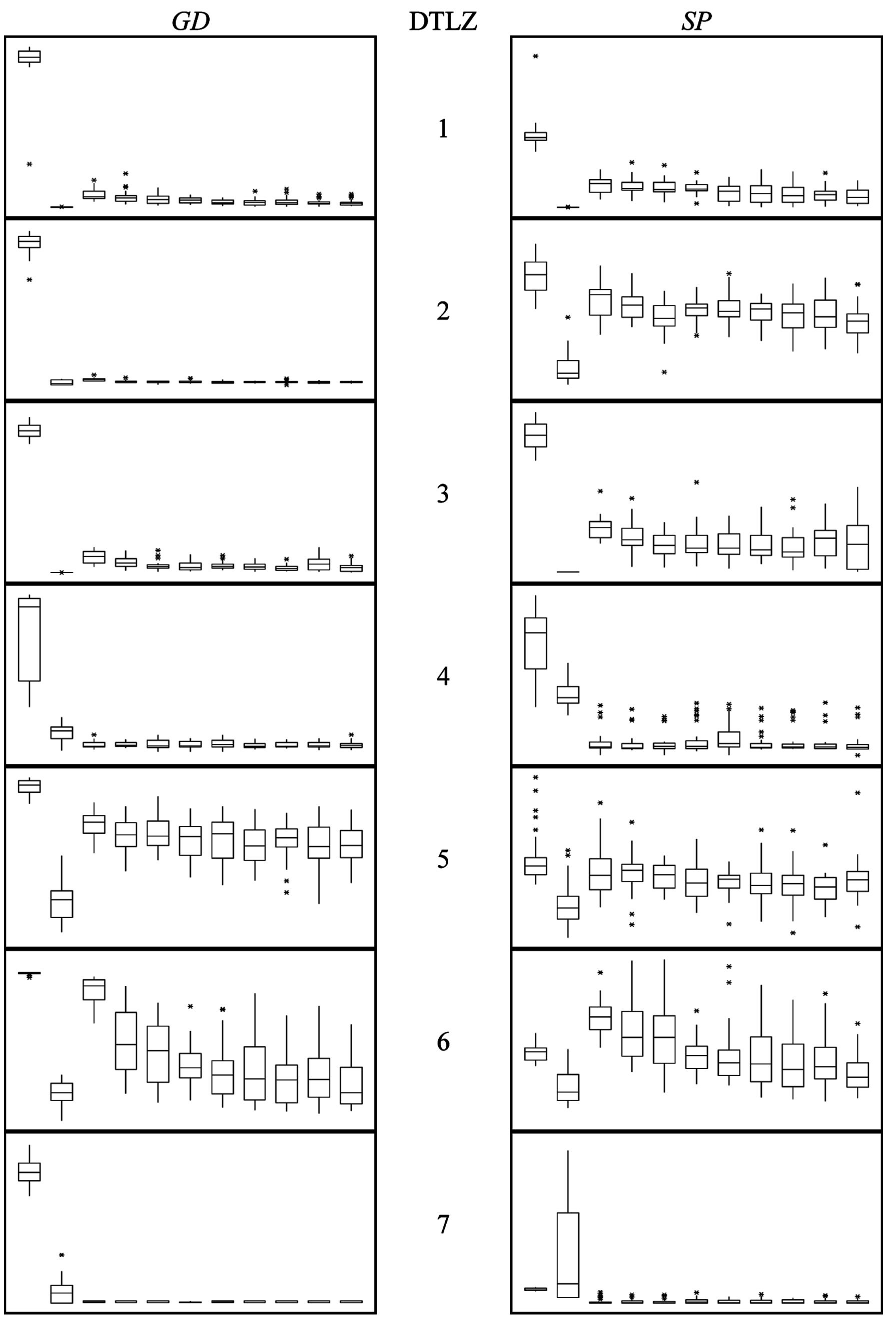


Fig. 6 Box plots of SPEA2, SPEA2-CC and SPEA2-CE-KR for generational distance (*GD*) and spacing (*SP*) with 5 objectives

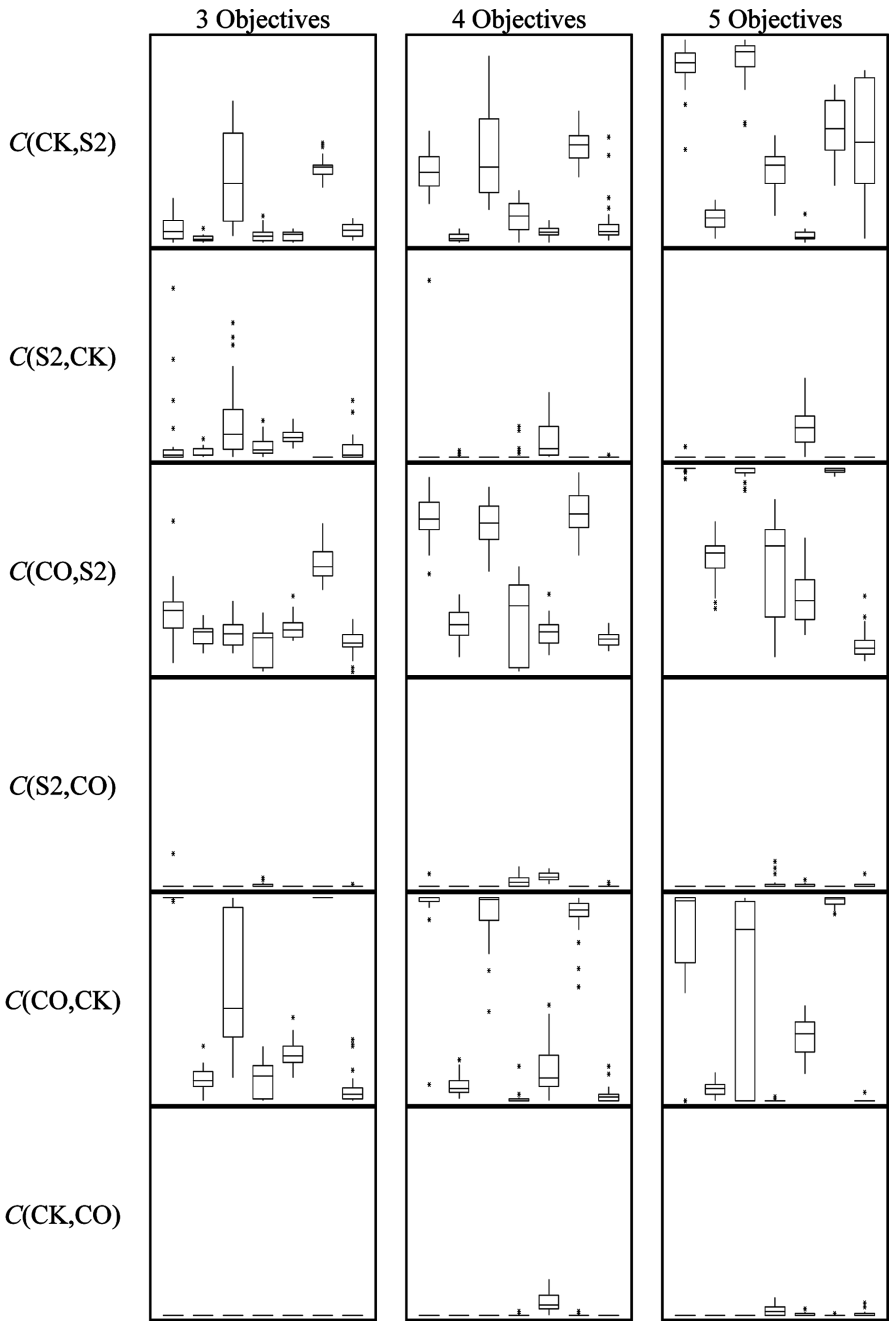


Fig. 7 Box plots of SPEA2, SPEA2-CC and SPEA2-CE-KR with 90 selected opponents for coverage with 3 to 5 objectives

Coverage (*C*): The box plots of *C* values from all seven problems are shown in Fig. 7. In each rectangle, from the leftmost box to the rightmost box represent the test problems from DTLZ1 to DTLZ7. The box plots of SPEA2-CE-KR with 90 opponents has been selected to be presented in Fig. 7, because the results show that this proposed algorithm with 90 opponents has a better coverage level compared to other *K* values.

SPEA2-CC is superior to SPEA2 in all the DTLZ test problems with three to five objectives. As can be noticed from the figure, all the box plots are approximately close to or at the bottom of the rectangle for *C*(SPEA2, SPEA2-CC). On the other hand, in all DTLZ problems with 3 objectives, the box plots obtained clearly indicate that SPEA2-CC outperformed SPEA2-CE-KR and also almost all the box plots have a *C* metric value of 0. In the 4-objective problems, SPEA2-CC has excellent coverage because most of the box plots are very close to or at the base of the plot. In the 5-objective DTLZ4 and DTLZ7 problems, SPEA2-CE-KR weakly dominated the nondominated solutions obtained by SPEA2-CC. However, SPEA2-CC provided nondominated solutions with very good coverage for DTLZ1, DTLZ2, DTLZ3, DTLZ5 and DTLZ6 problems.

Furthermore, SPEA2-CE-KR shows regular coverage of nondominated solutions for DTLZ test problems with 3 objectives compared to SPEA2. But for 4 and 5 objectives, the obtained nondominated solutions found by SPEA2-CE-KR clearly dominated the obtained nondominated solutions found by SPEA2 in almost all of the DTLZ test problems. The SPEA2-CE-KR's weakest results against SPEA2 are in DTLZ5. As can be seen from the box plots of the 4- and 5-objective problems, some box plots in the column *C*(SPEA2, SPEA2-CE-KR) are at the bottom of the rectangle, which indicates that the proposed algorithm is completely free from the domination of SPEA2.

t-tests: Tables 4 through 17 include the t-test results for generational distance and spacing respectively. For generational distance with 3 to 5 problems, it is clear from these results that SPEA2-CC and SPEA2-CE-KR are significantly better than SPEA2 in almost all of the test problems. On the other hand, for the spacing with 3-objective test problems, SPEA2-CE-KR is significantly worse in DTLZ2, DTLZ4, DTLZ5 and DTLZ7, whilst SPEA2-CC is significantly worse only in DTLZ4. However, when the number of objectives is increased to 4 and 5, SPEA2-CC and SPEA2-CE-KR are again significantly better.

9 Discussion

In this chapter, the performances of the proposed algorithms are evaluated by comparing them against SPEA2 since SPEA2 is currently one of the best performing MOEAs. SPEA2 has already been clearly shown to outperform other MOEAs such as NSGA-II and PESA [29]. To date, a very large majority of new studies still utilize SPEA2 as a common benchmark for comparing their proposed algorithms, some of which are still unable to outperform SPEA2 comprehensively [8, 14]. Hence, the proposed algorithms which show improvements over the original

SPEA2 will similarly compare favorably to other algorithms that have been outperformed by SPEA2.

From the experimental analysis, the KR competitive coevolution helps to enhance the SPEA2 optimization performance in terms of convergence, diversity and coverage level. These results may be explained by the fact that SPEA2-CE-KR is proposed as a different model to SPEA2 in terms of fitness evaluation of the individuals. In the SPEA2, individuals are evaluated simply by using a fitness function to gain the raw fitness values. However in SPEA2-CE-KR, although individuals are also evaluated by using a fitness function, this is only the first part of the evaluation process. The actual fitness values of each individual are based on the reward function determined by comparing the raw fitness of each individual with that of a set of opponents from the archive.

On the other hand, the new mechanism added to SPEA2 in the form of SPEA2-CC is the utilization of multiple populations that work independently of each other during the optimization process. This essentially resulted in the niching of optimization effectiveness for the individual populations to a certain area of the search space of the problem. In other words, each population has specialized itself to optimize only certain variables of the optimization problems. Hence, this is the main reason why SPEA2-CC performed better than SPEA2 in that each individual population could concentrate its optimization effort on searching and finding the best values for a limited number of variables within its niched area of optimization. In contrast, SPEA2 is not able to separate its optimization efforts to focus only on certain parts of the search space since it has neither the means nor the capacity of achieving this through its heterogeneous composition of individuals within a single combined population, which are unable to isolate themselves from other competing individuals in the single population. Consequently, the optimization effort of SPEA2 is diluted over the entire search space while in contrast, SPEA2-CC is focused only a small, niched area of the search space.

