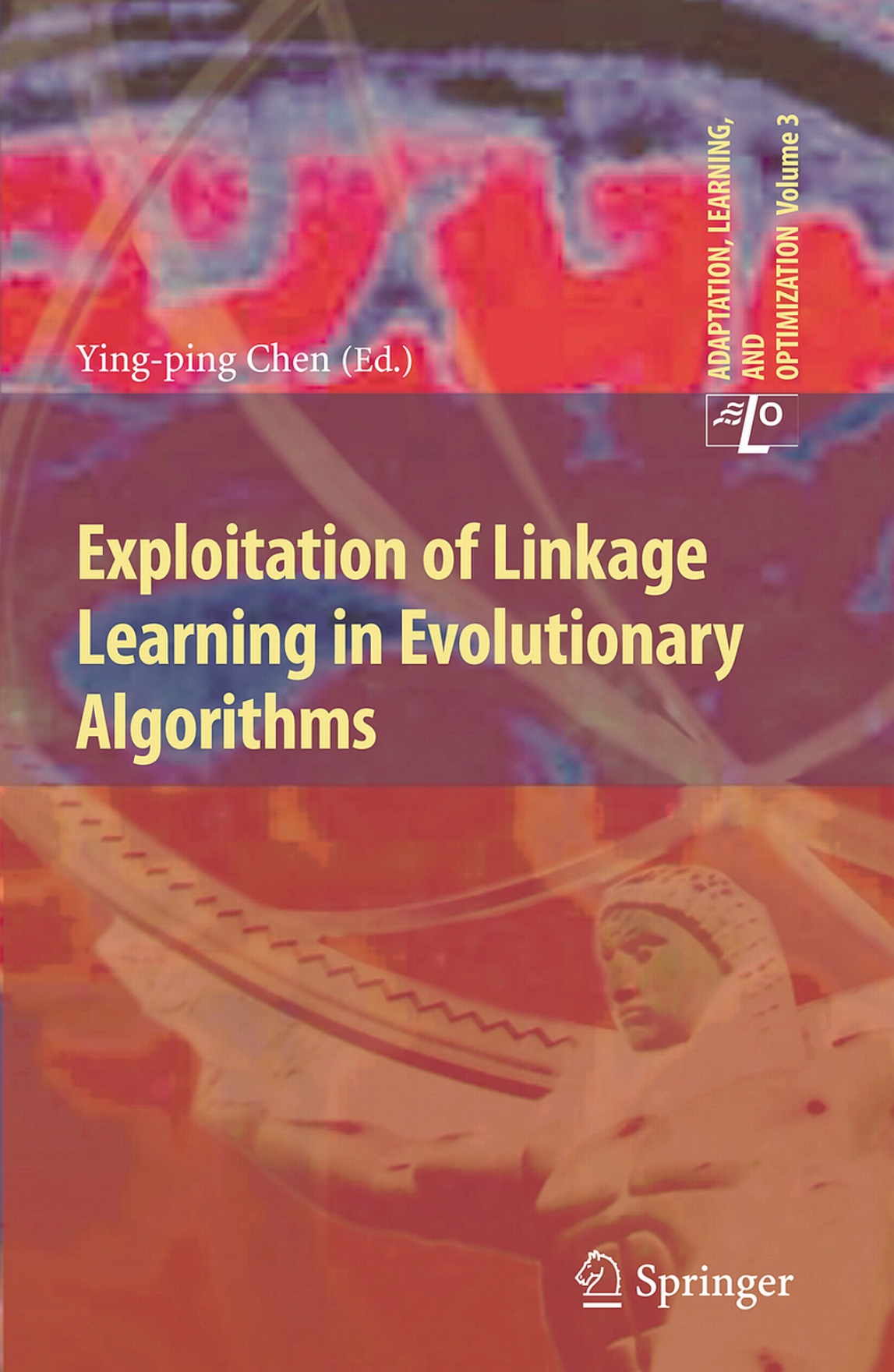

Ying-ping Chen (Ed.)
ADAPTATION, LEARNING, AND OPTIMIZATION Volume 3
Exploitation of Linkage Learning in Evolutionary Algorithms
Springer

Adaptation, Learning, and Optimization, Volume 3

**Series Editor-in-Chief**

Meng-Hiot Lim
Nanyang Technological University, Singapore
E-mail: emhlim@ntu.edu.sg

Yew-Soon Ong
Nanyang Technological University, Singapore
E-mail: asysong@ntu.edu.sg

---

Further volumes of this series can be found on our homepage: springer.com

Vol. 1. Jingqiao Zhang and Arthur C. Sanderson
Adaptive Differential Evolution, 2009
ISBN 978-3-642-01526-7

Vol. 2. Yoel Tenne and Chi-Keong Goh (Eds.)
Computational Intelligence in
Expensive Optimization Problems, 2010
ISBN 978-3-642-10700-9

Vol. 3. Ying-ping Chen (Ed.)
Exploitation of Linkage Learning in Evolutionary Algorithms, 2010
ISBN 978-3-642-12833-2

Ying-ping Chen (Ed.)

# Exploitation of Linkage Learning in Evolutionary Algorithms

Springer

Ying-ping Chen
Natural Computing Laboratory
Department of Computer Science
National Chiao Tung University
1001 Ta Hsueh Road
HsinChu City 300
Taiwan
E-mail: ypchen@cs.nctu.edu.tw

ISBN 978-3-642-12833-2 e-ISBN 978-3-642-12834-9

DOI 10.1007/978-3-642-12834-9

Adaptation, Learning, and Optimization ISSN 1867-4534

Library of Congress Control Number: 2010926027



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

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

# Preface

As genetic and evolutionary algorithms (GEAs) have been employed to handle complex optimization problems in recent years, the demand for improving the performance and applicability of GEAs has become a crucial and urgent issue. The exploitation of linkage is one of many mechanisms that have been integrated into GEAs. This concept draws an analogy between the genetic linkage in biological systems and the variable relationships in optimization problems. Most of the papers on the subjects of detecting, understanding, and exploiting linkage in GEAs are scattered throughout various journals and conference proceedings. This edited volume serves as an archive of the theoretical viewpoints, insightful studies, and the state-of-art development of linkage in GEAs.

This book consists of papers written by leading researchers who have investigated linkage in GEAs from different points of view. The 11 chapters in this volume can be divided into 3 parts: (I) Linkage & Problem Structures; (II) Model Building & Exploiting; and (III) Applications. Part I consists of 4 chapters that deal primarily with the nature and properties of linkage and problem structures. Thorough understanding of linkage, which composes the target problem, on the fundamental level is a must to devise GEAs better than what are available today. The next 4 chapters in Part II discuss issues regarding depicting linkage structures by establishing probabilistic models or presenting insights into relationship networks. These chapters develop adequate techniques for processing linkage, facilitating the analysis of problem structures and optimization tasks. Part III consists of 3 chapters that present applications that incorporate intermediate analysis solutions, allowing linkage to be exploited by, and incorporated into, practical problem-solving. More work on applying linkage to real-world problems should be encouraged, and this edited volume represents a significant step in that direction.

I hope that this book will serve as a useful reference for researchers working in the areas of detecting, understanding, and exploiting linkage in GEAs. This compilation is also suitable as a reference textbook for a graduate level course focusing on linkage issues. The collection of chapters can quickly

expose practitioners to most of the important issues pertaining to linkage. For example, practitioners looking for advanced tools and frameworks will find the chapters on applications a useful guide.

I am very fortunate and honored to have a group of distinguished contributors who are willing to share their findings, insights, and expertise in this edited volume. For this, I am truly grateful.

Hsinchu City, Taiwan Ying-ping Chen
December 2009

# Contents

# Part I
# Linkage and Problem Structures

# Linkage Structure and Genetic Evolutionary Algorithms

Susan Khor

**Abstract.** This chapter reviews and expands our work on the relationship between linkage structure, that is how decision variables of a problem are linked with (dependent on) one another, and the performance of three basic types of genetic evolutionary algorithms (GEAs): hill climbing, genetic algorithm and bottom-up self-assembly (compositional). It explores how concepts and quantitative methods from the field of social/complex networks can be used to characterize or explain problem difficulty for GEAs. It also re-introduces two novel concepts – *inter-level conflict* and *specificity* – which view linkage structure from a level perspective. In general, the basic GEAs performed well on our test problems with linkage structures resembling those empirically observed in many real-world networks. This is a positive indication that the structure of real-world networks which evolved without any central organization such as biological networks is not only influenced by evolution and therefore exhibit non-random properties, but also influences its own evolution in the sense that certain structures are easier for evolutionary forces to adapt for survival. However, this necessarily implies the difficulty of certain other structures. Hence, the need to go beyond basic GEAs to what we call GEAs with "brains", of which linkage-learning GEAs is one species.

## 1 Introduction

Research over the last two decades has uncovered evidence that evolved networks spanning across many domains, including social, technological and biological realms, share common structural properties [1, 28]. From this observation, one may ask the following question: What is the relationship between the structural properties of a network and the network's evolution and ability to survive through self-organization and adaptation? A similar question arises in the field of genetic evolutionary algorithms (GEAs). It is intuitive to view a problem's set of decision variables and their linkages or interactions as a network. What then is the relationship between the structural properties of a problem's interaction network and the ability of a GEA to evolve a solution for the problem? This chapter reports and expands on work we have done that addresses these twin questions in an abstract

Susan Khor
Concordia University, Montréal, Québec, Canada
e-mail: slc.khor@gmail.com

Y.-p. Chen (Ed.): Exploitation of Linkage Learning, ALO 3, pp. 3–23.
springerlink.com 

manner within the model of three basic GEAs: hill climbing, genetic algorithm and bottom-up self-assembly. We define basic GEAs as those that do not go beyond the primary tenets of biological evolution, i.e. random variation, genetic inheritance and competitive survival.

By examining the relationship between linkage structure of problems and basic GEA performance, we compiled a non-exhaustive list of structural characteristics and accompanying circumstances relevant to basic GEA performance. These include: modularity, degree distribution, clustering, path length, hub nodes, centrality, degree mixing pattern, inter-level conflict and specificity. Evidence of most, if not all, of these structural characteristics can be found in real-world networks. Interestingly, the basic GEAs performed well on our test problems with linkage structures resembling those empirically observed in many real-world networks, e.g. right-skewed heavy-tailed degree distribution, modularity and disassortativity. This is a positive indication that the structure of real-world networks which evolved without any central organization such as biological networks is not only influenced by evolution and therefore exhibit non-random properties, but also influences its own evolution in the sense that certain structures are easier for evolutionary forces to adapt for survival.

On the other hand, the structural characteristics can also help identify challenging problem instances for basic GEAs, and simultaneously, build a case for going beyond basic GEAs, that is to GEAs that have memory and the explicit ability to learn, to understand itself (self-reflection), make inferences and long-term strategies, in short "GEAs with brains".

The work presented here is distinct from those in [6, 7, 34 and 35] for example, which also investigate the relationship between problem structure and hardness, but not in the context of GEAs. Research has also been done on network topology and neural network behavior [30].

This chapter is organized as follows: section 2 describes the four test problems that will be referred to throughout the chapter; section 3 defines two basic structural characteristics and compares the four test problems in these terms; section 4 focuses on the hill climbing and genetic algorithm GEAs; section 5 focuses on the compositional (bottom-up self assembly) GEA; and section 6 concludes.

## 2 Test Problems

This section describes the four test problems directly referred to in this chapter. A test problem involves maximizing the number of satisfied if-and-only-if (*iff*) constraints defined on $S = \{0, 1\}^N$ where $iff(i, j) = \begin{cases} 1 & i = j; \\ 0 & otherwise. \end{cases}$. An *iff* constraint is a *symmetric interaction* or *linkage* between a unique pair of unique variables. A test problem can be viewed as a network (graph) of nodes and links (edges) where

each node represents a problem variable and each link denotes an *iff* constraint. We call such networks *interaction networks*.

How the set of *iff* constraints or linkages are placed on the set of variables, and their weights differentiate the four test problems **- C**, **II**, **M**, and **M1 -** used in this chapter. The adjacency matrix *A* for each of the four test problems when N=8 is given in Fig. 1. $A_{ij}$ is the weight associated with the linkage between variables *i* and *j*. Fitness of a string can be calculated by summing up the weights of the satisfied *iff* constraints, although a more concise method is given at the end of this section.

A test problem's adjacency matrix represents its interaction network. For a given N, test problem **C** has the most and maximum number of links: N (N-1) / 2; while test problems **M** and **M1**, the least and the minimum number of links: N-1 (for the set of variables to be connected). The **M1** linkage pattern is similar to **M**'s except **M**'s linkages above level 1 are shifted to the right by ε = (level – 2) variables. There is no reason for choosing level 3 as the level to begin the displacement except that by doing so we get a different degree distribution (section 3.2). For problems with fewer than 3 levels, there is no distinction between **M** and **M1**.

The problem size N is restricted to values of a power of 2 so that *S* can be recursively partitioned into levels and nested blocks of variables as shown in Fig. 2. A problem of size $N = 2^i$ where $i \in \mathbf{Z+}$ has $\log_2 N$ levels and N-1 blocks. The size of a block $|b|$ is the number of variables encompassed by the block. A block at level $\lambda$ encompasses $2^\lambda$ variables. The set of linkages belonging to a block *b* includes all linkages between *b*'s variables, and excludes all linkages belonging to *b*'s direct and indirect sub-blocks. The linkages are weighted and placed such that the maximum (optimal) fitness of a block is 1.0. Hence the optimal fitness of a string is N-1, and the two optimal strings are the all-zeroes (000…000) and the all-ones (111…111) strings, making problem difficulty for pure random search the same for all four problems of the same size.

Fitness of a string $F(S)$ is the sum over all block fitness values $f(b)$ as follows:

$$F(S) = \begin{cases} 0 & |S| = 1; \\ f(b) + F(S_L) + F(S_R) & |S| > 1. \end{cases}$$

$S_L$ and $S_R$ are the left and right halves of *S* respectively, and block *b* comprises all variables in *S* in each recursion. The difference between the fitness functions of the four test problems lies in the calculation of block fitness. For **C**, $f(b) = (p \times q) + (1 - p) \times (1 - q)$ where *p* and *q* are the proportion of ones in the left and right halves of *b* respectively [36]. For **II**, $f(b) = \frac{2}{|b|} \times \left( \sum_i iff\left(b_i, b_{|b|/2+i}\right) \right)$ where $0 \le i < |b| / 2$ and $|b|$ is the size of block *b* [12]. For **M**, $f(b) = iff(b_0, b_{|b|/2})$ [15]. For **M1**, $f(b) = iff(b_{0+\varepsilon}, b_{|b|/2+\varepsilon})$ where ε = (level – 2), level is level of *b*, and ε = 0 if level ≤ 2 [16].

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1/2 | 1/8 | 1/8 | 1/32 | 1/32 | 1/32 | 1/32 |
| 1 | 1/2 | 0 | 1/8 | 1/8 | 1/32 | 1/32 | 1/32 | 1/32 |
| 2 | 1/8 | 1/8 | 0 | 1/2 | 1/32 | 1/32 | 1/32 | 1/32 |
| 3 | 1/8 | 1/8 | 1/2 | 0 | 1/32 | 1/32 | 1/32 | 1/32 |
| 4 | 1/32 | 1/32 | 1/32 | 1/32 | 0 | 1/2 | 1/8 | 1/8 |
| 5 | 1/32 | 1/32 | 1/32 | 1/32 | 1/2 | 0 | 1/8 | 1/8 |
| 6 | 1/32 | 1/32 | 1/32 | 1/32 | 1/8 | 1/8 | 0 | 1/2 |
| 7 | 1/32 | 1/32 | 1/32 | 1/32 | 1/8 | 1/8 | 1/2 | 0 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1/2 | 1/4 | 0 | 1/8 | 0 | 0 | 0 |
| 1 | 1/2 | 0 | 0 | 1/4 | 0 | 1/8 | 0 | 0 |
| 2 | 1/4 | 0 | 0 | 1/2 | 0 | 0 | 1/8 | 0 |
| 3 | 0 | 1/4 | 1/2 | 0 | 0 | 0 | 0 | 1/8 |
| 4 | 1/8 | 0 | 0 | 0 | 0 | 1/2 | 1/4 | 0 |
| 5 | 0 | 1/8 | 0 | 0 | 1/2 | 0 | 0 | 1/4 |
| 6 | 0 | 0 | 1/8 | 0 | 1/4 | 0 | 0 | 1/2 |
| 7 | 0 | 0 | 0 | 1/8 | 0 | 1/4 | 1/2 | 0 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1/2 | 1/2 | 0 | 1/2 | 0 | 0 | 0 |
| 1 | 1/2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1/2 | 0 | 0 | 1/2 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1/2 | 0 | 0 | 0 | 0 | 0 |
| 4 | 1/2 | 0 | 0 | 0 | 0 | 1/2 | 1/2 | 0 |
| 5 | 0 | 0 | 0 | 0 | 1/2 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 1/2 | 0 | 0 | 1/2 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1/2 | 0 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1/2 | 1/2 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1/2 | 0 | 0 | 0 | 0 | 1/2 | 0 | 0 |
| 2 | 1/2 | 0 | 0 | 1/2 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1/2 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 1/2 | 1/2 | 0 |
| 5 | 0 | 1/2 | 0 | 0 | 1/2 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 1/2 | 0 | 0 | 1/2 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1/2 | 0 |

**Fig. 1** (top to bottom) Adjacency matrices for test problems C, II, M and M1, N=8

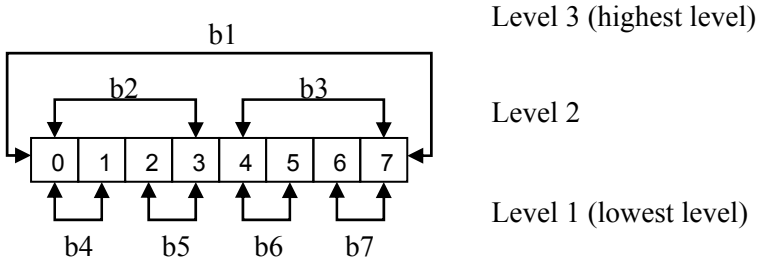



**Fig. 2** Decomposition of a string comprising 8 variables (0-7) into 7 blocks (b1-b7) and 3 levels (1-3)

## 3 Structure

This section defines two concepts used to characterize the interaction network or linkage structure of the test problems. Link weights are considered in the modularity quantification, but ignored in the degree distribution characterization.

### *3.1 Modularity*

When a network has identifiable subsets of nodes with higher link density amongst themselves than with nodes of other subsets within the same network, the network is said to be modular. This chapter uses the method introduced in [29] to quantify modularity of an interaction network. This method produces a real value Q within [0.0, 1.0] where a larger value indicates greater modularity. An example of how to calculate Q for the test problems in section 2 can be found in [16]. For a given problem size, the Q values for the four test problems in section 2 are identical and close to 1.0, e.g. Q = 0.9843 when N=128, and Q = 0.9922 when N=256. Therefore, these test problems are highly modular.

### *3.2 Degree Distribution*

The *degree* of a node is the number of links incident on the node. A network's degree distribution gives the probability $P(k)$ of a randomly selected node having degree $k$. Regular graphs like **C**'s and **II**'s interaction networks have single-point degree distributions since all nodes of a **C** or **II** interaction network have uniform degree. **M1**'s interaction network most resembles the degree distribution of classical random graphs which are scaled and forms a bell-shape curve (Poisson distribution for large N). Scale-free networks are those whose degree distributions can be approximated by a power-law. The degree distribution of scale-free networks is highly right-skewed with a heavy-tail denoting very many more nodes with small degree than nodes with large degree (*hubs*), and a wide degree value range. **M**'s interaction network is not scale-free, but compared to the other three test problems for a given N, it is most right-skewed and has the widest

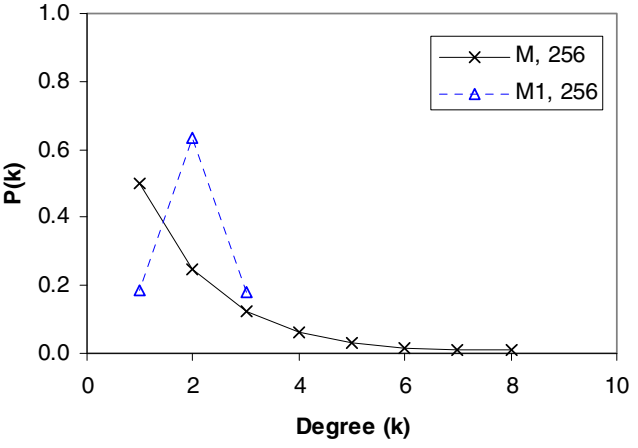



**Fig. 3** Degree distribution for M and M1, N=256

degree value range. For comparison, Fig. 3 gives the degree distributions of **M** and **M1** when N=256.

## 4 Hill Climbing and Genetic Algorithm

This section reviews our published results and presents some new developments in our work related to linkage structure and problem difficulty for hill climbing and genetic algorithm GEAs. The long suspected connection between modularity and problem difficulty for hill climbers and accompanying problem easiness for genetic algorithms [25] was clarified with the H-IFF problem [36]. The H-IFF problem demonstrated the importance of inter-module links as a factor in creating problem non-separability and frustration for two types of hill climbers: the Random Mutation Hill Climber (RMHC) [5] and the Macro-mutation Hill Climber (MMHC) [10]. Nevertheless, the H-IFF problem is not a piece of cake either for genetic algorithms. The genetic algorithm that successfully solved H-IFF worked explicitly to maintain genetic diversity in its population with the aid of deterministic-crowding [22].

In [16], several variations of the H-IFF problem were presented to investigate the relationship between linkage structure and problem difficulty for hill climbers and a genetic algorithm called upGA. One of the objectives of this investigation was to reduce the dependence of the genetic algorithm on explicit diversity maintenance and instead rely on mutation to produce genetic diversity in a population, as in the original design of genetic algorithms [8]. Additionally, as in biological evolution, both mutation and crossover play important roles in upGA success. Since upGA uses only one population, there is no teleological expectation, nor explicit manipulation to achieve an outcome such that different sub-populations evolve different parts of a solution for subsequent recombination into a whole solution. Details of the upGA algorithm can be found in [15].

The investigation reported in [16] found test problems amongst its test set which are easier for upGA than RMHC to solve, and that these test problems are modular like H-IFF, but unlike H-IFF which like **C** has single-point degree distribution, have broad right-skewed degree distributions like **M**. The investigation also looked at two non-structural factors: the Fitness Distance Correlation [9] for both Hamming and crossover distance, and the fitness distribution of the **C**, **II**, **M** and **M1** test problems, and found degree distribution to be a distinguishing factor in upGA performance. The most striking example is the test problem pair **M** and **M1**, which has identical Q values and identical fitness distributions, close FDC values, but upGA is more than twice more successful at solving **M** than **M1** within the given parameters. This difference in upGA performance is attributed to premature convergence, more specifically the synchronization problem [33] due to weak mutation in **M1** populations. Mutation success, as explained below, is related to the existence of hubs in a network.

The test problems in [16] were all predefined by hand. In [17], a looser approach is taken and test problems with interaction networks randomly generated to fit certain criteria (of degree distribution and modularity) [18] were used. Two kinds of interaction networks were generated: random and "scale-free" (allowing for finite size of networks) and experiments similar to that in [16] were made with these interaction networks. This second study confirmed that problems with "scale-free" interaction networks were easier for both hill climbers and upGA to optimize than problems with random interaction networks, and this difference is more apparent when the networks are modular [17].

To understand the role of linkage structure for the above result, [17] took a closer look at high degree nodes of the interaction networks (the degree of the high degree nodes in random interaction networks is expectedly, smaller than the degree of the hub nodes in "scale-free" interaction networks) and found significant differences in terms of path length or shortest distances between high degree nodes and the centrality of high degree nodes. Node (betweeness) centrality refers to the number of shortest paths that passes through the node. The average path length between nodes of high degree is significantly shorter in the modularized "scale-free" networks than in the modularized random networks. Hubs in the modularized "scale-free" networks also occupy a much more central position in inter-node communication on a network than in the modularized random networks. Further, in the "scale-free" networks, hubs mutate successfully less frequently than non-hub nodes. This is understandable since changing the value of a hub node can cause large changes to fitness.

We hypothesize that the aforementioned three factors combined help both hill climbers and upGA to be more successful (find a global optimum within the given parameters) on problems with the "scale-free" interaction networks. Shorter distances facilitate rapid inter-node transmission of information, and in turn synchronization of hubs, which helps a GEA to avoid the synchronization problem (different modules optimize to different global optima and cannot be put together to create a global optimum because the inter-module constraints are unsatisfied). Being more robust to mutation and occupying a central position in the network enables the hubs nodes to transmit a consistent message to the other non-hub

nodes so that they all optimize towards the same global optimum. To summarize, hubs exert a coordinating, directing and stabilizing force over the adaptation of a genotype, which is helpful for conducting search in frustrating fitness landscapes. The preceding analysis is successfully applied (holds true) for **M** and **M1** test problems (Table 1).

**Table 1**

| Links | N=256 | Degree | | Path Length | | | | Degree Centrality | RMHC SUCC | upGA SUCC |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | Average | Median | | | |
| 255 | **M1** | 1 | 3 | 1 | 39 | 17.85 | 18 | 0.5490 | 0/30 | 10/30 |
| 255 | **M** | 1 | 8 | 1 | 15 | 7.03 | 7 | 0.7908 | 0/30 | 27/30 |

Walsh [34] reports a number of real-world benchmark problems have non-random interaction networks, and that graphs with right-skewed degree distributions are easier to color than random graphs [35]. This is good news for the practicality of GEAs in the light of the above discussion which suggests GEAs such as hill climbing and genetic algorithms are more suited for solving problems with non-random interaction networks.

However, Walsh [34] also found that shorter path lengths, a side effect of high clustering, tend to increase difficulty for graph-coloring problems. This observation appears contrary to what we have proposed here so far. Nonetheless, it is expected since high clustering tend to create large cliques and the chromatic number of a graph is intimately related to the size of the largest clique. For the graph-coloring problem, we also found that degree-degree correlation affects the number of colors used by a complete algorithm DSATUR [2] and by a stochastic algorithm HC, which is similar to RMHC. Given similar conditions (i.e. number of nodes, number of links, degree distribution and clustering) and an unlimited color palette, fewer colors are needed to color disassortative than assortative networks [19]. We attribute this result to shorter path lengths amongst nodes of high degree in more assortative networks.

By preferring to fix the color of high degree nodes, which DSATUR does explicitly in its algorithm and HC does implicitly (negative correlations are recorded between node degree and time of last successful mutation, and between node degree and number of successful mutations), the number of colors used increases more slowly and less unnecessarily. However, if nodes of high degree have high probability of being directly linked with each other, a graph coloring algorithm would have little choice but to use more colors. Nodes of low degree have more color choices (are less constrained) and their exact color can be determined later within the existing color range. As such, a network would be colorable with fewer colors if nodes of high degree were separated from each other but still connected to one another via nodes of lower degree which are less constrained in their color choices. Longer path lengths amongst nodes of high degree reflect networks with such characteristics, as do negative degree-degree correlation or disassortative degree mixing pattern.

Section 4 has presented several cases where examining linkage structure of a problem has provided clues about the difficulty of a problem for a GEA, and have suggested several criteria borrowed from the field of complex networks to characterize problem linkage structure, e.g. modularity, degree distribution, path length, centrality, hub nodes, clustering and degree mixing pattern. One possible challenge for the future is to design GEAs with different capabilities to address difficulties posed by problems with different linkage characteristics, or more ambitiously, one "super-GEA" to dynamically tailor itself to a problem's linkage idiosyncrasies. Conversely, the challenge could be to design or evolve the interaction networks of problems so that they are easily solved by GEAs. In either case, one must first know what characteristic(s) to watch out for in problems, and how to quantify it. We believe the approach we propose here paves a way towards this goal.

## 5 Compositional Gea

This section investigates how linkage structure influences bottom-up evolution modeled by a compositional GEA called $\mathcal{J}$ (after the Roman God Janus). Unlike the hill climbers and upGA discussed in the previous section, the $\mathcal{J}$ GEA simultaneously composes and evolves a solution through a bottom-up self-assembly process. SEAM [37] and ETA [21] are two examples of GEAs using bottom-up compositional evolutional.

Starting with an initial pool of atomic entities with randomly generated genotypes, $\mathcal{J}$ creates interaction opportunities, in the form of joins and exchanges, between randomly selected entities in a population. $\mathcal{J}$ follows the rationale that when two or more entities interact with one another, they either repel (nothing happens), are attracted to each other as wholes (a join is made) or there is partial attraction (an exchange of entity parts is made). Section 5.1 explains how $\mathcal{J}$ decides whether a join or an exchange succeeds. The total amount of genetic material in a $\mathcal{J}$ population remains constant throughout a run, although the number of entities may fluctuate; typically number of entities decrease as entities assemble themselves into larger entities. Entities are also selected at random to undergo random bit-flip mutation. Section 5.2 explains how $\mathcal{J}$ determines if a mutation succeeds.

The $\mathcal{J}$ algorithm used here (section 5.3) is, for the most part, the one described in [14] which is a significant revision of an earlier version published in [13]. The revisions were mainly made to clarify part-fitness calculations. The only difference between the $\mathcal{J}$ algorithm used in this chapter and that in [14] is the use of random selection instead of fitness-proportionate selection when choosing an entity for mutation. We believe this second modification to be more realistic in terms of biological evolution where variations such as mutation (seemingly) occur at random. Detailed explanations and the reasoning behind the design of $\mathcal{J}$ are documented in [14].

## 5.1 Joins and Exchanges

This section illustrates how $\mathcal{J}$ decides whether a join or an exchange succeeds. The general rule is entities stay in their current context until there is clear incentive, in the form of increase to their own fitness, to change context. In addition, a join or an exchange must benefit all participant entities to succeed. Thus, successful joins and exchanges are instances of *synergistic cooperation* in $\mathcal{J}$.

To illustrate, consider two entities *a* and *b* with respective genotypes and fitness values as given in Table 2.

**Table 2** Fitness details for entities to illustrate joins and exchanges

| Entity | Genotype | **C** fitness | **II** fitness | **M** fitness |
|---|---|---|---|---|
| a | 0001 | 1.5 | 1.5 | 2.0 |
| b | 1000 | 1.5 | 1.5 | 1.0 |
| c | 0001 1000 | 3.625 | 3.5 | 3.0 |
| d | 1000 0001 | 3.625 | 3.5 | 3.0 |
| e | 0001 1001 | 2.5 | 2.25 | 2.0 |
| f | 1111 1000 | 4.75 | 4.75 | 5.0 |
| g | 1001 | 0.5 | 0.0 | 0.0 |
| h | 1111 | 3.0 | 3.0 | 3.0 |

A join between *a* and *b* can yield either *c (a + b)* or *d (b + a)*. For ease of discussion, let the join be *a* + *b* and the *inter-entity relationship* be **C**. This join creates a new context for both *a* and *b* in the form of entity *c*. We say that *a* and *b* are *part-entities* of *composite entity c*. In their original context, the fitness of both *a* and *b* is 1.5. In the new context of *c*, the fitness of both *a* and *b* is 1.5 + [3.625 – (1.5 + 1.5)] ÷ 2 = 1.8125. Since fitness of both *a* and *b* increases in the context of composite entity *c*, the join succeeds, i.e. *c* continues to exist because it is beneficial for both *a* and *b* to remain in *c*. If the relationship was **M**, the join would fail because neither *a* nor *b* increases their fitness by remaining in *c*.

Suppose an exchange is made between composite entities *e* and *f* and each of these two composite entities are decomposed for the purpose of the exchange into two equal halves: *e* into *a* and *g*, and *f* into *h* and *b* ($\mathcal{J}$ decides on the size of the part-entities participating in an exchange and the size of the resultant/new composite entity). Further, let the composite entity created by the exchange be *d* = (*b* + *a*). This new entity *d* will survive if both *a* and *b* have higher fitness by remaining in *d* than in their respective original contexts, i.e. *f* and *e*. Table 3 summarizes the changes to *a*'s and *b*'s fitness values when their context is changed from *e* and *f* respectively to *d*. The exchange succeeds only under **C** where *a*'s fitness increases from 1.75 to 1.8125 and *b*'s fitness increases from 1.625 to 1.8125. Under

**II**, the exchange fails because *a*'s fitness decreases from 1.875 to 1.75. The exchange also fails under **M** because *b*'s fitness decreases from 1.5 to 1.0.

**Table 3** Change in fitness for *a* and *b* as they move from composite entities *e* and *f* respectively to form composite entity *d*

| | **C** | | | | **II** | | | | **M** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *e* | *f* | *d* | | *e* | *f* | *d* | | *e* | *f* | *d* |
| *a* | 1.5 | +0.25 | - | +0.3125 | 1.5 | +0.375 | - | +0.25 | 2.0 | 0.0 | - | 0.0 |
| *b* | 1.5 | - | +0.125 | +0.3125 | 1.5 | - | +0.125 | +0.25 | 1.0 | - | +0.5 | 0.0 |

Even though entity *d* is fitter than *e* and less fit than *f* (Table 2), the exchange succeeds under **C** (i.e. both *e* and *f* are destroyed to create *d*) and fails under **II** and **M** (i.e. neither *e* nor *f* is destroyed to create *c*) because only *part-fitness* or fitness from the perspective of part-entities matters.

## *5.2 Mutation and Inter-level Conflict*

Inter-level conflict occurs when changes which are good (immediately adaptive or fitness improving) for one level is not so for another level. In *bottom-up inter-level conflict*, changes which are adaptive for lower levels are maladaptive for higher levels. *Top-down inter-level conflict* occurs when changes which are adaptive for higher levels are maladaptive for lower levels. Michod [24] describes the resolution of inter-level conflict as a fitness transfer from one level to another in the sense that lower (higher) levels are able to increase their fitness because higher (lower) levels give up some of their fitness. Because there is some sacrifice for another's good here, successful mutation is an occasion where *altruistic cooperation* can occur in $\mathcal{J}$.

### 5.2.1 *R1* Selection Scheme and Bottom-Up Inter-level Conflict

Bottom-up inter-level conflict is present in all four relationships – **C**, **II**, **M** and **M1**. This is confirmed with RMHC's low success rates on these relationships [16]. RMHC selects on the basis of total fitness of a genotype, and due to the modular linkage structure of **C**, **II**, **M** and **M1**, tends to favour optimization of sub-modules (part-entities) over optimization of the whole genotype (composite entity).

For example, suppose $\mathcal{J}$ mutates entity *e* by flipping the rightmost bit and as a result, transforms part-entity *g* into *b* and produces the mutant entity *c* (Table 4). This mutation succeeds for all three relationships, i.e. RMHC selects mutant *c* to replace *e*, because *c* is at least as fit as *e*. In $\mathcal{J}$, this selection scheme is named *R1*. Under **II**, the success of this mutation creates bottom-up inter-level conflict since the increase in fitness at lower levels is accompanied by a decrease in fitness at a higher level. Level 3 fitness drops from 0.75 to 0.5, while fitness at levels 1 and 2 increase (Table 4).

**Table 4** Fitness for composite entities *e* and *c*

| Entity | Genotype | **C** | **II** | **M** |
|---|---|---|---|---|
| $e = a + g$ | 0001 1001 | ⟨0.5, 1.0, 1.0 ⟩ 2.5 | ⟨0.75, 0.5, 1.0⟩ 2.25 | ⟨0.0, 1.0, 1.0⟩ 2.0 |
| $c = a + b$ | 0001 1000 | ⟨0.625, 1.0, 2.0⟩ 3.625 | ⟨0.5, 1.0, 2.0⟩ 3.5 | ⟨0.0, 1.0, 2.0⟩ 3.0 |

⟨, , ,…⟩ notes fitness by level from the highest (left) to the lowest (right) level.

Bottom-up inter-level conflict threatens the existence of a composite entity because it can transfer fitness at higher levels which is shared by all part-entities within a composite entity, to fitness at lower levels which benefits only some part-entities, and thereby weaken the bonds that bind part-entities together in a composite entity. The fitness-barrier, preventing part-entities from switching context when the opportunity arises is lowered.

### 5.2.2 *R2* Selection Scheme and Top-Down Inter-level Conflict

An alternative to the selection scheme in section 5.2.1 is one that mediates conflict in favour of higher levels. RMHC2 [11] is one such a selection scheme. In RMHC2, fitness of an entity (genotype) is broken down into levels and compared level wise from the highest level down. A mutant entity is chosen by RMHC2 if it is fitter than its parent at level λ and as fit as its parent at any level higher than λ, even though it may be less fit than its parent in total. In $\mathcal{J}$, this selection scheme is named *R2*.

For example, suppose $\mathcal{J}$ mutates entity *e* by flipping the rightmost bit and as a result, transforms part-entity *g* into *b* and produces the mutant entity *c* (Table 4). This mutation succeeds for **C** and **M** only, even though *c* if fitter than *e* for all three relationships. The mutation fails for **II** because *c* is less fit than *e* at level 3. If instead, the mutation is from *c* to *e*, this mutation succeeds under **II** even though *e* is less fit than *c* overall because *e* is fitter than *c* at a higher level. This transformation is also an instance of top-down inter-level conflict since increase in fitness at a higher level has come with a (possibly larger) decrease in fitness at a lower level.

Conflict mediation in favour of higher levels might seem like a good idea since it transfers fitness from lower to higher levels where it can be shared by all part-entities thereby strengthening the bonds that bind part-entities of a composite entity. Michod proposes it as one way biological aggregates maintain stability and develop their integrity [24]. However given the blind (without foresight) nature of evolution, there is no guarantee that fitness transfer from lower to higher levels will lead to optimal composite entities in the long term [27]. This is most evident when the relationship has top-down inter-level conflict. A test using RMHC2 (Table 5) reveals that of the four relationships examined in this chapter, **II** and **M1** have propensity for top-down inter-level conflict.

**Table 5** Number of runs out of the attempted 30 which found an optimal solution within the allocated number of function evaluations

| N=128 | RMHC2 | | | |
|---|---|---|---|---|
| $P_m$ | **C** | **II** | **M** | **M1** |
| 0.25 | - | 0 | - | 0 |
| 0.125 | - | 0 | - | 0 |
| 0.0625 | 30 | 0 | 30 | 0 |

## *5.3 The 𝒥 Algorithm*

The 𝒥 GEA attempts a join, an exchange or a mutation operation per iteration until either an optimal entity of the target size N is formed, or it reaches the maximum number of iterations specified. 𝒥's parameters are listed in Table 6.

**Table 6** Parameters for 𝒥

| Parameter | Symbol | Value |
|---|---|---|
| Atomic entity size | `q` | 2 |
| Target entity size | `N` | 256 |
| Number of entities per join or exchange | `p` | 2 |
| Maximum number of iterations | `MaxIters` | 1,000,000 |
| Initial population size | `PS` | 512 |
| Mutation rate | $P_m$ | 0.03125 |
| Join rate | $P_j$ | 0.5 |
| Selection scheme | `R` | 1, 2 |

**Main algorithm for 𝒥**

```
Create PS atomic entities each with a random genotype.
While number of iterations < MaxIters
    Increment number of iterations by 1
    If number of iterations is divisible by 50
        Record statistics.
        If fittest entity is optimal and of the target size, N
            Stop.
    With probability P_J
        Chose p distinct entities at random.
      If the p entities are all the same size and
            their combined size is ≤ N, then with probability 0.5,
            attempt a join with the p entities.
      Else
            Attempt an exchange with the p entities.
    Otherwise
        Select a random entity e. Attempt a mutation on e.
End.
```

The join operation enables p randomly chosen distinct entities of the same size to form a new composite entity *e* not larger than N. A join succeeds if entities increase their fitness in the context of the new composite entity (section 5.1).

**Join p entities**.

Create a new entity *e*.
    *e*.genotype is the concatenation of the genotypes of all p entities.
If *e* is fitter than the combined fitness of all p entities, the join succeeds.
    Remove the p entities from the population.
    Add *e* to population.
Else the join fails
    Discard *e* (restore the p entities).

The exchange operation enables entities belonging to p distinct entities chosen at random, to form a new composite entity *e*. At least one of the p distinct entities must be a composite entity, all entities exchanged are the same size (for simplicity), and the size of the new composite entity is the size of the largest of the p entities. The exchange succeeds if every entity that comprises *e* is fitter in *e* than in their respective original context (section 5.1). If an exchange succeeds, the new composite entity *e*, and the remaining entities not in *e* are added to the population.

**Exchange between p entities**

Determine smallest, the size of the smallest entity in the p entities.
Determine largest, the size of the largest entity in the p entities.
If every one of the p entities is atomic, stop.

Determine levels, the number of levels in the smallest entity.
    levels is $\log_q$ smallest.
Determine part size, the size of all part-entities, by chosing an integer *n* at random from within [1, levels]. Part size is $q^n$.
Use part size to split the p entities into part-entities.

Determine fitness of each part-entity in their respective original context. Let this fitness be old-fitness.

From the pool of part-entities, randomly select enough part-entities to create a new composite entity *e* of size largest. *e*.genotype is the concatenation of the genotypes of the selected part-entities.

Determine fitness for each part-entity in *e*. Let this fitness be new-fitness.
For each part-entity in *e*, compare new-fitness with old-fitness.
If for every pair, new-fitness > old-fitness, the exchange succeeds.
    Remove the p entities from the population.
    Add *e* and the unused part-entities to population.
Else the exchange fails.
    Discard *e* (restore the p entities).

The mutate operation flips at least 1 to $k$ number of bits of an existing entity, chosen at random from the population, and replaces the original (parent) entity with the mutant (child) entity if the mutant is not less fit than its parent in the sense defined by either the *R1* or the *R2* selection scheme (section 5.2).

**Mutate entity *e***

Create entity *f* whose genotype = *e*'s genotype.
Flip $k$ bits of *f*'s genotype chosen uniformly at random with replacement.
$k$ = maximum of (1, [1, $P_m \times f$.size])
If selection scheme is *R1*
If *f* is fitter than or as fit as *e*, the mutation succeeds
Replace *e* with *f* in the population.
Else, the mutation fails
Discard *f*.
Else if selection scheme is *R2*
For each level$\lambda$, starting from the highest level down, compare *e*'s and *f*'s fitness at level$\lambda$ as follows:
If *f* is fitter than *e* at level $\lambda$, the mutation succeeds
Replace *e* with *f* in the population. Stop.
Else if *e* is fitter than *f* at level $\lambda$, the mutation fails
Discard *f*. Stop.
If no decision has been made yet, the mutation succeeds (*e* and *f* have equal fitness for all levels)
Replace *e* with *f* in the population.

## *5.4 Results*

Fifty $\mathcal{J}$ runs using the parameter values listed in Table 6 and a different random number seed each time were made with both *R1* and *R2* selection schemes. The results are summarized in Table 7 and Figs. 4a and 4b.

When the *R1* selection scheme is used, $\mathcal{J}$ achieved close to 100% success for all four test problems (Table 7). In terms of number of iterations, $\mathcal{J}$ performed equally well for **C**, **II** and **M**, but took significantly longer (more iterations) for **M1** (Fig. 4a). The distribution of successful runs by iterations for **M1** has a longer tail on the right than the others (Fig. 4b). This performance difference is noteworthy because the **M** and **M1** relationships are very similar to each other (same number of links, identical Q values and fitness distribution – number of unique genotype configurations by fitness value) with one exception, their degree distributions (section 3.2).

When the *R2* selection scheme which does multi-level selection in favour of higher levels is used, $\mathcal{J}$ achieved 100% success for both **C** and **M**, but performed poorly on **II** and **M1** (Table 7). This is expected since both **II** and **M1** have top-down inter-level conflict (section 5.2.2) while both **C** and **M** do not.

In terms of number of iterations, there is no significant difference at the 99% confidence interval between **C** and **M** (Fig. 4a). However, 82% of **M** runs completed in less than 20,000 iterations compared with only 34% of **C** runs (Fig. 4b).

**Table 7** Number of successful $\mathcal{J}$ runs out of 50 and their average iterations

| | *R1* | | *R2* | |
|---|---|---|---|---|
| N=256 | Succ | Avg. iterations | Succ | Avg. iterations |
| **C** | 50 | 12,200 (1,003) | 50 | 21,500 (4,094) |
| **II** | 50 | 12,490 (1,044) | 3 | 483,300 (392,900) |
| **M** | 50 | 14,440 (6,381) | 50 | 18,420 (14,660) |
| **M1** | 49 | 49,150 (10,420) | 3 | 264,800 (438,000) |

One standard deviation is given in parentheses.

Nonetheless, the remaining 66% of **C** runs completed in less than 30,000 iterations while the remaining 18% of **M** runs took up to 70,000 iterations to complete. Hence, $\mathcal{J}$ could evolve optimal **M** entities faster than **C** entities with the *R2* selection scheme (but given enough time, there is no significant difference). This is another noteworthy difference. Both **C** and **M** do not have top-down inter-level conflict (section 5.2.2), but their interactions networks differ substantially not just in terms of number, weight and distribution of links, but also in what we term, *specificity* (section 5.4).

If we now compare the *R1* and *R2* results, none of the test problems seem to benefit significantly from the *R2* selection scheme. A different result was obtained in [14] where entities were randomly selected for mutation using a fitness-proportionate scheme. [14] found using *R2* significantly reduced the time (number of iterations) for $\mathcal{J}$ to evolve **M** entities, while significantly increased the time to evolve **C** entities. In [14], we concluded that conflict mediation in favour of higher levels can enhance (speed-up) bottom-up evolution, but that its usefulness is influenced by how entities relate to or interact with one another. Blindly giving higher levels priority over lower levels to adapt need not enhance bottom-up evolution, even when the relationship has no top-down inter-level conflict (e.g. **C**). We will come back to this point in section 5.5.

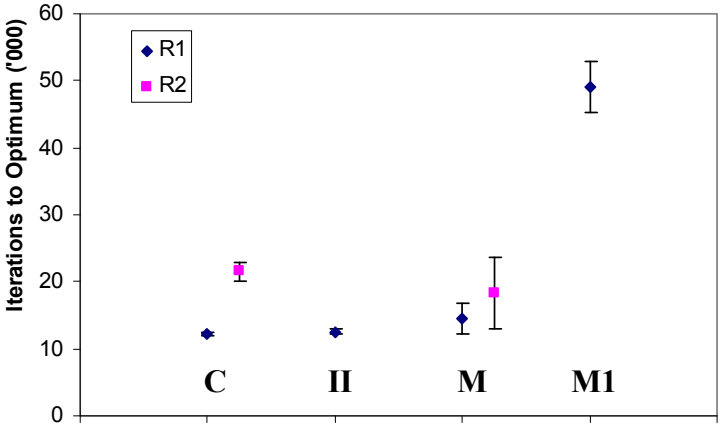



**Fig. 4a** Number of iterations to evolve an optimum entity of target size N=256, averaged over successful runs. Error bars indicate the 99% confidence interval. The R2 averages for II and M1 are excluded

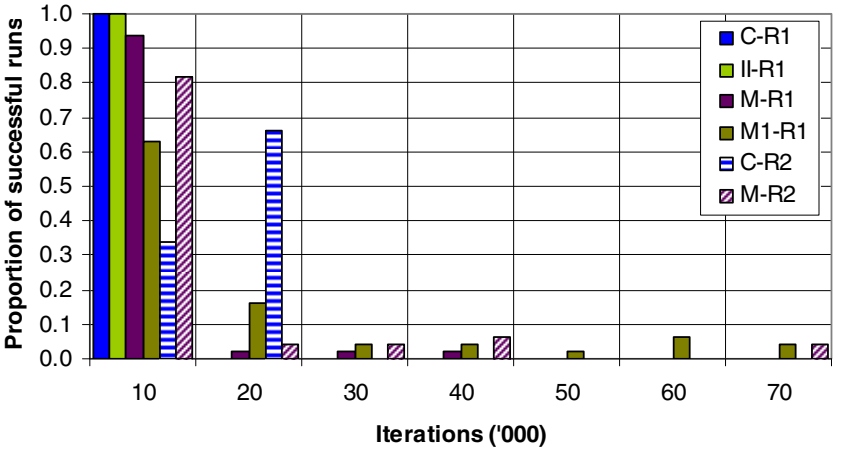



**Fig. 4b** Distribution of successful runs by number of iterations to evolve an optimum entity of target size 256

## *5.5 Specificity*

*Specificity* is a property of inter-entity interactions. Inter-entity interactions are more specific when there are fewer (but still some) interactions between entities. Specificity for a given level λ is the number of unique genotype configurations whose fitness at level λ is 0.0. A relationship with more genotypes with zero level fitness is more specific.

Table 8 illustrates for the four relationships discussed in this chapter. For $\lambda > 1$, **C < II < M** where '**<**' means is less specific than. **M** and **M1** are equally specific. Specificity can also be deduced from the adjacency matrices (section 2). Sparser matrices tend to produce more specific relationships.

**Table 8** Relationship specificity

| | **C** | **II** | **M** | **M1** |
|---|---|---|---|---|
| Number of genotypes with zero fitness at level λ. | $2^{N/2^{\lambda}}$ | $2^{N/2}$ | $2^{N\,(1\,-\,1/2^{\lambda})}$ | $2^{N\,(1\,-\,1/2^{\lambda})}$ |
| Number of genotypes with zero fitness at the highest level, i.e. $\lambda = \log_2 N$. | 2 | $2^{N/2}$ | $2^{N-1}$ | $2^{N-1}$ |
| Example of genotypes with zero $\lambda = 3$ fitness. | 1111 0000<br>0000 1111 | 0001 1110<br>1111 0000 | 0111 1010<br>0000 1111 | 0111 0010<br>1011 0100 |

Though related to linkage, specificity describes a different aspect of network structure than degree distribution. The degree distributions of **M** and **M1** starkly differ (section 3.2), but they are equally specific. Specificity is also different from modularity since all four relationships in Table 8 are equally modular, i.e. have identical Q values (section 3.1).

Relationships with high specificity make joins and exchanges amongst random entities more difficult to succeed since there are fewer ways to generate fitness above the sum of fitness of part-entities. But once a composite entity is formed, because of the specificity of the relationship, the composite entity is more difficult to destroy and hence is more stable (or less promiscuous). This line of analysis is carried further in [14]. Stability of intermediate aggregates is a corner stone of a bottom-up self-assembly process [32] and also of evolution [4]. "… The complex forms can arise from the simple ones by purely random processes. … Direction is provided to the scheme by the stability of the complex forms, once these come into existence. But this is nothing more than survival of the fittest – that is, of the stable." [32, p.93] "Darwin's 'survival of the fittest' is really a special case of a more general law of *survival of the stable*. The universe is populated by stable things. A stable thing is a collection of atoms that is permanent enough or common enough to deserve a name." [4, p.12]

Specificity has been defined as physical or structural isolation of parts [23] and this aspect of specificity is why when using *R2* in [14], $\mathcal{J}$ significantly reduced the time (number of iterations) to evolve **M** entities, while significantly increased the time to evolve **C** entities. $\mathcal{J}$ performs well when the modular structure of a relationship is respected [14]. By giving preference to optimization of higher levels (essentially inter-module constraints), *R2* can override the modular structure of a relationship, unless the structure of the relationship prevents it. By virtue of being more specific, the modular organization of the **M** relationship is more robust to such attacks than the **C** relationship.

Section 5.4 reported that the absence of top-down inter-level conflict in a relationship is insufficient to ensure that conflict mediation will be useful for bottom-up evolution, and that the efficacy of conflict mediation in favour of higher levels is also influenced by the structure of inter-entity interactions. Here, we give a characterization of such structure: high specificity. We propose that in the absence of top-down inter-level conflict, inter-entity interactions with high specificity stand to benefit more from conflict mediation in favour of higher levels than inter-entity interactions with low specificity.

Section 5 has discussed the performance of a bottom-up GEA, $\mathcal{J}$, and introduced two concepts: inter-level conflict and specificity. It finds that problem structure, especially when viewed from a level perspective, affects the evolutionary performance of $\mathcal{J}$. It might be interesting to see how the performance of $\mathcal{J}$ relates to upGA, and whether the two aforementioned concepts are applicable to genetic algorithms since genetic algorithms are also believed to work on the basis of combining low-order partial solutions into higher-order solutions (the building-block hypothesis) [8].

## 6 Conclusion

This chapter began with two related questions: (i) What is the relationship between the structural properties of a network and the network's evolution and ability to survive through self-organization and adaptation?; and (ii) What is the relationship between the structural properties of a problem's interaction network and the ability of a GEA to evolve a solution for the problem? It then summarized our work on the second question in sections 4 and 5. In this final section, we discuss the work presented so far and relate it to the first question.

Overall, structural properties of a problem's interaction network influence the ability of a GEA to evolve a solution for the problem. This finding in itself is not surprising, given previous work on epistasis (essentially linkages or dependencies between problem variables) and problem hardness for GEAs [3, 26] for example. The most significant contribution of this chapter is a quantifiable way to characterize different *kinds* of epistasis via concepts such as degree distribution, modularity, inter-level conflict and specificity. This is a shift from previous ways to look at problem difficulty for GEAs and to quantify epistasis. Naudts finds that when it comes to problem difficulty and GEAs "It is not the *amount* of interaction, but the kind of interaction that counts" [26, p.3]. The new way of seeing and characterizing linkage structure offered in this chapter could prompt new test problems particularly for linkage learning GEAs such as Estimation of Distribution Algorithms [20, 31].

This chapter also makes an observation with regards to the first question which is distinct from the question of network formation addressed in [1 and 28] for example. Basic GEAs performed well on our test problems with linkage structures resembling those empirically observed in many real-world networks, e.g. right-skewed heavy-tailed degree distribution with high modularity. This is a positive indication that the structure of real-world networks which evolved without any central organization such as biological networks is not only influenced by evolution and therefore exhibit non-random properties, but also *influences its own evolution* in the sense that certain structures are easier for evolutionary forces to adapt for survival. Our plan is to work with dynamic networks to investigate this point further.

## References

[1] Albert, R., Barabási, A.-L.: Statistical mechanics of complex networks. Reviews of Modern Physics 74(1), 47–97 (2002)

[2] Brélaz, D.: New methods to color the vertices of a graph. Communications of the ACM 22(4), 251–256 (1979)

[3] Davidor, Y.: Epistasis variance: a viewpoint of GA-hardness. In: Foundations of Genetic Algorithms, vol. 1, pp. 23–35. Morgan Kaufmann, San Francisco (1991)

[4] Dawkins, R.: The Selfish Gene. Oxford University Press, Oxford (2006)

[5] Forrest, S., Mitchell, M.: Relative building-block fitness and the building-block hypothesis. In: Foundations of Genetic Algorithms, pp. 109–126. Morgan Kaufmann, San Francisco (1993)

[6] Gomes, C., Walsh, T.: Randomness and Structure. In: Rossi, F., van Beek, P., Walsh, T. (eds.) Handbook of Constraint Programming. Elsevier, Amsterdam (2006)
[7] Hogg, T.: Refining the phase transition in combinatorial search. Artificial Intelligence 81, 127–154 (1996)
[8] Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor (1975)
[9] Jones, T., Forrest, S.: Fitness Distance Correlations as a measure of problem difficulty for genetic algorithms. In: 6th International Conference on Genetic Algorithms, pp. 184–192. Morgan Kaufmann, San Francisco (1995)
[10] Jones, T.: Evolutionary Algorithms, Fitness Landscapes and Search. PhD Dissertation, University of New Mexico, New Mexico, USA (1995)
[11] Khor, S.: Rethinking the adaptive capability of accretive evolution on hierarchically consistent problems. In: IEEE Symposium on Artificial Life, pp. 409–416. IEEE Press, Los Alamitos (2007)
[12] Khor, S.: HIFF-II: A hierarchically decomposable problem with inter-level interdependency. In: IEEE Symposium on Artificial Life, pp. 274–281. IEEE Press, Los Alamitos (2007)
[13] Khor, S.: How different hierarchical relationships impact evolution. In: Randall, M., Abbass, H.A., Wiles, J. (eds.) ACAL 2007. LNCS (LNAI), vol. 4828, pp. 119–130. Springer, Heidelberg (2007)
[14] Khor, S.: Problem Structure and Evolutionary Algorithm Difficulty. Ph.D. Dissertation, Concordia University, Montreal, Canada (2008)
[15] Khor, S.: Where genetic drift, crossover and mutation play nice in a free-mixing single-population genetic algorithm. In: IEEE World Congress on Computational Intelligence, pp. 62–69. IEEE Press, Los Alamitos (2008)
[16] Khor, S.: Exploring the influence of problem structural characteristics on evolutionary algorithm performance. In: IEEE Congress on Evolutionary Computation, pp. 3345–3352 (2009)
[17] Khor, S.: Effect of degree distribution on evolutionary search. In: Genetic and Evolutionary Computation Conference, pp. 1857–1858 (2009)
[18] Khor, S.: Generating hierarchically modular networks via link switching. arXiv:0903.2598 (2009)
[19] Khor, S.: Graph coloring and degree-degree correlation (2009) (Unpublished to date)
[20] Larranga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A new tool for evolutionary computation. Kluwer Academic Publishers, Dordrecht (2002)
[21] Lenaerts, T., Defaweux, A.: Solving hierarchically decomposable problems with the Evolutionary Transition Algorithm. In: Chiong, R., Dhakal, S. (eds.) Natural intelligence for scheduling, planning and packing problems. Springer, Berlin (2009)
[22] Mahfoud, S.: Niching Methods for Genetic Algorithms. Ph.D. Dissertation, University of Illinois (1995)
[23] Maslov, S., Sneppen, K.: Specificity and stability in topology of protein networks. Science 296, 910–913 (2002)
[24] Michod, R.E.: Cooperation and conflict in the evolution of complexity. In: Computational Synthesis: From basic building blocks to high level functionality: Papers from the AAAI Spring Symposium: 3–10. Technical Report SS-03-02. The AAAI Press, Menlo Park (2003)
[25] Mitchell, M., Forrest, S., Holland, J.H.: The Royal Road for genetic algorithms: fitness landscapes and GA performance. In: 1st European Conference on Artificial Life, pp. 245–254. MIT Press, Cambridge (1992)

[26] Naudts, B.: Measuring GA-Hardness. Ph.D. Dissertation, University of Antwerp, Belgium (1998)
[27] Nedelcu, A.M., Michod, R.E.: Evolvability, modularity and individuality during the transition to multicellularity in Volvocalean green algae. In: Schlosser, G., Wagner, G. (eds.) Modularity in Development and Evolution. University of Chicago Press (2003)
[28] Newman, M.E.J.: The structure and function of complex networks. SIAM Review 45, 167–256 (2003)
[29] Newman, M.E.J.: Modularity and community structure in networks. arXiv:physics/0602124v1 (2006)
[30] Oikonomou, P., Cluzel, P.: Effects of topology on network evolution. Nature Physics 2, 532–536 (2006)
[31] Pelikan, M.: Hierarchical Bayesian Optimization Algorithm: Toward a new generation of evolutionary algorithms. Springer, Heidelberg (2005)
[32] Simon, H.A.: The Sciences of the Artificial. The MIT Press, Cambridge (1969)
[33] Van Hoyweghen, C., Naudts, B., Goldberg, D.E.: Spin-flip symmetry and synchronization. In: Evolutionary Computation, vol. 10(4), pp. 317–344. MIT Press, Cambridge (2002)
[34] Walsh, T.: Search in a small world. In: International Joint Conference on Artificial Intelligence, pp. 1172–1177. Morgan Kaufmann, San Francisco (1999)
[35] Walsh, T.: Search on high degree graphs. In: Proc. of the 17th Int. Joint Conf. on Artificial Intelligence, pp. 266–274 (2001)
[36] Watson, R.A., Hornby, G.S., Pollack, J.B.: Modeling building-block interdependency. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 97–108. Springer, Heidelberg (1998)
[37] Watson, R.A.: Compositional Evolution: The Impact of sex, symbiosis and modularity on the gradualist framework of evolution. The MIT Press, Cambridge (2006)

# Fragment as a Small Evidence of the Building Blocks Existence

Chalermsub Sangkavichitr and Prabhas Chongstitvatana

**Abstract.** Building Blocks (BBs) can be considered as a plausible explanation for the success of Genetic Algorithms. The schema theorem can be interpreted as a support for Building Block Hypothesis. However, due to the nature of BBs that are dependent on the problems and the encoding of the chromosome, their behaviors are difficult to analyze. The aim of this work is to show the behavior of BBs processing. Toward this goal, a simplified definition of BBs, called Fragments is proposed. Fragments are similar contiguous bits found in highly fit chromosomes. Using this concept, genetic operations are designed to avoid disruption of BBs. Two operators are proposed, *Fragment identification* and *Fragment composition*. Experiments are designed to illustrate two aspects. One is the behavior of BBs processing and the other is the performance of the proposed GA incorporating these operators. The results of the experiments give a clear view of BBs processing. The performance of the proposed algorithm is shown to be superior to the competing algorithms for the Additively Decomposable Functions.

## 1 Introduction

In the early stage of the development of GAs, there are many works strive to find an answer how GA work. The schema theorem was proposed by Holland and was popularized by Goldberg [1]. It explains how GAs keep improving the population. It was interpreted that "Short, low-order, and highly fit schemata are sampled, recombined, and resampled to form strings of potential higher fitness" [2]. These short, low-order and highly fit schemata are called Building Blocks (BBs) and this interpretation is called the Building Block Hypothesis (BBH). This theorem becomes the fundamental of GAs. Later, there is an extended version of schema theorem that bases on a concept of the effective fitness. It shows that the chance of highly fit schema that is fitter than the average effective fitness will increase at the exponential rate and the length of the fit schema does not need to be short or low-order [3, 4].

The schema theorem assumes a positive effect of the selection that can maintain the good schema, and shows the negative effect of the crossover and the

Chalermsub Sangkavichitr · Prabhas Chongstitvatana
Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Thailand
e-mail: penockio@gmail.com, prabhas@chula.ac.th

Y.-p. Chen (Ed.): Exploitation of Linkage Learning, ALO 3, pp. 25–44.
springerlink.com 

mutation that disrupt the good schema. However there is no guideline how to process the BBs and the analysis is limited to the progress made in one generation. In practice, the crossover operation is expected to play a major role in "mixing" BBs. To understand this process, the fitness landscape called Royal Road function was designed to capture the idealized BBs form and many experiments show that the crossover operator has the ability to recombine the schemata into the better solutions [5, 6].

The normal selection process relies on the fitness value of chromosomes. In hard problems [7], the fitness landscape allures the schema away from the desired solution. These problems are called the deceptive functions. In order to explain the behavior of GAs in solving these problems, the Static Building Block Hypothesis (SBBH) is proposed. It states that "Given any low-order, short-defining-length hyperplane [i.e., schema] partition, a GA is expected to converge to the hyperplane [in that partition] with the best static average fitness (the 'expected winner')" [8]. This is proposed by Grefenstette and has not been proven. The SBBH shows the characteristic of GAs when deals with the deceptive problem. Moreover it also shows that the bias from the selection method in each generation should be considered carefully.

For the real world problems, it is hard to use BBH as an explanation of GAs success. The real structure of BBs is unknown and is dependent on the encoding scheme and it is very much dependent on the problem. So it is difficult to design a crossover operator that works well from the BBH perspective. One way to achieve the desired solution is to ensure that the rate of BBs construction is higher than the rate of BBs destruction. There is an effort to measure quantity of the BBs [9]. Many problems are analyzed: OneMax, Trap, Parabola and TSP problem. Two encoding scheme are used: the binary encoding and the gray encoding. The results show that BBs exist in OneMax, Trap, Parabola (the gray coding) and TSP (with third encoding scheme: binary matrix). This can imply that the BBs existence also depends on the encoding scheme. There are many factors that affect BBs such as the selection method, the identification algorithm, the recombination procedure and the measurement criterion.

If we hold the belief that the BBs existed, the rules to design a GA are available. Goldberg et al. proposed a principle for design competence GAs with six rules [10, 11, 12]. All of them concern with BBs but they are not easily realizable in practice due to lacking of information about the BBs. One obvious solution is finding a way to identify the BBs explicitly. This will help to manage the BBs effectively. The BBs can be regarded as the linkage between two or more alleles [13]. There are many ways to determine the linkage association such as loosely or tightly. A model of the linkage can be built in several manners and can be identified explicitly. In general, the meaning of the linkage model is equivalent to the BBs.

There are many ways to identify the BBs. An approach that concerns with explicit BBs is the messy GA (mGA) [14]. The mGA allows schema redundancy, and uses cut and splice technique as recombination operators. The mGA's mechanism and its BBs outperform the simple GAs (sGA) in many problems. Later, the mGA is improved in various versions [15, 16]. Another concept is the linkage

learning genetic algorithm (LLGA) [17, 18, 19]. For the LLGA, the chromosome is represented as a circular structure and the probabilistic expression mechanism is used for interpreting the chromosome. The recombination process uses the exchange crossover which performs linkage skew and linkage shift. Performance of the LLGA is superior to the simple GA on exponentially-scaled problems.

Recently, the field has evolved and one of the popular paradigm is the estimation of distribution algorithms (EDAs) that are claimed to solve the hard problem efficiently [20, 21]. The main concept of the EDA is sharing knowledge through a model. The model of distribution of population is created and is used to sample the next generation population. However most of them need some prior knowledge to identify relationships between individuals in a population and to build a model. The BBs are extracted explicitly in term of a probabilistic model. The main advantage of the EDAs comes from knowledge sharing in both model building and model sampling process to create the new offspring.

Typically, most of GAs operations such as crossover or mutation, are not designed to beware of BBs. They are designed with inspiration from nature. Even though a number of algorithms mentioned previously can demonstrate the schema of potential BBs, they are too complicated to use for studying the behavior of BBs. There is no explicit evidence that the BBs follow the BBH. Fortunately, a hint exists in the BBH that the short and low-order schema represents the picture of BBs. This point inspires us to find a way to present the BBs and their operation in a simple form.

This paper proposes a concept that simplifies the BBs identification and composition process. It can be applied in several ways. A simple algorithm is designed and demonstrated to validate the approach. The paper is organized as follows. The next section gives a definition of the simplified BBs. Section 3 demonstrates how to apply the proposed concept. Section 4 validates the algorithm with experimental results. Finally, Section 5 offers discussion and conclusion.

## 2 Fragment: A Simplified Definition of BBs

The study of GAs operators leads to the Building Block Hypothesis which explains the mechanism behind their success. Basically, GAs try to search for the suitable BBs and compose them to produce better solutions. In order to understand the behavior of GAs from the point of view of BBs creation and composition, there is a need for a very simple and direct representation of BBs. In this paper, we propose a new way to look at BBs called “Fragments”. The structure of Fragments is simple. The proposed definition can be applied directly in several ways both in the identification and the composition process. A Fragment can be regarded as a subset of the BB structure. The Fragment is defined as follows.

Given a sequence of chromosomes $C^k$ of length $l$

$$C^k = c_1^k c_2^k \ldots c_l^k : k \in \mathrm{I}^+, c_n^k \in \{0,1\}, 1 \le n \le l \tag{1}$$

Given a set of reference index of a chromosome of length $l$ from the position $f$ to position $t$

$$(f_i, t_i) = \left\{ z_i : 1 \le f_i \le z_i \le t_i \le l,\ f_i, t_i, z_i \in \mathrm{I}^+ \right\} \tag{2}$$

$$\text{and } \forall i, j \in \mathrm{I}^+ : i \ne j, (f_i, t_i) \cap (f_j, t_j) = \phi$$

Given a subsequence of the chromosome $C^{k_1}$

$$F_i^k = c_{f_i}^k c_{f_i+1}^k \ldots c_{t_i}^k \tag{3}$$

A set of the non overlap subsequence in a chromosome $C^{k_1}$ is defined as

$$S^k = \left\{ F_i^k : i \le l \right\} \tag{4}$$

We call the contiguous subsequence *F* as "Fragment".