10 Conclusions and Future Work

Two new coevolutionary SPEA2 algorithms for multiobjective optimization called SPEA2-CE-KR and SPEA2-CC were proposed. The performances of these augmented algorithms were compared to the original SPEA2. The main findings as a result of the work conducted in this study are summarized below:

i SPEA2-CC was the most successful performer compared to SPEA2-CE-KR and SPEA2. It can achieve very good convergence and diversity values as well as the solution set coverage.

ii SPEA2-CE-KR was also successful and outperformed SPEA2 in terms of the average distance from the obtained optimal solutions to the true Pareto front, the distribution of the obtained solution set and the solution set coverage.

iii As the number of dimensions to be optimized increased, the performance of the proposed coevolution-augmented algorithms also increased in the large majority of cases, compared to the original SPEA2.

Based on the experimental results, both the competitive and cooperative coevolution can greatly assist in enhancing the optimization performance of SPEA2, especially using the cooperative coevolution method. Overall, the introduction of the coevolutionary learning effectively improved the performance of an evolutionary multiobjective optimizer in terms of the convergence to the true Pareto front, the diversity distribution of the obtained nondominated solutions, coverage level and for better scalability to higher dimensional problems.

This research has generated many questions which require further investigation. It would be interesting to investigate whether a hybrid of competitive coevolution and cooperative coevolution would be able to further improve the performance of MOEAs, since cooperative coevolution can focus on the individual objectives using speciation and competitive coevolution can focus on the opposition-based learning [22] between individual solutions. It would also be highly informative to conduct further tests of scalability to higher dimensions for the proposed algorithms since the current results already demonstrate that proposed algorithms perform better than the original algorithm as the number of objectives was increased from 3 to 5 dimensions.

Also, it has been shown only very recently that algorithm Optimized Multi-Objective Particle Swarm Optimization (OMOPSO) is able to outperform SPEA2 [6]. As future work, it would be interesting to compare these proposed algorithms against OMOPSO.

Acknowledgments. The first author is supported by a National Science Fellowship (NSF) from the Ministry of Science, Technology and Innovation (MOSTI), Malaysia.

Appendix

Table 4 Summarization of mean and standard deviation of generational distance (*GD*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ1 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	9.692000	2.987000	33.740000	5.610000	46.520000	6.510000
CO	**0.016100**$^+$	0.013250	**0.033040**$^+$	0.027840	**0.032420**$^+$	0.031640
CK10	2.306000$^+$	0.782000	3.247000$^+$	1.034000	3.910000$^+$	1.711000
CK20	1.619000$^+$	0.818000	1.817000$^+$	0.974000	3.194000$^+$	2.004000
CK30	1.948000$^+$	1.529000	1.666000$^+$	0.947000	2.566000$^+$	1.446000
CK40	1.619000$^+$	2.002000	1.439000$^+$	0.563000	2.190000$^+$	0.900000
CK50	1.309000$^+$	1.124000	1.212600$^+$	0.533600	1.614000$^+$	0.842000
CK60	1.114000$^+$	0.868000	1.301000$^+$	0.722000	1.575000$^+$	1.053000
CK70	1.394000$^+$	1.642000	1.399000$^+$	0.854000	1.659000$^+$	1.305000
CK80	1.046000$^+$	1.114000	1.113000$^+$	0.847000	1.402000$^+$	0.828000
CK90	0.725600$^+$	0.428700	0.851700$^+$	0.533500	1.401000$^+$	1.000000

Table 5 Summarization of mean and standard deviation of generational distance (*GD*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ2 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	0.003689	0.001755	0.062900	0.012560	0.141180	0.009030
CO	**0.001734**$^+$	0.002298	0.006845$^+$	0.002789	**0.012570**$^+$	0.002200
CK10	0.002249$^+$	0.000264	0.006647$^+$	0.000544	0.015267$^+$	0.001333
CK20	0.002051$^+$	0.000193	0.006527$^+$	0.000626	0.013589$^+$	0.001257
CK30	0.002007$^+$	0.000212	0.006280$^+$	0.000628	0.013094$^+$	0.000968
CK40	0.002036$^+$	0.000265	0.006388$^+$	0.000612	0.013241$^+$	0.001240
CK50	0.001973$^+$	0.000219	0.006352$^+$	0.000515	0.012942$^+$	0.000957
CK60	0.001942$^+$	0.000224	0.006105$^+$	0.000427	0.012947$^+$	0.000829
CK70	0.001948$^+$	0.000218	0.006107$^+$	0.000381	0.013064$^+$	0.001247
CK80	0.001964$^+$	0.000227	0.006164$^+$	0.000443	0.012837$^+$	0.000919
CK90	0.001886$^+$	0.000256	**0.006066**$^+$	0.000429	0.012938$^+$	0.000924

Table 6 Summarization of mean and standard deviation of generational distance (*GD*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ3 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	19.110000	5.381000	73.000000	7.870000	118.610000	5.700000
CO	**0.060900**$^+$	0.060000	**0.056520**$^+$	0.049800	**0.059380**$^+$	0.049440
CK10	6.941000$^+$	4.228000	7.842000$^+$	2.849000	13.03100$^+$	5.146000
CK20	7.545000$^+$	4.646000	5.356000$^+$	2.584000	8.401000$^+$	4.094000
CK30	5.625000$^+$	4.256000	4.178000$^+$	2.129000	5.783000$^+$	3.705000
CK40	6.056000$^+$	4.071000	4.459000$^+$	2.777000	5.627000$^+$	3.746000
CK50	7.120000$^+$	5.740000	3.533000$^+$	1.701000	5.793000$^+$	3.216000
CK60	6.262000$^+$	4.028000	3.222000$^+$	1.696000	5.331000$^+$	2.950000
CK70	6.673000$^+$	5.469000	2.374000$^+$	1.451000	4.089000$^+$	2.540000
CK80	7.371000$^+$	4.611000	4.371000$^+$	3.957000	8.040000$^+$	6.340000
CK90	7.569000$^+$	5.444000	3.154000$^+$	2.405000	4.422000$^+$	3.528000

Table 7 Summarization of mean and standard deviation of generational distance (*GD*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ4 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	0.003235	0.001623	0.047410	0.029130	0.125830	0.038940
CO	0.003257	0.003263	0.014550$^+$	0.007080	0.026360$^+$	0.007950
CK10	0.001949$^+$	0.000512	0.006746$^+$	0.001131	0.015870$^+$	0.003744
CK20	0.001778$^+$	0.000435	0.006551$^+$	0.001080	0.016043$^+$	0.002673
CK30	0.001937$^+$	0.000424	0.006018$^+$	0.001097	0.016413$^+$	0.004181
CK40	**0.001758**$^+$	0.000395	0.006022$^+$	0.001062	0.015638$^+$	0.003222
CK50	0.001917$^+$	0.000387	**0.005852**$^+$	0.001155	0.016380$^+$	0.003657
CK60	0.001799$^+$	0.000389	0.006244$^+$	0.001189	**0.015299**$^+$	0.002971
CK70	0.001870$^+$	0.000327	0.005869$^+$	0.001108	0.015377$^+$	0.003034
CK80	0.001903$^+$	0.000336	0.006196$^+$	0.001285	0.015659$^+$	0.003024
CK90	0.001823$^+$	0.000350	0.006131$^+$	0.001036	0.015561$^+$	0.003229

Table 8 Summarization of mean and standard deviation of generational distance (*GD*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ5 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	0.000378	0.000105	0.137610	0.003870	0.168450	0.005550
CO	**0.000021**$^+$	0.000002	0.077180$^+$	0.012320	**0.082630**$^+$	0.014660
CK10	0.000503$^-$	0.000102	0.067310$^+$	0.009430	0.140890$^+$	0.009030
CK20	0.000407	0.000081	**0.055040**$^+$	0.008940	0.131820$^+$	0.012600
CK30	0.000492	0.000290	0.056210$^+$	0.016760	0.133220$^+$	0.012570
CK40	0.000356	0.000067	0.080160$^+$	0.039710	0.128740$^+$	0.013820
CK50	0.000413	0.000142	0.078390$^+$	0.040110	0.128630$^+$	0.016290
CK60	0.000349	0.000065	0.078030$^+$	0.038570	0.123830$^+$	0.014280
CK70	0.000364	0.000062	0.088880$^+$	0.044920	0.127430$^+$	0.013990
CK80	0.000385	0.000151	0.083890$^+$	0.034450	0.124230$^+$	0.016860
CK90	0.000371	0.000085	0.070140$^+$	0.035950	0.125510$^+$	0.013660

Table 9 Summarization of mean and standard deviation of generational distance (*GD*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ6 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	0.105450	0.034230	0.516590	0.049630	0.806390	0.005660
CO	**0.000409$^+$**	0.001718	0.202600$^+$	0.079100	**0.253740$^+$**	0.052970
CK10	0.015030$^+$	0.001044	0.092410$^+$	0.028960	0.728800$^+$	0.060900
CK20	0.014478$^+$	0.001036	0.078690$^+$	0.009460	0.485100$^+$	0.150900
CK30	0.014149$^+$	0.001120	0.070310$^+$	0.010540	0.437000$^+$	0.140900
CK40	0.014515$^+$	0.001231	0.070240$^+$	0.009010	0.374600$^+$	0.095900
CK50	0.014206$^+$	0.000972	0.069210$^+$	0.010440	0.352300$^+$	0.123500
CK60	0.013800$^+$	0.001118	0.068540$^+$	0.007930	0.340600$^+$	0.139300
CK70	0.013869$^+$	0.001069	**0.065610$^+$**	0.008830	0.325100$^+$	0.124800
CK80	0.014164$^+$	0.001340	0.067870$^+$	0.009310	0.327900$^+$	0.124300
CK90	0.013555$^+$	0.000732	0.069020$^+$	0.006420	0.291700$^+$	0.108200

Table 10 Summarization of mean and standard deviation of generational distance (*GD*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ7 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	0.021630	0.015890	0.670800	0.138700	1.571800	0.148600
CO	0.005040$^+$	0.019080	0.028000$^+$	0.071300	0.135900$^+$	0.143500
CK10	0.001613$^+$	0.000451	0.006187$^+$	0.000827	0.024740$^+$	0.010030
CK20	0.001646$^+$	0.000846	0.005505$^+$	0.000789	0.021870$^+$	0.009090
CK30	**0.001419$^+$**	0.000488	0.005266$^+$	0.000930	0.023990$^+$	0.010150
CK40	0.002002$^+$	0.001753	0.005358$^+$	0.001250	**0.018890$^+$**	0.009220
CK50	0.001692$^+$	0.001306	0.005376$^+$	0.000776	0.023730$^+$	0.010280
CK60	0.002886$^+$	0.002978	0.005342$^+$	0.000761	0.021530$^+$	0.009880
CK70	0.002132$^+$	0.001894	0.005464$^+$	0.001020	0.021100$^+$	0.010060
CK80	0.001952$^+$	0.001602	**0.005124$^+$**	0.000923	0.021520$^+$	0.009880
CK90	0.002731$^+$	0.002518	0.005265$^+$	0.001200	0.023100$^+$	0.010140