Given a schema of a chromosome with the length *l*

$$H^k = h_1^k h_2^k \ldots h_l^k : k \in \mathrm{I}^+, h_n^k \in \{0,1,*\}, * \in \{0,1\}, 1 \le n \le l \tag{5}$$

The Fragment can be defined in term of the schema as

$$F_i^k = h_{f_i}^k h_{f_i+1}^k \ldots h_{t_i}^k : h_{z_i}^k \ne * \tag{6}$$

The definition above will be illustrated by an example in Fig.1. A schema *H* (Eq. 5) of 10-bit chromosome composes of 1*110***01 is shown. There are three Fragments (Eq. 6) in this schema as follows: 1, 110 and 01 (F1, F2 and F3 respectively).

There are many possible patterns of Fragments in a chromosome shown in Fig. 2. The minimum size of a Fragment is one allele and the maximum size equals to the chromosome length.

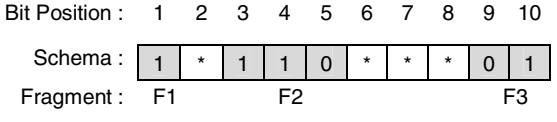



**Fig. 1** An example of Fragments in a schema

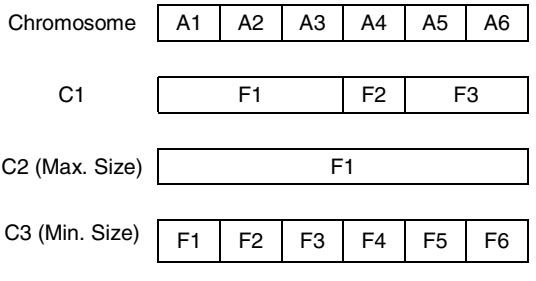



**Fig. 2** An example of possible patterns of Fragments in a chromosome

### *2.1 Fragments and BBs*

A schema can be separated into Fragments and Fragments can be combined into a schema. Fragments can be regarded as BBs under the interpretation of BBH because Fragments are short and low-order. There are many ways to compose a schema but most of them disrupt the structure of the schema. By defining Fragment, it is easier to understand the schema disruption. These substructures are easy to assemble. They have more diversity and can be combined in many different ways. The proposed method is simple and it is consistent with the interpretation of the BBH. This is the key to comprehend BBs and their processing.

### *2.2 Fragments and Linkage*

The smallest unit of a schema and a Fragment is one allele (one bit). This is the only case that there is no linkage. If there are two or more alleles, there may be a linkage among them. There may be a hierarchy of linkage. The clustering of alleles indicates that there is linkage (the closer they are, the tighter the linkage). If a common allele pattern occurs in many schemata, it implies that the linkage is robust. The clustering factor depends on the chromosome encoding. Generally it is not known what encoding is suitable for a problem. A Fragment is considered as a tight-linkage because it is a contiguous subsequence. Most recombination methods are based on crossover operators which have random cut points. Therefore the short, low-order and tight-linkage substructures have a higher potential to survive the crossover. This leads to an expectation that Fragments will survive and will become an important genetic material for producing the better chromosome. The problems where linkages are non-contiguous are considered as difficult problems for GAs [2, 22, 23].

## 3 Operations on Fragments

There are many methods to identify and to compose Fragments. The canonical GAs pays no attention to BBs and imposes no restriction on the crossover point. For the BBs mixing process, the crossover operation alone is sufficient. A traditional crossover operator does not require any special knowledge. In this section, a simple method for the Fragment identification and composition based on the crossover operation is proposed.

### *3.1 Fragment Identification*

The information theory supplies a tool to measure the information from data. In GAs, each chromosome holds some information about the solution. It is generally accepted that good solutions can guide a search method to the desired solution because they contain some useful information. The problem is how to extract such information. The common knowledge between good solutions (the mutual

information) can be observed. In the case of two chromosomes, the similarity of bits in the same position is their mutual information. Although there are many different chromosomes that have the same fitness value, there will be some repeat pattern (common knowledge) between them. If the size of population is large enough, this mutual information will be reliable. This increases the chance to find common subsequences and maintains diversity of common patterns. The Fragment identification process is described as follows:

Given a common subsequence between two chromosomes $C^{k_1}$ and $C^{k_2}$

$$F_i^{k_1,k_2} = c_{f_i}^{k_1} c_{f_i+1}^{k_1} \ldots c_{t_i}^{k_1} \text{ such that } c_{z_i}^{k_1} = c_{z_i}^{k_2} \tag{7}$$

where $z_i$ is defined in Eq.2. Other subsequences between two chromosomes $C^{k_1}$ and $C^{k_2}$ are

$$F_{j,1}^{k_1,k_2} = c_{f_j}^{k_1} c_{f_j+1}^{k_1} \ldots c_{t_j}^{k_1} \text{ such that } c_{z_j}^{k_1} \neq c_{z_j}^{k_2} \tag{8}$$

$$F_{j,2}^{k_1,k_2} = c_{f_j}^{k_2} c_{f_j+1}^{k_2} \ldots c_{t_j}^{k_2} \text{ such that } c_{z_j}^{k_1} \neq c_{z_j}^{k_2} \tag{9}$$

The contiguous subsequences between two chromosomes (Fragments) are defined as:

$$S_i^{k_1,k_2} = \left\{ (f_i, F_i^{k_1,k_2}) : i \leq \left\lceil \frac{l}{2} \right\rceil \right\} \tag{10}$$

$$S_j^{k_1,k_2} = \left\{ (f_j, F_{j,1}^{k_1,k_2}), (f_j, F_{j,2}^{k_1,k_2}) : j \leq 2\left\lceil \frac{l}{2} \right\rceil \right\} \tag{11}$$

$$S^{k_1,k_2} = \left\{ S_i^{k_1,k_2} \cup S_j^{k_1,k_2} : \left|S_i^{k_1,k_2}\right| + \left|S_j^{k_1,k_2}\right| \leq \left\lfloor \frac{l}{2} \right\rfloor + 2\left\lceil \frac{l}{2} \right\rceil \right\} \tag{12}$$

The definition above will be illustrated by an example in Fig. 3. Note that the string is indexed from left to right and starting from the position 1. Given two 10-bit chromosome sequences (Eq. 1) $C^1$ = (1,0,1,1,0,1,0,1,0,1) and $C^2$ = (1,1,1,1,0,0,1,0,0,1), then the index ranges (Eq. 2) of substructures between $C^1$ and $C^2$ are $(f_1,t_1)$ = (1,1), $(f_2,t_2)$ = (2,2), $(f_3,t_3)$ = (3,5), $(f_4,t_4)$ = (6,8), and $(f_5,t_5)$ = (9,10) and then the Fragments of $S^{1,2}$ are $F_1^{1,2} = (c_1^1)$ = (1), $F_{2,1}^{1,2} = (c_2^1)$ = (0), $F_{2,2}^{1,2} = (c_2^2)$ = (1), $F_3^{1,2} = (c_3^1 c_4^1 c_5^1)$ = (110), $F_{4,1}^{1,2} = (c_6^1 c_7^1 c_8^1)$ = (101), $F_{4,2}^{1,2} = (c_6^2 c_7^2 c_8^2)$ = (010), $F_{5,1}^{1,2} = (c_9^1 c_{10}^1)$ = (01) consecutively. The F1, F3

and F5 are the common Fragments (Eq. 7), and the F2 and F4 are the other Fragments (Eq. 8, 9).

## *3.2 Fragment Composition*

The common Fragments are regarded as the high potential good substructure or the BBs because they appear identically in two selected chromosomes which are assumed to be good or highly fit. Therefore they will be retained in the original structures. On the other hand, the other Fragments are considered as ambiguous substructures that may be or may not be the good substructures; however they come from the good chromosomes. In this case, they should not be disrupted.

The next problem is how to compose these Fragments. The traditional operation is the crossover operator. It performs well in various problems. There are many variations of the crossover operator. They differ in the number of cross-point and the criterion to choose the cross-point. Traditionally the one-point crossover is widely-used with good results. The two-point crossover is claimed to have the least disruption but there is no reliable evidence to support this claim [24]. The uniform crossover is most disruptive and it has uncertain performance depending on particular encoding and problem [25, 26, 27, 28]. A suitable number of cross-point is difficult to determine. Other special crossover methods are more elaborate and designed for special purpose.

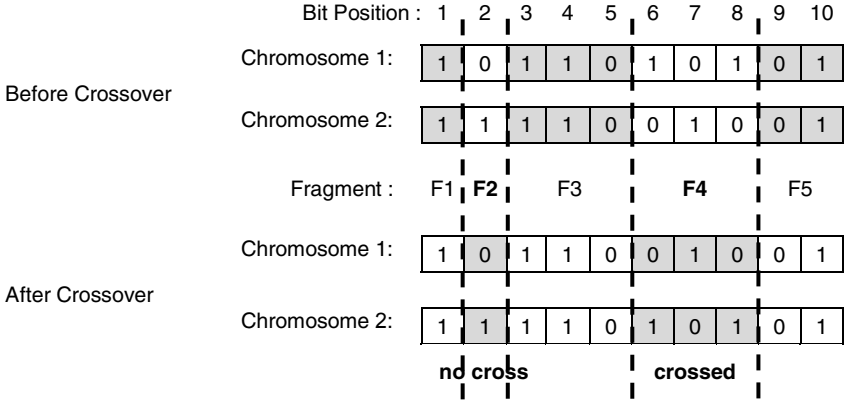



**Fig. 3** An example of the Fragment identification and composition between two chromosomes. The Fragment F4 is crossed

The main purpose of the crossover operator is the BBs recombination. But it often disrupts the BBs because it has not been designed with the knowledge about BBs. Thus if BBs can be identified explicitly, they should not be disrupted and they should be mixed properly to explore better solutions. Fragments will be exchange in the crossover process with no disruption as shown in Fig. 3-4. Each Fragment is crossed independently with the same crossover rate. The common

Fragments do not move. This method gains the recombination power at highest rate without the BBs disruption. The proposed method (Fragment identification and composition) is called Fragment Crossover (FC). In the Fig. 3, crossover occurs with the Fragment F4 only.

***R*** denotes selected chromosome.
***F*** denotes common and uncommon subsequence (Fragments).

**Algorithm** Fragment_Crossover
{The comparison between individual $\boldsymbol{R_1}$ and $\boldsymbol{R_2}$}
$F^{R_1,R_2} \leftarrow$ Compare $\boldsymbol{R_1}$ to $\boldsymbol{R_2}$;
(Identify common and uncommon substructure)
**For** $i$ = 1 to $k$ **do** (There are $k$ uncommon Fragments)
**If** Crossover **do**
$F_i^{R_1,R_2} \leftarrow swap(F_{i,1}^{R_1,R_2}, F_{i,2}^{R_1,R_2})$;
(Exchange the uncommon Fragment)
**EndFor**
**End.**

**Fig. 4** Pseudocode of the Fragment Crossover

## 4 Experimental Settings and Results

The experiment will be separate into two parts: the first part aims to study the behavior of GA in processing BBs, the second part studies performance of the proposed algorithm using a set of benchmark problems. The proposed algorithm is based on Simple Genetic Algorithm (sGA). The difference lies in the use of Fragment Crossover. This algorithm, named Simple GA with Fragment Crossover (sGA-FC), combines the BBs identification with the BBs composition. The first experiment is setup to show the processing of BBs. Two problems with known BB structures are used. They are Royal Road and Trap-5 functions. The results are compared with Simple Genetic Algorithm (with one-point crossover). The second experiment compares sGA-FC with many advanced algorithms. They are: Chi-square matrix (CSM) [29], Bayesian optimization algorithm (BOA) [30] and hierarchical Bayesian optimization algorithm (hBOA) [31]. The problem set consists of OneMax, Royal Road, Deceptive-3 (Dec3) [30], Exponential-Deceptive-3 (Exp-Dec3) [32], Trap-5, Hierarchical-if-and-only-if (HIFF) [33], and hierarchical Trap-1 (hTrap-1) [34] functions. The mutation operator is not used in all the experiments because the experiments are aimed to test capability of the schema processing (the BBs recombination) so it is better to avoid another source of genetic material. The details of experiments are described in the following sections.

### *4.1 Test Problems*

The BBs validation test problems can be separated into two classes: non deceptive problem and deceptive problem, both of them are additively decomposable functions (ADFs). Generally the deceptive problem is harder to solve than the non deceptive one; however this is also dependent on a particular algorithm. It is hard to claim what algorithm is suitable for a particular class of problem. Nevertheless the experimental results can be conducted to support the statement.

There are two BBs validation test problems in this experiment.

The Classical problem Royal Road function is designed for testing the ability of GAs to compose BBs [5]. The general *k*-bit Royal Road is define as

$$E_k(b_1,\ldots,b_k) = \begin{cases} f & ;\text{if } u = k \\ 0 & ;\text{otherwise} \end{cases} \tag{13}$$

Where $b_i$ is in $\{0,1\}$, $u = \sum_{i=1}^{k} b_i$ and $f = k$. The additively decomposable functions, denoted by $E_{m\times k}$ are defined as

$$E_{m\times k}(k_1 \ldots k_m) = \sum_{i=1}^{m} E_k(k_i), \quad k_i \in \{0,1\}^k \tag{14}$$

The *m* and *k* are varied to produce a number of test functions. The difficulty of this problem is that there is no hint about BBs. Its optimal solution is the solution that composed of all ones. This problem is a representative of problems that have a simple BB structure.

The well-known Trap functions are designed for studying BBs and the linkage problems in the GAs [17]. The general *k*-bit trap functions are defined as:

$$E_k(b_1,\ldots,b_k) = \begin{cases} f_{high} & ;\text{ if } u = k \\ f_{low} - \left((u \times f_{low})/(k-1)\right) & ;\text{otherwise} \end{cases} \tag{15}$$

Where $b_i$ is in $\{0,1\}$, $u = \sum_{i=1}^{k} b_i$ and $f_{high} > f_{low}$. Usually, $f_{high}$ is set at $k$ and $f_{low}$ is set at $k$-1. The Trap problem is defined as Equation 14.

The Trap functions fool the gradient-based optimizers to favor zeros, but the optimal solution is composed of all ones.

For the performance test of sGA-FC, seven problems are used. These problems are also the problems which have known BB structures. The details of these problems can be found in the references. The last two problems, HIFF and hTrap-1 are the hierarchical decomposable functions (HDFs) that are generally harder than the ADFs problems.

### *4.2 Measurement*

All tested problems are performed with 30 independent runs in both success case and failure case, and all algorithms are required to find the optimal solution in all of 30 runs in the success case and conversely for the failure case. In the success case, the minimum population size is used to achieve the optimum in all runs. But in failure case, the maximum population size is reported. The number of function evaluations and the population size are limited to be not greater than one million and fifty thousand consecutively. The sGA and sGA-FC use the same tournament selection method (tournament-size is 4). In the failure case, behavior of sGA and sGA-FC are shown only for the first one hundred generations. The crossover rate of sGA is set to 1.00 for the best result and sGA-FC is set to 0.50 for no bias. There is no mutation in the both algorithms. These parameters setting are the same for all test problems.

The BBs are classified into three classes to clarify behavior of the BBs processing as follows: Pure BB, Mixed BB and Non BB. An example is shown in Fig. 5. The Pure BB means the pattern in a chromosome that is corresponded to the ideal BB in the problem. The Mixed BB means that there is at least one allele of the ideal BBs in its structure; therefore it can be regarded as substantial source of diversity or genetic material in the recombination process. The Non BB means that there is no allele of the ideal BBs in its structure, and can be considered as a barrier for achieving the desired BBs. The number of the Pure BB depends on a particular problem and its encoding length, and is used to indicate the performance of algorithms directly. In this experiment, the binary encoding length of Royal Road and Trap-5 functions are 64 bits and 60 bits consecutively. The number of the Pure BB in each problem is 8 and 12 respectively. The number of classified BBs is measured in each generation with entire population. For example, if a population size is 100 for the Trap-5 60 bits problem, then there are 12 (BBs) × 100 (Chromosomes) equals to 1,200 BBs (Pure + Mixed + Non BBs) in each generation.

Fragments are independent contiguous subsequence in a chromosome so they are considered as parts of BBs. However they show the different point of view because Fragments are not exactly the BB. Fragments and BBs are different in size and structure. Fragments have two types: common and uncommon. The common Fragment is the similar and contiguous bits found in two good chromosomes. The uncommon Fragment is other contiguous bits found in good chromosomes. The numbers of common and uncommon Fragments are calculated from each crossover operation for example, there are three common Fragments and two uncommon Fragments in Fig. 3. The change in the number of both types of Fragments is an indicator of the behavior of BBs processing. For example, when the size of the common Fragment is larger, it can be interpreted two ways: one is that BBs have been combined into larger BBs and the other is that the evolution has been converged to a local optimum. The number of Fragments can be a good indicator of the diversity in the population. The ratio between the common and uncommon Fragment indicates the competition between alternate schemas. Therefore Fragments can be used to illustrate the schema processing and BBs construction.

Normally, the performance of GAs is compared through the number of fitness evaluations (#FE). So the results of all algorithms in the experiments are evaluated

by using the #FE. The competence Genetic Algorithms (CSM, BOA and hBOA) are used in the performance comparison. The parameters setting and results are found in the published works (CSM [30], BOA and hBOA [31]).

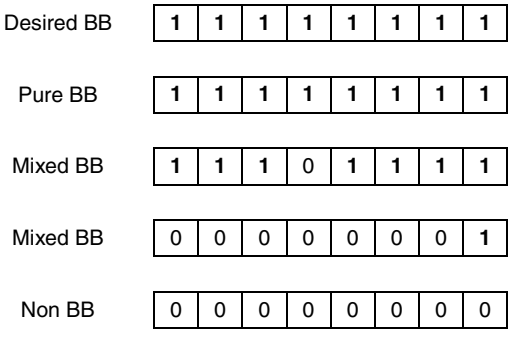



**Fig. 5** An example of the BBs classification

## *4.3 Results*

All results are averaged from 30 runs with the same parameters setting. The parameters setting and results of the Royal Road and the Trap-5 are shown in Table 1. In terms of #FE in the Royal Road function, sGA-FC performs worse than sGA. But in the Trap-5 function, the sGA-FC performs better than sGA. The population size in the success case and the failure case can be interpreted as upper bound and lower bound of an initial source of the diversity because there is no mutation or other source of genetic material during the process. The success case requires minimum number of population size to achieve the optimal solution in all runs. This is interpreted as an upper bound of diversity for reliable results. On the other hand, the failure case needs a maximum number of populations that still cannot find the optimal in all runs. This implies that if there is more population, it can find the optimal solution in at least one run. The population size in the failure case can be interpreted as a lower bound of diversity. In the success case, the population size of sGA-FC equals to sGA in the Royal Road function and is less than sGA in the Trap-5 function. In the failure case, the population size of sGA-FC is less than sGA in Royal Road function and is equal to the sGA in Trap-5 function. These indicate that sGA-FC can maintain diversity better than sGA in both problems.

**Table 1.** Experimental parameters setting and results

| Parameters | | Success 30 run | | | | Failure 30 run | |
|---|---|---|---|---|---|---|---|
| | | sGA | | sGA-FC | | sGA | sGA-FC |
| Problems | Size (bit) | #pop. | #FEs. | #pop. | #FEs. | #pop. | #pop. |
| Royal Road | 64 | 1,200 | 11,900 | 1,200 | 13,500 | 200 | 100 |
| Trap-5 | 60 | 2,300 | 28,400 | 1,600 | 20,850 | 300 | 300 |

Note: #pop denotes the population size and #FEs denotes the number of function evaluations.

Now turn the attention to the behavior of BBs processing. For the Royal Road function, the behavior of sGA is shown in Fig. 6. There is only the competition between Pure BBs and Mixed BBs in Fig. 6(a-b) because the Royal Road is not the deceptive problem. In the success case, sGA can combine good substructures into BBs. But in the failure case, sGA cannot reach the optimal solution due to diversity limitation.

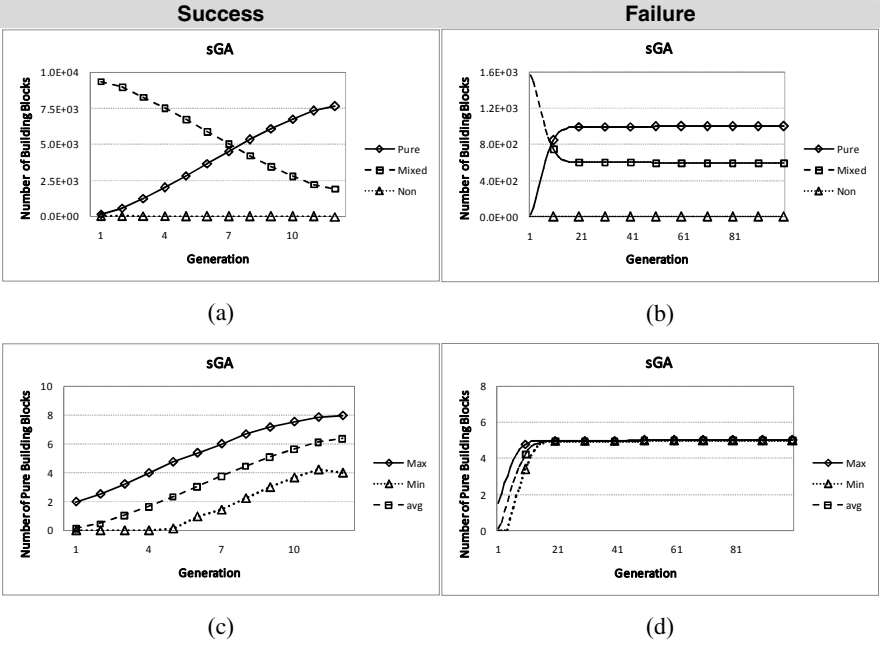



**Fig. 6** The sGA result of the Royal Road 64-bit: (a)(b) the number of the BBs in each generation and (c)(d) the number of the ideal BBs (Pure BBs) in each generation

For the sGA-FC, the results are shown in Fig. 7. The BBs processing behavior is similar to the sGA; furthermore in Fig. 7(a), Pure BBs and Mixed BBs cross at the generation 10 compared to the sGA which cross at the generation 7 (Fig. 6a). In the failure case (Fig. 7b), Pure BBs is close to Mixed BBs more than the sGA (Fig. 6b). These facts implied that the sGA-FC can maintain diversity better than the sGA. Fig. 7(e-f) show the average size of the common Fragments which increases continuously. In the success case, this growth of Fragments refers to the BBs combination but in the failure case, the common Fragments size continue to increase, this indicates that the diversity loss and the population will converge to a local optimum. Fig. 7(g-h) show the number of Fragments gradually decreases in the success case but it rapidly drops in the failure case due to the diversity loss.

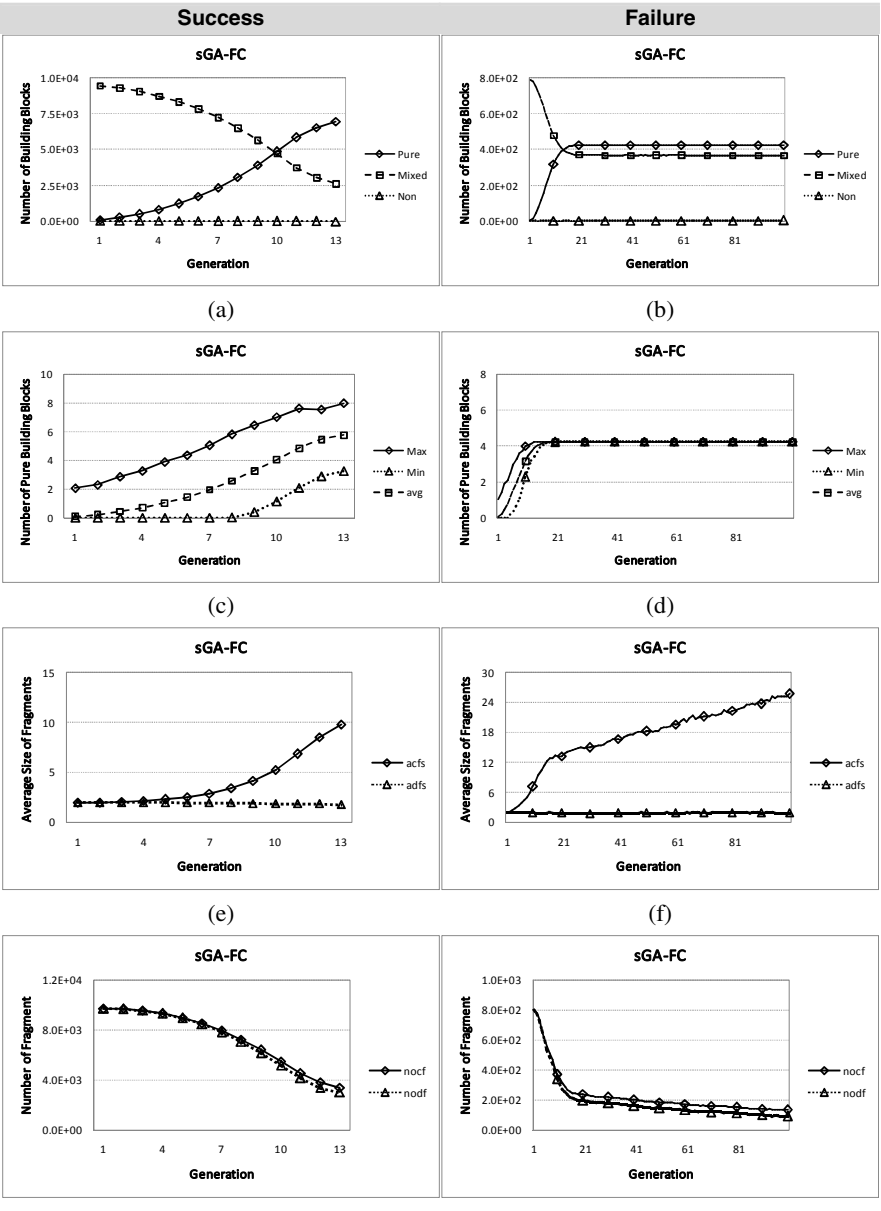



**Fig. 7** The sGA-FC result of the Royal Road 64-bit: (a)(b) the number of the BBs in each generation, (c)(d) the number of ideal BBs (Pure BBs) in each generation, (e)(f) the average size of Fragments in each generation and (g)(h) the number of Fragments in each generation. Note: acfs denotes the average common fragment size, adfs denotes the uncommon fragment size, nocf denotes the number of common fragment and nodf denotes the number of uncommon fragment

For the Trap-5 function, the overall behavior of the sGA (Fig. 8.) is still the same as the Royal Road function except there is an additional competition from the Non BBs due to the deceptive bias. The Non BBs grow at the lower rate comparing to the Pure BBs in both success case and failure case. The deceptive bias affects the BBs process slightly because the sGA has least bias and disruption from the one-point crossover. The Trap-5 60-bit is not too big for sGA thus the effect of the deceptive bias is not overwhelmed.

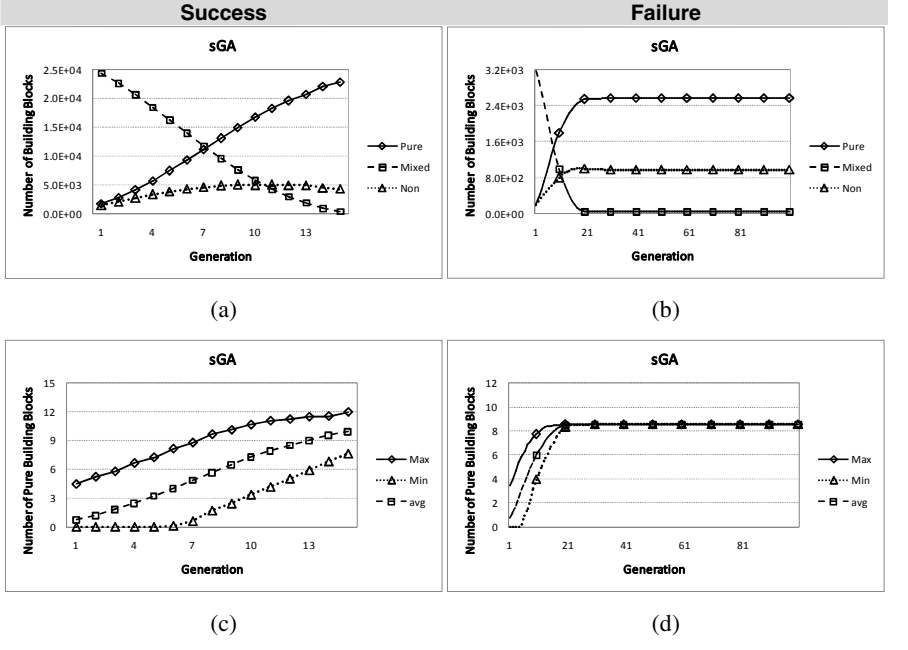



**Fig. 8** The sGA results of the Trap-5 60-bit: (a)(b) the number of the BBs in each generation and (c)(d) the number of ideal BBs (Pure BBs) in each generation

The results of sGA-FC are shown in Fig. 9. The BBs processing resembles to the sGA; moreover in Fig. 9(a), the Non BBs compete stronger than sGA in Fig 8(a). In the success case, the deceptive structures (the Non BBs) tend to be favor from the beginning to the cross point at generation 9 and after that sGA-FC can distinguish the ideal structure from the deceptive structure. The deception has more influence on sGA-FC than sGA because there is more BBs mixing in the sGA-FC and the deceptive structures are sensitive to the variation. The Fragments behavior (Fig. 9e-h) is similar to the Royal Road function in Fig. 7(e-h). In the success case (Fig. 9e), the uncommon Fragments size increases at the same rate as the common Fragments until the generation 11 and then level off. In case of the failure (Fig. 9f), the diversity of Fragments falls off sharply since the substructures are allured to the deceptive rapidly.

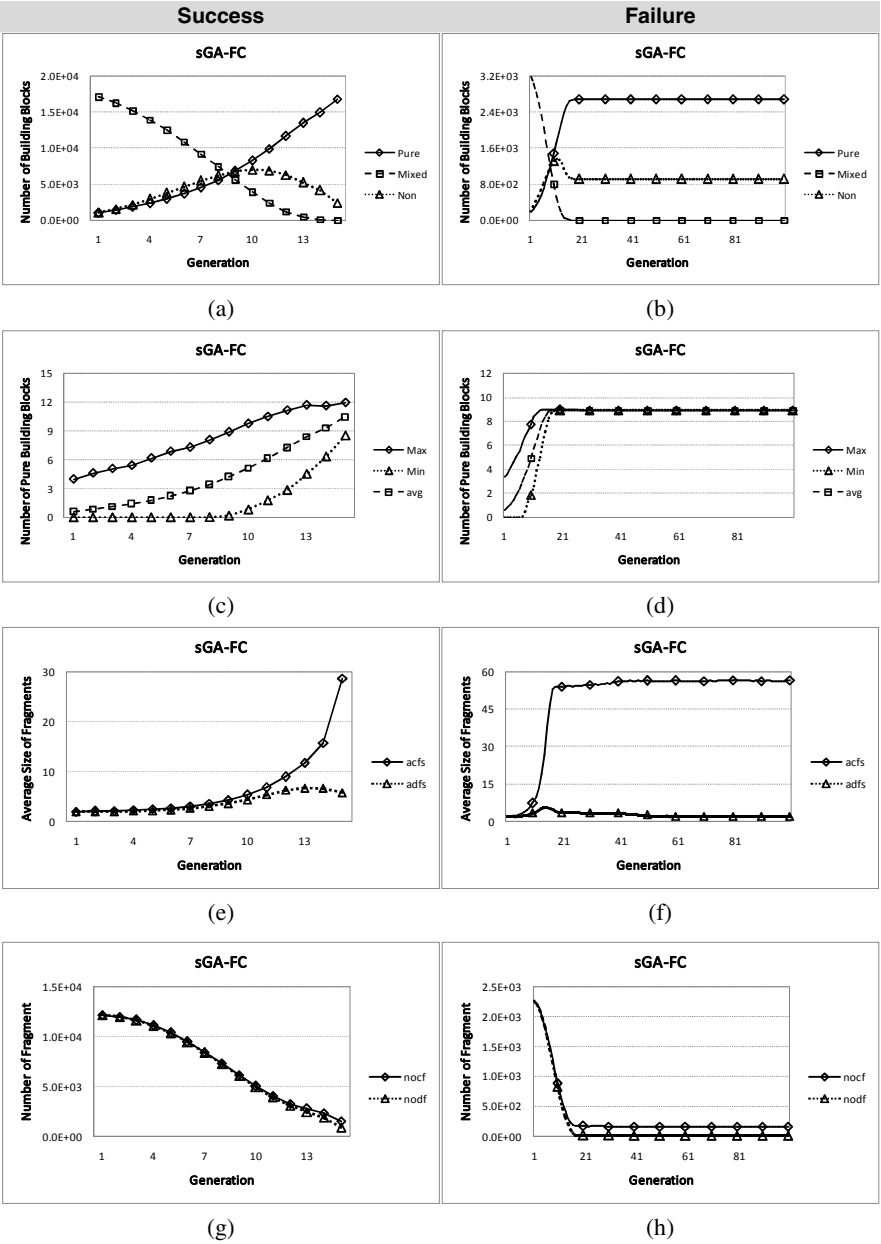



**Fig. 9** The sGA-FC results of the Trap-5 60-bit: (a)(b) the number of the BBs in each generation, (c)(d) the number of ideal BBs (Pure BBs) in each generation, (e)(f) the average size of Fragments in each generation and (g)(h) the number of Fragments in each generation. Note: acfs denotes the average common fragment size, adfs denotes the uncommon fragment size, nocf denotes the number of common fragment and nodf denotes the number of uncommon fragment

In Table 2-3, the benchmark performance results show that the sGA-FC performs better than most of the competing algorithms in the ADFs problems (except Royal Road 64-bit). However in the HDFs problems, sGA-FC is better in HIFF function but is worse in the hTrap-1 function. This indicates that some extra knowledge about the linkage is required to solve the HDFs problems efficiently.

## 5 Discussion and Conclusions

Normally, we do not know the real BBs structure but in the experiment to study the BBs processing the BBs structure are assumed to exist and are known. The results from the experiments show the behavior of BBs processing clearly. These results give an insight into the process of GAs. However for a general problem, it is not clear how to show the existence of BBs or their development. From the schema theorem, the BBs are observed from generation to generation, so the statistics about their size and their structure are not reliable and are not exactly like the ideal BBs in the experiment. However the real BBs can be regarded as substructures or subsequences of good chromosomes that have potential to be a part of the desired solution. Fragment is a drawing of a simple and basic substructure that composes of contiguous fixed bits of a schema. This can represent the real BBs as a part of the optimal solution. When there are large numbers of Fragments in the population, there is a good chance to compose them to form better solutions.

In the experiment, Fragments are defined as common substructures between selected chromosomes. The common Fragments are considered as a part of the Pure BBs. The uncommon Fragments can be considered as the Mixed BBs and the Non BBs, which act as source of diversity. In both the Royal Road and the Trap-5 functions, the size of Fragments grows continuously but at the different rate. In the success case, the Fragment size gradually grows in the early generations and then dramatically increases. In early generations, BBs are ambiguous and difficult to identify because there is not enough information and there are too many Fragments in the recombination process. This can be noticed from the number of Mixed BBs that is maximum at beginning and then decreases continuously. Later, when there is more information and the variety of Fragments becomes lower, it is not difficult to distinguish BBs from useless or deceptive structures. This happens after the cross time of Pure BBs and Mixed BBs. In the Royal Road function, Mixed BBs turn into Pure BBs only but in the Trap-5 function, Mixed BBs become Pure BBs and Non BBs due to deceptive bias. In case of the failure, both Royal Road and Trap-5 function indicate that if there is enough population or other source of diversity, it can reach the optimal solution. Although there is not enough diversity to find the solution, the both sGA and sGA-FC can distinguish the Pure BBs from the Mixed BBs and Non BBs.

This observation implies that if the best solution is not required, the genetic algorithms are capable of finding the better solutions.

In Table 3, the performance of the proposed method is superior to the competing algorithms in the ADFs problems. This indicates that Fragment Crossover is suitable for the ADFs problems which each BB is independent from the others. The performance of sGA-FC is lower than sGA only in Royal Road 64 bits problems but not much. Overall, it is much better in both 128 bits and 256 bits problems because sGA-FC can maintain diversity better than sGA. Generally in a low order or small size problem, more diversity maintenance means slower convergence due to more variation. But in a higher order or larger size problem, the search space will grow exponentially so the exploration power is needed to find better solutions and to prevent the deceptive biases. This requires capability to maintain diversity. The Fragment Crossover can be regarded as a self adaptive multi-point crossover which the number of cross-point is varied in each crossover operation. This mechanism accelerates the BBs recombination process and simultaneously prevents BBs from the disruption. However in the HDFs problems, the performance of Fragment Crossover is good for non-deceptive problems and is poor for deceptive problems. The BBs relationship in HDFs is more complex than in ADFs problems. Information about linkage is required in order to identify the Fragments for solving HDFs problems effectively.

**Table 2** The population size of sGA and sGA-FC in the success case

| Parameters | | sGA | sGA-FC | Parameters | | sGA | sGA-FC |
|---|---|---|---|---|---|---|---|
| Problems | Size (bits) | #pop. | #pop. | Problems | Size (bits) | #pop. | #pop. |
| OneMax | 100 | 1,300 | 300 | Trap-5 | 100 | 4,500 | 2,500 |
| | 150 | 2,800 | 400 | | 150 | 13,000 | 4,500 |
| | 200 | 4,500 | 500 | | 200 | 28,000 | 7,000 |
| | 250 | 8,500 | 500 | | 250 | N/A | 10,500 |
| Royal Road | 64 | 1,200 | 1,200 | HIFF | 32 | 900 | 500 |
| | 128 | 3,000 | 1,600 | | 64 | 4,000 | 1,000 |
| | 256 | 12,000 | 2,500 | | 128 | 40,000 | 2,800 |
| Dec3 | 60 | 1,500 | 1,000 | | 256 | N/A | 8,500 |
| | 120 | 9,500 | 2,200 | Htrap-1 | 27 | 2,400 | 1,500 |
| | 180 | 20,000 | 3,000 | | 81 | N/A | N/A |
| | 240 | N/A | 4,000 | | 243 | N/A | N/A |
| Exp-Dec3 | 60 | 7,500 | 2,700 | | | | |
| | 90 | 18,000 | 3,500 | | | | |
| | 120 | N/A | 5,500 | | | | |

Note: #pop denotes the population size and N/A denotes that the data is not available because it cannot find the optimal solution under the limited population size (#pop $\leq$ 50,000).

**Table 3** Experimental parameters setting and results of the benchmark problem set

| Parameters | | sGA | CSM | BOA | hBOA | sGA-FC |
|---|---|---|---|---|---|---|
| Problems | Size (bits) | #FE | #FE | #FE | #FE | #FE |
| OneMax | 100 | 22,200 | 14,000 | 5,100 | - | **3,800** |
| | 150 | 64,300 | 32,500 | 8,300 | - | **6,600** |
| | 200 | 128,300 | 60,000 | 12,500 | - | **9,800** |
| | 250 | 271,200 | 80,000 | 15,700 | - | **11,500** |
| Royal Road | 64 | **11,900** | - | - | - | 13,500 |
| | 128 | 49,500 | - | - | - | **29,900** |
| | 256 | 304,400 | - | - | - | **75,200** |
| Dec3 | 60 | 25,600 | - | 27,000 | - | **14,100** |
| | 120 | 217,200 | - | 80,000 | - | **51,100** |
| | 180 | 626,000 | - | 180,000 | - | **92,100** |
| | 240 | N/A | - | 235,000 | - | **147,000** |
| Exp-Dec3 | 60 | 181,500 | - | 165,000 | - | **55,300** |
| | 90 | 732,000 | - | 480,000 | - | **112,600** |
| | 120 | N/A | - | 1,000,000 | - | **247,500** |
| Trap-5 | 100 | 83,250 | 65,000 | 99,000 | - | **47,900** |
| | 150 | 305,500 | 165,000 | 220,000 | - | **114,000** |
| | 200 | 784,900 | 310,000 | 320,000 | - | **215,600** |
| | 250 | N/A | 750,000 | 490,000 | - | **375,900** |
| HIFF | 32 | 4,800 | 3,300 | - | **2,100** | 2,600 |
| | 64 | 38,800 | 14,500 | - | **7,800** | 11,800 |
| | 128 | 584,000 | 51,000 | - | **27,000** | 45,900 |
| | 256 | N/A | 370,000 | - | **90,000** | 222,700 |
| Htrap-1 | 27 | 11,400 | **3,000** | - | 3,400 | 8,000 |
| | 81 | N/A | 35,000 | - | **30,000** | N/A |
| | 243 | N/A | 310,000 | - | **225,000** | N/A |

Note: #FEs denotes the number of function evaluations and N/A denotes that the data is not available because it cannot find the optimal solution under the limited number of function evaluation (#FE ≤ 1,000,000). The bold face indicates the best value.

In summary, the good substructures can be interpreted as common subsequences between two highly fit chromosomes called Fragments. These Fragments can be regarded as the BBs because they are short, low-order and come from the highly-fit chromosomes. Therefore the building block can be identified explicitly. The results from the experiments indicate that there are good substructures existed in good individuals and Fragment Crossover can identify and recombine them to create better solutions. This shows the existence of the BBs and supports the understanding of the mechanism in evolutionary process based on the BBH. The proposed idea is simple to implement and it is easy to tune as it introduces no new

parameter. It is also effective in solving ADFs problems. To apply this method to real world problems, any additional information can be integrated into the algorithm. For example, the mutation operator is not used in the experiment but there is no reason to prohibit it. The Fragment identification and composition can also be modified to include extra knowledge in the problem domain.

## References


1. Holland, J.H.: Adaptation in natural and artificial systems. University of Michigan Press, Michigan (1975)
2. Goldberg, D.E.: Genetic algorithms in search optimization and machine learning. Addison-Wesley, Reading (1989)
3. Stephens, C.R., Waelbroeck, H., Aguirre, R.: Schemata as building blocks: does size matter? Found Genet Algor, 117–133 (1999)
4. Stephens, C.R., Waelbroeck, H.: Schemata evolution and building blocks. IEEE Con. Evolut. Comput., 109–124 (1999)
5. Mitchell, M., Forrest, S., Holland, J.H.: The royal road for genetic algorithms: fitness landscapes and GA performance. In: Proc. Conf. Artif. Life, pp. 245–254 (1992)
6. Watson, R.A., Jansen, T.: A building block royal road where crossover is provably essential. In: Proc. Genet. and Evolut. Comput. Conf., pp. 1452–1459 (2007)
7. Stephanie, F., Mitchell, M.: What makes a problem hard for a genetic algorithm? Some anomalous results and their explanation. Mach Learn., 285–319 (1993)
8. Grefenstette, J.J.: Deception considered harmful. Found Genet. Algor., 75–91 (1993)
9. Aporntewan, C., Chongstitvatana, P.: A quantitative approach for validating the building-block hypothesis. Congr. Evolut. Comput., 1403–1409 (2005)
10. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic Algorithms, noise, and the sizing of populations. Applica. Artif. Intel. Eng., 3–16 (1991)
11. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic algorithms, noise, and the sizing of populations. J. Complex Syst., 333–362 (1992)
12. Goldberg, D.E.: The design of innovation: Lessons from and for competent genetic algorithms. Kluwer, Boston (2002)
13. Chen, Y.P., Yu, T.L., Sastry, K., et al.: A survey of linkage learning techniques in genetic and evolutionary algorithms (2007), `http://www.illigal.uiuc.edu/web/technical-reports/2007/03/12/a-survey-of-linkage-learning-techniques-in-genetic-and-evolutionary-algorithms/` (Accessed March 12, 2007)
14. Goldberg, D.E., Korb, B., Deb, K.: Messy genetic algorithms: Motivation, analysis, and first results. Complex Syst., 493–530 (1989)
15. Goldberg, D.E., Deb, K., Kargupta, H., et al.: Rapid, accurate optimization of dif-ficult problems using fast messy genetic algorithms. In: Proc. Conf. Genet. Algor., pp. 56–64 (1993)
16. Kargupta, H.: The gene expression messy genetic algorithm. In: IEEE Int. Proc. Conf. Evolut. Comput., pp. 814–819 (1996)
17. Harik, G.R., Goldberg, D.E.: Learning linkage. Found Genet. Algor., 247–262 (1996)

18. Harik, G.R.: Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms (1997), http://www.illigal.uiuc.edu/web/technical-reports/1997/01/24/learning-linkage-to-efficiently-solve-problems-of-bounded-difficulty-using-genetic-algorithms-135pp/ (Accessed January 24, 1997)
19. Harik, G.R., Goldberg, D.E.: Learning linkage through probabilistic expression. Comput. Method Appl. Mech. Eng., 295–310 (2000)
20. Larrañaga, P., Lozano, J.A.: Estimation of distribution algorithms: A new tool for evolutionary computation. Kluwer, Boston (2001)
21. Pelikan, M., Sastry, K., Cantú-Paz, E.: Scalable optimization via probabilistic modeling. Springer, Heidelberg (2006)
22. Goldberg, D.E., Deb, K., Thierens, D.: Toward a better understanding of mixing in genetic algorithms. J. Societ Instrum. Control Eng., 10–16 (1993)
23. Thierens, D.: Analysis and design of genetic algorithms. Doctoral dissertation, Katholieke Universiteit Leuven, Leuven, Belgium (1995)
24. DeJong, K.A., Spears, W.M.: An analysis of the interacting roles of population size and crossover in genetic algorithms. Proc. Parallel Probl. Solving, 38–47 (1991)
25. DeJong, K.A., Spears, W.: A formal analysis of the role of multi-point cross over in genetic algorithms. Ann. Math. Artif. Intel., 1–26 (1992)
26. Lin, G., Yao, X.: Analysing crossover operators by search step size. In: IEEE Int. Conf. Evolut. Comput., pp. 107–110 (1997)
27. Falkenauer, E.: The worth of the uniform [uniform crossover]. Congr. Evolut. Comput., 776–782 (1999)
28. Kinner, S., Riddle, P.J.: Expected rates of building block discovery, retention and combination under 1-point and uniform crossover. In: Int. Conf. Parallel Probl. Solving Nat., pp. 121–130 (2004)
29. Aporntewan, C., Chongstitvatana, P.: Chi-Square matrix: an approach for building-block identification. In: Asian Comput. Sci. Conf., pp. 63–77 (2004)
30. Pelikan, M., Goldberg, D.E., Cantu-Paz, E.: BOA: The Bayesian optimization algorithm. In: Proc. Genet. Evolut. Comput. Conf., pp. 525–532 (1999)
31. Pelikan, M.: Bayesian optimization algorithm: From single level to hierarchy (2002), http://www.illigal.uiuc.edu/web/technical-reports/2002/04/20/bayesian-optimization-algorithm-from-single-level-to-hierarchy/ (Accessed April 20, 2002)
32. Rudnick, W.M.: Genetic algorithms and fitness variance with an application to the automated design of artificial neural networks. Doctoral Dissertation Oregon Graduate Institute of Science & Technology, Beaverton, USA (1992)
33. Watson, R.A., Hornby, G.S., Pollack, J.B.: Modeling building block interdependency. In: Proc. Parallel Probl. Solving Nat., pp. 97–106 (1998)
34. Watson, R.A., Pollack, J.B.: Hierarchically consistent test problems for genetic algorithms. In: Proc. Congr. Evolut. Comput., pp. 292–297 (1999)

# Structure Learning and Optimisation in a Markov Network Based Estimation of Distribution Algorithm

Alexander E.I. Brownlee, John A.W. McCall, Siddhartha K. Shakya, and Qingfu Zhang

**Abstract.** Linkage learning has been a focus of research interest since the early days of evolutionary computation. There is a strong connection between linkage learning and the concept of structure learning, which is a crucial component of a multivariate Estimation of Distribution Algorithm. Structure learning determines the interactions between variables in the probabilistic model of an EDA, based on analysis of the fitness function or a population. In this chapter we apply three different approaches to structure learning in an EDA based on Markov networks and use measures from the information retrieval community (precision, recall and the F-measure) to assess the quality of the structures learned. We present observations and analysis of the impact that structure learning has on optimisation performance and fitness modelling.

## 1 Background

The concept of linkage learning has been a focus of research interest since the early days of evolutionary computation. Linkage is defined in relation to the biological notion of epistasis: informally the effect on fitness of any one "gene" is related in a complex way to the values of other "genes". A strongly-related concept is that of fitness landscape: the geometric representation of the distribution of fitness over the solution space. So linkage can be thought of as a relationship between highly fit regions in the fitness landscape and the particular joint configurations of solution variables that locate those regions.

Alexander E.I. Brownlee · John A.W. McCall
IDEAS Research Institute, Robert Gordon University, Aberdeen, UK
e-mail: {sb,jm}@comp.rgu.ac.uk

Siddhartha K. Shakya
Intelligent Systems Research Centre, BT Group, Ipswich, UK
e-mail: sid.shakya@bt.com

Qingfu Zhang
Department of Computer Science, University of Essex, Colchester, Essex, UK
e-mail: qzhang@essex.ac.uk

Y.-p. Chen (Ed.): Exploitation of Linkage Learning, ALO 3, pp. 45–69.
springerlink.com 

A broad strand of effort in evolutionary algorithms (EAs) is aimed at optimisation. EAs may be compared empirically in terms of their performance in optimising a variety of benchmark problems. An EA maintains a population of points which follow a trajectory over the fitness landscape as the evolution progresses. The trajectory is a function of the operators used by the evolutionary algorithm. Optimisation performance may be generally measured in terms of the efficiency with which the trajectory of a population evolves from a random start to concentration in highly fit regions of the solution space.

The relationship between linkage and optimisation performance has lain at the heart of much theoretical research in EAs over the years. Essentially this relationship is determined by the interaction between the linkage and the particular selection and recombination operators used by the algorithm. If the operators of the evolutionary algorithm are well-suited to producing those variable value combinations typical of highly fit solutions, the selection and recombination process will quickly move the population into highly fit regions and optimisation performance will be high. The algorithm can be said to have "detected" or "discovered" the linkage. Conversely if the operators are not so well-suited, the evolution will struggle or fail to concentrate the population in highly fit regions and performance will deteriorate.

It is important to mention at this point that the choice of solution encoding and definition of the fitness function are also highly influential on the success or failure of a particular algorithm. Indeed representation theory is a whole strand of research in itself [1]. We assume however for the purposes of this chapter that the substantial body of literature on this subject has been used to good effect and that we have sensible choices of solution representation and a fitness function that embodies all knowledge of the problem in a meaningful way. Therefore we concentrate on the discovery of the linkage in the problem.

An early and influential approach to understanding the relationship of linkage to Genetic Algorithm (GA) performance was the Schema Theorem [2]. In terms of linkage discovery, the Schema Theorem can be interpreted as a relationship between particular subsets of fixed solution variable values and high fitness. This view was encapsulated in the Building Block Hypothesis which theorised that GA performance could be understood in terms of the selection and recombination of high fitness schemata with small subsets of defined solution variables closely positioned in the solution encoding.

In [3], the authors identified the phenomenon of hitchhiking in Royal Road GA. In hitchhiking, undesirable combinations of solution variables that were present in solutions containing building blocks became fixed in the population through being selected alongside the building blocks. This revealed some subtleties of the interaction between linkage and evolutionary operators that the schema approach cannot adequately address because the fitness of any particular solution cannot be ascribed uniquely to any particular schema of which the solution is a member.

Another approach to linkage detection is to explicitly model the effect that operators have on the distribution of fitness. In [4], Greffenstette attempted to predict the performance of a GA given a representation, a fitness landscape and a set of

genetic operators. More precisely, performance was measured as the change in mean population fitness over the course of a GA run.

A key element of Greffenstette's approach is to construct models that predict how a particular recombination operator will affect the distribution of fitness. Greffenstette was able to demonstrate that reasonable predictive models for population fitness over the course of an evolution could be derived for some problem cases.

In [5], Mühlenbein and Schlierkamp-Voosen presented the Breeder Genetic Algorithm (BGA). Here the focus was on designing an algorithm with known statistical properties. The key idea here was to analyse the specific effects on fitness of particular operators with a particular encoding and using this to predict GA performance. This approach made no use of schemata or any other explicit reference to linkage. Instead, strong assumptions were made throughout about normal distribution of fitness under application of operators.

In [6], Vose developed a Markov Chain approach to analysing the evolution of a GA. Essentially, the evolution step of a simple GA was analysed as a stochastic chain of transitions between the set of all possible populations (of a given size M or of infinite size). This work led to powerful insight into GA performance and convergence properties. However computing the transition matrix for any particular problem is impractical due to its combinatorially huge size.

These developments set the stage for the emergence of Estimation of Distribution Algorithms (EDA). In an EDA, the traditional genetic operators of crossover and mutation are replaced by a probabilistic model that generates a successor population of solutions. Selection is used to provide a set of relatively high quality solutions from which to build a model. The aim then is to develop a model-based sampling operator which distributes solutions in a high fitness region of the solution space.

Early EDAs such as PBIL [7] and UMDA [8] constructed probabilistic models based on the marginal distributions of solution variable values present in selected solutions. As the field developed, more complex models based on the joint distribution of solution variable values were proposed. Much of this work uses the theory of probabilistic graphical models [9] which relates a joint probability distribution (JPD) to a graph of solution variables and their interactions. Inevitably within the EDA community, linkage learning has taken on the precise meaning of determining the structure of a probabilistic graphical model from a population of solutions.

Therefore EDAs can be seen as inheriting from a number of key strands of GA theory: the concept of linkage and its relation to genetic operators; the idea of modelling operator fitness distribution and the abstraction of an EA as a sequence of stochastic evolutionary steps.

The remaining sections of this chapter are arranged as follows. In Section 2, we introduce notation to describe the probabilistic models used in EDA with a particular focus on Markov Network Models. Section 3 describes the main approaches we will apply to structure learning in the Markov Network EDA, DEUM and introduces measures of the quality of structure learning. Section 4 describes the Distribution Estimation Using Markov networks (DEUM) framework and describes how

structure learning is applied in this approach. Section 5 describes the Ising problem and attempts to solve it using EDAs. Sections 6-8 describe experiments on the Ising problem with DEUM using three different structure learning approaches. Results are provided on the optimisation performance and the quality of structure learning for each approach. The chapter concludes in Section 9.

## 2 EDAs and Approaches to Probabilistic Modelling

We begin with some generalities on EDAs and probabilistic models. An EDA regards a solution $x = x_1 \ldots x_n$ as composed of a set of values $x_i$ taken by a set of random variables, $X = X_i$. The aim of an EDA is to estimate a joint probability distribution (JPD), denoted $P(X) = P(X_1, \ldots, X_n)$, in such a way that high fitness solutions may be sampled from this distribution with high probability. An EDA has the following general structure:

1. Initialise a population of N solutions
2. While stopping criteria are not satisfied
    2.1 Select a subset of $M < N$ solutions from P
    2.2 Build a probabilistic model P(X) from the selected solutions
    2.3 Sample P(X) to generate N new solutions to replace P
3. Return best solution found

EDAs begin by initializing a population of solutions, P, usually with uniform probability. Then a subset of solutions is selected from P, typically by truncation or tournament selection. This subset is then used to estimate a probabilistic model of the JPD, P(X). P(X) is then sampled to generate the next population. It is an assumption of EDA that, since P(X) was estimated from relatively high fitness solutions, sampling P(X) will result in an improved distribution of solution fitness in succeeding populations. The process iterates until stopping criteria are satisfied and the best solution found is returned.

The performance of an EDA heavily depends on how successfully it estimates P(X). In general, the computation of P(X) for a bitstring variable encoding, $x \in \{0,1\}$ , involves the computation of probabilities for all $2^n$ configurations of x. This is not computationally feasible in most problems of interest. However, in many cases, a good approximation to P(X) can be obtained by factorising the distribution in terms of marginal and conditional probabilities of variables, thus reducing the costs of distribution estimation and sampling. An excellent review of this can be found in [10].

The concept of a Probabilistic Graphical Model (PGM) provides an efficient and effective tool to represent the factorisation of the JPD, and therefore has an important role in EDAs. Any PGM has a graph component that represents the dependency structure between variables and a parameter component that can be used to compute P(X). Full information on PGMs can be obtained from [11].

The most commonly used PGM in the EDA community is the Bayesian Network (BN). Formally, a BN is a pair $(D,\Theta)$, where D is a Directed Acyclic Graph (DAG) and $\Theta$ is a set of conditional probabilities for each variable $X_i$ conditioned

on its parents $\pi(X_i)$ in D. The graph is realised algebraically as a factorisation of the JPD in terms of these conditional probabilities (1).

$$P(X) = \prod_{i=1}^{n} P(X_i \mid \pi(X_i)) \tag{1}$$

We now present an alternative PGM and describe how it can be used in an EDA. A Markov Network (MN) is a pair $(G,\Psi)$, where G is the structure and $\Psi$ is the parameter set of the network. G is an undirected graph where each node corresponds to a solution variable and each edge corresponds to a joint dependency between variables. We use $N = \{N_1, \ldots, N_n\}$ to denote the neighbourhood system derived from G , where each $N_i$ is the set of nodes adjacent to node $X_i$ in G [12], [13]. The JPD, for a Markov Network must satisfy the following conditions for each solution x and for each random solution variable $X_i$:

$$0 < p(x) < 1 \tag{2.1}$$

$$\sum_{x} p(x) = 1 \tag{2.2}$$

$$p(X_i \mid X \setminus \{X_i\}) = p(X_i \mid N_i) \tag{2.3}$$

Condition (2.3), known as the Markovianity property, states that the conditional probability distribution of any variable is completely determined by the values of its neighbours. The Hammersley-Clifford Theorem ([14], [13]), states that the JPD of a Markov Network can always be factorised as a Gibbs Distribution. The general form of this JPD is:

$$p(x) = \frac{e^{-U(x)/T}}{\sum_{y} e^{-U(y)/T}} \tag{3}$$

This will be familiar to many readers as a Boltzmann distribution with temperature coefficient T. The function U(x) is called the energy function and has a particular form that is determined by the neighbourhood structure of the Markov Network. We now introduce some terminology that is useful in describing the JPD in a Markov Network.

A clique $K_{i_1,\ldots,i_k} = \{X_{i_1},\ldots,X_{i_k}\}$ is a set of nodes in G that are mutual neighbours. The order of a clique is the number of nodes that belong to it. Formally, we also regard each singleton $\{X_i\}$ as a clique of order 1 and the empty set, $\varnothing$, as the unique clique of order 0.

We define a set of clique potential functions $V_K(x)$ for each clique K and each solution x as follows:

$$V_{\varnothing}(x) = 1 \quad \forall x \tag{4.1}$$

$$V_{\{X_i\}}(x) = \begin{cases} 1 & \text{if } \mathrm{x_i} = 1 \\ -1 & \text{if } \mathrm{x_i} = 0 \end{cases} \tag{4.2}$$

$$V_K(x) = \prod_{X_i \in K} V_{\{X_i\}}(x) \tag{4.3}$$

It is a consequence of the Hammersley-Clifford Theorem that the energy function can be written as a sum of clique potential functions as follows:

$$U(x) = \sum_K \alpha_K V_K(x) \tag{5}$$

where the $\alpha_K$ are real-valued coefficients known as Markov Network parameters.

We refer to (5) as the General Markov Network Model and make the following observations. Substitution of (5) into (3) gives a factorization of the JPD in terms of exponents of the individual summands of (5). The clique potential functions are determined by the structure, G, of the Markov Network. The summation in (5) is over all cliques K present in G. The set of real-valued parameters $\alpha_K$ therefore completely determines the JPD for a given structure G.

## 3 Structure Learning in the DEUM Markov Network EDA

Markov networks have been proposed as the probabilistic model for a number of different EDAs [15], [16], [17], [18]. In [15],[16],[17] the variable interaction (independence) graph is learned from data using a statistical independence test. The structure is then refined to reduce its density and maximal cliques are found. Finally, a junction graph is learned in the case of MN-FDA [15],[16] or a Kikuchi Approximation is learned by MN-EDA [17] to approximate the distribution. In [19] an algorithm is proposed which uses the Linkage Detection Algorithm [20] to discover interactions when building a Boltzmann distribution of the fitness function.

In this section we describe the main approaches we will apply to structure learning in the Markov Network EDA, DEUM and introduce measures of the quality of structure learning. The fitness modelling approach of DEUM allows us to make observations as to the quality of the structure learned and its effect on the fitness modelling capability of the resulting model which should be of interest to others using undirected graphical models and the wider EDA community.

We investigate the impact of three structure learning approaches on the fitness modelling and optimisation capability of EDAs that use the DEUM framework [18]. This framework is fully described in Section 4 but a key feature is the expression of P(X) as a model of fitness. This will allow us to analyse the relationship between the quality of the structure learned and its effect on the fitness modelling capability of the resulting model, thus explicitly relating structure learning to optimisation.

We achieve this by extending the DEUM framework to include a structure learning step rather than relying on the existing knowledge of the problem structure. We use measures borrowed from the information retrieval community to assess the structures learned: specifically *precision* and *recall* combined in the *F-measure* [21]. We compare these with the fitness modelling and optimisation capabilities of the resulting models. This allows us to make observations on the structures learned by each algorithm which will be of relevance for other algorithms using an undirected structure.

The three different versions of DEUM with structure learning are:

- DEUM-LDA, a single-generation algorithm which incorporates the Linkage Detection Algorithm of Heckendorn and Wright [20]
- $\text{DEUM}_{\chi 2}$, a single-generation algorithm which incorporates an independence test structure learner using Pearson's Chi-Square statistics
- ev$\text{DEUM}_{\chi 2}$, a multi-generation algorithm variant of $\text{DEUM}_{\chi 2}$

The results we present here relate to the algorithm's performance on optimising the Ising spin glass problem. This has previously been used as a benchmark problem for EDAs [22], [23],[24], [15], [16], [17], [18] due to interesting properties such as symmetry and a large number of plateaus. The problem exhibits an undirected network of interactions between variables and consequently EDAs using Markov networks are naturally suited to it and should perform well. In [18] it was shown that by supplying the underlying lattice structure to the algorithm DEUM was able to efficiently optimise the 2D Ising problem.

The structure of the Markov Fitness Model in our previous work has always been fixed and supplied prior to running the algorithm. In earlier work it was univariate - the model containing only one term for each variable in the problem. In [18], [25] the model was extended to incorporate terms representing bivariate and trivariate interactions among variables. The algorithm used in this work incorporates an additional step during which the structure is learned.

## *3.1 How Good Is the Structure?*

Other work has been done to analyse structures learned in EDAs. The study of the Learning Factorized Distribution Algorithm (LFDA) carried out in [26] included an analysis of structure learning capability. A discussion of the importance of higher-order structure in EDAs is presented in [27]. It is known that not all interactions which are present in a problem will necessarily be required in the model for the algorithm to rank individuals by fitness and find a global optimum. This concept is similar to the idea of benign and malign interactions [28] and is related to the concept of unnecessary interactions [29]. This is in addition to the idea of spurious correlations [30], [31] which are false relationships in the model resulting from selection.

Here we build on these concepts by comparing the structure learned by the algorithm to what we call the *perfect model structure*. The perfect model structure includes exactly those interactions which are present in the underlying fitness function. This does not include all possible interactions but does include those

which influence the absolute fitness value. In the example version of the Ising problem shown in Figure 1, these are:

$x_1x_2$, $x_2x_3$, $x_3x_4$, $x_1x_4$, $x_5x_6$, $x_6x_7$, $x_7x_8$, $x_5x_8$, $x_9x_{10}$, $x_{10}x_{11}$, $x_{11}x_{12}$, $x_9x_{12}$, $x_{13}x_{14}$, $x_{14}x_{15}$, $x_{15}x_{16}$, $x_{13}x_{16}$, $x_1x_5$, $x_2x_6$, $x_3x_7$, $x_4x_8$, $x_5x_9$, $x_6x_{10}$, $x_7x_{11}$, $x_8x_{12}$, $x_9x_{13}$, $x_{10}x_{14}$, $x_{11}x_{15}$, $x_{12}x_{16}$, $x_1x_{13}$, $x_2x_{14}$, $x_3x_{15}$, $x_4x_{16}$.

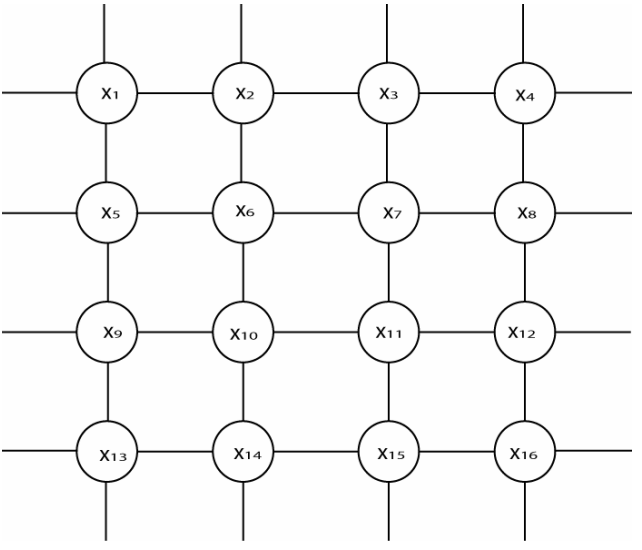



**Fig. 1** 16 bit 2D Ising Lattice

These interactions are required to perfectly fit the model to the fitness function. In general of course it is only possible to identify the perfect structure in this way for predefined test problems where the interactions are explicitly known. In the case of onemax there are no interactions. For the 2D Ising problem, an interaction exists wherever there is a coupling between two spin variables. In the example above, for a 16 bit 2D Ising problem the model will have 49 parameters in total including the univariate parameters and the constant.

In assessing the structures learned by the algorithm we use two measures from the information retrieval community: Precision (p) and Recall (r) [21]. These are defined in (6) and (7).

$$\text{precision} = \frac{\text{True interactions found}}{\text{Total interactions found}} \tag{6}$$

$$\text{recall} = \frac{\text{True interactions found}}{\text{Total true interactions present}} \tag{7}$$

That is, precision measures how much of the learned structure comprises correctly identified interactions and recall measures how many of the interactions present in

the problem have been found. Both are proportions ranging from 0 to 1, with 1 being the best (if both measures are 1 then the learned structure perfectly matches the true structure). These can be combined to form the F-measure (8) [21].

$$F = \frac{2pr}{p+r} \tag{8}$$

This particular definition is sometimes referred to as the $F_1$ measure in which p and r are equally weighted. The F-measure ranges from 0 to 1, with 1 achieved if p and r are both 1.

## 4 Distribution Estimation Using Markov Networks

First we will describe the general framework of the DEUM algorithm, before going on to show the different approaches taken to incorporate a structure learning step.

### *4.1 General Model*

Previous publications on DEUM [18], [25] have described how the Markov network is used to model the distribution of energy across the set of variables in a problem. Energy follows a negative log relationship with fitness, so the Markov network can be used as a model of the fitness function which we call the Markov Fitness Model (MFM). From (5), this relationship may be written in general as (9).

$$-\ln f(x) = U(x) = \sum_K \alpha_K V_K(x) \tag{9}$$

For each individual in the population, $x = x_1 \ldots x_n$, and given a particular neighbourhood structure on the random variables $X_i$, we can derive from (9) an equation linking the fitness, f(x) of x to the values of the $x_i$. (10).

$$-\ln f(x) = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n + \alpha_{1,2} x_1 x_2 + \ldots \tag{10}$$

Here, in a slight abuse of notation, we will use (unordered) juxtapositions of the variable symbols $x_i$ in a clique K in place of the clique potential functions $V_K(x)$ for simplicity of presentation. The right-hand side of (10) can always be evaluated accurately by interpreting the 0 and 1 bit values as -1 and +1 respectively in accordance with (4.2) and (4.3). In (9) each term represents the energy contribution due to a clique on the Markov network. We have presented terms representing 1-cliques such as $\alpha_1 x_1$ and 2-cliques such as $\alpha_{1,2} x_1 x_2$. Additional terms relating to higher order cliques will in general also be present (for example see [25] where 3-cliques form part of the structure for 3-CNF MAX-SAT problems). However, for this study we restrict the model to 2-cliques to match the level of complexity of the 2D Ising problem. As we vary the number of spins, n, of the Ising lattice, The number of correct structure elements grows as $O(n^2)$. With the restriction noted

above, the search space of possible structures has size $2^{|K|}$ where |K| is the size of the set of all possible cliques of order 0 to 2. As we vary the number of spins, n, of the Ising lattice, |K| itself grows as $O(n^2)$.