Table 11 Summarization of mean and standard deviation of spacing (*SP*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ1 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	9.964000	5.157000	25.372000	4.639000	39.550000	8.690000
CO	**0.085000**$^+$	0.077700	**0.101950**$^+$	0.052190	**0.142200**$^+$	0.128000
CK10	9.519000	4.736000	13.18000$^+$	5.550000	12.14800$^+$	4.196000
CK20	6.225000$^+$	3.949000	8.590000$^+$	5.520000	11.34600$^+$	4.244000
CK30	7.748000	5.287000	7.904000$^+$	4.832000	10.91400$^+$	4.268000
CK40	5.677000$^+$	4.375000	8.019000$^+$	4.281000	10.38900$^+$	3.195000
CK50	5.791000$^+$	4.235000	6.780000$^+$	3.958000	8.201000$^+$	4.702000
CK60	4.890000$^+$	4.263000	7.552000$^+$	5.101000	8.030000$^+$	5.760000
CK70	5.255000$^+$	5.456000	7.025000$^+$	4.762000	7.173000$^+$	4.842000
CK80	3.684000$^+$	3.363000	5.832000$^+$	4.625000	6.866000$^+$	4.012000
CK90	3.486000$^+$	4.046000	3.990000$^+$	3.556000	6.195000$^+$	4.347000

Table 12 Summarization of mean and standard deviation of spacing (*SP*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ2 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	**0.025130**	0.002828	0.105250	0.016970	0.188760	0.018550
CO	0.027560	0.013850	**0.060860**$^+$	0.021290	**0.082100**$^+$	0.016820
CK10	0.056490$^-$	0.006580	0.107560	0.011310	0.159290$^+$	0.021840
CK20	0.057880$^-$	0.009250	0.106140	0.011250	0.154670$^+$	0.016860
CK30	0.053980$^-$	0.007510	0.105410	0.012680	0.141060$^+$	0.019560
CK40	0.058720$^-$	0.010500	0.102490	0.013380	0.149950$^+$	0.013820
CK50	0.055440$^-$	0.007620	0.103920	0.010090	0.151160$^+$	0.016610
CK60	0.056230$^-$	0.007960	0.101030	0.012330	0.147530$^+$	0.013810
CK70	0.054930$^-$	0.007000	0.100560	0.012550	0.142850$^+$	0.017940
CK80	0.055560$^-$	0.009810	0.100240	0.010830	0.144540$^+$	0.021450
CK90	0.053190$^-$	0.007890	0.096240$^+$	0.013010	0.136720$^+$	0.018290

Table 13 Summarization of mean and standard deviation of spacing (*SP*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ3 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	19.650000	8.920000	64.150000	10.360000	103.780000	9.920000
CO	**0.2239000**$^{+}$	0.325600	**0.3590000**$^{+}$	0.387400	**0.3163000**$^{+}$	0.214400
CK10	21.280000	7.800000	23.910000^{+}	7.340000	33.260000^{+}	8.470000
CK20	20.240000	10.850000	21.290000^{+}	8.960000	27.340000^{+}	11.530000
CK30	16.460000	10.330000	18.430000^{+}	9.130000	20.450000^{+}	8.580000
CK40	16.830000	10.420000	17.910000^{+}	9.260000	22.400000^{+}	13.090000
CK50	20.680000	9.640000	19.010000^{+}	11.840000	20.780000^{+}	9.710000
CK60	18.440000	9.180000	15.940000^{+}	10.480000	21.660000^{+}	11.800000
CK70	19.740000	10.910000	11.150000^{+}	8.300000	19.580000^{+}	12.840000
CK80	19.120000	10.130000	16.240000^{+}	11.350000	24.010000^{+}	11.930000
CK90	18.630000	9.130000	13.550000^{+}	10.160000	20.650000^{+}	18.050000

Table 14 Summarization of mean and standard deviation of spacing (*SP*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ4 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	**0.020280**	0.011180	0.077580	0.042130	0.170310	0.046350
CO	0.031080^{-}	0.018570	0.057850	0.015020	0.089900^{+}	0.018510
CK10	0.039890^{-}	0.022820	**0.035320**$^{+}$	0.025380	0.019710^{+}	0.018860
CK20	0.036810^{-}	0.026750	0.035830^{+}	0.025080	**0.016780**$^{+}$	0.014820
CK30	0.048220^{-}	0.022330	0.046900^{+}	0.021310	0.017550^{+}	0.015340
CK40	0.038040^{-}	0.026630	0.040350^{+}	0.020760	0.022970^{+}	0.021730
CK50	0.050300^{-}	0.024360	0.047040^{+}	0.024660	0.026470^{+}	0.022980
CK60	0.042800^{-}	0.025890	0.038070^{+}	0.026050	0.018260^{+}	0.015630
CK70	0.051110^{-}	0.024040	0.047170^{+}	0.025620	0.020430^{+}	0.018760
CK80	0.044430^{-}	0.024800	0.036720^{+}	0.024380	0.017460^{+}	0.016090
CK90	0.043890^{-}	0.024900	0.044530^{+}	0.024310	0.016910^{+}	0.016340

Table 15 Summarization of mean and standard deviation of spacing (*SP*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ5 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	0.005045	0.000556	0.087290	0.023240	0.153300	0.034530
CO	**0.004684**	0.000520	**0.053480**$^+$	0.012000	**0.091490**$^+$	0.026050
CK10	0.015992$^-$	0.005078	0.086360	0.012440	0.135960	0.032300
CK20	0.016060$^-$	0.008690	0.070680$^+$	0.012950	0.135090	0.026560
CK30	0.018090$^-$	0.010840	0.072470$^+$	0.016850	0.130680$^+$	0.016560
CK40	0.015450$^-$	0.007410	0.075970	0.024600	0.122080$^+$	0.021270
CK50	0.016158$^-$	0.004952	0.068730$^+$	0.016130	0.123490$^+$	0.016690
CK60	0.015590$^-$	0.006710	0.072170$^+$	0.019250	0.122920$^+$	0.024530
CK70	0.016270$^-$	0.011220	0.072260	0.023260	0.119010$^+$	0.024650
CK80	0.014040$^-$	0.005860	0.078800	0.028770	0.115720$^+$	0.017900
CK90	0.016820$^-$	0.007700	0.068610$^+$	0.014970	0.125250$^+$	0.028930

Table 16 Summarization of mean and standard deviation of spacing (*SP*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ6 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	0.047210	0.013340	0.329610	0.040750	0.533700	0.056900
CO	**0.008510**$^+$	0.017160	0.167100$^+$	0.106300	**0.292900**$^+$	0.116100
CK10	0.029610$^+$	0.008310	0.125610$^+$	0.030270	0.788100$^-$	0.126100
CK20	0.026600$^+$	0.007370	0.103040$^+$	0.017620	0.683500$^-$	0.212400
CK30	0.029700$^+$	0.016570	0.097870$^+$	0.018060	0.646300	0.241600
CK40	0.025280$^+$	0.004798	0.094650$^+$	0.016940	0.514200	0.125600
CK50	0.024665$^+$	0.004728	0.100610$^+$	0.021540	0.501600	0.198200
CK60	0.025970$^+$	0.010340	0.094400$^+$	0.016750	0.498100	0.212400
CK70	0.025350$^+$	0.006090	**0.090630**$^+$	0.018780	0.454800	0.200300
CK80	0.024084$^+$	0.004731	0.097290$^+$	0.018670	0.470100	0.188200
CK90	0.024180$^+$	0.005930	0.094290$^+$	0.014670	0.396500$^+$	0.144000

Table 17 Summarization of mean and standard deviation of spacing (*SP*) between SPEA2, SPEA2-CC and SPEA2-CE-KR for DTLZ7 test problem with 3 to 5 objectives

Algorithm	3 Objectives		4 Objectives		5 Objectives	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
S2	**0.046800**	0.022270	0.164590	0.029300	0.294940	0.026320
CO	0.066000	0.180000	0.148700	0.410200	0.994000^-	1.000000
CK10	0.060870^-	0.018470	0.101470^+	0.054240	0.038600^+	0.065400
CK20	0.068610^-	0.013110	0.109700^+	0.044780	0.032790^+	0.047390
CK30	0.068020^-	0.013500	0.099190^+	0.043070	0.035700^+	0.054800
CK40	0.066900^-	0.011590	0.108500^+	0.042540	0.054100^+	0.059500
CK50	0.067150^-	0.012790	$\mathbf{0.095710^+}$	0.048720	0.033130^+	0.043700
CK60	0.063200^-	0.013920	0.100220^+	0.045890	0.038870^+	0.050960
CK70	0.065690^-	0.014890	0.112270^+	0.053140	0.038200^+	0.041560
CK80	0.070240^-	0.011120	0.125730^+	0.033250	0.036680^+	0.048520
CK90	0.064500^-	0.017210	0.118580^+	0.044200	$\mathbf{0.032510^+}$	0.042730

Table 18 Summarization of mean and standard deviation of coverage (*C*) between SPEA2 and SPEA2-CE-KR with 90 opponents for DTLZ test problems with 3 to 5 objectives. $C(X', X'') > C(X'', X')$, then X' is better than X''

DTLZ	*C* Metric	3 Objectives		4 Objectives		5 Objectives	
		Mean	St Dev	Mean	St Dev	Mean	St Dev
1	*C*(CK,S2)	0.072100	0.060500	0.355200	0.098800	0.867600	0.105400
	C(S2,CK)	0.069700	0.174600	0.029000	0.158800	0.001670	0.009130
2	*C*(CK,S2)	0.018670	0.015480	0.029670	0.021730	0.113330	0.050330
	C(S2,CK)	0.021330	0.022700	0.002670	0.006910	0.000000	0.000000
3	*C*(CK,S2)	0.335000	0.227800	0.441300	0.229600	0.899300	0.107300
	C(S2,CK)	0.179300	0.189700	0.000000	0.000000	0.000000	0.000000
4	*C*(CK,S2)	0.038670	0.036360	0.132300	0.075900	0.354700	0.096400
	C(S2,CK)	0.050670	0.052650	0.013000	0.036020	0.000000	0.000000
5	*C*(CK,S2)	0.032330	0.022080	0.052670	0.025720	0.034670	0.029560
	C(S2,CK)	0.100000	0.041850	0.085300	0.100600	0.151000	0.105900
6	*C*(CK,S2)	0.368700	0.056700	0.474000	0.084100	0.568300	0.133500
	C(S2,CK)	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
7	*C*(CK,S2)	0.063330	0.033150	0.094700	0.114400	0.506000	0.277000
	C(S2,CK)	0.046700	0.078800	0.000667	0.002537	0.000000	0.000000

Table 19 Summarization of mean and standard deviation of coverage (C) between SPEA2 and SPEA2-CC for DTLZ test problems with 3 to 5 objectives. $C(X', X'') > C(X'', X')$, then X' is better than X''

DTLZ	C Metric	3 Objectives		4 Objectives		5 Objectives	
		Mean	St Dev	Mean	St Dev	Mean	St Dev
1	C(CO,S2)	0.299000	0.122900	0.758800	0.103100	0.996200	0.010100
	C(S2,CO)	0.005330	0.029210	0.002000	0.010950	0.000000	0.000000
2	C(CO,S2)	0.184330	0.053800	0.237000	0.072000	0.565300	0.106300
	C(S2,CO)	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3	C(CO,S2)	0.190300	0.067300	0.729700	0.101900	0.983330	0.030890
	C(S2,CO)	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4	C(CO,S2)	0.136300	0.085500	0.265700	0.182200	0.518000	0.230900
	C(S2,CO)	0.005000	0.009740	0.024670	0.025960	0.011000	0.028330
5	C(CO,S2)	0.216000	0.052630	0.190000	0.065900	0.357300	0.135100
	C(S2,CO)	0.000000	0.000000	0.047670	0.022690	0.004670	0.007300
6	C(CO,S2)	0.533300	0.082400	0.786300	0.104200	0.988000	0.009970
	C(S2,CO)	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
7	C(CO,S2)	0.146300	0.060500	0.159330	0.037870	0.128700	0.069500
	C(S2,CO)	0.000667	0.002537	0.001667	0.005307	0.005000	0.011370

Table 20 Summarization of mean and standard deviation of coverage (C) between SPEA2-CC and SPEA2-CE-KR with 90 opponents for DTLZ test problems with 3 to 5 objectives. $C(X', X'') > C(X'', X')$, then X' is better than X''

DTLZ	C Metric	3 Objectives		4 Objectives		5 Objectives	
		Mean	St Dev	Mean	St Dev	Mean	St Dev
1	C(CO,CK)	0.999000	0.004030	0.957700	0.167300	0.761000	0.398800
	C(CK,CO)	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	C(CO,CK)	0.110330	0.054300	0.074000	0.045070	0.058330	0.036210
	C(CK,CO)	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3	C(CO,CK)	0.574300	0.304400	0.921000	0.133600	0.598700	0.447700
	C(CK,CO)	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4	C(CO,CK)	0.108300	0.086300	0.011000	0.031220	0.001000	0.004030
	C(CK,CO)	0.000000	0.000000	0.001670	0.005310	0.025000	0.025560
5	C(CO,CK)	0.228300	0.060200	0.153300	0.117300	0.316300	0.095000
	C(CK,CO)	0.000000	0.000000	0.061670	0.043870	0.006330	0.010330
6	C(CO,CK)	1.000000	0.000000	0.914700	0.097800	0.985000	0.021770
	C(CK,CO)	0.000000	0.000000	0.001330	0.004340	0.000330	0.001830
7	C(CO,CK)	0.060000	0.083500	0.027330	0.038680	0.001330	0.007300
	C(CK,CO)	0.000000	0.000000	0.000000	0.000000	0.009330	0.013370

References

1. Coello Coello, C.A.: Evolutionary multi-objective optimization: a critical review. In: Sarker, R., Mohammadian, M., Yao, X. (eds.) Evolutionary optimization, pp. 117–146. Kluwer Academic, Massachusetts (2002)
2. Coello Coello, C.A.: Recent trends in evolutionary multiobjective optimization. In: Abraham, A., Jain, L., Goldberg, R. (eds.) Evolutionary multiobjective optimization: theoretical advances and applications, pp. 7–32. Springer, Berlin (2005)
3. Coello Coello, C.A., Reyes Sierra, M.: A coevolutionary multi-objective evolutionary algorithm. In: Proceedings of the congress on evolutionary computation, pp. 482–489 (2003)
4. Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley, New York (2001)
5. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. KanGAL report 2001001. Kanpur Genetic Algorithms Laboratory (KanGAL), Department of Mechanical Engineering, Indian Institute of Technology Kanpur, Kanpur (2001)
6. Durillo, J.J., Nebro, A.J., Coello Coello, C.A., Luna, F., Alba, E.: A comparative study of the effect of parameter scalability in multi-objective evolutionary algorithms. In: Proceedings of the IEEE congress on evolutionary computation, pp. 1893–1900 (2008)
7. Fonseca, C.M., Fleming, P.J.: An overview of evolutionary algorithms in multiobjective optimization. Evolutionary Computation 3(1), 1–16 (1995)
8. Groşan, C.: Multiobjective adaptive representation evolutionary algorithm (marea) – a new evolutionary algorithm for multiobjective optimization. In: Proceedings of the world on-line conference on soft computing in industrial application, applied soft computing technologies: the challenge of complexity advances in soft computing, pp. 113–121. Springer, Berlin (2006)
9. Iorio, A.W., Li, X.-D.: A cooperative coevolutionary multiobjective algorithm using non-dominated sorting. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 537–548. Springer, Heidelberg (2004)
10. Keerativuttitumrong, N., Chaiyaratana, N., Varavithya, V.: Multi-objective co-operative co-evolutionary genetic algorithm. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 288–297. Springer, Heidelberg (2002)
11. Knowles, J., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK Report 214. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Zurich (2006)
12. Lohn, J.D., Kraus, W.F., Haith, G.L.: Comparing a coevolutionary genetic algorithm for multiobjective optimization. In: Proceedings of the IEEE congress on evolutionary computation, pp. 1157–1162 (2002)
13. Maneeratana, K., Boonlong, K., Chaiyaratana, N.: Multi-objective optimisation by co-operative co-evolution. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 772–781. Springer, Heidelberg (2004)
14. Maneeratana, K., Boonlong, K., Chaiyaratana, N.: Compressed-objective genetic algorithm. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 473–482. Springer, Heidelberg (2006)

15. McFadden, C.H., Keeton, W.T.: Biology: an exploration of life. W. W. Norton & Company, New York (1995)
16. Panait, L., Luke, S.: A comparative study of two competitive fitness functions. In: Langdon, W.B., Cantú-Paz, E., Mathias, K.E., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M.A., Schultz, A.C., Miller, J.F., Burke, E.K., Jonoska, N. (eds.) Proceedings of the genetic and evolutionary computation conference, pp. 503–511. Morgan Kaufmann, California (2002)
17. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994)
18. Potter, M.A., De Jong, K.A.: Cooperative coevolution: an architecture for evolving coadapted subcomponents. Evolutionary Computation 8(1), 1–29 (2000)
19. Santana-Quintero, L.V., Coello Coello, C.A.: An algorithm based on differential evolution for multi-objective problems. International Journal of Computational Intelligence Research 1(2), 151–169 (2005)
20. Schott, J.R.: Fault tolerant design using single and multicriteria genetic algorithm optimization. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Massachusetts (1995)
21. Tan, K.C., Lee, T.H., Yang, Y.J., Liu, D.S.: A cooperative coevolutionary algorithm for multiobjective optimization. In: Proceedings of the IEEE international conference on systems, man and cybernetics, pp. 1926–1931 (2004)
22. Tizhoosh, H.R.: Opposition-based learning: a new scheme for machine intelligence. In: Proceedings of the international conference on computational intelligence for modelling control and automation, vol. 1, pp. 695–701 (2005)
23. Tomassini, M.: Evolutionary algorithms. In: Sanchez, E., Tomassini, M. (eds.) Proceedings of the international workshop on towards evolvable hardware, the evolutionary engineering approach, pp. 19–47. Springer, Berlin (1996)
24. Van Veldhuizen, D.A., Lamont, G.B.: On measuring multiobjective evolutionary algorithm performance. In: Proceedings of the congress on evolutionary computation, vol. 1, pp. 204–211. IEEE Service Center, Piscataway (2000)
25. Walker, M.: Literature review (2003), `http://www.massey.ac.nz/~mgwalker/work/lit-review.pdf` (accessed on December 12, 2007)
26. Wiegand, R.P., De Jong, K.A.: Understanding coevolution-theory and analysis of coevolutionary algorithms: preface. In: Barry, A.M. (ed.) Proceedings of the genetic and evolutionary computation conference workshop on coevolution: understanding coevolution. AAAI, New York (2002)
27. Wiegand, R.P., Liles, W.C., De Jong, K.A.: An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In: Spector, L., Goodman, E.D., Wu, A., Langdon, W., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M.H., Burke, E. (eds.) Proceedings of the genetic and evolutionary computation conference, pp. 1235–1242 (2001)
28. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. Evolutionary Computation 8(2), 173–195 (2000)
29. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength pareto evolutionary algorithm. Technical report 103. Computer Engineering and Network Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Zurich (2001)

Evolutionary Optimization for Multiobjective Portfolio Selection under Markowitz's Model with Application to the Caracas Stock Exchange

Feijoo Colomine Duran, Carlos Cotta, and Antonio J. Fernández

Abstract. Several problems in the area of financial optimization can be naturally dealt with optimization techniques under multiobjective approaches, followed by a decision-making procedure on the resulting efficient solutions. The problem of portfolio optimization is one of them. This chapter studies the use of evolutionary multiobjective techniques to solve such problems, focusing on Venezuelan market mutual funds between years 1994 and 2002. We perform a comparison of different evolutionary multiobjective approaches, namely NSGA-II, SPEA2, and IBEA, and show how these algorithms provide different optimization profiles. The subsequent step of solution selection is done using Sharpe's index as a measure of risk premium. We firstly show that NSGA-II provides similar results to SPEA2 on mixed and fixed funds, and better (according to Sharpe's index) solutions than SPEA2 on variable funds, indicating that NSGA-II provides a better coverage of the region containing interesting solutions for Sharpe's index. Furthermore, IBEA outperforms both NSGA-II and SPEA2 in terms of index value attained. Finally, we also show that this procedure results in a more profitable solution than an indexed portfolio by the Caracas Stock Exchange.

1 Introduction

Finance is a branch of Economics that studies the flow of money and other assets, their acquisition and management by a company, individual or state, and the markets in which they are traded. In other words, it comprises studies concerning the collection and management of money and other valuables such as securities, bonds, etc.

Feijoo Colomine Duran
Universidad Nacional Experimental del Táchira (UNET) Laboratorio de Computación de Alto Rendimiento (LCAR), San Cristóbal, Venezuela
e-mail: fcolomin@unet.edu.ve

Carlos Cotta · Antonio J. Fernández
Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, University of Málaga, Campus de Teatinos, 29071 - Málaga, Spain
e-mail: {ccottap,afdez}@lcc.uma.es

One of the main challenges for the administration of financial resources is to maintain the profitability and liquidity at times in which the simple act of leaving the money deposited in a bank makes it lose value.

This work focuses on the study of the components that promote an acceptable (above inflation) economic return, as well as the need to obtain better risk diversification as determined by the degree of abhorrence of each investor [15]. Plainly speaking, diversifying amounts using a mechanism *by which all eggs are not put in one basket*, that is, investing in a range of financial sectors whose economic activities result in some benefit and whose economic cycles behave differently from each other. In this context, the risk diversification is achieved by creating a portfolio of investments in several of these financial instruments or sectors.

The area of financial management encompasses a number of theoretical elements and field studies regarding the risk/performance relationship. There is no static optimal solution, and the best portfolio always depends on market evolution. In very general terms, this implies that simultaneous risk minimization and performance maximization are the obvious desired goals. Needless to say, these goals are partially opposed to each other. Several proposals can be found in the literature in this regard. For example, Markowitz's model [18] has become an essential theoretical reference for portfolio selection. However, its practical application has not been as broad as it could, mostly due to the complexity of the method: on one hand, being a quadratic parameterized model its resolution is not trivial; on the other hand, the number of variables involved is high.

The topic addressed by Markowitz relates to the selection of investments, namely the problem of allocating resources among the various options available for that purpose. Prior to the popularization of Markowitz's approach, investment selection involved a costly process of collecting and processing a wide range of information about the companies issuing the assets (primarily shares). This information included, among other things, balance sheets and financial statements, status of the company within the industry and within the market as a whole, the quality of company management, dividend policy, and so on. Markowitz's approach significantly simplified the selection problem by considering asset performance as a stochastic process, focusing solely on the historical log of returns of the issuing companies, and more precisely on three statistical measures of these data: mean, variance and covariance of return rates.

Markowitz developed the model based on the rational behavior of the investor. In other words, the investor wants to maximize her profit and rejects the risk. Therefore, a portfolio will be efficient for her if it provides the highest possible return for a given risk, or equivalently, if it presents the least possible risk for a given level of profitability. The collection of portfolios offering such a combination of risk/profitability is termed the *efficient frontier*, and once known the investor can select her optimal portfolio according to her preferences.

If no additional considerations are made and a specific risk/profitability profile is known, the optimization problem can be solved using quadratic programming. However, this is not usually the case. On one hand, several constraints such as cardinality constraints (i.e., a limit on the number of different investments in the portfolio) or minimum transaction lots can be considered, thus making quadratic programming or other exact techniques infeasible. On the other hand, if no profitability target is fixed a priori (or if a more general investment strategy is sought) the task of finding (or approximating as much as possible) the whole efficient frontier in an efficient way requires the use of powerful optimization techniques. In this scenario the use of metaheuristic techniques is the general norm [3]. These techniques cannot provide optimality proofs for the solutions they obtain, but if adequately crafted, they will likely provide optimal or near-optimal solutions to a wide range of continuous and combinatorial optimization problems.