A system of equations can be formed by substituting into (10) the variable values and fitness for each individual within a population. Then, a least squares approach may be used to estimate values for the parameters $\alpha$. The neighbourhood structure and the set of parameters completely define the MFM. This can then be sampled to generate new individuals. Previously reported results for optimisation using DEUM have been found using the above techniques in the same broad workflow:

1. Generate random population
2. Repeat until termination criteria met:
   2.1 Compute model parameters
   2.2 Sample model to generate new population

A number of different techniques have been used to sample the MFM. In [32] the MFM was used to update a probability vector similar to that used in PBIL [7]. [33] described a technique to directly sample the marginal probabilities from a univariate MFM. [34], [18] employed a zero-temperature Metropolis method to sample the MFM and finally [18], [25] used variations of a Gibbs sampler. It was found empirically that the Gibbs sampler approach produced the best results for higher complexity problems and consequently it is only the Gibbs sampler which is used here - specifically the version used when optimising the Ising problem in [18]. Depending on the problem and algorithm parameters we found that this approach would often find a global optimum in the first generation. This leads us to the single-step framework in Sections 6 and 7.

## *4.2 Fitness Prediction Correlation*

In our results we also report the fitness prediction correlation $C_r$ for each model. This measure is described in detail in [25] and [35]. In contrast to precision, recall and F measure, fitness prediction correlation is defined specifically in terms of the MFM approach so cannot be used to compare directly to other approaches. However fitness prediction correlation gives a helpful indication of the fitness information being learned by the model. Section 4 showed that DEUM models the fitness function directly (9). Therefore, in addition to sampling the model to find an optimum we can also use it to predict the fitness of individuals. This ability can be exploited to measure how closely the MFM models the fitness function. This in turn acts as a predictor for the optimisation capability of the algorithm.

Fitness prediction correlation is defined as the statistical correlation between the true fitness and the model-predicted fitness of a population of randomly generated individuals. The experiments described here use Spearman's rank correlation [36]. As $C_r$ approaches +1 the model has an increasingly strong positive correlation with the fitness function.

## 5 The Ising Problem and EDAs

The general Ising spin glass problem can be defined as an energy function H(X) over a set of spin variables X = {$X_1$, … , $X_n$} and a set of coupling constants $h_i$ and $J_{ij}$ as

$$H(X) = -\sum_{i \in L} h_i(X_i) - \sum_{i<j \in L} J_{ij} X_i X_j \tag{11}$$

Each coupling constant $h_i$ and $J_{ij}$ relates to a single spin $X_i$ and pair of spins {$X_i$,,$X_j$} respectively. Each spin variable can be either +1 or -1 and L represents a lattice of n spins.

Given a particular pair of values of the coupling constants $h_i$ and $J_{ij}$, the task is to find values for all $X_i$ that minimise the energy H. For the purposes of this chapter, we consider only instances of the problem where each $J_{ij}$ takes values in {-1, +1}, with the spins arranged in a two dimensional lattice of n spins. This is represented graphically in Figure 1.

In [23], [22], [24], it was shown that the hierarchical Bayesian Optimization Algorithm (hBOA) was able to efficiently solve the Ising problem, outperforming other algorithms. In [15], [16] the Ising problem was used as a benchmark for the Markov Network Estimation of Distribution Algorithm (MN-EDA) and the Markov Network Factorized Distribution Algorithm (MN-FDA). The latter of these papers argued that the Kikuchi approximation used to estimate the distribution used by MN-EDA gave an advantage for the 2D Ising problem as it was able to represent the bivariate dependencies as an exact factorisation. Finally [18] demonstrated that this also applied to the DEUM algorithm as it could represent the exact factorisation from the structure in the form of potential functions. DEUM was demonstrated to perform very well - solving the problem with only a single generation. In that case the structure was supplied to the algorithm. By adding a structure learning component, the work presented in this chapter will allow a fairer comparison with other EDAs on the Ising problem. In the experiments which follow, we use the same four instances of Ising at each size as those used in [18], with the results aggregated into a mean for each size.

## 6 DEUM LDA

This approach adds the Linkage Detection Algorithm [20] to the DEUM framework, giving us the following workflow.

1. Run LDA using the fitness function
2. Generate random initial population P, of size 4.4N
3. Select a subset σ, the top 1.1N of P
4. Use σ to build the MFM
5. Calculate fitness prediction correlation, $C_r$
6. Sample new population from MFM using random walk Gibbs sampler

Step 5 is not an integral part of the algorithm. It is only included in the above workflow to show the point at which we calculate the value for $C_r$.

The proportion of the population selected and population size used were determined by earlier experiments described in [35]. There we found that fitness modelling capability of the MFM increases greatly when the number of individuals selected is more than the number of parameters in the model, denoted by N. Once the structure is known, we then set the number to be selected to 1.1N to ensure that this is the case (the system of equations generated from (10) is thus an over-specified system). The population size |P| is then set to be large enough to achieve the desired selection pressure. The results in [35] revealed that fitness modelling capability improves as the selection pressure is increased. In this case we set the proportion of the population selected to 0.25 - that is, the population is 4.4N, four times the number of individuals required to build the MFM. This represents a good tradeoff between a useful selective pressure and an excessively large population.

The cooling rate parameter for the Gibbs sampler was 0.0005, following [18]. The iteration cap of the sampler was increased from 500 to 2000 - this improved the success rate of the algorithm and decreased the total number of iterations required to find the global optimum.

## *6.1 Fitness Model*

First we will look at the model generated by the algorithm for each problem size. In Table 1 we see the mean precision p and recall r for the structures found over 30 runs, with corresponding standard deviations (p-SD and r-SD). This is followed by the F-measure F combining the mean precision and recall. This is shown alongside the mean fitness prediction correlation $C_r$ for each size of the problem with its corresponding standard deviation $C_r$ -SD.

With a precision and recall (and hence F) of 1.0, the structure discovered matches the perfect structure for the problem. As a consequence, we would expect similar optimisation results to those seen in [18].

**Table 1** DEUM-LDA Fitness Model Statistics over 30 runs

| PS | p | p-SD | r | r-SD | F | $C_r$ | $C_r$ -SD |
|---|---|---|---|---|---|---|---|
| 16 | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 | 0.887 | 0.091 |
| 25 | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 | 0.924 | 0.049 |
| 36 | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 | 0.932 | 0.034 |
| 49 | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 | 0.939 | 0.037 |
| 64 | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 | 0.949 | 0.023 |
| 100 | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 | 0.942 | 0.027 |
| 256 | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 | 0.945 | 0.018 |
| 324 | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 | 0.943 | 0.015 |
| 400 | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 | 0.940 | 0.022 |

## 6.2 Optimisation

Table 2 shows the results for optimising using the algorithm.

**Table 2** DEUM-LDA Optimisation Statistics over 30 runs

| PS | $C_r$ | LDA-FE | FE | FE-SD | IT | IT-SD | SR |
|---|---|---|---|---|---|---|---|
| 16 | 0.887 | 480 | 210 | 4 | 163 | 310 | 100 |
| 25 | 0.920 | 1200 | 330 | 1 | 989 | 532 | 100 |
| 36 | 0.932 | 2520 | 483 | 25 | 821 | 1995 | 100 |
| 49 | 0.905 | 4704 | 647 | 3 | 1675 | 1234 | 100 |
| 64 | 0.939 | 8064 | 846 | 1 | 902 | 688 | 100 |
| 100 | 0.944 | 19800 | 1446 | 221 | 11989 | 21296 | 87 |
| 256 | 0.950 | 130560 | 4342 | 862 | 130995 | 118419 | 67 |
| 324 | - | - | - | - | - | - | 0 |
| 400 | - | - | - | - | - | - | 0 |

The $C_r$ value is given to show the fitness prediction power of the model in the context of optimisation. The number of fitness evaluations needed by LDA (LDA-FE) for each problem size is given next. Then the mean number of additional function evaluations (FE) and internal iterations of the Gibbs sampler (IT) are given along with their standard deviations (FE-SD and IT-SD). FE includes the evaluations needed to estimate model parameters and the evaluations needed to confirm the fitness of individuals generated by the Gibbs sampler. All of these values only include the successful runs; the success rate (SR) is given as a percentage of times the algorithm found a known global optimum over 30 independent runs.

The results were unexpectedly poor given the perfect structure learned by LDA. In [18], DEUM was able to optimise 2D Ising when supplied with the perfect structure. We increased the number of individuals selected from the population to estimate the model parameters to 2N (that is, twice the number of parameters in the model). Part of our work on the fitness information content in a population [35] indicated that Ising requires a larger number of individuals than other fitness functions we have looked at to obtain a good model of fitness. The results for population size 2N are given in Table 3. It can be seen that this resulted in a marked improvement of the optimisation capability, even for the largest instances of the problem that we used.

**Table 3** DEUM-LDA Optimisation Statistics with increased population size over 30 runs

| PS | $C_r$ | LDA-FE | FE | FE-SD | IT | IT-SD | SR |
|---|---|---|---|---|---|---|---|
| 100 | 0.993 | 19800 | 2409 | 7 | 8015 | 6825 | 100 |
| 256 | 0.993 | 130560 | 6166 | 48 | 37705 | 82985 | 100 |
| 324 | 0.993 | 209304 | 7786 | 7 | 18947 | 11962 | 80 |
| 400 | 0.992 | 319200 | 9688 | 86 | 160237 | 154439 | 100 |

One problem with this approach is that the linkage detection algorithm requires a large number of fitness evaluations to learn the structure. This could be mitigated by recycling the solutions generated by the LDA run for estimating the model parameters but this would not make a large difference. The number of possible interactions for a particular problem size n is given in (12); given that LDA requires four evaluations for each possible interaction, the number of evaluations required by LDA at a particular problem size n is given in (13). This can be reduced by caching individuals but not by a large amount and this comes with a rapidly growing space complexity.

$$Possible = \frac{n(n-1)}{2} \tag{12}$$

$$Evals = 2n(n-1) \tag{13}$$

In addition to this, without some threshold it will include any interactions which are not useful for optimisation and this will lead to a large and overly complex model which will result in the algorithm being overloaded and potentially resulting in poor performance. An effect similar to this was seen with the noisy chemotherapy problem used for benchmarking hBOA in [37]. The 2D Ising problem is in comparison very "clean" - because of the nature of the problem any interactions which LDA discovers are important for optimisation with the result that the structure found is perfect. This allows DEUM to build a model which closely matches the fitness function. For other problems this may not be possible which provides motivation for the independence test approach.

# 7 DEUM-$\chi^2$

## *7.1 The Algorithm*

The workflow for this algorithm is very similar to that for DEUM incorporating LDA. The only additions are the extra selection step to choose individuals for the Chi-Square structure learning algorithm and the structure refinement step, both taken from [15], [16], [17].

1. Generate random initial population P
2. Select a subset $\sigma_1$, the top 25% of P
3. Run Chi-Square edge detection algorithm to search for statistical dependencies apparent in $\sigma_1$
4. Refine structure
5. Select a subset $\sigma_2$, the top 1.1N of P
6. Use $\sigma_2$ to build MFM
7. Calculate $C_r$

Sample new population from MFM using random walk Gibbs sampler.

The structure is learned by performing a first-order Chi-Square independence test on each possible pairing of variables; an interaction is assumed where the test exceeds a threshold of 0.75. The number of individuals for the structure learning selection was set to the top 25%; the selection step for estimating the MFM parameters selected the top 1.1N individuals. These figures were found to represent a good balance between fitness modelling capability and function evaluations required by a series of experiments following the pattern described in [35]. The structure refinement step is the same as that used in [15], [16], [17]: in these works a limit is imposed on the number of edges (interactions) incident to a node (variable) on the graph. For any node exceeding this limit, the edges with the lowest Chi-Square scores are removed until the limit is reached. For 2D Ising we set this limit to 4 - that is, each variable can have only four neighbours as is the case in the 2D Ising lattice. The clique finding step from those papers is not applied here as we know that the 2D Ising problem has a bivariate structure and we wish to keep the structure at this level of complexity. The population size was set to be the problem size multiplied by 100. As in the previous section, the cooling rate for the Gibbs sampler was 0.0005 and the iteration cap was set to 2000.

## *7.2 Fitness Model*

Again, before looking at optimisation results we will look at the model generated by the algorithm for each problem size. These are shown in Table 4; the column headings are the same is in Table 1.

**Table 4** DEUM-$\chi^2$ Fitness Model Statistics over 30 runs

| PS | p | p-SD | r | r-SD | F | $C_r$ | $C_r$ -SD |
|---|---|---|---|---|---|---|---|
| 16 | 0.785 | 0.183 | 0.979 | 0.028 | 0.988 | 0.997 | 0.01 |
| 25 | 0.749 | 0.213 | 0.968 | 0.029 | 0.980 | 0.992 | 0.015 |
| 36 | 0.730 | 0.166 | 0.970 | 0.022 | 0.982 | 0.994 | 0.008 |
| 49 | 0.754 | 0.101 | 0.963 | 0.019 | 0.976 | 0.990 | 0.012 |
| 64 | 0.741 | 0.107 | 0.958 | 0.018 | 0.974 | 0.990 | 0.008 |
| 100 | 0.695 | 0.116 | 0.948 | 0.018 | 0.966 | 0.985 | 0.010 |
| 256 | 0.589 | 0.092 | 0.914 | 0.009 | 0.940 | 0.968 | 0.007 |
| 324 | 0.584 | 0.083 | 0.908 | 0.011 | 0.935 | 0.964 | 0.008 |
| 400 | 0.529 | 0.085 | 0.894 | 0.011 | 0.924 | 0.956 | 0.007 |

We can see that increasing problem size results in a decrease in both p and r, reflected in a steadily decreasing F. This indicates that with increasing problem size the algorithm finds it more difficult to correctly identify all interactions and also begins to match some false positives. This also results in a corresponding decrease in the fitness prediction power of the model, revealed in the decreasing $C_r$ values.

We can see that these fall off very quickly with a comparatively small decrease in r; this highlights the importance of finding a good model structure. The experiment was rerun with population sizes equal to the number of fitness evaluations required by LDA at each step; the results for this are given in Table 5.

We can see that the structures are considerably better at the larger sizes; for the smaller sizes the number of evaluations required by LDA was actually less than 100 times the problem size as used in the previous experiment so the resulting structures are poorer. The increased $C_r$ values for this algorithm on large problem sizes are a by-product of this algorithm's workflow. In DEUM-LDA the parameter estimation step selected individuals from a new population of size 4.4N rather than recycling that produced in the course of the LDA run. DEUM-$\chi^2$ uses the structure learning population, which is very large. The high selective pressure results in a slightly better model of fitness with the same model structure, in line with the results reported in [35].

**Table 5** DEUM-$\chi^2$ Fitness Model Statistics with LDA-equivalent population size over 30 runs

| PS | p | p-SD | r | r-SD | F | $C_r$ | $C_r$ -SD |
|---|---|---|---|---|---|---|---|
| 16 | 0.800 | 0.037 | 0.703 | 0.052 | 0.854 | 0.573 | 0.163 |
| 25 | 0.896 | 0.051 | 0.816 | 0.053 | 0.921 | 0.733 | 0.095 |
| 36 | 0.954 | 0.020 | 0.890 | 0.027 | 0.976 | 0.869 | 0.050 |
| 49 | 0.992 | 0.011 | 0.961 | 0.023 | 0.989 | 0.941 | 0.047 |
| 64 | 0.995 | 0.007 | 0.984 | 0.009 | 1.000 | 0.974 | 0.014 |
| 100 | 1.000 | 0.000 | 1.000 | 0.002 | 1.000 | 0.992 | 0.004 |
| 256 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.993 | 0.002 |
| 324 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.993 | 0.002 |
| 400 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.993 | 0.001 |

## *7.3 Optimisation Results*

We know from the results in section 6 that the algorithm can find the global optimum when the learned structure has F equal to 1.0 relative to the true structure so the optimisation experiment was instead run on the structures learned by the Chi-Square algorithm with the smaller population. Statistics for this were shown in Table 4. The motivation for this is that if the global optimum can be found using these imperfect structures then we have a significant saving in function evaluations over the LDA based algorithm.

Now we run the full optimisation algorithm incorporating the Chi-Square structure learner and report the results in Table 6. The headings are the same as Table 2.

**Table 6** DEUM-$\chi^2$ Optimisation Statistics over 30 runs

| PS | $C_r$ | FE | FE-SD | IT | IT-SD | SR |
|---|---|---|---|---|---|---|
| 16 | 0.814 | 219 | 17 | 646 | 1074 | 90 |
| 25 | 0.831 | 340 | 32 | 1278 | 2828 | 77 |
| 36 | 0.830 | 479 | 21 | 887 | 1621 | 60 |
| 49 | 0.807 | 722 | 88 | 6863 | 6948 | 50 |
| 64 | 0.816 | 1005 | 256 | 17435 | 26404 | 43 |
| 100 | - | - | - | - | - | 0 |
| 256 | - | - | - | - | - | 0 |
| 324 | - | - | - | - | - | 0 |
| 400 | - | - | - | - | - | 0 |

We can see that with the decrease in the fitness prediction capability of the model there is a marked decrease in the optimisation capability of the algorithm. Indeed, it is clear that the $C_r$ values for the runs which proved successful (Table 4) were on average higher than those found across all runs (Table 6). To improve on these results, the population size was again increased to 2N for the larger instances of the problem and the experiment rerun. The results are presented in Table 7.

**Table 7** DEUM-$\chi^2$ Optimisation Statistics with increase population size over 30 runs

| PS | $C_r$ | FE | FE-SD | IT | IT-SD | SR |
|---|---|---|---|---|---|---|
| 16 | 0.951 | 381 | 5 | 598 | 230 | 100 |
| 25 | 0.971 | 599 | 5 | 2122 | 2142 | 100 |
| 36 | 0.951 | 852 | 9 | 1425 | 543 | 100 |
| 49 | 0.940 | 1166 | 13 | 8192 | 14735 | 100 |
| 64 | 0.936 | 1505 | 16 | 1875 | 1941 | 100 |
| 100 | 0.939 | 2799 | 744 | 316472 | 525816 | 70 |
| 256 | - | - | - | - | - | 0 |
| 324 | - | - | - | - | - | 0 |
| 400 | - | - | - | - | - | 0 |

In contrast to the previous section, we see that the results are still poor. This can be attributed to the imperfections in the structure learned by the independence test method. The recall values of around 0.9 indicate that up to 10% of the interactions present in the perfect structure are missing; precision values of around 0.96 indicate that up to 4% of the interactions which were added to the model are not present in the perfect structure.

# 8 EVDEUM

Based on the poor results for the single-step algorithm it is worth determining whether an evolutionary approach would be able to overcome the issues with poor structure.

One important change was a reduction in the run time for the Gibbs sampler. With this algorithm there is no longer an assumption that a model with a close fit to the fitness function will be found in the first generation. This means that areas of high probability within the model will not necessarily be areas of high fitness, and running the Gibbs sampler slowly to convergence is likely to result in an individual of inferior fitness to the global optimum. The algorithm was initially run with a fixed value for cooling rate and maximum number of iterations. It was found that performance was improved by varying these parameters over the course of the evolution - reducing the cooling rate and increasing the maximum number of iterations with each generation. This allowed the algorithm to be balanced towards exploration in early generations and exploitation in later ones as the model fits more closely to the fitness function. For each generation g, the cooling rate r was calculated according to equation (14) and the maximum number of iterations of the Gibbs sampler I was calculated according to equation (15). The parameters for these were determined empirically to yield the best results.

$$r = 1 + \frac{g^2}{10} \tag{14}$$

$$I = \frac{1}{20g} \tag{15}$$

We also adopted a steady-state approach for the algorithm, replacing 5% of the population each generation. This allows us to use a large population for the structure learning component and maintain diversity for as long as possible.

1. Generate random initial population P
2. Select a subset $\sigma_1$, the top 25% of P
3. Run Chi-Square edge detection algorithm to search for statistical dependencies apparent in $\sigma_1$
4. Refine structure
5. Select a subset $\sigma_2$, the top 1.1N of P
6. Use $\sigma_2$ to build MFM
7. Calculate $C_r$
8. Sample R new individuals from MFM using random walk Gibbs sampler and replace poorest R individuals in P with these

## *8.1 Fitness Model*

As there are now multiple models being created, there are multiple figures for p, r, F and $C_r$. For ease of interpretation, the values over the course of evolution for three instances of the problem (25 bit, 100 bit and 256 bit) are represented graphically in Figures 2, 3 and 4.

The four measures can be shown relative to the same y-axis as they all have a range of zero to one (Strictly speaking $C_r$ has a range of -1 to +1 but in the examples here it is always positive). 25 bits was chosen for the first problem to look at rather than 16 bits because the algorithm was often able to solve the 16 bit instances of the problem in a single or very small number of generations so the chance to observe an effect over many generations was reduced.

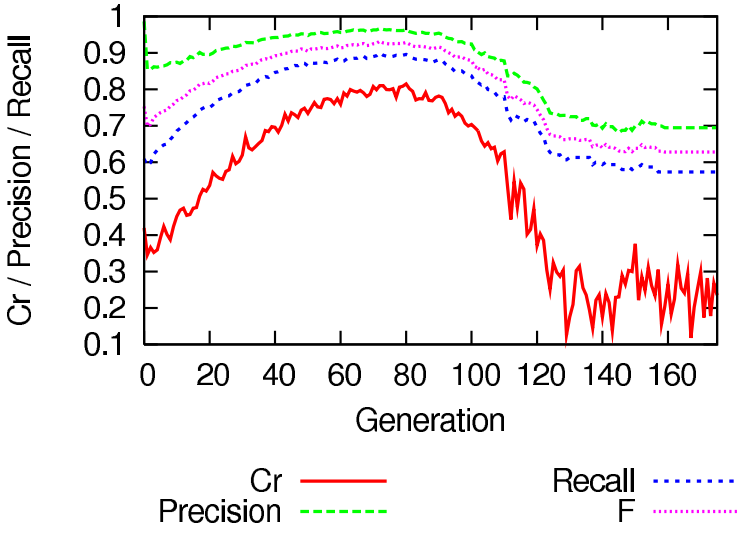



**Fig. 2** Fitness Model Statistics for EvDEUM-$\chi^2$ on 25bit 2D Ising lattice over 30 runs

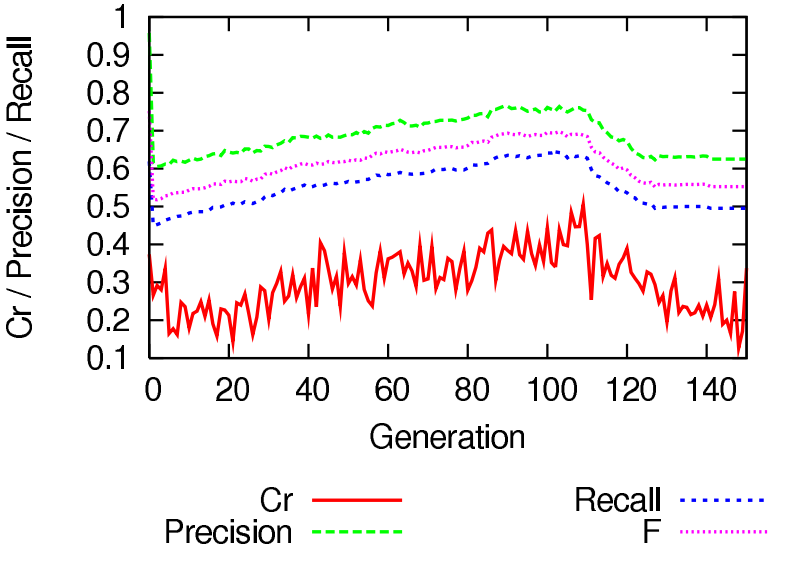



**Fig. 3** Fitness Model Statistics for EvDEUM-$\chi^2$ on 100bit 2D Ising lattice over 30 runs

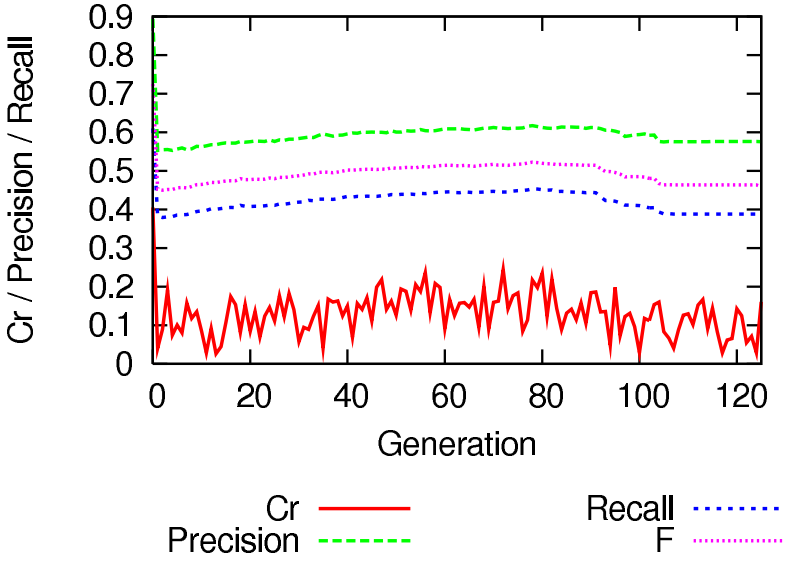



**Fig. 4** Fitness Model Statistics for EvDEUM-$\chi^2$ on 256bit 2D Ising lattice over 30 runs

We can see that the structure quality and consequently the fitness modelling capability of the model rise to begin with and then fall off as the population converges and diversity decreases. Further work is needed to determine the factors which affect the point at which this occurs. With increasing problem size the maximum values reached for structure quality and fitness prediction capability becomes lower, never exceeding an F of 0.5 or a $C_r$ of 0.3 as evolution proceeds. This means the model is unlikely to be good enough to allow the algorithm to find the global optimum. It is also notable that in the first generation the precision and recall of the structure and $C_r$ for the model are all relatively high, this then drops off immediately in the second generation.

## *8.2 Optimisation Results*

The results for the optimisation capability of the algorithm are shown in Table 8. As before, the mean number of function evaluations (FE) and internal iterations of the Gibbs sampler (IT) are given along with their standard deviations (FE-SD and IT-SD). All of these values only include the successful runs; the success rate (SR) is given as a percentage over 30 independent runs. No $C_r$ values are present this time because that value varied over the course of the evolution.

**Table 8** EVDEUM-$\chi^2$ Optimisation Statistics over 30 runs

| PS | FE | FE-SD | IT | IT-SD | SR |
|---|---|---|---|---|---|
| 16 | 2465 | 3254 | 4292 | 10233 | 100 |
| 25 | 21135 | 2861 | 2564001 | 2010019 | 83 |
| 36 | 25913 | 911 | 3226402 | 1154861 | 100 |
| 49 | 33321 | 1826 | 8193559 | 3778421 | 67 |
| 64 | - | - | - | - | 0 |
| 100 | - | - | - | - | 0 |
| 256 | - | - | - | - | 0 |
| 324 | - | - | - | - | 0 |
| 400 | - | - | - | - | 0 |

The results in this section reflect the poor quality of the models learned. We can see that even for small instances of the problem, the success rate is below 100% and the total number of function evaluations used by the algorithm is higher than for the single step algorithms. As the problem size increases, the perfect structure is never found and the algorithm is unable to find the global optimum. In each generation, the model build time is reduced because the structure learned has fewer interactions, meaning fewer terms in the model. The sampling times are greatly reduced in comparison with the single step approach as we are deliberately lowering the iteration cap on the sampler to encourage diversity in the population. Both of these benefits are lost when repeated over many generations. While the number of function evaluations is reduced in each generation (compared to the number required by the single-step structure learning algorithms), the evolutionary approach does not allow the structure to be found with reduced data.

## 9 Conclusion

In this chapter we have shown how the general concept of linkage learning is realised as structure learning in EDAs. In particular, we have extended the DEUM Markov EDA framework to incorporate a structure learning step, exploring three different approaches. using the measures precision, recall and the F-measure. We believe these are useful measures by which to compare structure learning algorithms on known benchmark problems and our results bear this out in the case of 2D Ising problems.

Precision, recall and F Measure are well-known in machine learning but we believe have not been studied before in relation to structure learning in EDAs. They provide a different prespective from existing measures of structure learning such as benign/malign and unnecessary interactions - those terms describe the influence an interaction has on fitness and whether an interaction is required by the model for optimisation. Precision, recall and the F-measure refer specifically to the number of interactions that the structure learning algorithm has found relative to the

known structure and allow a precise measurement of the effectiveness of a structure learning algorithm.

Our results show that the DEUM algorithms are able to optimise successfully, but the overhead (model build and sampling time) is expensive. The MFM requires a near-perfect structure to be supplied and a large population for optimisation to be successful. An incremental approach to the model building step like that of iBOA [38] offers a potential improvement and suggests a direction for future work on Markov Network EDAs. Additionally, other ways of making use of the fitness model may give better results than the Gibbs sampler. These include guided operators in a hybrid algorithm incorporating guided operators [39] and surrogate fitness models [40], [41], [42], [43].

An interesting observation to come out of this work is the drop-off in fitness modelling capability as evolution proceeds. We attribute this to loss of diversity in the population. The problem was mitigated by the use of a steady-state approach but did still prove a hindrance over time. This relates to other work on diversity loss [44], [45], [46], [47], [48] and another avenue for future work is mutation of the probabilistic model, niching [49] or other technique to reduce the effect of diversity loss and improve optimisation performance.

With the simple addition of a maximal clique finding algorithm it will also be possible to apply this approach to problems with higher order interactions such as SAT. Further work is needed to determine whether the effects described here hold true for other important problem classes.