We consider the particular case of nature-inspired metaheuristics, or more precisely, evolutionary algorithms. Not only do these techniques hold an impressive successive-record on different hard optimization tasks; they have also been shown to be extremely effective in solving multiobjective optimization problems. As such, they are quite appropriate to deal with the combined risk/performance optimization. We consider several state-of-the-art second generation approaches for evolutionary multiobjective optimization, and compare them on the basis of sound performance metrics defined in the literature. We also address the subsequent selection step: once the efficient frontier has been identified, there remains the problem of selecting one particular solution according to the risk profile determined by the investor. This latter approach is considered here, and as it will be shown, using Sharpe's index as a guiding measure we are able to identify solutions better than those currently used in indexed portfolios by the Caracas Stock Exchange, a Latin American exchange operating in Venezuela.

2 Background

The work presented in this chapter deals with real investments that are conditioned by two main parameters: (i) profitability, i.e., the returns on the investment, and (ii) risk, i.e., the chances of low (or even negative) returns. Obviously, profitability is a positive element for the investor whereas risk is a negative one. This means that an investor wishes to maximize profitability and minimize risk [1]. This will be formalized within Markowitz's model in Section 2.2. Before that, a brief overview of mutual funds will be provided first in Section 2.1.

2.1 Mutual Funds

Mutual funds are instruments that combine the money invested by a group of persons. They are handled by a management office specializing in the administration of investment portfolios, which takes decisions on the purchase of shares, bonds, and other instruments of the market. By combining the money of several investors,

[1] Other parameters such as liquidity or political control of a company might be considered as well.

mutual funds allow them to participate in larger portfolios than those they could buy individually. There are several types of mutual funds:

1. **Fixed revenue:** The aim of these funds is to invest in state bonds, private bonds, and other instruments that offer a predictable performance if they are maintained up to their expiration. Fixed funds are a way of adjusting to different investment horizons, e.g., they tend to specialize in investments on assets that operate in either a short term, a middle term, or a long term basis. Purchasing assets that participate in fixed funds has diverse advantages such as risk diversification, professional administration, (management), and accessibility (i.e., small investors have access to high investments).
2. **Variable revenue:** These funds try to maximize the profit by investing in shares of companies that quote in the Stock exchange. Because of the very nature of the assets in which they are invested in, these funds have the highest risk associated with them. From a conceptual point of view, the investments in shares, if correctly chosen, is the most profitable in the longer term. However, this also implies a higher volatility of the investments (i.e., increased risk).
3. **Mixed revenue:** These funds represent a combination of fixed revenue and variable revenue. They aim to diversify the investment in stocks of both fixed and variable revenue. Their composition is thus a combination of the different types of assets, and their risk / profitability ratio is intermediate between that of fixed funds and variable funds. The risk obviously depends on the proportion of the investment made on the different mutual funds.

Whichever type of fund considered, the investor and/or manager is faced with an optimization problem regarding the composition of the portfolio. One of the most widely used and conspicuous method of addressing this problem is Markowitz's model, described in next section.

2.2 Markowitz's Model and Sharpe's Index

Markowitz's model [18] is a pioneering model in the selection of assets to construct an ideal portfolio. It assumes that the future performance a specific investment can offer can be determined from both experience and investigation. This model is thus applied with the aim of obtaining an optimal portfolio selection. The basic idea is that by analyzing the expected profitabilities of the individual financial assets one can make a correct portfolio selection. Two main components have to be taken into account: profitability and the risk to be assumed by the investor. The investor needs to know the risk level, that is to say, the degree of profitability variation which is measured by the variance defined as:

$$\sigma_i^2(\mathbf{R}) = \sum_{t=1}^{T} \frac{[R_{it} - E(R_i)]^2}{n} \tag{1}$$

where $\mathbf{R} = \{R_{it}\}$, $1 \leqslant i \leqslant n$, $1 \leqslant t \leqslant T$, is a matrix containing the profitability of each asset at each time interval t, $E(R_i)$ is the mean profitability of the i-th asset, and T is the number of intervals in the time horizon. The overall risk of the portfolio is then defined as a weighted quadratic combination of the covariances of the assets included in it, i.e.,

$$\sigma^2(\mathbf{R}|\mathbf{W}) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \sigma_{ij}(\mathbf{R}) \tag{2}$$

where $\mathbf{W} = \{w_i\}$, $1 \leqslant i \leqslant n$, is a vector comprising the fraction of the budget allocated to each asset ($w_i \geqslant 0$), and $\sigma_{ij}(\mathbf{R})$ is the covariance of the performance of the i-th asset and the j-th asset, defined as:

$$\sigma_{ij}(\mathbf{R}) = \sum_{t=1}^{T} \frac{[R_{it} - E(R_i)]\,[R_{jt} - E(R_j)]}{T} \tag{3}$$

Similar to the risk, the profitability $E(\mathbf{R}|\mathbf{W})$ of a portfolio is defined as the weighted average of the assets involved, i.e.,

$$E(\mathbf{R}|\mathbf{W}) = \sum_{i=1}^{n} w_i E(R_i) \tag{4}$$

Generally speaking, the investor looks for the curve of utility with $E(\mathbf{R}|\mathbf{W}) = \infty$ and $\sigma^2(\mathbf{R}|\mathbf{W}) = 0$, but this not a realistic option as this curve is limited by the existing assets that never have this nature. We note that for the assets without risk (i.e., those with null profit-variance), the utility is equal to the expected profitability because there is no penalization due to the risk.

To evaluate the quality of a portfolio we have to define a measure that accounts for both the profitability and the risk of the assets involved. Such a measure can also allow the comparison between different portfolios. To this end, we have considered Sharpe's index [25], that determines the performance according to the ratio of excess profitability and risk. More precisely,

$$S(\mathbf{R}|\mathbf{W}) = \frac{E(\mathbf{R}|\mathbf{W}) - R_0}{\sigma(\mathbf{R}|\mathbf{W})} \tag{5}$$

where R_0 is the performance of a portfolio without risk. $E(\mathbf{R}|\mathbf{W}) - R_0$ is therefore the excess performance (that is, the extra profit obtained by taking some risks), which is divided by the risk of the portfolio (measured as the standard deviation of returns). Basically, the index indicates how much performance is expected with respect to the risk. The higher the value returned is, the higher the success of the fund management is.

2.3 Related Work

An early reference on portfolio optimization with MOEAs is the work of Veradajan *et al.* [29]. They describe the use of NSGA (non-dominated sorting genetic algorithm) [26] to optimize investment portfolios, as an alternative to quadratic programming techniques. In addition to the typical objectives of increasing performance and decreasing risk, a third objective involving the costs of the transactions is also considered. Several variants of the problem involving the presence of additional constraints can be also found in the literature. For example, Chang *et al.* [5] consider limits on the number of assets and their proportion within the portfolio, and use different metaheuristics (tabu search, genetic algorithms, and simulated annealing) to solve the problem. Busetti [4] also consider tabu search and genetic algorithms, in this case for solving the problem with cardinality constraints and transaction costs. Streichert *et al.* [27] deal with a cardinality constrained portfolio selection problem too, using NSGA and evolution strategies. They actually compare different representations of solutions (pure binary, gray binary, and real-valued). Fieldsend *et al.* [11] also deal with this variation of the problem, and more specifically with the case in which the analyst does not know a priori how many instruments should be included in the portfolio, or the degree of risk-performance that can be accepted. They also propose the addition of the cardinality constraints as a third objective to be minimized. Lin *et al.* [17] consider a variant of the problem with fixed transaction costs and minimum transaction lots. They demonstrated that in this case, the selection of the portfolios becomes more complicated because the problem model has to manage mixed integer variables and nonlinear objectives.

From a more general point of view, Mukerjee *et al.* [20] utilize NSGA-II to implement a decision-making multicriteria model used in the risk/performance negotiation by a bank loan manager. Two models with respect to this negotiation were considered. Also in a bank context, Schlottmann and Seese [23] present a survey of different financial applications that can be handled via MOEAs, and encourage the use of specific problem knowledge and hybridization techniques to obtain better algorithms. A more recent perspective on multiobjective evolutionary optimization of portfolios can be found in [21].

3 Material and Methods

Once the problem scenario has been presented, this section is devoted to providing a more precise formulation of the optimization task. Subsequently, the data used in the experiments (corresponding to real market data from a Latin American exchange), as well as the algorithms considered will be described.

3.1 Problem Setting

As stated before, Markowitz's model is based on the assumption that the investor abhors risk, which can be represented as the variability of returns for a certain investment. At the same time, she wants to maximize her profits. Hence, we can consider a portfolio as efficient if it achieves the profit sought by the investor at the minimum risk. The set of efficient portfolios can be calculated by solving the following parametric nonlinear equation:

$$\min \sigma^2(\mathbf{R}|\mathbf{W}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \sigma_{ij}(\mathbf{R}) \tag{6}$$

subject to:

$$E(\mathbf{R}|\mathbf{W}) = \sum_{i=1}^{n} w_i E(R_i) = V^* \tag{7}$$

$$\sum_{i=1}^{n} w_i = 1 \tag{8}$$

Note that by varying the parameter V^* the optimal solution in each case minimizes the risk of the portfolio for a given target profit. This consideration leads naturally to a multiobjective scenario in which the whole efficient frontier is sought rather than just solving the above equations for different target profits. This efficient frontier comprises Pareto-optimal portfolios, i.e., portfolios whose profitability cannot be increased without increasing the risk as well (and vice versa, the risk cannot be reduced without decreasing the expected return). This bi-objective problem is thus formulated as

$$\min \sigma^2(\mathbf{R}|\mathbf{W}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \sigma_{ij}(\mathbf{R}) \tag{9}$$

$$\max E(\mathbf{R}|\mathbf{W}) = \sum_{i=1}^{n} w_i E(R_i) \tag{10}$$

subject to:

$$\sum_{i=1}^{n} w_i = 1 \tag{11}$$

This basic model corresponds to unconstrained portfolios, in which the investor can allocate any number of investments she desires, and these can be as large or small as wanted. Additional constraints can at any rate be posed on the composition of the portfolio, e.g., cardinality constraints (at most K assets can be included in the portfolio), or size constraints (the fraction of the portfolio allocated to an asset is bounded [2]). We are interested in analyzing carefully the performance of different multiobjective optimizers on the problem, in particular with respect to finding highly desirable solutions according to Sharpe's index. For this reason, we will focus initially on the case of unconstrained portfolios since they provide a more unbiased arena for performance evaluation, and will pave the way for subsequent experimentation with other variants of the problem.

[2] Michaud [19] considers that the use of historical data to estimate risk and expected returns introduces an important bias: efficient portfolios can be composed of few, largely uncorrelated assets. Such a portfolio can be unattractive for some investors. However, this problem can be solved by considering constraints on the maximum percentage of the portfolio that a certain asset can represent.

3.2 Data: Venezuelan Mutual Funds

The data used in the experiments is taken from the Caracas Stock Exchange (*Bolsa de Valores de Caracas* - BVC), the only securities exchange operating in Venezuela. More precisely, we have considered data corresponding to the last five years. This time interval is large enough to be representative of the evolution of shares, and not too large to include irrelevant –for prediction purposes- data (the status of funds can fluctuate in the long term, commonly making old data useless for forecasting the future evolution of shares). According to this, our sample – $\sim 35,000$ daily prices of different mutual funds: fixed, variable, and mixed– comprises those funds no older than five years and still available in the BVC [1]. To be precise, we have used weekly market data from year 1994 to year 2002, corresponding to 26 Venezuelan mutual funds: 12 fixed funds, 7 variable funds, and 7 mixed funds. Data up to year 2001 is used for training purposes, whereas data corresponding to the year 2002 will be used for testing the obtained portfolios with respect to an investment portfolio indexed in the BVC. The relative ratio of share values in successive weeks is calculated to compute the profitability of each fund. This is done for each week in the year, and subsequently averaged to yield the annual weekly mean and thus obtain the annual profit percentage. The covariance matrix of these profitability values is also computed, as a part of Markowitz's model.

3.3 Evolutionary Multiobjective Approaches

The multiobjective portfolio optimization problem posed in this section will be solved via multiobjective evolutionary algorithms (MOEAs). Indeed, multiobjective evolutionary optimization nowadays provides powerful tools for dealing with this kind of problems. A detailed survey of this field is beyond the scope of this work. We refer the reader to [6, 7, 8, 9, 12, 30, 36] among other works for more comprehensive information about this topic. Let us anyway note for the sake of completeness that MOEA approaches can be classically categorized under three major types [36]: (i) aggregation/scalarization, (ii) criterion-based, and (iii) Pareto-dominance based. A fourth class has been defined more recently, namely indicator-based, and will be discussed later. Firstly, let us describe the basis of the three classical approaches.

Aggregation approaches are based on constructing a single scalar value using some function that takes the multiple objective values as input. This is typically done using a linear combination, and the method exhibits several drawbacks, e.g., the difficulty in determining the relative weight of each objective, and the inadequate coverage of the set of efficient solutions, among others. As to the criterion-based approaches, they try to switch priorities between the objectives during different stages of the search (Schaffer's VEGA approach [22] pioneered this line of attack, using each objective to select a fraction of solutions for breeding). This does not constitute a full solution to the problem of approximating the whole efficient front though. Such a solution can be nevertheless obtained via Pareto-based approaches. These are based on the notion of Pareto-dominance. Let f_i, $1 \leqslant i \leqslant n$, represent each of the n objective functions, and let $f_i(x) \prec f_i(y)$ denote that x is better than y according to the i-th objective value. Then, abusing of the notation we use $x \prec y$ to denote that x dominates y when

$$x \prec y \Leftrightarrow [(\exists i : f_i(x) \prec f_j(y)) \wedge (\nexists i : f_i(y) \prec f_j(x))] \tag{12}$$

The Pareto front (i.e., the efficient front) is therefore the set of non-dominated solutions, i.e., $P = \{x \mid \nexists z : z \prec x\}$. Pareto-based MOEAs use the notion of Pareto-dominance for determining the solutions that will breed and/or the solutions that will be replaced.

In this work we consider three state-of-the-art MOEAs, namely NSGA-II (Non-dominated Sorting Genetic Algorithm II) [10], SPEA2 (Strength Pareto Evolutionary Algorithm 2) [34] and IBEA (Indicator-Based Evolutionary Algorithm) [31]. The first two fall within the Pareto-based class, and are the second-generation version of two previous algorithms –NSGA [26], and SPEA [33] respectively. As such, they rely on the use of elitism (an external archive of non-dominated solutions in the case of SPEA2, and a plus-replacement strategy –keeping the best solutions from the union of parents and offspring– in the case of NSGA-II). More precisely, the central theme in these algorithms is assigning fitness to individuals according to some kind of non-dominated sorting, and preserving diversity among solutions in the non-dominated front. NSGA-II does this by sorting the population in non-domination levels. First of all, the set of non-dominated solutions is extracted from the current population $\mathscr{P}$; let this set be termed $\mathscr{F}_1$, and let $\mathscr{P}_1 = \mathscr{P} \setminus \mathscr{F}_1$. Subsequently, while there exist solutions in $\mathscr{P}_i$, $i \geqslant 1$, a new front $\mathscr{F}_{i+1}$ is extracted, and the procedure repeated. This way, each solution is assigned a rank, depending on the front it belongs to (the lower, the better). Such a rank is used for selection. To be precise, a binary tournament is conducted according to the domination level, and a crowding distance is utilized to break domination ties (thus spreading the front).

As to SPEA2, it uses an external archive of solutions that is used to calculate the "strength" of each individual i (the number of solutions dominated by or equal to i, divided by the population size plus one). Selection tries to minimize –via binary tournaments– the combined strength of all individuals not dominated by competing parents. This fitness calculation is coarse-grained, and may not always be capable of providing adequate guidance information. For this reason, a fine-grained fitness assignment is used, (i) taking into account both the external archive and the current population, and (ii) incorporating a nearest-neighbor density estimation technique (to spread the front). As a final addition with respect to SPEA, a sophisticated archive update strategy is used to preserve boundary conditions (see [34]).

The third algorithm considered is IBEA, which as its name indicates falls within the indicator-based class. Algorithms in this class approach multiobjective optimization as a procedure aimed at maximizing (or minimizing) some performance indicator. Many such indicators are based on the notion of Pareto-dominance and hence, this class of algorithms is in many respects related to these Pareto-based approaches. Nevertheless, it is necessary to note that they deserve separate treatment due to the philosophy behind them. Actually, in some sense, indicator-based algorithms can be regarded as a collective approach, where selective pressure is exerted to maximize the performance of the whole population. Consider, for example, an IBEA approach based on the hypervolume indicator. This indicator provides information on the hypervolume of the fitness space that is dominated by a certain set of solutions. This definition includes singletons (sets of a single solution), and therefore can be used to compare two individuals. This way, it can be used for selection purposes. However when it comes to replacement, a global perspective is used: the solution whose substitution results in the best value of the indicator for the whole population is taken out. In this work, we have considered an IBEA based on the ε-indicator [35].

In all the algorithms considered, solutions, i.e., a vector of rational values in the $[0,1]$ range indicating the fraction of the portfolio devoted to each fund, are represented as binary strings. Each fund is assigned 10 bits, yielding a raw weight $\bar{w}_i$. These weights are subsequently normalized as $w_i = \bar{w}_i / \sum_j \bar{w}_j$ to obtain the actual composition of the portfolio. Evaluation is done by computing the risk and return of the portfolio using the formulation depicted before. As to reproduction, we consider standard operators such as two-point crossover and bit-flip mutation.

4 Results

The experiments were conducted with the three algorithms described earlier, namely NSGA-II, SPEA2 and IBEA. We have utilized the PISA library (A Platform and Programming Language Independent Interface for Search Algorithms) [2], which provides an implementation of these two algorithms. The crossover rate is $P_x = 0.8$, the mutation rate is $P_m = 1/\ell$, and the population size is 2ℓ, where ℓ is the total number of bits in a solution. The algorithms run for a maximum number of 100 generations. The number of runs per data set is 30.

4.1 Front Analysis

The first part of the experimentation deals with the analysis of the Pareto fronts obtained. The results obtained are graphically depicted in Figs. 1–3. As can be seen, the grand fronts generated by either algorithm seem to be very similar, although the grand front found by IBEA appears to be slightly more spread for fixed funds. To analyze the extent of the significant difference in the performance more carefully we have considered two well-known performance indicators: the hypervolume indicator [32] and the R_2 indicator [13]. As mentioned before, the first one provides an indication of the region in the fitness space that is dominated by the front (and hence the larger, the better). As to the second indicator, it estimates the extent to which a certain front approximates another one (the true Pareto-optimal front if known, or a reference front otherwise). We have considered the unary version of this indicator, taking the combined NSGA-II/SPEA2/IBEA Pareto front as a reference set. Being a measure of distance to the reference set, the lower a R_2 value, the better.

Figs. 4 and 5 show the distribution of these two indicators for the experiments realized. Let us first consider the hypervolume distribution. SPEA2 appears to provide slightly worse values of this indicator with respect to NSGA-II. Actually, NSGA-II

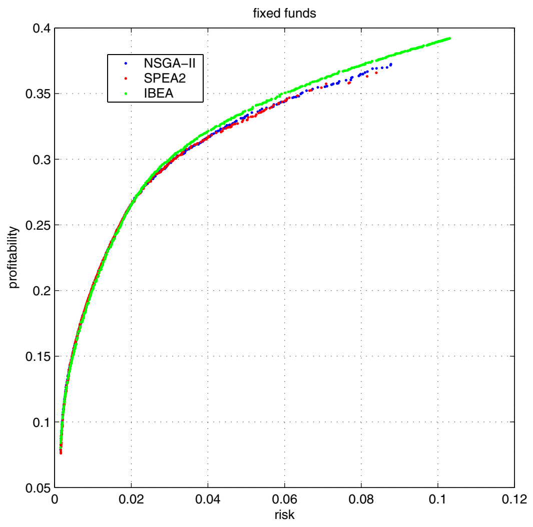


Fig. 1 Comparison of the Pareto fronts found by NSGA-II, SPEA2 and IBEA on fixed funds

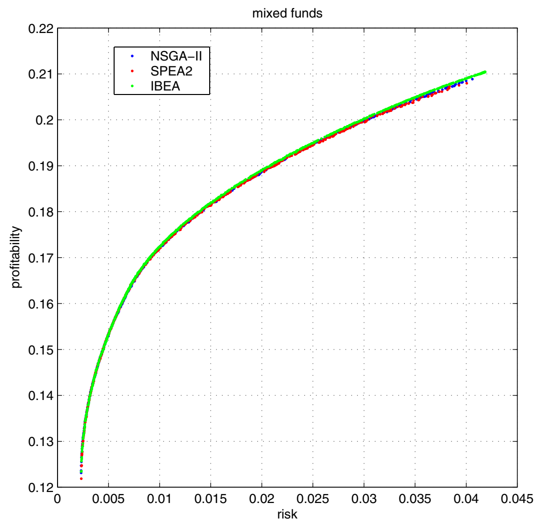


Fig. 2 Comparison of the Pareto fronts found by NSGA-II, SPEA2 and IBEA on mixed funds

is better (with statistical significance at the standard 0.05 level, according to a Wilcoxon ranksum test [16]) on fixed and mixed funds, and provides a negligible difference on variable funds. On the other hand, IBEA exhibits an interesting behavioral pattern with notably better results than both NSGA-II and SPEA2 on fixed funds, no difference on mixed funds, and clearly worse results on variable funds (in all cases, with statistical significance as before). A similar pattern is observed when the R_2 indicator is considered. NSGA-II compares favorably to SPEA2 in all the three types of funds, and IBEA varies from providing the best results on fixed

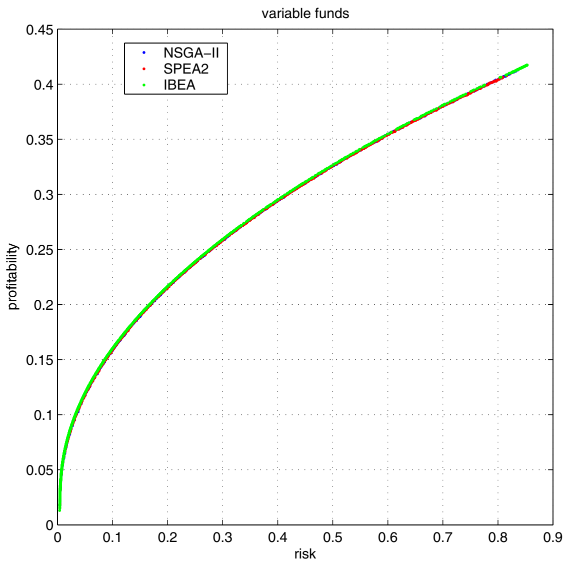


Fig. 3 Comparison of the Pareto fronts found by NSGA-II, SPEA2 and IBEA on variable funds

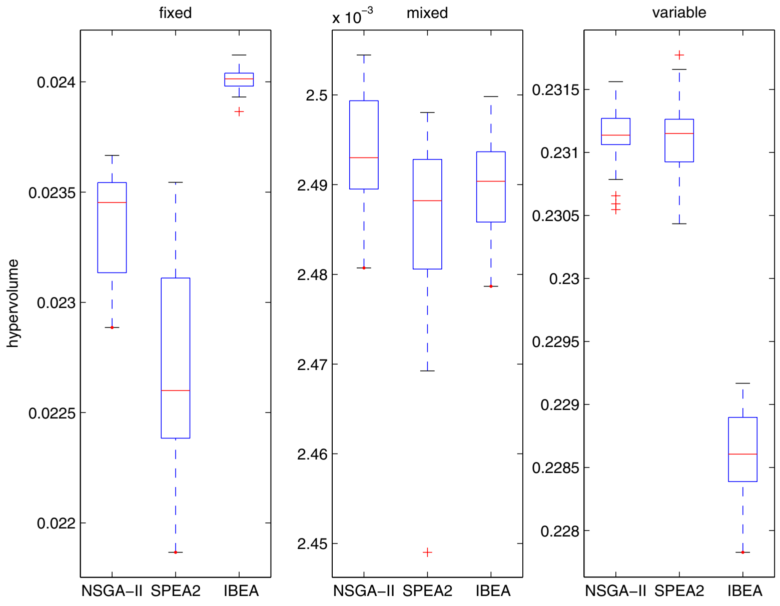


Fig. 4 Boxplot of the hypervolume indicator for NSGA-II, SPEA2 and IBEA

funds to the worst ones on variable funds. Notice that all differences are statistically significant, except SPEA2 vs IBEA on variable funds.