## References

[1] Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms. Springer, Heidelberg (2006)

[2] Holland, J.H.: Adaptation in natural and artificial systems. MIT Press, Cambridge (1992)

[3] Mitchell, M., Holland, J.H., Forrest, S.: When will a Genetic Algorithm Outperform Hillclimbing? In: Cowan, J.D., Tesauro, G., Alspector, J. (eds.) Advances in Neural Information Processing Systems, vol. 6. Morgan-Kaufmann, San Francisco (1994)

[4] Greffenstette, J.J.: Predictive Models using Fitness Distributions of Genetic Operators. In: Foundations of Genetic Algorithms, vol. 3. Morgan Kauffmann, San Francisco (1995)

[5] Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive Models for the Breeder Genetic Algorithm. Evolutionary Computation 1(1), 25–49 (1993)

[6] Vose, M.D.: The Simple Genetic Algorithm: Foundations and Theory. The MIT Press, Cambridge (1999)

[7] Baluja, S., Caruana, R.: Removing the genetics from the standard genetic algorithm, pp. 38–46. Morgan Kaufmann Publishers, San Francisco (1995)

[8] Mühlenbein, H., Paß, G.: From recombination of genes to the estimation of distributions I. Binary Parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)

[9] Lauritzen, S.L.: Graphical models, vol. 17. Clarendon Press, New York; Oxford University Press, Oxford (1996)

[10] Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Boston (2002)
[11] Jordan, M.I. (ed.): Learning in Graphical Models. NATO Science Series. Kluwer Academic Publishers, Dordrecht (1998)
[12] Besag, J.: Spatial Interaction and the Statistical Analysis of Lattice Systems. Journal of the Royal Statistical Society 36(2), 192–236 (1974)
[13] Li, S.Z.: Markov Random Field Modeling in Computer Vision. Springer, London (1995)
[14] Hammersley, J.M., Clifford, P.: Markov Fields on Finite Graphs and Lattices (1971) (unpublished manuscript)
[15] Santana, R.: A Markov network based factorized distribution algorithm for optimization. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 337–348. Springer, Heidelberg (2003)
[16] Santana, R.: Probabilistic modeling based on undirected graphs in estimation of distribution algorithms, Ph.D. dissertation, Institute of Cybernetics, Mathematics and Physics, Havana, Cuba (2003)
[17] Santana, R.: Estimation of distribution algorithms with Kikuchi approximations. Evolutionary Computation 13(1), 67–97 (2005)
[18] Shakya, S.K., McCall, J.A.W., Brown, D.F.: Solving the Ising spin glass problem using a bivariate EDA based on Markov random fields. In: Proceedings of IEEE Congress on Evolutionary Computation. IEEE Press, Los Alamitos (2006)
[19] Wright, A.H., Pulavarty, S.: On the convergence of an estimation of distribution algorithm based on linkage discovery and factorization. In: GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 695–702. ACM Press, New York (2005)
[20] Heckendorn, R.B., Wright, A.H.: Efficient linkage discovery by limited probing. Evolutionary computation 12(4), 517–545 (2004)
[21] Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann Series in Data Management Sys. Morgan Kaufmann, San Francisco (2005)
[22] Pelikan, M., Goldberg, D.: Hierarchical BOA solves Ising spin glasses and MAXSAT. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1271–1282. Springer, Heidelberg (2003)
[23] Pelikan, M.: Bayesian optimization algorithm: from single level to hierarchy. Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL (2002)
[24] Pelikan, M., Ocenasek, J., Trebst, S., Troyer, M., Alet, F.: Computational complexity and simulation of rare events of Ising spin glasses. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 36–47. Springer, Heidelberg (2004)
[25] Brownlee, A.E.I., McCall, J.A.W., Brown, D.F.: Solving the MAXSAT prob-lem using a multivariate EDA based on Markov networks. In: GECCO 2007: Proceedings of the 2007 GECCO Conference on Genetic and Evolutionary computation, pp. 2423–2428. ACM, New York (2007)
[26] Mühlenbein, H., Höns, R.: The estimation of distributions and the minimum relative entropy principle. Evolutionary Computation 13(1), 1–27 (2005)
[27] Zhang, Q.: On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm. IEEE Transactions in Evolutionary Computation 8(1), 80–93 (2004)

[28] Kallel, L., Naudts, B., Reeves, R.: Properties of fitness functions and search landscapes. In: Kallel, L., Naudts, B., Rogers, A. (eds.) Theoretical Aspects of Evolutionary Computing, pp. 177–208. Springer, Heidelberg (2000)
[29] Hauschild, M., Pelikan, M., Lima, C.F., Sastry, K.: Analyzing probabilistic models in hierarchical BOA on traps and spin glasses. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and Evolutionary Computation, pp. 523–530. ACM, New York (2007)
[30] Mühlenbein, H., Mahnig, T.: Evolutionary optimization using graphical models. New Gen. Comput. 18(2), 157–166 (2000)
[31] Santana, R., Larrañaga, P., Lozano, J.A.: Challenges and open problems in discrete EDAs. Department of Computer Science and Artificial Intelligence, University of the Basque Country, Tech. Rep. EHU-KZAA-IK-1/07 (October 2007), http://www.sc.ehu.es/ccwbayes/technical.htm
[32] Shakya, S.K., McCall, J.A.W., Brown, D.F.: Updating the probability vector using MRF technique for a univariate EDA. In: Proceedings of STAIRS 2004, pp. 15–25. IOS Press, Amsterdam (2004)
[33] Shakya, S.K., McCall, J.A.W., Brown, D.F.: Estimating the distribution in an EDA. In: Proceedings of the International Conference on Adaptive and Natural computing Algorithms (ICANNGA 2005), pp. 202–205. Springer, Heidelberg (2005)
[34] Shakya, S.K., McCall, J.A.W., Brown, D.F.: Incorporating a Metropolis method in a distribution estimation using Markov random field algorithm. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 2576–2583. IEEE, Los Alamitos (2005)
[35] Brownlee, A., McCall, J., Zhang, Q., Brown, D.: Approaches to selection and their effect on fitness modelling in an estimation of distribution algorithm. In: Zurada, J.M., Yen, G.G., Wang, J. (eds.) Computational Intelligence: Research Frontiers. LNCS, vol. 5050. Springer, Heidelberg (2008)
[36] Lucey, T.: Quantatitive Techniques: An Instructional Manual, Eastleigh, Hampshire. D. P. Publications, UK (1984)
[37] Brownlee, A.E., Pelikan, M., McCall, J.A., Petrovski, A.: An application of a multivariate estimation of distribution algorithm to cancer chemotherapy. In: GECCO 2008: Proceedings of the 10th annual conference on Genetic and evolutionary computation, pp. 463–464. ACM, New York (2008)
[38] Pelikan, M., Sastry, K., Goldberg, D.E.: iBOA: The incremental Bayesian optimization algorithm. In: GECCO 2008: Proceedings of the 10th annual conference on Genetic and Evolutionary Computation, pp. 455–462. ACM, New York (2008)
[39] Zhang, Q., Sun, J., Tsang, E.: An evolutionary algorithm with guided mutation for the maximum clique problem 9(2), 192–200 (2005)
[40] Pelikan, M., Sastry, K.: Fitness inheritance in the Bayesian optimization algorithm. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 48–59. Springer, Heidelberg (2004)
[41] Lima, C.F., Pelikan, M., Sastry, K., Butz, M.V., Goldberg, D.E., Lobo, F.G.: Substructural neighborhoods for local search in the Bayesian optimization 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. 232–241. Springer, Heidelberg (2006)

[42] Sastry, K., Lima, C., Goldberg, D.E.: Evaluation relaxation using substructural information and linear estimation. In: Proceedings of the 8th annual Conference on Genetic and Evolutionary Computation GECCO-2006, pp. 419–426. ACM Press, New York (2006)
[43] Orriols-Puig, A., Bernadó-Mansilla, E., Sastry, K., Goldberg, D.E.: Substructural surrogates for learning decomposable classification problems: implementation and first results. In: GECCO 2007: Proceedings of the 2007 GECCO Conference on Genetic and Evolutionary Computation, pp. 2875–2882. ACM, New York (2007)
[44] Ochoa, A., Soto, M.R.: Linking entropy to estimation of distribution algorithms. In: Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.) Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms, pp. 1–38. Springer, Heidelberg (2006)
[45] Handa, H.: Estimation of distribution algorithms with mutation. In: Raidl, G.R., Gottlieb, J. (eds.) EvoCOP 2005. LNCS, vol. 3448, pp. 112–121. Springer, Heidelberg (2005)
[46] Mahnig, T., Mühlenbein, H.: Optimal mutation rate using Bayesian priors for estimation of distribution algorithms. In: Steinhöfel, K. (ed.) SAGA 2001. LNCS, vol. 2264, pp. 33–48. Springer, Heidelberg (2001)
[47] Branke, J., Lode, C., Shapiro, J.L.: Addressing sampling errors and diversity loss in UMDA. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 508–515. ACM, New York (2007)
[48] Posik, P.: Preventing premature convergence in a simple EDA via global step size setting. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 549–558. Springer, Heidelberg (2008)
[49] Dong, W., Yao, X.: Niching EDA: Utilizing the diversity inside a population of EDAs for continuous optimization. In: IEEE World Congress on Computational Intelligence 2008 (CEC 2008), pp. 1260–1267 (2008)

# DEUM – A Fully Multivariate EDA Based on Markov Networks

Siddhartha Shakya, Alexander Brownlee, John McCall, François Fournier, and Gilbert Owusu

**Abstract.** Recent years have seen an increasing interest in Markov networks as an alternative approach to probabilistic modelling in estimation of distribution algorithms (EDAs). Distribution Estimation Using Markov network (DEUM) is one of the early EDAs to use this approach. Over the years, several different versions of DEUM have been proposed using different Markov network structures, and are shown to work well in a number of different optimisation problems. One of the key similarities between all of the DEUM algorithms proposed so far is that they all assume the interaction between variables in the problem to be pre-given. In other words, they do not learn the structure of Markov network, and assume that it is known in advance. This work presents a recent development in DEUM framework - a fully multivariate DEUM algorithm that can automatically learn the undirected structure of the problem, automatically find the cliques from the structure and automatically estimate a joint probability model of the Markov network. This model is then sampled using Monte Carlo samplers. The chapter also reviews some of the key

Siddhartha Shakya
BT Innovate & Design, Ipswich, UK
e-mail: sid.shakya@bt.com

Alexander Brownlee
School of Computing, Robert Gordon University, Aberdeen, UK
e-mail: sb@comp.rgu.ac.uk

John McCall
School of Computing, Robert Gordon University, Aberdeen, UK
e-mail: jm@comp.rgu.ac.uk

François Fournier
ODS-Petrodata, Aberdeen, UK
e-mail: ffournier@ods-petrodata.com

Gilbert Owusu
BT Innovate & Design, Ipswich, UK
e-mail: gilbert.owusu@bt.com

Y.-p. Chen (Ed.): Exploitation of Linkage Learning, ALO 3, pp. 71–93.
springerlink.com 

works on use of Markov networks in EDAs, and explains the fitness modelling concept used by DEUM. The proposed DEUM algorithm can be applied to any general optimisation problems even when the structure is not known.

## 1 Introduction

Estimation of Distribution Algorithm (EDA) [25][17] belongs to a class of population based optimisation algorithm that iteratively evolves solution to a problem by means of estimating a probabilistic model of good solutions, and sampling from them to get new solutions. Much research in EDA focuses on different approach to probabilistic modelling and sampling. Particularly, directed graphical models (Bayesian networks) [31] have been widely studied and are well established as a useful approach for modelling the distribution in EDAs. Some of the well known instances of Bayesian network based EDA includes Bayesian Optimisation Algorithm (BOA) [32], hierarchical Bayesian Optimisation Algorithm (hBOA) [33], Estimation of Bayesian Network Algorithm (EBNA) [9][16] and Learning Factorised Distribution Algorithm (LFDA) [23]. Recent years have seen an increasing interest in the use of undirected graphical models (Markov networks) [2][20][27] in EDAs [37][43][44][38][41][39][42][6][7]. Some of the well known instances of Markov network based EDA includes Distribution Estimation Using Markov Networks (DEUM) algorithm [41], Markov Network EDA (MN-EDA) [38], Markov Network Factorised Distribution Algorithm (MN-FDA) [37] and Markovianity based Optimisation Algorithm (MOA) [46, 47].

In this chapter, our focus is on DEUM algorithms [43][41]. They are a family of Markov network based EDA that builds a model of fitness function from the undirected graphs, and use this model to estimate the parameters of the Gibbs distribution. Markov chain Monte Carlo simulations, including Gibbs sampler [10] and Metropolis sampler [22], are then used to sample new solutions from the Gibbs distribution. Several variants of DEUM have been proposed and are found to perform well, in comparison to other EDAs of their class, in a range of different test problems, including Ising Spin Glass and SAT. [43][44][45][5]. However, there is one key similarity between all of the DEUM algorithms proposed so far. That is they all assume the interaction between variables in the problem to be pre-given. In other words, they do not learn the structure of the problem and assume that it is known in advance. Therefore, they may not be classified as full DEUM algorithms.

In this chapter, we present a fully multivariate DEUM algorithm that can automatically learn the undirected structure of the problem, automatically find the cliques from the structure and automatically estimate the joint probability model of the Markov network. This model is then sampled using Monte Carlo samplers. The proposed DEUM algorithm can be applied to any general optimisation problem even when the structure is not known.

The outline of the chapter is as follows. Section 2 describes Markov network and its key properties. Section 3 reviews the use of Markov networks in EDAs. Section 4 gives background on DEUM framework and also describes the fitness

modelling approach to estimating the parameters of the Markov network within different instances of this framework. Section 5 presents a detailed workflow of the proposed full DEUM algorithm, together with the structure learning and clique finding algorithms used by it. Section 6 presents the experimental results on the performance of the proposed algorithm on several instances of well known Ising Spin Glass problems. Section 7 highlights some future work and concludes the chapter.

## 2 Probabilistic Graphical Models in EDA

An EDA regards a solution, $x = \{x_1, x_2, .., x_n\}$, as a set of values taken by a set of variables, $X = \{X_1, X_2, ..., X_n\}$. A general EDA begins by initialising a population of solutions, $P$. A set of promising solutions $D$ is then selected from $P$, and is used to estimate a probabilistic model of $X$. The model is then sampled to generate the next population. Figure 1 shows the general EDA workflow.

**Estimation of Distribution Algorithm**

1. Generate initial (parent) population $P$ of size $M$
2. Select set $D$ from $P$ consisting of $N$ solutions, where $N <= M$
3. Estimate the probability distribution of variables in the solution from $D$
4. Sample distribution to generate offspring, and replace parents
5. Go to step 2 until termination criteria are met

**Fig. 1** The workflow of the general Estimation of Distribution Algorithm

The estimation of probability distribution lies in the very heart of EDA, and its effectiveness largely depends on how well it estimates and samples the distribution. This is where probabilistic graphical models [18] can be useful. Probabilistic graphical models provide an efficient and effective tool to represent the probability distribution of random variables. They can be seen as a merger of two disciplines, probability theory and graph theory [13]. They are mainly categorised into two groups [1].

1. Directed models (Bayesian networks)
2. Undirected models (Markov networks / Markov Random Fields)

### *2.1 Bayesian Networks*

A Bayesian network can be regarded as a pair $(B, \Theta)$, where $B$ is the structure of the model and the $\Theta$ is a set of parameters of the model. The structure $B$ is a *Directed*

[1] There are several other categories of probabilistic graphical model, such as factor graph and mixture models. However, for the purpose of this chapter, we limit them to two categories.

*Acyclic Graph (DAG)*[2], where each node corresponds to a variable in the modelled data set and each edge corresponds to a conditional dependency. A set of nodes $\Pi_i$ is said to be the parent of $X_i$ if there are edges from each variable in $\Pi_i$ pointing to $X_i$. The parameter $\Theta = \{p(x_1|\Pi_1), p(x_2|\Pi_2), ..., p(x_n|\Pi_n)\}$ of the model is the set of conditional probabilities, where each $p(x_i|\Pi_i)$ is the set of probabilities associated with a variable $X_i = x_i$ given it's parent variables $\Pi_i$. A Bayesian network is characterized in terms of the joint probability distribution of the variables in the modelled dataset as

$$p(x) = \prod_{i=1}^{n} p(x_i|\Pi_i) \tag{1}$$

## 2.2 Markov Networks

A Markov network is a pair $(G, \Psi)$, where $G$ is the structure and the $\Psi$ is the parameter set of the network. $G$ is an undirected graph where each node corresponds to a random variable in the modelled data set and each edge corresponds to a conditional dependency between two variables. However, unlike Bayesian networks, the edges in Markov networks are undirected. Here, the relationship between two nodes should be seen as a *neighbourhood relationship*, rather than a parenthood relationship. We use $N = \{N_1, N_2, ..., N_n\}$ to define a *neighbourhood system* on $G$, where each $N_i$ is a set of nodes neighbouring to node $X_i$. Figure 2 shows an example of a Markov network structure on 6 random variables. Here, variable $X_1$ has 2 neighbours, $N_1 = \{X_2, X_3\}$. Similarly, variable $X_2$ has 4 neighbours $N_2 = \{X_1, X_3, X_4, X_5\}$.

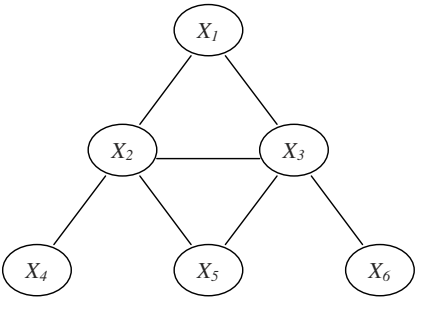



**Fig. 2** A Markov network structure on 6 random variables

### 2.2.1 Local Markov Property

A Markov network is characterised in terms of neighbourhood relationship between variables by its *local Markov property* known as *Markovianity* [2][20], which states

[2] A DAG is a graph where each edge joining two nodes is a *directed edge*, and also there is *no cycle* in the graph, i.e. it is not possible to start from a node and, travelling towards the correct direction, return back to the starting node.

that the conditional probability of a node $X_i$ given the rest of the variables can be completely defined in terms of it's conditional probability given only its neighboring states $N_i$. $N_i$ is sometimes referred to as *Markov Blanket* for $X_i$ [26]. In terms of probability, it can be written as

$$p(x_i|x - \{x_i\}) = p(x_i|N_i) \tag{2}$$

### 2.2.2 Global Markov Property

A Markov network is also characterised in terms of *cliques*[3] in the undirected graph by its global property, the joint probability distribution, and can be written as

$$p(x) = \frac{1}{Z}\prod_{i=1}^{m}\psi_i(c_i) \tag{3}$$

where, $\psi_i(c_i)$ (or more precisely $\psi_i(C_i = c_i)$) is a *potential function* on clique $C_i \in X$, $m$ is the number of cliques in the structure $G$. $Z = \sum_{x\in\Omega}\prod_{i=1}^{m}\psi_i(c_i)$ is the normalising constant known as the *partition function* which ensures that $\sum_{x\in\Omega} p(x) = 1$. Here, $\Omega$ is the set of all possible combination of the variables in $X$.

Equivalently, using Hammersley-Clifford theorem [11], the global Markov property can also be written in terms of Gibbs distribution as

$$p(x) = \frac{e^{-U(x)/T}}{Z} \tag{4}$$

where,

$$Z = \sum_{y\in\Omega} e^{-U(y)/T} \tag{5}$$

is a normalising constant, $T$ is a parameter of the Gibbs distribution known as the *temperature* and $U(x)$ (or more precisely $U(X = x)$) is known as the *energy* of the distribution.

Given an undirected graph, $G$, on $X$, energy, $U(x)$, is defined as a sum of *potential functions* over the cliques, $C_i$, in $G$.

$$U(x) = \sum_{i=1}^{m} u_i(c_i) \tag{6}$$

Here, $u_i(c_i)$ (or more precisely $u_i(C_i = c_i)$) is a potential function defined over a clique $C_i \in X$. Equation (4), in terms of clique potential function, can also be written as

$$p(x) = \frac{e^{-\sum_{i=1}^{m} u_i(c_i)/T}}{Z} \tag{7}$$

Note that the relationship between $\psi_i(c_i)$ in (3) and $u_i(c_i)$ in (7) is defined as

[3] Given an undirected graph $G$, a clique is a fully connected subset of the nodes. For example, in Figure 2, variables $\{X_1, X_2, X_3\}$ define a clique.

$$\psi_i(c_i) = e^{-u_i(c_i)/T} \tag{8}$$

The clique potential function, $u_i(c_i)$, captures the way variables interact in the clique $c_i$. It should be carefully defined in order to get a desired behaviour from a Markov network.

## 3 Markov Network Based EDAs

Markov network based EDAs can be categorised into two groups depending upon whether local (2) or global (3) Markov property is exploited in modelling and sampling the distribution.

### *3.1 Global Markov Property Based EDAs*

Most of the EDAs based on Markov network use its global property (3) in one form or another. More precisely, they factorise the joint probability distribution in terms of cliques in the undirected graph and sample it to generate new solutions.

Three main categories can be distinguished in this class of Markov network based EDAs. They are:

1. Factorised distribution algorithm (FDA)
2. Markov Network Estimation of Distribution Algorithm (MN-EDA), Markov network Factorised Distribution Algorithm (MN-FDA)
3. Distribution Estimation using Markov network algorithm (DEUM)

FDA is one of the early EDAs proposed by [24]. Based on *running intersection property* [18] of an undirected graph, it first identifies *residuals* and *separators* from the undirected structure and constructs a junction tree [19] that completely specifies the joint probability distribution. Junction tree is then sampled using Probabilistic logic sampling (PLS) [12] to generate new solution. An FDA able to learn a junction tree from the data was introduced in [28].

MN-EDA [38] and MN-FDA [37] are based on the idea of making an approximation to the joint probability distribution in terms of cliques in the undirected graph. MN-EDA does so by means of Kikuchi approximation [14] of the joint distribution and uses a Gibbs sampler to sample the new solutions. Similarly, MN-FDA constructs a junction graph [37] from the undirected structure that approximates the joint probability, which is then sampled using PLS to generate new solutions.

DEUM [43][41] is a family of Markov network based EDA that builds a model of fitness function in terms of the cliques in the undirected graph and factorises joint probability as a Gibbs distribution. The parameters of the fitness model is then estimated from the population of solutions and Markov chain Monte Carlo simulations, including Gibbs sampler [10] and Metropolis sampler [22], are used to sample new solutions. Several variants of DEUM have been proposed and are found to perform well in comparison to other EDAs of their class in range of different test problems, including Ising Spin Glass and SAT. [43][44][45].

### 3.2 Local Markov Property Based EDAs

Some recent works in EDA have focused on using local Markov property (2) for modelling and sampling from Markov network. As opposed to global Markov property based EDAs, they do not factorise joint probability distribution. Instead, they directly sample from local conditional probabilities defined by undirected graph.

Two main categories can be distinguished in this class of Markov network based EDAs. They are:

1. Markovianity based Optimisation Algorithm (MOA)
2. Markovian Learning Estimation of Distribution Algorithm (MARLEDA)

Both MOA[46, 47] and MARLEDA [1] estimate conditional probabilities defined by neighbourhood relationship in the undirected graph and directly sample from them to generate new solutions. The difference is, however, in the way they estimate undirected structure and sample from it. MALDERIA use chi-square test to find the undirected structure. MOA however use a mutual information based approach to do the same. Also, MOA use Gibbs sampler algorithms to sample new solution, and use a *temperature* based annealing schedule to balance the exploration and exploitation of the search space.

## 4 Fitness Modelling and DEUM Algorithms

In this chapter, our focus is on global Markov property based EDAs, more precisely on DEUM algorithms. As with other EDAs, a general DEUM starts by initialising a population of parent solutions, $P$. It then selects a set of promising solutions $D$ from $P$, which is then used to estimate the Markov network structure. It then builds a model of fitness function from the undirected relationship captured by the structure, and fits the model to the selected set of solution to estimate the model parameters. These parameters fully specify the joint probability of the Markov network. The build Markov network is then sampled to generate new solutions. These new solutions replace the parent solutions and this iteration continues.

The general workflow of DEUM algorithm is similar to that of other Markov network based EDAs. However, there is one noticeable characteristic that is specific to DEUM - it builds a model of fitness function and uses it to estimate the parameters of the Markov network. Next, this concept is described.

### 4.1 Fitness Modelling

Assuming that the probability of a solution is proportional to its fitness, the jpd, $p(x)$, can be modelled in terms of fitness as

$$p(x) = \frac{f(x)}{Z} \tag{9}$$

where, $Z = \sum_{y \in \Omega} f(y)$ is the partition function and $\Omega$ is the set of all possible solutions.

Now, from (4) and (9), we can deduce following equivalence of jpd for Markov netowrks in terms of fitness function.

$$p(x) = \frac{e^{-U(x)/T}}{\sum_{y \in \Omega} e^{-U(y)/T}} \equiv \frac{f(x)}{\sum_{y \in \Omega} f(y)} \tag{10}$$

From which, following relationship between fitness and the energy can be deduced [4].

$$-ln(f(x)) = U(x) \tag{11}$$

For simplicity, here we assume $T$ from (10) to be 1. In other words, (11) defines the equivalence shown in (10). We refer to (11) as **M**arkov network **F**itness **M**odel **(MFM)**. From (6), MFM can also be written in terms of potential functions as:

$$-ln(f(x)) = \sum_{i=1}^{m} u_i(c_i) \tag{12}$$

Energy, $U(x)$, in MFM (11) gives the full specification of the jpd (4), so MFM can be regarded as a probabilistic model of the fitness function. Also notice that, minimising $U(x)$ here is equivalent to maximising $f(x)$.

At this point, it is important to notice that the log-linear form of MFM (12) is the result of our assumption of jpd as a mass distribution of fitness over solution space, as shown in (9). We could easily get different relationship between $f(x)$ and $U(x)$ by making different assumption about mass distribution of fitness function. For example, assuming $p(x) = \frac{e^{-f(x)}}{\sum_{y \in \Omega} e^{-f(y)}}$, we would get a linear MFM as $f(x) = \sum_{i=1}^{m} u_i(c_i)$.

In general, the form of energy, $U(x)$ in MFM models the different order of interaction between variables in $X$.

## 4.2 Univariate MFM in DEUM$_{pv}$ and DEUM$_d$

The two initial univariate DEUM algorithms, DEUM with probability vector (DEUM$_{pv}$) [43] and DEUM with direct sampling from Gibbs distribution (DEUM$_d$) [44], used univariate MFM as their model, i.e. they assumed each variables $X_i \in X$ to be independent. The graph $G$ for such structure will be an edge less graph. Therefore, the set of maximal cliques, $C$, in $G$ would consist of $n$ singleton cliques $C_i = \{X_i\}$. For each clique, $\{X_i\}$, a potential function is associated as follows:

$$u_i(x_i) = \alpha_i x_i \tag{13}$$

From (11) and (12), the univariate MFM can then be written as:

$$-ln(f(x)) = U(x) = \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n x_n \tag{14}$$

Here, $\alpha_i$ are the parameters associated with each cliques $\{X_i\}$. $\alpha_i$ being the only unknown parameters of the potential function (13), completely specifies $U(x)$ and therefore completely specifies the Gibbs distribution (4). Therefore, they are known as Markov network (or Markov Random Field) parameters (MRF parameters) [20]. We use $\theta$ to refer to vector of all MRF parameters in the model. For univariate case, the vector $\theta = \alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$. In terms of MFM, (11), a MRF parameter measures the effect that the interaction between variables in a clique have on the fitness of the solution, $f(x)$. Obviously, in univariate case (14), $\alpha_i$ measures the effect of a single variable, $X_i$, on fitness.

### *4.3 Multivariate MFM in Is-DEUM*

An improved DEUM algorithm proposed in [45], known as Ising DEUM (Is-DEUM), used more complex MFM, which considered higher order cliques with two variables. Figure 3 shows the structure assumed by Is-DEUM where a variable interacted with 4 of its immediate Neighbours.

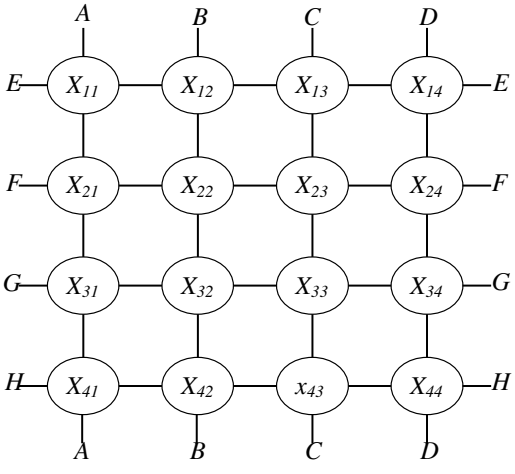



**Fig. 3** A structure showing the interaction between variables in a two dimensional lattice

This structure can also be seen as an instance of *Ising model* on two dimensional lattice [15]. The set of maximal cliques, $C$, in this case, contains $2 \times 4^2$ bivariate cliques $C_{ij,i'j'} = \{X_{ij}, X_{i'j'}\}$. This structure can be generalised to $n = l \times l$ variables, where $C$ will contain $m = 2l^2$ bivariate cliques. For each clique $\{X_{ij}, X_{i'j'}\}$, a potential function can be assigned as $\beta_{ij,i'j'} x_{ij} x_{i'j'}$, where, each $\beta_{ij,i'j'}$ is the MRF parameter associated with bivariate clique $\{X_{ij}, X_{i'j'}\}$. The energy, $U(x)$ in MFM (11) for such $X$ will therefore be

$$-ln(f(x)) = U(x) = \sum_{i=1}^{l} \sum_{j=1}^{l} \beta_{ij,(i+1)j} x_{ij} x_{(i+1)j} + \beta_{ij,i(j+1)} x_{ij} x_{i(j+1)}) \qquad (15)$$

The vector $\theta = \beta$ is used to denote the set of all $2n$ bivariate MRF parameters $\beta_{ij,i'j'}$. [5] extended this idea further to include cliques with three variables for solving 3-SAT problems.

### 4.4 Estimating MRF Parameters

Once the MFM is built, next step in DEUM is to estimate the set of all MRF parameters, $\theta$ . In general, each solution in a given population provides an equation satisfying the MFM ( eg. (14), or (15)), where $\theta$ are the unknows. Selecting a set of solution $D$ consisting of $N$ promising solutions from a population $P$ therefore allows us to estimate these parameters by solving the system of equations [4]:

$$F = A\theta^T \tag{16}$$

Here, $F$ is the vector containing $-\ln(f(x))$ of all solutions in $D$, $\theta$, the unknown part of the equation is the vector of all MRF parameters and $A$ is the matrix of allele values in $D$. In other words, DEUM fit the MFM to a dataset $D$ and approximate the parameter of the Markov networks, $\theta$.

### 4.5 Sampling Markov Networks

Once the MRF parameters are estimated, the joint probability distribution is fully specified. It is then sampled to generate new solution. By its definition, a Markov network structure may contain cycles. Apart from some restricted set of undirected structures, for example those that satisfy running intersection properties and can be formulated as a directed acyclic graph, most of the Markov networks do not satisfy the ancestral ordering of variables needed by Probabilistic logic sampling (PLS) used by Bayesian network based EDAs. DEUM use Markov Chain Monte Carlo (MCMC) [22] methods for sampling. MCMC are a iterative sampling approach that does not require the ancestral ordering of variables. An instance of it is described later in the chapter.

## 5 A Fully Multivariate General DEUM Algorithm

All instances of DEUM algorithm proposed so far (including those mentioned in previous section) have one key similarity. They all assume the interaction between variables in the problem to be pre given. In other words they do not learn the structure of the problem and assume that it is known in advance. For example, DEUM$_{pv}$ and DEUM$_d$ both use univariate model of probability distribution, i.e., assume each variables to be independent. Therefore, they do not need to learn the structure. Is-DEUM, which was proposed as the enhancement to these univariate DEUMs use a multivariate model of distribution, but still assume the structure of the Markov

[4] The current implementation of DEUM use Singular Value Decomposition (SVD) [36] to solve the system of equations.

network to be fixed (as shown in Figure 3). Therefore, it still restricts itself to types of problem that satisfies the structure assumed by it.

Next, we present a fully multivariate DEUM algorithm that can automatically learn the undirected structure of the problem, automatically find the cliques from the structure and automatically estimate a joint probability model of the Markov network. The implemented workflow is shown in Figure 4. It starts by initialising a

**Distribution Estimation using Markov networks (DEUM)**

1. Generate parent population $P$
2. Select a set of solutions $D$ from $P$
3. Estimate undirected structure $G$ of the Markov network from $D$
4. Find all the cliques in structure $G$ and build a model of fitness function (MFM)
5. Estimate parameters of the Markov network by fitting the build MFM to $D$
6. Sample Markov network to generate new solutions
7. Go to step 2 until termination criteria are meet

**Fig. 4** The pseudo-code of the fully multivariate Distribution Estimation Using Markov network (DEUM) algorithm

population of parent solution $P$. It then selects a set of promising solution $D$ from $P$. It then estimates an undirected graph $G$ from $D$ that defines the interaction between variables in the problem. It then finds a set of all cliques in $G$ and assigns potential function to them to build a MFM. MFM is again fitted to $D$ (or its subset) to find the parameters of the Markov networks. These parameters then specify the joint probability distribution, which is then sampled using Monte Carlo sampling techniques.

The estimation of undirected structure and building of a MFM (Step (3) and (4) in Figure 4) is the additional part in proposed DEUM algorithm that has not been so far implemented in other DEUM instances.

Let us describe these key features of the proposed algorithm in more detail.

## *5.1 Estimation of Undirected Structure*

A number of different approaches can be used to estimate an undirected structure from data. For the purpose of this chapter we implement a entropy based approach, initially described in [46]. More precisely, we estimate mutual information of each pairs of variable in the solution to create a matrix of mutual information. The pairs with the mutual information higher than a certain threshold are then made neighbours. Also, in order to avoid an overly complex network, we limit the number of neighbours that a variable can have to a certain number. Figure 5 describes the implemented Markov network structure learning algorithm. We note that other general statistical tests, such as chi square (also use by [1]), could also be used as the measurer for mutual information.

**Estimating structure in DEUM**

1. Create a matrix of mutual information, $MI$, by estimating mutual information for each pairs of variable in the solution. Mutual information between two random variables, A and B, is given by

$$I(A,B) = \sum_{a,b} p(a,b) log \quad \frac{p(a,b)}{p(a) \cdot p(b)}$$

   where sum is over all possible combinations of A and B, and $p(a,b)$ is the joint probability of $A = a$ and $B = b$ computed from $D$

2. Create an edge between two variables, if the mutual information between them is higher than the given threshold. Here we compute the threshold, $TR$ as $TR = avg(MI) * sig$, where $avg(MI)$ is the average of the elements of the MI matrix and $sig$ is the significance parameter, which is set to 1.5.
3. If the number of neighbours to a variable is higher than the maximum number, $MN$, allowed, only keep $MN$ neighbours that have the highest mutual information.

**Fig. 5** The workflow of an undirected structure learning algorithm

In our experiment, we also implement the structure learning algorithm that calculates difference in joint probability of two variables against the product of marginal probabilities of individual variables as a measure for mutual information. We call it Joint-Equals-Marginal-Product (JEMP) measure. JEMP for two random variables, A and B, can be estimated as

$$JEMP(A,B) = \sum_{a,b} |p(a,b) - (p(a) \cdot p(b))| \tag{17}$$

## 5.2 Finding Cliques and Assigning Potentials

Once the undirected structure is found, next step is to find all the cliques in the structure. In the proposed implementation of DEUM, we use Bron-Kerbosch algorithm [3] to find the set of all the maximal cliques in the undirected graph. We do not go into details of this algorithm. Interested readers are suggested to see [3]. Once the set of maximal cliques are found, the potential functions are assigned to them. There are three possibilities while defining clique potentials.

1. Define potentials to all the maximum cliques
2. Define potentials to all the maximum cliques and all the subcliques[5].

[5] Subcliques are defined as the smaller cliques within the maximal cliques. For example, in Figure 2, variables $\{X_1, X_2\}$ defines a sub-clique within clique $\{X_1, X_2, X_3\}$. Furthermore, variable $\{X_1\}$ also defines a singleton sub-clique within $\{X_1, X_2, X_3\}$.

3. Define potentials to all the maximum cliques and only some of the sub-cliques within it

The first option is the simplest and most efficient one, which requires only one potential function to be assigned per clique and, therefore require only one MRF parameter to be defined per clique. The second option is the safest one since it considers all possible sub interaction with the cliques. At the same time, it is the most expensive way to define MFM, since the number of MRF parameter grows significantly depending upon the order of the maximal cliques. The third option is the compromise between first and second, however, it may not be obvious to choose the sub-cliques required to build the model. Further research could be done in this area. For the purpose of this work, we use the first option.

### *5.3 Sampling New Solution*

Once the MFM is built, it is then fitted to the set of selected solution $D$ (as described earlier in section (4.4)) and the MRF parameters are estimated. These MRF parameters fully specify the joint distribution (4), which is then sampled to generate new solutions. Similar to other DEUM instances, here we use a MCMC approach to sampling. Particularly, we implement a Gibbs sampler [10] to sample from the Markov networks, which is described next.

Let us use $x^+$ to denote a solution $x$ having a particular $x_i = +1$ and $x^-$ to denote a solution $x$ having $x_i = -1$. The probability that the value of the variable in position $i$ is equal to 1 given its neighbours, $p(x_i = 1|N_i)$, can then be written as

$$p(x_i = 1|N_i) = \frac{p(x^+)}{p(x^+) + p(x^-)} \tag{18}$$

Substituting $p(x)$ from (4) and cancelling the $Z$, we get

$$p(x_i = 1|N_i) = \frac{e^{-U(x^+)/T}}{e^{-U(x^+)/T} + e^{-U(x^-)/T}} \tag{19}$$

Since, $U(x^+)$ and $U(x^-)$ agree in all terms other than those containing $x_i$, the common terms in both $U(x^+)$ and $U(x^-)$ drop out and we get much simpler expression as the estimate of the marginal probability for $x_i = 1$ conditional upon $N_i$:

$$p(x_i = 1|N_i) = \frac{1}{1 + e^{W_i/T}} \tag{20}$$

Here, $W_i$ is the difference in two energies, $U(x^+)$ and $U(x^-)$, after substituting the $x_i$ to 1 for all the remaining terms in $U(x^+)$ and to $-1$ for all remaining terms in $U(x^-)$. For example, $W_i$ for the univariate MFM (14) simplifies to

$$W_i = 2\alpha_i \tag{21}$$

Similarly, $W_i$ (or more precisely $W_{ij}$) for bivariate MFM (15) simplifies to

**Random Walk Gibbs Sampler (RWGS)**

1. Generate a solution $x^o = \{x_1^o, x_2^o, .., x_n^o\}$
2. Set the initial value for $T$.
3. Repeat $IT$ times:
   a. Randomly select a variable $x_i^o$ from $x^o$ and set $x_i^o = 1$ with probability $p(x_i^o = 1|N_i^o)$
   b. Decrease $T$
4. Terminate with answer $x^o$.