Among the three types of funds, it is clear that the front corresponding to variable funds is the longest one, spreading from very low risk/low profit solutions to high

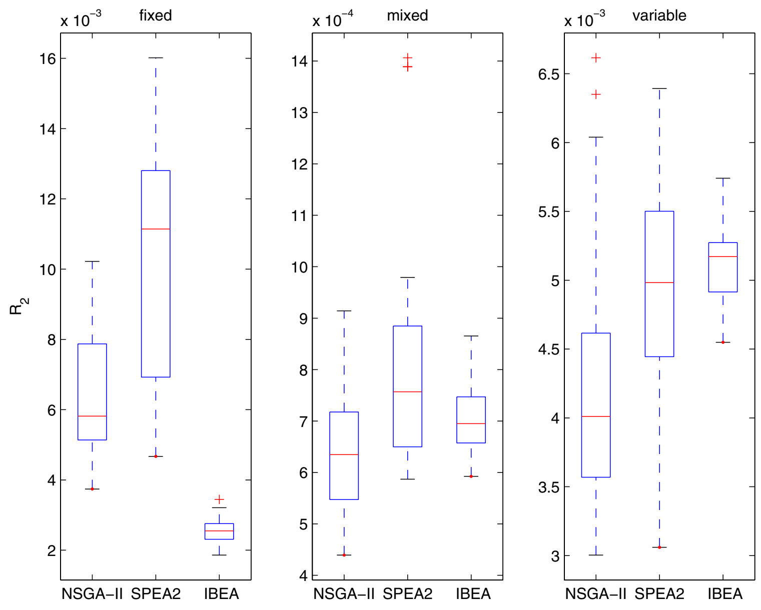


Fig. 5 Boxplot of the R_2 indicator for NSGA-II, SPEA2 and IBEA

risk/high profit portfolios. On the contrary, the front generated for mixed funds is much more focused on a regime than can be described as low risk/moderate profit. As to fixed funds, they cover a risk spectrum similar to that of variable funds, but the extreme points of attainable profit are well within the range of profit values found for variable funds. A more precise perspective of the particular risk/profit tradeoffs attained by each of the algorithms on the different types of funds will be provided in next section via the use of Sharpe's index.

4.2 Use of Sharpe's Index

Sharpe's index has been used for decision-making purposes, enabling the selection of a single solution out of the whole efficient front. Recall that this index measures how much excess profit per risk unit is attained by a certain portfolio. Depending on the particular shape of the observed front (which depends on the assets that can be potentially included in the portfolio), this solution can correspond to different risk/profit combinations.

This is illustrated in Figs. 6–8, where the best final solution (according to its Sharpe's index) provided by each algorithm on each of the 30 runs is shown for each type of fund. Best solutions tend to be arranged close to a line whose slope is the optimal value of Sharpe's index. Moreover, solutions are generally clustered in a relatively small range of risk/profit combinations. This indicates all algorithms typically provide solutions with a stable risk/profit profile. Indeed, the composition of portfolios tends to be stable as well, as shown in Figs. 9–10: NSGA-II, SPEA2

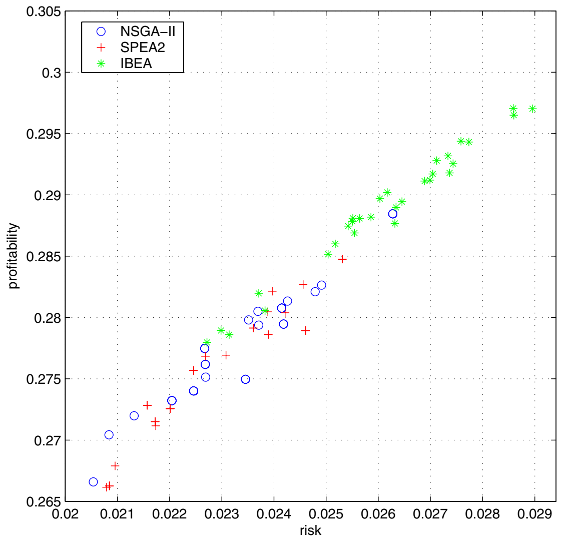


Fig. 6 Best solution (fixed funds) in each run (according to Sharpe's index) found by NSGA-II, SPEA2 and IBEA

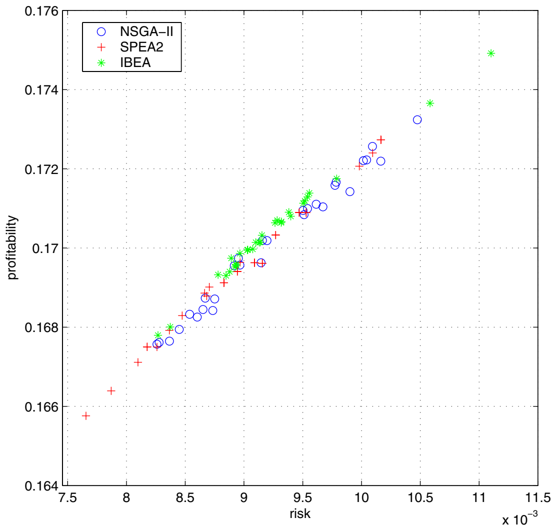
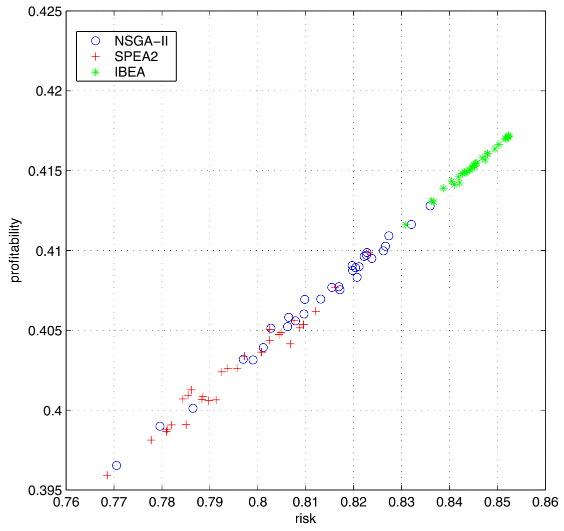


Fig. 7 Best solution (mixed funds) in each run (according to Sharpe's index) found by NSGA-II, SPEA2 and IBEA

and IBEA agree on which funds should be included in the portfolio in each situation, and the variability of percentages (viz. the vertical size of boxes in the boxplot) is small, particularly in variable funds (where investments are mainly concentrated in fund #4, *Mercantil*) and mixed funds (where investments are stably distributed among three funds, *Ceiba*, *Mercantil*, and *Provincial*). In the case of fixed funds there seems to be a higher variability in the percentages of two funds (*Exterior RF* and *Primus RF*) due to their similar profiles.

Fig. 8 Best solution (variable funds) in each run (according to Sharpe's index) found by NSGA-II, SPEA2 and IBEA



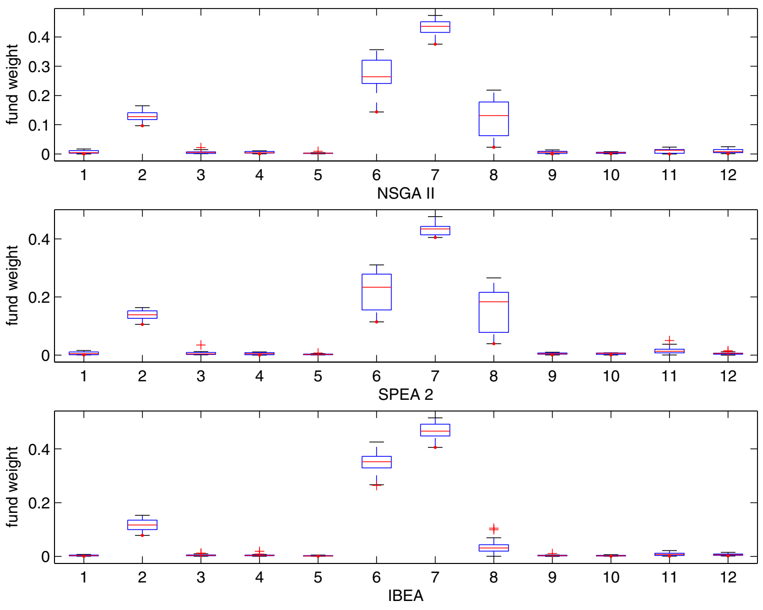


Fig. 9 Portfolio distribution (fixed funds) in solutions selected according to Sharpe's index. (Top) NSGA-II (middle) SPEA2 (bottom) IBEA.

Another interesting aspect concerns the distribution of Sharpe's index values obtained in each run. Fig. 12 shows a boxplot of Sharpe's index values for the 30 runs of each algorithm on each type of fund. NSGA-II and SPEA2 perform similarly

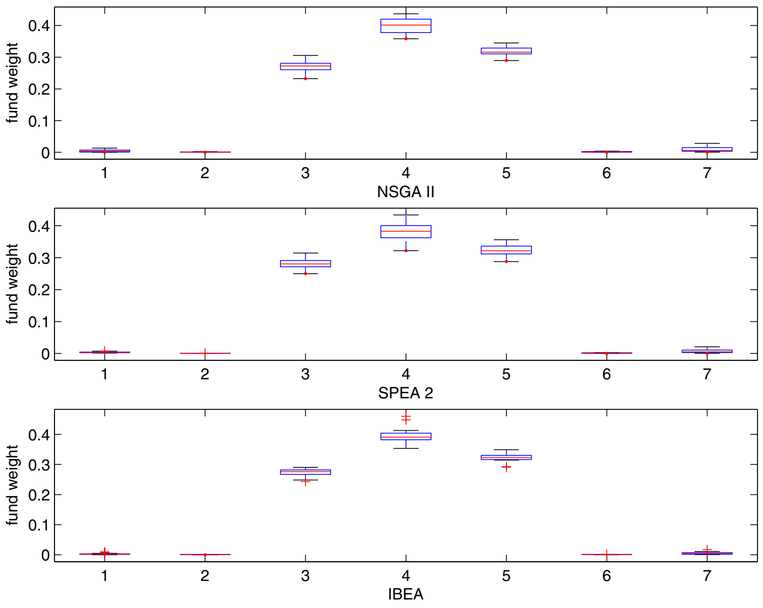


Fig. 10 Portfolio distribution (mixed funds) in solutions selected according to Sharpe's index. (Top) NSGA-II (middle) SPEA2 (bottom) IBEA.

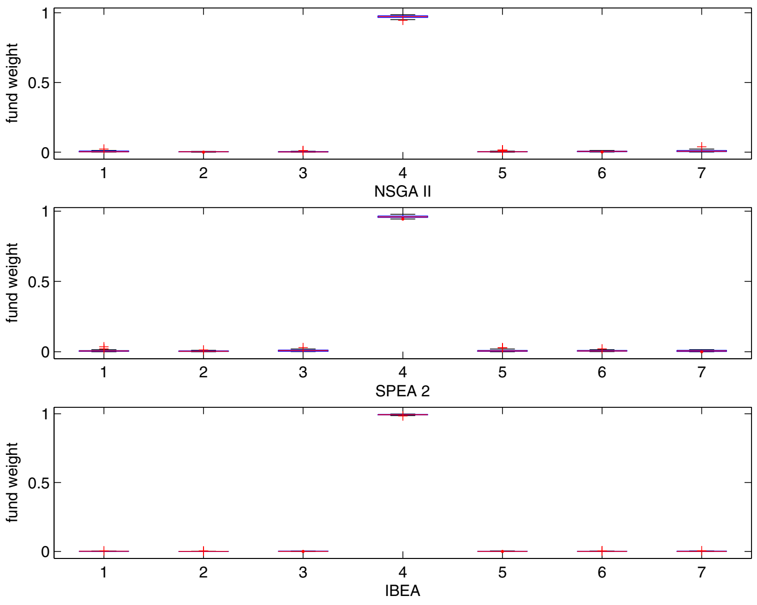


Fig. 11 Portfolio distribution (variable funds) in solutions selected according to Sharpe's index. (Top) NSGA-II (middle) SPEA2 (bottom) IBEA.