**Fig. 6** The pseudo-code of the Random Walk Gibbs Sampler

**Bit-Wise Gibbs Sampler (BWGS)**

1. Generate a solution $x^o = \{x_1^o, x_2^o, .., x_n^o\}$ at random.
2. set $r = 0$ and also set the initial value for $T$.
3. Repeat:
   a. Set $x^{tmp} = x^o$.
   b. For $i = 1$ to $n$
      i. Increase $r$ by 1
      ii. Decrease $T$
      iii. Set $x_i^o = 1$ with probability $p(x_i^o = 1|N_i)$

   :Until $x^{tmp} = x^o$.
4. Terminate with answer $x^o$.

**Fig. 7** The pseudo-code of the Bitwise Gibbs Sampler

$$W_{ij} = \beta_{ij,(i+1)j} x_{(i+1)j} + \beta_{ij,i(j+1)} x_{i(j+1)} + \beta_{(i-1)j,ij} x_{(i-1)j} + \beta_{i(j-1),ij} x_{i(j-1)} \quad (22)$$

As we said earlier, temperature, $T$, has a very important role in Gibbs distribution. It controls the *convergence* of the distribution. In equation (20), as $T \to 0$, the value of $p(x_i = 1|N_i)$ tends to a limit depending on the $W_i$. If $W_i > 0$, then $p(x_i = 1|N_i) \to 0$ as $T \to 0$. Conversely, if $W_i < 0$, then $p(x_i = 1|N_i) \to 1$ as $T \to 0$. If $W_i = 0$, then $p(x_i = 1|N_i) = 0.5$ regardless of the value of $T$. Therefore, the $W_i$ are indicators of whether the $x_i$ at the position $i$ should be 1 or $-1$. This indication becomes stronger as the temperature is cooled towards zero. This forms the basis for the implemented Gibbs sampler algorithm, which is shown in Figure 6. We call it Random Walk Gibbs Sampler (RWGS).

For our experiments, we also implement another version of the Gibbs sampler, which systematically iterates through each variable in the solution, as oppose to randomly selecting the variable in RWGS Figure 7. We call it the Bit-wise Gibbs Sampler (BWGS).

Notice that each run of Gibbs sampler will sample a single solution. Running it multiple times will give us a population of solution, which then replaces the parent population and the next iteration of DEUM follows.

## 6 Experimental Results

We apply proposed DEUM algorithm to a well known instance of Ising spin glass problem [15]. Due to their interesting properties, such as symmetry and a large number of plateaus, Ising spin glass problem have been widely studied by the GA (and EDA) community [34, 38].

Here, we consider spin glass system on a two dimensional lattice consist of $n = l \times l$ sites, where each spin variable interacts only with its nearest neighbouring variables on a *toroidal* lattice[6]. The Hamiltonian specifying the energy for this system can be written as

$$H(\sigma) = -\sum_{i=1}^{l}\sum_{j=1}^{l} J_{ij,(i+1)j}\sigma_{ij}\sigma_{(i+1)j} + J_{ij,i(j+1)}\sigma_{ij}\sigma_{i(j+1)}\big) \tag{23}$$

where, $i+1 = 1$ if $i = l$ and $j+1 = 1$ if $j = l$.

Here, each $J_{ij,i'j'}$ is the coupling constant in two dimensional lattice relating to spin $\sigma_{ij}$ and $\sigma_{i'j'}$. The task in Ising spin glass problem is to find the value for each $\sigma_i$ that minimises the energy, $H$.

In the context of EDAs, spin glass systems on a two dimensional lattice have been of particular interest to researchers. In particular, [34, 35] showed that hBOA could efficiently solve these problems outperforming other algorithms. [37] used the Ising spin glass problem as a test problem for MN-EDA and MN-FDA and showed that their performance is better then that of other EDAs based on Bayesian networks. Also [24] stated that, although the two dimensional Ising spin glass problem is in the class of *Additively Decomposable Functions* (ADF), it cannot be efficiently represented as a *junction tree*. This is because, the junction tree based EDA has a *triangular structure* of dependency and therefore requires cliques to be of order 3. However, the two dimensional Ising spin glass problem has maximum cliques of order 2. [38] argues that the Kikuchi approximation approach to estimate the distribution used by MN-EDA can accurately represent this dependency, and therefore has an advantage over junction tree based EDAs. Also, it has been shown that Is-DEUM, the previous version of the DEUM algorithm, which although required structure to be pre given, could also accurately represent the bivariate dependency, performed very well in this problem, and compared well with the rest of the EDAs.

[6] Figure 3 can be seen as the structure of the Ising spin glass on toroidal latttice.

### 6.1 Experimental Setup

We set up a series of experiments with two structure learning algorithm. One based on mutual inforamtion, $I$, and another based on JEMP measure. Also, we set up the experiment with two sampling algorithms described earlier, RBGS and RWGS. A further experiment is performed to examine the effect of population size on the algorithm.

Experiments were conducted with three different sizes of Ising Spin Glass problem: $10 \times 10$ (n = 100), $16 \times 16$ (n = 256) and $20 \times 20$ (n = 400). Four random instances of each problem size were used for the experiment. Each instance was generated by randomly sampling the coupling constant $J_{ij} \in \{+1, -1\}$. The optimum solution for each instance was verified by using Spin Glass Ground server, provided by the group of Prof. Michael Juenger[7]. The parameters for each algorithm were chosen empirically.

We made 100 independent runs of DEUM for each of the 12 instances of the Ising spin glass problem and recorded the number of fitness evaluations needed to find the optimum. The population size used was 30000, 75000 and 150000 for the problem sizes 100, 256 and 400 respectively. We found that the DEUM was able to find the accurate structure for the problem in the initial generation, and, as with Is-DEUM, was able to find the solution in the single generation. It, however, required a higher population size. To improve the speed of the algorithm, a smaller number of best individuals was selected for the parameter estimation step (Step 5 in Figure 4) than for estimating the undirected structure. The number selected to estimate the structure was 5000,7000 and 8000 for each of the problem sizes; the number selected to estimate the parameters was 250, 700 and 1000. The maximum neighbour (MN) parameter used by the structure learning algorithm was set to 4. The temperature $T$ for the Gibbs sampler was set to $T = 1/0.0005r$, where $r$ is the current number of $x_i^o$ samplings done in the sampler. (see Figure 6). As $r$ increases, $T$ decreases and the solution $x^o$ will converge to a particular value for each $x_i^o$ . The maximum number of allowed repetitions, $R$, for both RBGS and RWGS was set to 500. DEUM was terminated if the optimum was found or $R$ repetitions of the sampler were done. As, at the end of each Gibbs sampler run, a fitness evaluation was done in order to calculate $f(x^o)$, the number of fitness evaluations was calculated as the sum of population size and the total repetitions of the Gibbs sampler needed before finding the optimum.

### 6.2 Results

In Tables 1,2,3 and 4, we show the performance results for the algorithms incorporating the different structure learning and sampling techniques. In each, PI is the specific instance of the problem. The following three columns show the mean number of fitness evaluations FE (with standard deviation FE-SD) required by the algorithm to find an optimum and the percentage of runs finding the global optimum over 100 runs.

[7] `http://www.informatik.uni-koeln.de/lsjuenger/research/sgs/sgs.html`

**Table 1** DEUM Ising Problem Optimisation, using structure learning based on mutual information and Bit-wise Gibbs Sampler

| PI | FE | FE-SD | SR |
|---|---|---|---|
| Ising-100-1 | 30007.3 | 5.885266658 | 100 |
| Ising-100-2 | 30001.84 | 1.134580419 | 100 |
| Ising-100-3 | 30010.72 | 49.68955747 | 99 |
| Ising-100-4 | 30001.77 | 1.246044246 | 100 |
| Ising-256-1 | 75005.84 | 5.991104517 | 100 |
| Ising-256-2 | 75002.77 | 14.78386713 | 100 |
| Ising-256-3 | 75018.3 | 60.11764561 | 99 |
| Ising-256-4 | 75001.85 | 1.24214705 | 100 |
| Ising-400-1 | 150029.28 | 55.56869986 | 99 |
| Ising-400-2 | 150012.32 | 50.55647507 | 99 |
| Ising-400-3 | 150032.63 | 104.1892853 | 97 |
| Ising-400-4 | 150013.21 | 53.68544655 | 99 |

**Table 2** DEUM Ising Problem Optimisation, using structure learning based on mutual information and Random Walk Gibbs Sampler

| PI | FE | FE-SD | SR |
|---|---|---|---|
| Ising-100-1 | 30007.74 | 8.094916724 | 100 |
| Ising-100-2 | 30001 | 0 | 100 |
| Ising-100-3 | 30004.64 | 4.482333447 | 100 |
| Ising-100-4 | 30001.26 | 0.543464383 | 100 |
| Ising-256-1 | 75007.66 | 13.61313982 | 100 |
| Ising-256-2 | 75001.6 | 1.034749762 | 100 |
| Ising-256-3 | 75037.08 | 98.38757415 | 97 |
| Ising-256-4 | 75002.53 | 2.536421563 | 100 |
| Ising-400-1 | 150055.57 | 107.586306 | 95 |
| Ising-400-2 | 150016.28 | 51.34417044 | 99 |
| Ising-400-3 | 150041.9 | 117.3730563 | 94 |
| Ising-400-4 | 150015.58 | 54.44303482 | 99 |

### *6.3 Analysis*

We can see that DEUM is able to learn the required Markov network structure and therefore solve the problem with a high success rate. The performance, in terms of success rate, was comparable to that reported for Is-DEUM in [45]. Also, we notice that the DEUM with mutual information based structure learner was slightly better than that with JEMP based structure learner. Similarly, we find that DEUM based on BWGS is slightly better that that based on RWGS.

One point worth further investigation is that the population sizes required by the algorithm are very large, in comparison to the results presented in [45]. As we mentioned earlier, this is because the dependency tests require a large population to learn useful structure in a single generation. To further illustrate the effect of the

**Table 3** DEUM Ising Problem Optimisation, using structure learning based on JEMP and Bit-wise Gibbs Sampler

| PI | FE | FE-SD | SR |
|---|---|---|---|
| Ising-100-1 | 30008.57 | 7.521356798 | 100 |
| Ising-100-2 | 30001.73 | 1.043062701 | 100 |
| Ising-100-3 | 30006.36 | 5.424448955 | 100 |
| Ising-100-4 | 30001.95 | 1.336171222 | 100 |
| Ising-256-1 | 75013.56 | 57.38780358 | 99 |
| Ising-256-2 | 75003.48 | 20.97496343 | 100 |
| Ising-256-3 | 75023.67 | 75.84780814 | 98 |
| Ising-256-4 | 75002.11 | 2.403259571 | 100 |
| Ising-400-1 | 150041.16 | 80.25863747 | 98 |
| Ising-400-2 | 150013.06 | 50.09341173 | 99 |
| Ising-400-3 | 150021.44 | 74.11216179 | 98 |
| Ising-400-4 | 150009.6 | 49.68974449 | 99 |

**Table 4** DEUM Ising Problem Optimisation, using structure learning based on JEMP and Random Walk Gibbs Sampler

| PI | FE | FE-SD | SR |
|---|---|---|---|
| Ising-100-1 | 30009.41 | 9.855619327 | 100 |
| Ising-100-2 | 30006 | 49.89909009 | 99 |
| Ising-100-3 | 30005.5 | 3.9376453 | 100 |
| Ising-100-4 | 30001.22 | 0.612661028 | 100 |
| Ising-256-1 | 75006.72 | 6.130038641 | 100 |
| Ising-256-2 | 75004.12 | 24.07281882 | 100 |
| Ising-256-3 | 75038.23 | 94.1580291 | 97 |
| Ising-256-4 | 75002.44 | 2.011683048 | 100 |
| Ising-400-1 | 150064.5 | 114.7484293 | 95 |
| Ising-400-2 | 150022.97 | 69.90649671 | 98 |
| Ising-400-3 | 150034.09 | 98.33575703 | 97 |
| Ising-400-4 | 150022.31 | 76.1903113 | 98 |

population size on the algorithm, we ran similar experiment as above on the four 100 bit instances of the problem with different population sizes. The experiment used DEUM with a Bit-wise Gibbs Sampler, with both the mutual information based structure learner and JEMP based structure learner. Figure 8 shows the success rate for the algorithm over varying population size for the two dependency tests. For this experiment, the selection operator for the dependency test selected the top 1/4 of the population. The selection operator for the parameter learning component selected the top 250 individuals as in the previous experiments. The results show a clear link between population size and whether the algorithm is able to find the global optimum. We believe that the population size can be significantly reduced for the proposed DEUM algorithm by incrementally learning the structure over multiple generations.

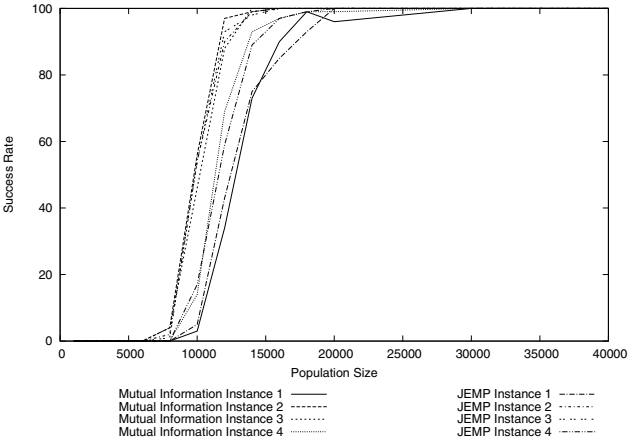



**Fig. 8** Success rate vs population size for different 100 bit Ising instances

## 7 Conclusion

In this chapter, we have presented a fully multivariate DEUM algorithm that incorporates automatic structure learning and model building to the existing DEUM framework. We have also tested the algorithm with a number of different instances of the Ising spin glass problem, and shown that it can learn the Ising structure from data. Also, we have shown that its performance is comparable to that of Is-DEUM that required the structure of the problem to be pre-given.

We note that, although the experiments presented in this work were performed with the Ising Problems, we believe that the algorithm will work at least as well as other DEUM algorithms on the wide range of other problems previously tackled using the DEUM framework [44, 45, 5, 49, 8].

The proposed DEUM algorithm fills the missing gap in the DEUM framework, and finally makes it a fully multivariate EDA. It also opens a number of promising future research directions in this area. One of the immediate tasks is to apply the full multivariate DEUM algorithm to other multivariate optimisation problems. Also, research should be done to improve the performance of DEUM, in particular, to incorporate better structure learning algorithms and better sampling techniques. The former is crucial, since current implementation of structure learning algorithm only looks at pair-wise dependency between variables. Extending it to higher order dependency test could improve the quality of the learned structure.

The results we present must be read within the context that there is some overhead associated with the fitness modelling approach as implemented in DEUM. In particular, the time to estimate parameters with SVD [36] grows rapidly ($O(n^3)$) as the number of unknowns in the model grows. With this in mind it is important to consider that the explicit model of fitness represented by the MFM brings additional benefits. These include applications such as surrogate fitness models [21, 40, 29]

and guided genetic operators [48, 50, 51, 30] as well as revealing the underlying dynamics of the fitness function [8]. There is still much to be explored in this area.

To summarise, the work presented in this chapter is an important extension to the general DEUM framework. We believe that the added automatic structure learning and model building process, together with the powerful fitness modelling and sampling techniques, in an undirected modelling environment, gives DEUM an increased versatility to tackle difficult real world optimisation problems.

## References

1. Alden, M.A.: MARLEDA: Effective Distribution Estimation Through Markov Random Fields. PhD thesis, Faculty of the Graduate Schoool, University of Texas at Austin, USA (December 2007)
2. Besag, J.: Spatial interactions and the statistical analysis of lattice systems (with discussions). Journal of the Royal Statistical Society 36, 192–236 (1974)
3. Born, C., Kerbosch, J.: Algorithms 457 - finding all cliques of an undirected graph. Communications of the ACM 16(6), 575–577 (1973)
4. Brown, D.F., Garmendia-Doval, A.B., McCall, J.A.W.: Markov Random Field Modelling of Royal Road Genetic Algorithms. In: Collet, P., Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M. (eds.) EA 2001. LNCS, vol. 2310, pp. 65–78. Springer, Heidelberg (2002)
5. Brownlee, A., McCall, J., Brown, D.: Solving the MAXSAT problem using a Multivariate EDA based on Markov Networks. In: A late breaking paper in GECCO 2007 : Proceedings of the 2007 conference on Genetic and Evolutionary Computation, Global Link Publishing (2007)
6. Brownlee, A., McCall, J., Zhang, Q., Brown, D.: Approaches to selection and their effect on fitness modelling in an estimation of distribution algorithm. In: Proceedings of the 2008 Congress on Evolutionary Computation CEC-2008, Hong Kong, pp. 2621–2628. IEEE Press, Los Alamitos (2008)
7. Brownlee, A.E.I.: Multivariate Markov networks for fitness modelling in an estimation of distribution algorithm. PhD thesis, The Robert Gordon University. School of Computing, Aberdeen, UK (2009)
8. Brownlee, A.E.I., Wu, Y., McCall, J.A.W., Godley, P.M., Cairns, D.E., Cowie, J.: Optimisation and fitness modelling of bio-control in mushroom farming using a Markov network EDA. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008), Atlanta, Georgia, USA, pp. 465–466. ACM, New York (2008)
9. Etxeberria, R., Larrañaga, P.: Global optimization using Bayesian networks. In: Ochoa, A., Soto, M.R., Santana, R. (eds.) Proceedings of the Second Symposium on Artificial Intelligence (CIMAF 1999), Havana, Cuba, pp. 151–173 (1999)
10. Geman, S., Geman, D.: Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. In: Fischler, M.A., Firschein, O. (eds.) Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, pp. 564–584. Kaufmann, Los Altos (1987)
11. Hammersley, J.M., Clifford, P.: Markov fields on finite graphs and lattices (1971) (Unpublished)
12. Henrion, M.: Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In: Lemmer, J.F., Kanal, L.N. (eds.) Uncertainty in Artificial Intelligence, vol. 2, pp. 149–163. North-Holland, Amsterdam (1988)

13. Jordan, M.I. (ed.): Learning in Graphical Models. NATO Science Series. Kluwer Academic Publishers, Dordrecht (1998)
14. Kikuchi, R.: A Theory of Cooperative Phenomena. Physical Review 81, 988–1003 (1951)
15. Kindermann, R., Snell, J.L.: Markov Random Fields and Their Applications. AMS (1980)
16. Larrañaga, P., Etxeberria, R., Lozano, J.A., Peña, J.M.: Combinatorial optimization by learning and simulation of Bayesian networks. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, Stanford, pp. 343–352 (2000)
17. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2002)
18. Lauritzen, S.L.: Graphical Models. Oxford University Press, Oxford (1996)
19. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society B 50, 157–224 (1988)
20. Li, S.Z.: Markov Random Field modeling in computer vision. Springer, Heidelberg (1995)
21. Lim, D., Jin, Y., Ong, Y.-S., Sendhoff, B.: Generalizing surrogate-assisted evolutionary computation. IEEE Transactions on Evolutionary Computation (2008)
22. Metropolis, N.: Equations of state calculations by fast computational machine. Journal of Chemical Physics 21, 1087–1091 (1953)
23. Mühlenbein, H., Mahnig, T.: FDA - A scalable evolutionary algorithm for the optimization of additively decomposed functions. Evolutionary Computation 7(4), 353–376 (1999)
24. Mühlenbein, H., Mahnig, T., Ochoa, A.R.: Schemata, distributions and graphical models in evolutionary optimization. Journal of Heuristics 5(2), 215–247 (1999)
25. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions: I. binary parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
26. Murphy, K.: Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis, University of California, Berkeley (2002)
27. Murray, I., Ghahramani, Z.: Bayesian Learning in Undirected Graphical Models: Approximate MCMC algorithms. In: Twentieth Conference on Uncertainty in Artificial Intelligence (UAI 2004), Banff, Canada, July 8-11 (2004)
28. Ochoa, A., Soto, M.R., Santana, R., Madera, J., Jorge, N.: The factorized distribution algorithm and the junction tree: A learning perspective. In: Ochoa, A., Soto, M.R., Santana, R. (eds.) Proceedings of the Second Symposium on Artificial Intelligence (CIMAF 1999), Havana, Cuba, March 1999, pp. 368–377 (1999)
29. Ong, Y.S., Nair, P.B., Keane, A.J., Wong, K.W.: Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In: In Knowledge Incorporation in Evolutionary Computation, pp. 307–332. Springer, Heidelberg (2004)
30. Peña, J., Robles, V., Larrañaga, P., Herves, V., Rosales, F., Pérez, M.: GA-EDA: Hybrid evolutionary algorithm using genetic and estimation of distribution algorithms. In: Orchard, B., Yang, C., Ali, M. (eds.) IEA/AIE 2004. LNCS (LNAI), vol. 3029, pp. 361–371. Springer, Heidelberg (2004)
31. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufman Publishers, Palo Alto (1988)
32. Pelikan, M.: Bayesian optimization algorithm: From single level to hierarchy. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, Also IlliGAL Report No. 2002023 (2002)

33. Pelikan, M., Goldberg, D.E.: Hierarchical problem solving by the Bayesian optimization algorithm. IlliGAL Report No. 2000002, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL (2000)
34. Pelikan, M., Goldberg, D.E.: Hierarchical BOA solves Ising spin glasses and MAXSAT. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1271–1282. Springer, Heidelberg (2003)
35. Pelikan, M., Ocenasek, J., Trebst, S., Troyer, M., Alet, F.: Computational complexity and simulation of rare events of ising spin glasses. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 36–47. Springer, Heidelberg (2004)
36. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C: The Art of Scientific Computing, 2nd edn. Cambridge University Press, Cambridge (1993)
37. Santana, R.: A Markov network based factorized distribution algorithm for optimization. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 337–348. Springer, Heidelberg (2003)
38. Santana, R.: Estimation of Distribution Algorithms with Kikuchi Approximation. Evolutionary Computation 13, 67–98 (2005)
39. Santana, R., Larrañaga, P., Lozano, J.A.: Mixtures of Kikuchi approximations. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 365–376. Springer, Heidelberg (2006)
40. Sastry, K., Lima, C., Goldberg, D.E.: Evaluation relaxation using substructural information and linear estimation. In: Proceedings of the Genetic and Evolutionary Computation COnference (GECCO 2006), pp. 419–426. ACM Press, New York (2006)
41. Shakya, S.: DEUM: A Framework for an Estimation of Distribution Algorithm based on Markov Random Fields. PhD thesis, The Robert Gordon University, Aberdeen, UK (April 2006)
42. Shakya, S., McCall, J.: Optimisation by Estimation of Distribution with DEUM framework based on Markov Random Fields. International Journal of Automation and Computing 4, 262–272 (2007)
43. Shakya, S., McCall, J., Brown, D.: Updating the probability vector using MRF technique for a univariate EDA. In: Onaindia, E., Staab, S. (eds.) Proceedings of the Second Starting AI Researchers' Symposium, Valencia, Spain, August 2004. Frontiers in Artificial Intelligence and Applications, vol. 109, pp. 15–25. IOS Press, Amsterdam (2004)
44. Shakya, S., McCall, J., Brown, D.: Using a Markov Network Model in a Univariate EDA: An Emperical Cost-Benefit Analysis. In: Proceedings of Genetic and Evolutionary Computation COnference (GECCO2005), Washington, D.C., USA, pp. 727–734. ACM, New York (2005)
45. Shakya, S., McCall, J., Brown, D.: Solving the Ising spin glass problem using a bivariate EDA based on Markov Random Fields. In: Proceedings of IEEE Congress on Evolutionary Computation (IEEE CEC 2006), Vancouver, Canada, pp. 3250–3257. IEEE Press, Los Alamitos (2006)
46. Shakya, S., Santana, R.: An EDA based on local Markov property and Gibbs sampling. In: Proceedings of Genetic and Evolutionary Computation COnference (GECCO 2008), Atlanta, Georgia, USA. ACM, New York (2008)
47. Shakya, S., Santana, R.: A markovianity based optimisation algorithm. Technical Report Technical Report EHU-KZAA-IK-3/08, Department of Computer Science and Artificial Intelligence, University of the Basque Country (September 2008)

48. Sun, J., Zhang, Q., Li, J., Yao, X.: A hybrid estimation of distribution algorithm for cdma cellular system design. International Journal of Computational Intelligence and Applications 7(2), 187–200 (2008)
49. Wu, Y., McCall, J., Godley, P., Brownlee, A., Cairns, D.: Bio-control in mushroom farming using a markov network eda. In: Proceedings of the 2008 Congress on Evolutionary Computation, pp. 2996–3001 (2008)
50. Zhang, Q., Sun, J.: Iterated local search with guided mutation. In: Proceedings of the IEEE World Congress on Computational Intelligence (CEC 2006), pp. 924–929. IEEE Press, Los Alamitos (2006)
51. Zhang, Q., Sun, J., Tsang, E.: An evolutionary algorithm with guided mutation for the maximum clique problem. IEEE Transactions on Evolutionary Computation 9(2), 192–200 (2005)

# Part II
# Model Building and Exploiting

# Pairwise Interactions Induced Probabilistic Model Building

David Iclănzan, D. Dumitrescu, and Béat Hirsbrunner

**Abstract.** The intrinsic feature of Estimation of Distribution Algorithms lies in their ability to learn and employ probabilistic models over the input spaces. Discovery of the appropriate model usually implies a computationally expensive comprehensive search, where many models are proposed and evaluated in order to find the best value of some model discriminative scoring metric. This chapter presents how simple pairwise interaction variable data can be extended and used to efficiently guide the model search, decreasing the number of model evaluations by several orders of magnitude or even facilitate the finding of richer, qualitatively better models. As case studies, first the $O(n^3)$ model building of the Extended Compact Genetic Algorithm is successfully replaced by a correlation guided search of linear complexity, which infers the perfect problem structures on the test suites. In a second study, a search technique is proposed for finding Bayesian network structures, capable of modeling complicated multivariate interactions, like the one exemplified by the parity function.

## 1 Introduction

Estimation of Distribution Algorithms (EDAs) extend the classical framework of Evolutionary Algorithms (EAs) by learning and using probabilistic models over the

David Iclănzan
Department of Electrical Engineering, Sapientia Hungarian University of Transylvania, Şoseaua Sighişoarei 1C, 540485, OP 9, CP 4, Romania
e-mail: david.iclanzan@gmail.com

D. Dumitrescu
Department of Computer Science, Babeş-Bolyai University, Kogălniceanu no. 1, Cluj-Napoca, 400084, Romania
e-mail: ddumitr@cs.ubbcluj.ro

Béat Hirsbrunner
Pervasive Artificial Intelligence Group, DIUF University of Fribourg, Bd de Perolles 90, CH-1700 Fribourg, Switzerland
e-mail: beat.hirsbrunner@unifr.ch

Y.-p. Chen (Ed.): Exploitation of Linkage Learning, ALO 3, pp. 97–122.
springerlink.com 

search variables, thus they are able to exploit dependencies and the modular structure of the search space. By replacing simple operators with a repeated selection–model-building–sampling process, these methods can solve problems that are intractable using fixed, problem independent operators and representations [15].

EDAs usually search for a model which fits an existing population according to some criteria, like the Minimum Description Length Principle [21] or Bayesian-Dirichlet metric [15]. The size of the population must be large enough to guarantee proper initial-supply, decision-making and accurate model-building. The search for an appropriate model in EDAs capable of modeling higher order dependencies, requires many model evaluations with regard to the population. Given the implied population sizes as the dimension of the problems increases, the computational cost of model building may quickly exceed economical practicality. Recent benchmarking and profiling results showed that easily more than 90% of EDAs running time may be spent in the model building phase [5].

To cope up with large problem sizes many enhancements and modifications of the original methods were proposed. To reduce the computational and/or memory cost investigations had considered parallelization [22, 14] and hybridization with local search methods [18], the usage of iterative [20] and sporadic model building [19] or incorporation of initial knowledge [1, 10].

More direct approaches aim to reduce the complexity of model building by restricting the search over a reduced set of variables in each epoch [5]. Heuristics of this type work well for problems where all or most of the variables are engaged in dependencies as randomly choosing and analyzing a partition will discover and exploit dependencies with a high probability. Nevertheless, for problems with many independent variables this heuristic may prove inefficient as many blindly chosen partitions will not reveal dependencies and in these cases resources are spent but the finding of a better model that could help focusing the search is postponed.

In the model building phase EDAs iteratively improve an initial model according to the scoring metric. Usually candidates for new models are obtained by stochastic or deterministic (pre-fixed order) alteration of the current model. In this way many poor model extensions are proposed and analyzed. A question of great interest relates to the possibility to build smarter search methods that could *dynamically* choose and propose only the best potential extensions. Such intelligent bias could greatly reduce the complexity of model building.

To achieve this desiderate one needs a method that can quickly predict which model extensions maximize the multivariate mutual information, reduce entropy. Only the most promising extensions are analyzed in detail with regard to the population data by the scoring metric which will determine their exact contribution and will also guard against over fitting.

In this chapter we propose and describe techniques for the use of pairwise interaction measures as means of determining the best model extension choices for binary EDAs. By computing the pairwise interaction between variables, one obtains a matrix containing some normalized measure of the strength of relationship between variables. We extend these pairwise relations to approximate the strength of interaction between arbitrary groups of variables. This approach proves to be

very efficient in guiding the model-building for binary problems. It can even detect highly non-linear multivariate variable interplay by mapping the variables and their interactions in a higher dimensional space, where dependencies are expressed in an easily detectable manner.

The following section presents the mathematical principle and reasoning behind the pairwise interaction guided model building and details it in a general setting. Section 3 presents a first case study, where the model building in the classic Extended Compact Genetic Algorithm (eCGA) and its hybridized version is replaced by a correlation guided search of linear complexity. The section also contains the description of our experimental setup, the results and a discussion of our findings related to the modified version of the eCGA. A Bayesian network model-building search technique, based on an extension of pairwise interactions to groups of variables, thus able to detect and model synergic multivariate interactions is described in section 4. Conclusion and outline of extension possibilities, future work is given in section 5.

## 2 Predicting Information Gain from Pairwise Interactions

Some definitions and basic concepts that will aid the discussion are presented.

*Covariance* is a measure of how much two variables vary together.

$$C(X,Y) = E(X \cdot Y) - \mu\nu \tag{1}$$

where $E$ is the expected value operator and $E(X) = \mu$, $E(Y) = \nu$.

In typical data analysis applications, one is usually interested in the degree of relationship between variables. The Pearson *correlation coefficient* between two variables represents the normalized measure of the strength of linear relationship.

$$R(X,Y) = \frac{C(X,Y)}{\overline{C(X,X) \cdot C(Y,Y)}} \tag{2}$$

This relation gives values between +1 and -1 inclusive. If there is perfect linear relationship with: a) positive slope between the two variables, $R(X,Y) = 1$; b) negative slope $R(X,Y) = -1$. A correlation coefficient of 0 means that there is no linear relationship between the variables.

The *entropy* is a measure of the average uncertainty in a random variable. It is the number of bits on average required to describe the random variable. The entropy of a random variable $X$ with a probability mass function $p(x)$ is defined by

$$H(X) = -\sum_{x \in X} p(x) \log_2(p(x)) \tag{3}$$

Entropy is the uncertainty of a single random variable. Conditional entropy $H(X|Y)$ is the entropy of a random variable conditional on the knowledge of another random variable. The reduction in uncertainty due to another random variable is called the *mutual information*:

$$I(X;Y) = H(X) - H(X|Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \quad \frac{p(x,y)}{p(x)p(y)} \tag{4}$$

Another way to think about mutual information is that it is a measure of how close the true joint distribution of X and Y is to the independent joint distribution. Thus mutual information is closely related to the *relative entropy* or *Kullback-Leibler divergence* which measures the "distance" between two distributions:

$$D_{KL}(p||q) = \sum_{x \in X} p(x) \log(\frac{p(x)}{q(x)}) \tag{5}$$

## 2.1 Information Gain on Binary Data

By incorporating and exploiting problem structure knowledge, a good model provides compression to the data, reduces entropy. The goal in linkage learning evolutionary algorithms is to find the group of variables with the highest interactions among them and exploit these linkages.

The relative entropy can be used as a measure of the information gain by considering a group of variables linked: $D_{KL}$ from Eq. 5 is applied as the expected discrimination information for the observed joint distribution of some variables $p(x_1, x_2, \ldots x_n)$ over the product of independent joint distribution $p_{X_1}(x_1)p_{X_2}(x_2)\ldots p_{X_n}(x_n)$. Because the $D_{KL}$ is zero if and only if $p(x_1, x_2, \ldots x_n) = p_{X_1}(x_1)p_{X_2}(x_2)\ldots p_{X_n}(x_n)$, it follows that this measure can capture all kind of dependencies between random variables.

Unfortunately, considering and evaluating all possible cases of variable combinations is impossible due to the combinatorial explosion.

In the following we consider the ways in which simple pairwise interaction information, like mutual information or various correlation coefficients can be used to predict which variable groups may maximize the information gain, which may be measured by relative entropy or other discriminative metric, for example ones based on the Minimum Description Length principle.

As two binary variables can only be linearly dependent, the analysis can be safely restricted to correlation coefficients, as in this case a correlation of zero implies statistical independence.