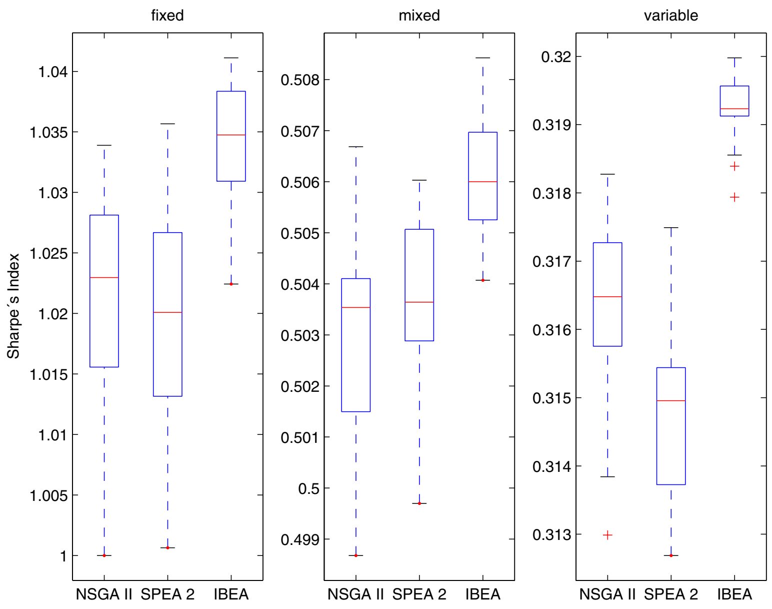


Fig. 12 Boxplots of Sharpe's index values attained by NSGA-II, SPEA2 and IBEA on fixed funds (right), mixed funds (middle) and variable funds (left)

Table 1 Comparison of the best solutions (according to Sharpe's index) found by NSGA-II, SPEA2 and IBEA

	Fixed Funds			Mixed Funds			Variable Funds		
	NSGA-II	SPEA2	IBEA	NSGA-II	SPEA2	IBEA	NSGA-II	SPEA2	IBEA
$E(\mathbf{R}\|\mathbf{W})$	.2775	.2821	.2881	.1697	.1690	.1697	.4128	.4099	.4172
$\sigma^2(\mathbf{R}\|\mathbf{W})$	.0227	.0240	.0255	.0090	.0087	.0089	.8359	.8232	.8523
Sharpe's index	1.034	1.036	1.041	.5067	.5060	.5084	.3183	.3175	.3200
$E_{2002}(\mathbf{R}\|\mathbf{W})$	.2367	.2457	.2365	.2455	.2468	.2453	.5392	.5342	.5432

except on variable funds, where NSGA-II is clearly better. However, IBEA outperforms both NSGA-II and SPEA2 on all types of funds (clear from visual inspection, and further verified by a Wilcoxon ranksum test).

Finally, the best overall solutions found by each of the algorithms are compared to an indexed portfolio in the Caracas Stock Exchange. To this end, we consider data for the year 2002, which was not seen during the optimization process. Table 1 displays the objective values for the best evolved portfolios, and the profit projection for 2002. As a reference, the mentioned indexed portfolio (IBC) achieves a profit of .1988 for 2002. It can thus be seen that the evolved portfolios are notoriously better that this latter portfolio.

5 Conclusions

Portfolio optimization is a natural arena for multiobjective optimizers. In particular, MOEAs have both the power and the flexibility required to successfully deal with this kind of problems. In this sense, this work has analyzed the performance of three state-of-the-art MOEAs, namely NSGA-II, SPEA2, and IBEA on portfolio optimization, using real-world mutual funds data taken from the Caracas Stock Exchange. Although the algorithms performed similarly from high level –with the exception of fixed funds, where IBEA provides a wider and deeper front– a closer look indicates that they offer different optimization profiles for this problem. NSGA-II is capable of advancing deeper towards some regions of the Pareto front (with statistical significance at the standard 0.05 level in the case of fixed funds and mixed funds), and IBEA lags behind the other two algorithms on variable funds.

Quite interestingly, when the subsequent decision-making step is approached and a single solution is selected from the Pareto front, the comparison turns out to be favorable to IBEA in all the cases. Furthermore, NSGA-II is better than SPEA2 on the problem scenario –variable funds– on which it did not achieve better quality indicators than the latter. More precisely, using Sharpe's index –based on a profit/risk ratio– to identify the best solution from the Pareto front provides significantly better values when using NSGA-II than SPEA2 on variable funds. This indicates a much better coverage of the region where such solutions lie. There is no statistically significant difference in the case of fixed and mixed funds. Likewise, IBEA provides much better solutions in this latter case, even when the quality indicators were worse than those of NSGA-II and SPEA2. This fact illustrates a recurrent theme in multiobjective optimization, i.e., the extent of the usefulness of approximating the whole Pareto front in practical problem scenarios. The fact that a deeper, wider, and more complete the Pareto front returned by an algorithm is better for any problem is based on a reasonable premise: providing the best set of solutions for the decision-maker to make the final selection. However, in some situations the details of how this decision-maker makes the decision cannot be ignored when evaluating the multiobjective optimizer. In other words, the best set of solutions is not necessarily the largest or the most diverse set, but the set that achieves a better coverage of the region in the search space that the decision-maker prefers. Portfolio optimization under Markowitz's model using Sharpe's index for selection is a good example of this situation.

Future work will be directed at analyzing other variants of the problem where additional constraints are introduced, e.g., cardinality constraints, minimum/maximum percentage of assets, etc. This analysis will pave the way for the development of ad hoc MOEAs, where we plan to integrate specific knowledge on the problem and on the subsequent decision-making procedure. Another line of future research concerns the measure of risk. While we have focused on variance here, this is by no means the unique available option. As an alternative, we may for example consider value at risk, i.e., the maximum loss that can take place at a certain confidence level. A related measure is the conditional value at risk, namely the expected shortfall in the worst $q\%$ of cases, where q is a parameter. Other possible measures are Jensen index [14], Treynor index [28], or models emanating from capital asset pricing theory (CAPM) [24], among others. An analysis of these alternatives is underway.

Acknowledgements. This work is partially supported by project TIN2008-05941 of the Spanish Ministry of Science and Innovation.

References

1. AVAF, Annual report of the venezuelan fund management association (2003), `http://www.avaf.org/`
2. Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E.: PISA—a platform and programming language independent interface for search algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 494–508. Springer, Heidelberg (2003)
3. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35, 268–308 (2003)
4. Bussetti, F.R.: Metaheuristic approaches to realistic portfolio optimisation. Master's thesis, University of South Africa (2000)
5. Chang, T.-J., Meade, N., Beasley, J., Sharaiha, Y.: Heuristics for cardinality constrained portfolio optimisation. Comput. Oper. Res. 27, 1271–1302 (2000)
6. Coello Coello, C., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Genetic Algorithms and Evolutionary Computation, vol. 5. Kluwer Academic Publishers, Dordrecht (2002)
7. Coello Coello, C.A.: 20 years of evolutionary multi-objective optimization: What has been done and what remains to be done. In: Yen, G.Y., Fogel, D.B. (eds.) Computational Intelligence: Principles and Practice, pp. 73–88. IEEE Computer Society Press, Los Alamitos (2006)
8. Coello Coello, C.A., Lamont, G.B.: Applications of Multi-Objective Evolutionary Algorithms. World Scientific, New York (2004)
9. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
10. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
11. Fieldsend, J., Matatko, J., Peng, M.: Cardinality constrained portfolio optimisation. In: Yang, Z.R., Yin, H., Everson, R.M. (eds.) IDEAL 2004. LNCS, vol. 3177, pp. 788–793. Springer, Heidelberg (2004)
12. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Forrest, S. (ed.) Fifth International Conference on Genetic Algorithms, University of Illinois at Urbana-Champaign, pp. 416–423. Morgan Kaufmann, San Francisco (1993)
13. Hansen, M., Jaszkiewicz, A.: Evaluating the quality of approximations to the nondominated set. Tech. Rep. IMM-REP-1998-7, Institute of Mathematical Modelling Technical University of Denmark (1998)
14. Jensen, M.C.: The performance of mutual funds in the period 1945 - 1964. J. Financ. 23, 383–417 (1968)

15. Knight, F.: Risk, Uncertainty, and Profit. Houghton Mifflin Company, Hart (1921)
16. Lehmann, E., D'Abrera, H.: Nonparametrics: Statistical Methods Based on Ranks. Prentice-Hall, Englewood Cliffs (1998)
17. Lin, D., Li, X., Li, M.: A genetic algorithm for solving portfolio optimization problems with transaction costs and minimum transaction lots. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3612, pp. 808–811. Springer, Heidelberg (2005)
18. Markowitz, H.M.: Portfolio selection. J. Financ. 7, 77–91 (1952)
19. Michaud, R.: The markowitz optimization enigma: Is optimized optimal? J. Financ. 45(1), 31–42 (1989)
20. Mukerjee, A., Biswas, R., Deb, K., Mathur, A.P.: Multiobjective evolutionary algorithms for the risk-return trade-off in bank loan management. Int. T. Oper. Res. 9, 583–597 (2002)
21. Radhakrishnan, A.: Evolutionary algorithms for multiobjective optimization with applications in portfolio optimization. Master's thesis, North Carolina State University (2007)
22. Schaffer, J.D.: Multiple objective optimization with vector evaluated geneticalgorithms. In: Grefenstette, J.J. (ed.) First International Conference on Genetic Algorithms, pp. 93–100. Lawrence Erlbaum, Mahwah (1985)
23. Schlottmann, F., Seese, D.: Hybrid multi-objective evolutionary computation of constrained downside risk-return efficient sets for credit portfolios. In: Eighth International Conference of the Society for Computational Economics, Computing in Economics and Finance, Aix-en-Provence, France (June 2002)
24. Sharpe, W.: Capital assets prices: A theory of market equilibrium under conditions of risk. J. Financ. 19, 425–442 (1964)
25. Sharpe, W.F.: Mutual fund performance. J. Bus. 39, 119–138 (1966)
26. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evol. Comp. 2, 221–248 (1994)
27. Streichert, F., Ulmer, H., Zell, A.: Evolutionary algorithms and the cardinality constrained portfolio selection problem. In: Operations Research Proceedings 2003, Selected Papers of the International Conference on Operations Research (OR 2003), pp. 3–5. Springer, Heidelberg (2003)
28. Treynor, J.: How to rate management of investment funds. Harvard Bus. Rev. 43, 63–75 (1965)
29. Vedarajan, G., Chan, L.C., Goldberg, D.: Investment portfolio optimization using genetic algorithms. In: Late Breaking Papers at the 1997 Genetic Programming Conference, pp. 255–263 (1997)
30. Veldhuizen, D.A.V., Lamont, G.B.: Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. Evol. Comp. 8(2), 125–147 (2000)
31. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)
32. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - a comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
33. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE T. Evol. Comp. 3(4), 257–271 (1999)

34. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm. In: Giannakoglou, K., et al. (eds.) EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Athens, Greece, pp. 95–100 (2002)
35. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE T. Evol. Comp. 7(2), 117–132 (2003)
36. Zitzler, E., Laumanns, M., Bleuler, S.: A Tutorial on Evolutionary Multiobjective Optimization. In: Gandibleux, X., et al. (eds.) Metaheuristics for Multiobjective Optimisation. Lecture Notes in Economics and Mathematical Systems, vol. 535 (2004)

Index

Author Index