Binary problems of real interest have many variables with complicated multivariate interactions among them. The dependency of a binary variable $X_e$ on a (noisy) feature expressed by several other variables of the problem can be formalized as follows:

$$\begin{aligned} \text{if} \quad & f_b(X_{v1}, X_{v2}, \ldots, X_{vl}) \quad [\text{and } noise(X)] \\ & \text{then} \qquad\qquad X_e = b \\ & [\text{else} \qquad\qquad X_e = \overline{b}] \end{aligned} \tag{6}$$

where $f_b$ is an arbitrary boolean function of $l$ binary variables of equal marginal distributions and $\overline{b}$ is the negation of the binary value $b$. As the relation must not be fully specified, the else branch is optional. The optional boolean $noise(X)$ function can be used to introduce stochasticity to the relation, to model external influences or factors which are not directly considered when evaluating the feature. This boolean function may prevent the expression of the feature even if the conditions are present, thus adding noise to the relation.

The boolean function $f_b$ analyzes if the input variables satisfy a certain feature or not. This feature can be an arbitrarily complex computable relationship; the only requirement is that the function must remain deterministic i.e for the same inputs the answer must be always the same. For example one can apply the variables to a binary polynomial and consider as feature, the value of the polynomial being a prime number.

Given this general setting, we are interested, in what conditions can correlation detect that there is a relation between some $X_v$ and $X_e$?

If the samples are drawn according to a discrete uniform distribution and we assume no noise, the exact probability $P_{f_b}$ of the realization of the feature can be defined. If there are $N$ samples the approximate number of times the feature will be present and expressed in $X_e$ is $P \approx NP_{f_b}$. On these $P$ samples, $K$ times $X_e = X_v$ and $P-K$ times $X_e = \overline{X_v}$.

Let us standardize $X_e$ and $X_v$ to $z_{X_e}$ and $z_{X_v}$ with mean 0 and standard deviation 1. As we assumed equal marginal distributions this accounts to replacing 0's with -1's. The correlation coefficient for $z$ scores can be expressed as:

$$R(z_{X_e}, z_{X_v}) = \sum_{i=1}^{N} \frac{z_{X_{e_i}} z_{X_{v_i}}}{N-1} \tag{7}$$

Assume that the samples are reordered in such way that the first $P$ samples are those where the feature is present and the next $N-P$ samples are unrelated (R=0). By further reordering, the first $K$ samples from $P$ are those where $z_{X_e} = z_{X_v}$ and the next $P-K$ samples are those where $z_{X_e} = \overline{z_{X_v}}$. Now the expectation of $R$ can be expressed as

$$\begin{aligned} E(R) = \tfrac{1}{N-1} \cdot \; & E \sum_{j=1}^{K} z_{X_{e_j}} z_{X_{v_j}} + E \sum_{j=K+1}^{P} z_{X_{e_j}} z_{X_{v_j}} \\ & + E \sum_{j=P+1}^{N} z_{X_{e_j}} z_{X_{v_j}} \end{aligned} \tag{8}$$

The first expectation is $K$ as $z_{X_e} = z_{X_v}$, the second one is $K-P$ as $z_{X_e} z_{X_v} = -1$ from $z_{X_e} = \overline{z_{X_v}}$ and the last one is 0. Consequently it follows that

$$E(R) \approx \frac{2K-P}{N-1} \tag{9}$$

Simply stated, the absolute value of $R$ conveys the probability that the binary paired values are identical/complemental due to a common source, regardless of what this source is. From the viewpoint of conditioning, the correlation coefficient measures the effectiveness of predicting one binary variable with the other. A $K$ close to 0 or P maximizes the mutual information in the boundary of $P_{f_b}$.

A zero value of the correlation coefficient means that considering just a single variable *alone*, will not carry any information about the other one. Nevertheless, it is not excluded that the preconditioning by *two or more* variables will present some information.

Let us analyze the case when $X_e$ is conditioned by a feature over several variables $X_C = \{X_{v1}, X_{v2}, \ldots, X_{vl}\}$ but no dependency is indicated by the correlation coefficient for any of the $X_C$-s i.e $R(z_{X_e}, z_{X_C}) = 0$ in all cases. From eq. 9 $E(R)$ is zero if and only if $K = P/2$. In this way whenever $f_b(X_C) = 1$ the value of 0 and 1 for $X_v$-s is still equiprobable. Thus the realization of the feature does not affect the probability mass functions of $X_v$-s – the feature can be explained only at a synergic level. It follows that $X_e$ is conditioned by a count function over $X_C$ and without loss of generality, we can assume that is conditioned by the number of 1's from $X_C$.

To show this, consider the cases when $f_b(X_C)$ is true, the feature is realized. If $X_e$ is independent of the number of 1's (implicitly independent of the number of 0's) from $X_C$ on these cases, as the probability mass functions for $X_v$-s are unperturbed, every configuration of $X_C$ has the same probability. Hence, the data follows a discrete uniform distribution and this contradicts the assumption that $X_e$ is conditioned by $X_C$.

For large and complicated features the deviations in the probability mass functions might be to small to safely detect with simple correlation analysis, or even nonexistent like in the case of the parity relation, which is expressed only at a synergic level. To be able to detect all kind of relationship we propose a technique where the variables and some simple statistics about the parity of 1's are mapped in a higher dimensional feature space and a new correlation analysis is performed there. To build the statistics we use the generalized or n-ary exclusive "or" operator:

$$\oplus_n(X_{ALL}) = X_1 \oplus X_2 \oplus \cdots \oplus X_l \tag{10}$$

where $X_i \in X_{ALL}$ and $i$ goes from 1 to the number of variables $l$. $\oplus_n$ is true if and only if it has an odd number of 1's in the inputs, a property we exploit to detect parity based dependencies.

The rationale behind this technique is that given a set of binary variables, the probability mass function of the outcome obtained by XOR-ing them can be predicted, provided that the variables are statistically independent.

Let $X_1$ and $X_2$ be two random variables with probability mass functions $p_{X_1}(x_1)$, $p_{X_2}(x_2)$ and $X_3 = X_1 \oplus X_2$. The probability mass function of $X_3 = 1$ is given by:

$$p(X_3 = 1) = p_{X_1}(X_1 = 1)p_{X_2}(X_2 = 0) + p_{X_1}(X_1 = 0)p_{X_2}(X_2 = 1) \tag{11}$$

This relation can be recursively applied to an arbitrary number of variables. For multivariate interactions, the actual values will deviate from the expectations,

enabling even the detection of parity relations: after all possible combinations $B$ of variables $\oplus_n(B)$ is calculated and placed in the feature space, for the parity based relations, whenever $f_b(X_C)$ is true, $\oplus_n(X_C)$ will yield the same result ($K = P$ or $K = 0$), thus $\oplus_n(B)$ and $X_e$ will be maximally correlated in the bounds of $P_{f_b}$. The strength of conditioning i.e the value of $P_{f_b}$ will be reflected in the pairwise interaction coefficient.

## 2.2 General Measurement of Module-Wise Interactions

The processing of the XOR operations and the calculation of correlations can be highly parallelized. Even so, for larger problem sizes the processing of all the mappings can become problematic. For $n$ variables, the dimension of the feature space is

$$S = n + \sum_{i=2}^{n} \binom{n}{i} \tag{12}$$

Therefore when searching for linkages, we can take into account second order interactions among group of variables up to a bounded size $k$.

The interaction between a variable and any other group of variables is quantified by:

$$d_I(X_i, B) = I(X_i, \oplus_n(B)) \tag{13}$$

where $B$ is a set of variables of maximum $k$ variables and not containing $X_i$, i.e $B \cap X_i = \emptyset$.

In this definition we use mutual information, so that this relation can be applied to higher order alphabets also, after appropriate replacement of the $\oplus_n$ operator.

For many problems, finding and exploiting all simple first order dependencies with correlation analysis might suffice. All widely spread probabilistic model building genetic algorithms begin their model building by looking for pairwise dependencies. Expressing the simpler interactions will hopefully heavily reduce the search space and after no improvements can be detected cheaply, model building can continue with classical, more complex and expensive search methods.

Let $M_R$ contain the absolute values from the correlation coefficient matrix but with self-correlations set to zero, i.e $M_R(x,x) = 0$. The first module-wise metric simply averages the different pairwise interactions:

$$d_1(X,Y) = \frac{\sum_{x \in X} \sum_{y \in Y} M_R(x,y)}{\binom{|X \quad Y|}{2}} \tag{14}$$

As it averages, $d_1$ is influenced by outliers, penalizing the incorporation of non-correlated components. For quantifying the strong interactions even in subsets of the modules, we introduce a metric which just sums up interactions:

$$d_2(X,Y) = \sum_{x \in X} \sum_{y \in Y} \sigma(x,y) \cdot M_R(x,y) \tag{15}$$

where

$$\sigma(x,y) = \begin{array}{ll} 1 & \text{, if } M_R(x,y) \text{ is statistically significant;} \\ 0 & \text{, otherwise.} \end{array} \quad (16)$$

In $d_2$ non-correlated components can not heavily bias the outcome. This also means that too complex module formation are not penalized. The two metrics are complemental and can be used together. If $d_2$ is high but $d_1$ has a small value, we should consider splitting the modules in multiple pieces: there are strong interactions but not all $(x,y)$ variable pairs are correlated.

In the next subsection we empirically depict how these metrics can guide the model search.

## 2.3 Examples

We consider $n = 8$ binary variables $X_1 \cdots X_8$, with $X_5$ conditioned by the first 4 variables $X_C = \{X_1 \cdots X_4\}$ in the following way: if $\sum_{i=1}^{4} X_i = 3$ then $X_5 = 1$, i.e whenever 3 out of the 4 variables are 1 $X_5$ is also 1. The probability of realization of the feature $f_b(X_C) = 4/16 = 0.25$. $X_5 = X_C$ in a $K = 3/4 = 0.75$ proportions and in a $1 - K = 0.25$ proportion $X_5 = \bar{X_C}$. From equation 9 it results that the theoretical expectancy for the conditioning is $E = (2K-1) * f_b = 0.125$.

$N = 2000$ samples are generated randomly with uniform distribution followed by the application of the above mentioned conditioning. The correlation coefficients are computed and their absolute values are stored in $M_R$. Self-correlation is set to 0. The hypothesis of no correlation is tested also, where the probability of getting a correlation as large as the observed value by random chance is computed using a t-test. The probabilities are stored in a matrix $M_T$.

The results for $M_R$ and $M_T$ after averaging on 10 independent runs are presented in table 1. The maximum values from $M_R$ clearly suggest that the strongest interactions are between $(X_3, X_5)$, $(X_2, X_5)$, $(X_4, X_5)$ and $(X_1, X_5)$. Please remark how these values are very close to the theoretical expectance of 0.125. These indications are also statistically significant, as the corresponding values from $M_T$ are very close to 0. For many applications a p-value less than 0.05 is considered significant. After $(X_3, X_5)$ are joined in a module, the metrics $d_1$ and $d_2$ will successfully measure that the joint model is most strongly connected to $X_2$ $X_4$ and $X_1$. Following the correlation guidance, the whole interdependence between $X_C$ and $X_5$ can be revealed.

In a second example we consider the conditioning:
if $s = \sum_{i=1}^{6} X_i$ is an odd number then $X_7 = 1$ for 10 variables. Computing $M_R$ on this example does not reveal any pairwise interactions as expected. Also, all values from $M_T$ are above the significance level of 0.05.

After extending the base 10 variables to the feature space by applying the $\oplus_n$ on all possible combinations of the variables and computing the correlation coefficients here, the strongest interaction according to $d_I$ is identified as between modules $X_7$ and $(X_1 \cdots X_6)$ with a significance level of $1.0750e-022$. Thus the underlying conditioning is again revealed in one (extended) step with great accuracy.

**Table 1** The absolute correlation coefficients values ($M_R$) and probabilities of no correlation testing ($M_T$) resulting from the $\sum_{i=1}^{4} X_i = 3) \mapsto (X_5 = 1)$ conditioning

$$M_R = \begin{matrix} - & 0.016 & 0.021 & 0.021 & 0.122 & 0.022 & 0.021 & 0.020 \\ 0.016 & - & 0.019 & 0.013 & 0.126 & 0.016 & 0.019 & 0.017 \\ 0.021 & 0.019 & - & 0.014 & 0.135 & 0.022 & 0.018 & 0.018 \\ 0.021 & 0.013 & 0.014 & - & 0.124 & 0.013 & 0.018 & 0.017 \\ 0.122 & 0.126 & 0.135 & 0.124 & - & 0.014 & 0.018 & 0.016 \\ 0.022 & 0.016 & 0.022 & 0.013 & 0.014 & - & 0.016 & 0.014 \\ 0.021 & 0.019 & 0.018 & 0.018 & 0.018 & 0.016 & - & 0.017 \\ 0.020 & 0.017 & 0.018 & 0.017 & 0.016 & 0.014 & 0.017 & - \end{matrix}$$

$$M_T = \begin{matrix} - & 0.531 & 0.425 & 0.427 & 0.000 & 0.412 & 0.428 & 0.441 \\ 0.531 & - & 0.457 & 0.602 & 0.000 & 0.511 & 0.476 & 0.512 \\ 0.425 & 0.457 & - & 0.573 & 0.000 & 0.461 & 0.511 & 0.473 \\ 0.427 & 0.602 & 0.573 & - & 0.000 & 0.605 & 0.506 & 0.516 \\ 0.000 & 0.000 & 0.000 & 0.000 & - & 0.587 & 0.485 & 0.552 \\ 0.412 & 0.511 & 0.461 & 0.605 & 0.587 & - & 0.562 & 0.574 \\ 0.428 & 0.476 & 0.511 & 0.506 & 0.485 & 0.562 & - & 0.503 \\ 0.441 & 0.512 & 0.473 & 0.516 & 0.552 & 0.574 & 0.503 & - \end{matrix}$$

Parity problems like the one presented are intractable for EDAs except if they especially model $b$-wise dependencies where $b$ is the block size with parity interaction [4]. If the EDA fail to discover these higher order dependencies, then variables will be modeled as independent and randomly distributed in consequent generations. To avoid this, the presented technique can be used as a preprocessor for EDAs as follows. Extending 14-16 variables to the feature space is still feasible and can be quickly achieved using parallel matrices. Therefore, the preprocessing would repeatedly choose random groups of 10-16 variables and would analyze their relationship in feature space. In this way the existence of first and second order interactions from the groups can be detected and this information can facilitate the building of a more adequate initial model.

The empirical preliminary testing confirms the pairwise interaction analysis can detect complex conditioning. The computation of the coefficient correlation matrix is a fast operation being faster then evaluating a single model fit on the population data.

## 3 Case Study on eCGA

The eCGA [8] is a multivariate extension of the Compact Genetic Algorithm [9] based on the key principle that learning a good probability distribution of the population is equivalent to the linkage learning process. The measure of a good distribution is quantified based on minimum description length (MDL) models. MDL is pillared on the concept that any regularity in a given set of data can be used to compress the data. The best hypothesis for a given set of data is the one that leads

**Table 2** MPM over 3 variables. $X_1$ and $X_3$ are linked together defining a joint distribution where $X_1$ having the same values as $X_3$ has probability 0.0. This, together with the information that $X_2$ is three times more likely to be 1 than 0 helps focusing the search

| $[X_1, X_3]$ | $[X_2]$ |
|---|---|
| $P(X_1 = 0 \text{ and } X_3 = 0) = 0.0$ | $P(X_2 = 0) = 0.25$ |
| $P(X_1 = 0 \text{ and } X_3 = 1) = 0.5$ | $P(X_2 = 1) = 0.75$ |
| $P(X_1 = 1 \text{ and } X_3 = 0) = 0.5$ | |
| $P(X_1 = 1 \text{ and } X_3 = 1) = 0.0$ | |

to the largest compression. Consequently, a tradeoff between model accuracy and complexity must be found.

MDL restriction reformulates the problem of finding a good distribution as minimizing both population representation (population complexity – $C_p$) and the cost of representing the model itself (model complexity – $C_m$). Hence the combined complexity criterion $C_c$ to be minimized is given by:

$$C_c = C_p + C_m \tag{17}$$

The probability distribution used by the eCGA belongs to the Marginal Product Model (MPM), a class of probability model. Subsets of variables can be modeled jointly as partitions, providing a direct linkage map. Partitions together with the products of marginal distributions over them they form the MPMs.

The MPM concept is illustrated in Table 2 over a 3 bit problem with $[1,3],[2]$ as partitions. The first and third bit are jointly distributed while variable $[2]$ is independent. The compound partition [1,3] can have four settings: $\{00, 01, 10, 11\}$. The probability distribution for the partitions is given by the frequency of the individuals in the population with those bit values.

Starting from a random population, the eCGA applies the process of evaluation, selection, MPMs based model-building and sampling until a halting criterion is met.

**Algorithm 1.** Model-building in eCGA

```
1  Build initial model m where each variable is an independent partition;
2  repeat
3      m_best ← m;
4      foreach [p,q] from the (|m| choose 2) set of possible pair partitions of m do
5          Form new model m' based on m but with p and q merged into a joint partition;
6          Evaluate combined complexity criterion C_c(m');
7          if m' improves over m_best then
8              m_best ← m';
9      m ← m_best;
10 until No improvement was found ;
```

In its model-building phase, the eCGA greedily searches the space of possible partitions guided by the $C_c$, evaluating all pairwise partition merges and always retaining the best one until no more improvements can be made. Given a partition configuration, their probability distribution are estimated by counting the frequencies of each different partition setting in the population.

The model building process is outlined in Algorithm 1. It involves a very expensive computational task as the determination of $C_c$ for each tested model, requires the model to be fitted against the (large) population. The method has $O(n^3)$ complexity over the combined complexity criterion evaluations as line 5 iterates over pairs of variables. This can be intuitively seen as $\binom{n}{2}$ has complexity $O(n^2)$.

## *3.1 Hybridization of eCGA*

EDAs in general through model learning and sampling implement an idealized crossover operator, being very efficient at combining high-quality substructures/building-blocks. As they have no explicit variational (mutation) operators to discover new building-blocks on the go, EDAs must rely only on large population sizes that ensure building-block supply and accurate decision making [7]. With too small population sizes EDAs may not be able to efficiently discover building-blocks.

To alleviate this issue, a commonly used efficiency enhancement mechanism is the hybridization with local-search techniques. In the case of the eCGA, incorporating local-search in the subsolution search space leads to better results and greater robustness [13].

Another study had shown that a simple bit-flipping hill-climber applied to all individuals can reduce the population requirements by an order of magnitude even on deceptive problems [5]. This kind of hybridization is easily parallelizable as independently improving each individual through local-search is an embarrassingly

**Algorithm 2.** Correlation guided model-building

```
1  Build initial model m where each variable is an independent partition;
2  Compute M_R and M_T;
3  if this is the first generation and there are no significant values in M_T then
4      Request a bigger population and suggest using univariate EDAs or performing
       search for higher order interactions;
5      Halt the search;
6  repeat
7      [p,q] ← StrongestInteraction(m, M_R, d);
8      Form new model m' based on m but with p and q merged into a joint partition;
9      Evaluate combined complexity criterion C_c(m');
10     if m' improves over m then
11         m ← m';
12 until No improvement was found ;
```

parallel task. Finally, as local-search moves all non-locally-optimal sub solutions to local-optima, it also reduces entropy, easing the model building process. As it affects variable correlation and entropy, we also study how local-search enhancement assist correlation guided model-building. We use a simple greedy search that processes the $b_p$ most probable partition configuration of each partition and retains the one that maximizes the objective function value.

### 3.2 Guided Linear Model Building

When searching for a proper MPM, eCGA greedily searches the space of possible models evaluating *all* pairwise partition merges. The model is extended sequentially with the best possible improvement obtained by joining modules. The main idea of the proposed search is to not process the list of possible extensions blindly and exhaustively: the best extension(s) to be considered are based on correlation analysis between the partitions. The search stops immediately when the model extension does not improve the combined complexity criterion. The reasoning behind this is that all other partition merges that are not analyzed would have (according to the correlation based measurements) lower degree of interactions then the last proposed extension. If that extension was rejected by the $C_c$ then all the remaining ones would be also discarded, there is no point to continue the analysis.

The correlation guided model-building is presented in Algorithm 2. $M_R$ contains the absolute values of the correlation coefficient matrix and has self-correlations set to zero ( $M_R(x,x) = 0$). $M_T$ is a result of a a t-test that contains the probabilities of getting the correlations observed in $M_R$ by random chance.

$d$ is a module-wise metric which operates on $M_R$ and takes into account only first order interactions like the ones described in subsection 2.2. The metric could be extended to cover all interactions but with the cost of the complexity overhead presented in Equation 12.

The method $StrongestInteraction$ identifies the best possible interaction(s) according to a module-wise metric $d$ which in turn is based on the raw data provided in $M_R$. Please note that this matrix has to be computed only once in the beginning. This computation has a complexity of $O(n^2)$ in terms of *elementary operations*[1]. Computing $M_R$ is cheaper then evaluating once the combined complexity criterion: as the $C_c$ has to evaluate the model fit according to the population it has a complexity of $O(Nn)$ where $N$ is the population size and $n$ the problem size. From population sizing theory [25] $N > n$, therefore $O(Nn) \geq O(n^2)$.

In terms of $C_c$ evaluations, the proposed method is very efficient, being linear.

If on the first run of the model-building no statistically significant interactions can be detected the method stops and request a bigger population/sample size for safe detection of first interactions (line 3-5 of Algorithm 2). For the cases where it may be possible to not have any useful interactions it also suggest applying other techniques:

[1] Not to be confused with the complexity in terms of $C_c$ evaluations.

- univariate models for simple problems like OneMax [6] where applying complex methods are an overkill;
- for hard problems, applying other search techniques able to detect higher order, multivariate interactions.

On noisy problems, line 7 can be changed to retrieve the best $j$ suggested model extensions. When $j$ is a constant, analyzing all $j$ extensions and retaining the best one does not change the complexity in terms of $C_c$ evaluations of the model-building.

## *3.3 Test Suite*

We test the eCGA with correlation guided model-building on two problems that combine the core of two well known problem difficulty dimensions for building blocks (BB):

- Intra-BB difficulty: *deception* due trap functions.
- Extra-BB difficulty: non-linear dependencies due to hierarchical structure, which at a single hierarchical level can be interpreted as exogenous *noise* – generated by interactions from higher levels.

### 3.3.1 Concatenated k-Trap Function

Deceptive functions are among the most challenging problems as they exhibit one or more deceitful optima located far away from the global optimum. The basins of attraction of the local-optima are much bigger than the attraction area of the optimal solutions, thus following the objective function gradient will mislead the search most of the time.

Order $k$ deceptive trap function or simply k-Trap, is a function of unitation (its value depends only on the numbers of 1's in the input string), based on two parameters $f_{high}$ and $f_{low}$ which define the degree of deception and the fitness signal-to-noise ratio.

Let $u$ be the unitary of the binary input string. Then the k-trap function is defined as:

$$trap_k(u) = \begin{array}{ll} f_{high} & \text{, if } u = k; \\ f_{low} \times \frac{k-1-u}{k-1} & \text{, otherwise.} \end{array} \tag{18}$$

Here we use a k-Trap function based on $k = 4$, $f_{high} = 1$ and $f_{low} = 0.75$.

Concatenating $m$ copies of this trap function gives a global additively separable, boundedly deceptive function over binary strings:

$$f_d(x) = \sum_{i=0}^{m-1} trap_k \sum_{j=ki}^{ki+k-1} x_j \tag{19}$$

### 3.3.2 Hierarchical XOR

Hierarchical problems have a gross-scale building-block structure but they are not additively separable. Higher level interactions may be only interpreted after solving

all the subproblems from lower levels. Therefore, the correct partitions can only be discovered sequentially, which is in contrast with the uniformly scaled case where the problem structure can be captured immediately with a sufficiently large sample size.

As building-blocks in hierarchical problems have multiple context-optimal settings - allowing multiple solutions for each subproblem, the contribution of building-blocks to the objective function is separated from their meaning. This conceptual separation induces non-linear dependencies between building-blocks: providing the same objective function contribution, a building-block might be completely suited for one context whilst completely wrong for another one. Therefore, the fitness of a building-block can be misleading if it is incompatible with its context. At particular hierarchical levels this fitness variance resulting from higher order non-linear interactions resembles the effect of exogenous noise.

The hierarchical test problem under consideration is the hXOR [24] is defined on binary strings of the form $x \in \{0,1\}^{2^p}$, where $p$ is the number of hierarchical levels. This problem is based on the complemental relation between two sub-modules. This relation is checked with the help of a Boolean function $h$ which determines if sub-blocks are valid in their current context, forming exclusive disjunction.

Let $L = x_1, x_2, \ldots, x_{2^{p-1}}$ be the first half of the binary string $x$ and $R = x_{2^{p-1}+1}, x_{2^{p-1}+2}, \ldots, x_{2^p}$ the second one. Then $h$ is defined as:

$$h(x) = \begin{array}{ll} 1 & \text{, if } p = 0; \\ 1 & \text{, if } h(L) = h(R) = 1 \text{ and } L = \bar{R}; \\ 0 & \text{, otherwise.} \end{array} \tag{20}$$

$\bar{R}$ stands for the bitwise negation of $R$.

Based on $h$ the hierarchical XOR is defined recursively:

$$hXOR(x) = hXOR(L) + hXOR(R) + \begin{array}{l} |x|, \text{ if } h(x) = 1; \\ 0, \text{ otherwise.} \end{array} \tag{21}$$

where $|x|$ denotes the length of $x$.

At each level $p > 0$ the $hXOR(x)$ function rewards a block $x$ if the two composing sub-blocks are valid and complemental. Otherwise the contribution is zero.

The hXOR has two global optima, composed by half zeros and half ones. At the lowest level the problem has $2^{n/2}$ local optima where $n$ is the problem size.

In order to be able to control the complexity of the problem we introduce the not fully hierarchical version of the $hXOR$ denoted by $lhXOR$. Here pairwise hierarchical combinations of blocks are evaluated and valid combinations are rewarded up to the level $l$.

The shuffled version of the problem is used, where the tight linkage is disrupted by randomly reordering the bits.

## 3.4 Performance of the Modified eCGA

The linear runtime of the correlation guided model-building in eCGA provides a huge *qualitative* advantage over the $O(n^3)$ classic model-building complexity. A heuristic based model-building with a $O(n^2)$ complexity had been shown to speed up the eCGA up to more than 1000 times [5].

Therefore, instead of providing a *quantitative* run-time comparison between eCGA with the proposed and the classical model-building, we concentrate the empirical investigation on the scaling, model quality – number of generations until convergence and the effect of hybridization.

### 3.4.1 Test Setup

We tested the correlation guided eCGA, henceforth denoted by eCGA$^*$ with and without local-search hybridization (denoted as eCGA$^*_h$) on concatenated 4-Trap and lhXOR with $l = 3$ for problem sizes $psize = \{32, 64, 256\}$. Population sizes are $15\,psize$ for eCGA$^*_h$ and $55\,psize$ for eCGA$^*$. These values were not tuned. A number of 10 runs were averaged for each test case. Results are presented in Figure 1 and discussed in the followings.

### 3.4.2 Analysis

In all cases the algorithms have found a global optima and the correct structures.

Figure 1 shows the number of function evaluations needed per problem size.

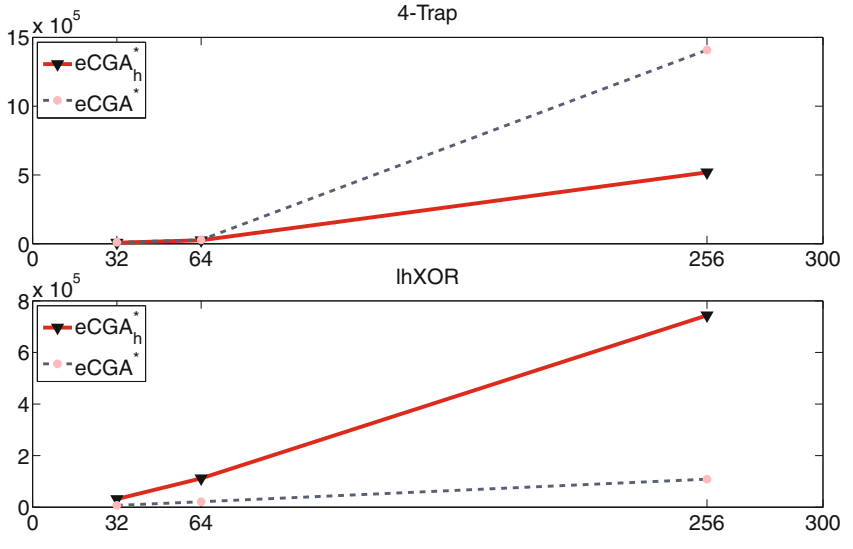



**Fig. 1** Scaling of the correlation guided eCGA$^*$ with an without local-search hybridization (eCGA$^*_h$) on concatenated 4-Trap and lhXOR with $l = 3$ for problem sizes $psize = \{32, 64, 256\}$

Local search reduces entropy thus as expected the eCGA$_h^*$ showed a very accurate - noise free model-building, all problems being solved within 3 generations. Nevertheless, local-search is expensive especially for big problem sizes.

eCGA showed somehow similar model-building qualities on hXOR solving the problems in 4-8 generations. On 4-Trap with 256 bit the convergence took a long time even if the correct linkages were detected. The reason for is that the optimal blocks formed by all ones are much more unlikely in the beginning of the search and there are also many noisy non-optimal configurations. In this cases it takes a long time for the best configuration to takeover.

We think there is a possibility to safely speed up this takeover by accelerating trends. Sub solutions whose probability diminishes over several epochs are replaced by sub solutions that are continually expanding. This operator can have the same effect as the local-search but without the computational burden of the local search.

## 4 Extended Bayesian Model Building

A Bayesian networks is a probabilistic graphical model that depicts a set of random variables and their conditional independence via a directed acyclic graph. It represents a factorization of a multivariate probability distribution that results from an application of the product theorem of probability theory and a simplification of the factors achieved by exploiting conditional independence statements of the form $P(A|B,X) = P(A|X)$, where A and B are attributes and X is a set of attributes.

The represented joint distribution is given by:

$$P(A_1,\ldots,A_n) = \prod_{i=1}^{n} P(A_i|par(A_i)) \tag{22}$$

where $par(A_i)$ denotes the set of parents of attribute $A_i$ in the directed acyclic graph that is used to represents the factorization.

Bayesian networks provide excellent means to structure complex domains and to draw inferences. They can be acquired from data or be constructed manually by domain experts (a tedious and time-consuming task).

One of the most challenging task in dealing with Bayesian networks is learning their structures, which is an NP-hard problem [2, 3]. Most algorithms for the task of automated network building from data, consist of two ingredients: a search method that generates alternative structures and an evaluation measure or scoring function to assess the quality of a given network by calculating the goodness-of-fit of a structure to the data.

Due to the computational cost implications, most of the algorithms that learn Bayesian network structures from data use a Greedy heuristic local search to find a good model, trading accuracy for tractability and efficiency.

The greedy hill-climbing methods traverse the search space starting in an initial solution and doing a finite number of steps. At each step the algorithm only considers local changes, i.e. neighbor network structures, and chooses that resulting in the greatest improvement of some discriminative metric. The algorithm stops when there is no local change yielding an improvement.

In Bayesian network learning, the search operators do arc addition, arc deletion and arc reversal. Search operators have to take care of avoiding to introduce directed cycles in the graph. There are $O(n^2)$ possible changes, $n$ being the number of variables. With respect to the starting solution, the empty network is usually considered although random starting points or perturbed local optima are also used, specially in the case of iterated local search.

By adding, reversing, removing only one arc at the time, these network search algorithms can not find and express multivariate interactions that only manifest at a synergic level like the parity function. Here, adding edges between less than $k$ nodes, where $k$ is the size of the block containing the multivariate interaction, will not result in any improvement.

EDAs replace the evolutionary search operators by learning a probability distribution from the population which encodes the relationships between the individual components and sampling new individuals from that learned distribution. Among multivariate models the usage of Bayesian networks for this task have been one of the most successful approaches [12, 17, 16].

The probability network encoded, representing the interdependency of variables is more complex than of other approaches, like the eCGA, which cluster variables components in independent groups and learn a joint probability distribution for each group. The rich modeling capabilities come at a cost, the computation time and complexity to construct the best Bayesian network hugely increases, being impossible to search through all possible models for larger problem sizes. A lots of current research in EDAs is focused on developing heuristics capable to quickly find good models.

In the following we describe a method based on the extension of pairwise interactions, that is able to select *all* relevant parents for an attribute in one step, thus enabling the finding and expression of relationships not manifesting at pairwise level.

### *4.1 Multi-parent Search*

The goal is to construct a Bayesian network by detecting the set $par(A_i)$ for each attribute. Adding the edges between and attribute and its parents must not result in a cycle.

In our proposed approach, we build a feature space as described in Section 2.1 by mapping interactions among group of variables up to a bounded size $k$, which is a parameter of the method. For each attribute $A_i$, we sequentially assign the potential parent set $par^*_j(A_i)$ to be the $j^{th}$ highest interacting subset of variables, quantified by $d_I$ defined in Equation 13. In this way we process a prefixed top $S_{nr}$ interacting subsets for each attribute. For every subset, we process each potential parent $p^*(A_i)$, $p^*(A_i) \in par^*_j(A_i)$, and if adding an edge between the attribute and its potential parent does not result in a cycle, $p^*(A_i)$ becomes a parent of $A_i$: $par_j(A_i) = par_j(A_i) \cup p^*(A_i)$. From all the obtained and tested parent subsets for each attribute, we choose attribute and its parents that maximizes a given discriminative scoring function, in our case the Bayesian Dirichlet metric [11].

The search stops when we determined the parents of each attribute, or when considering the extension of the network does not result in improvements. The outline of this parent search procedure is outlined in Algorithm 3.

**Algorithm 3.** Constructing a Bayesian network that is able to capture all kind of interactions up to a prefixed order $k$

**input :** $data$, $k$
**output:** $BN$

$BN \leftarrow EmptyNetwork()$;
**foreach** *c from the possible combinations of variables up to size k* **do**
  $F \leftarrow [F, \oplus(data(c))]$;
**foreach** *attribute A, iterating by i* **do**
  **foreach** *extended variable e from F, iterating by j* **do**
    $M(i,j) = I(A_i, e_j)$;
Sort by highest interactions first $M \leftarrow sort(M, 2, 'descend')$;
**repeat**
  **for** *i=1:n* **do**
    **if** *HasParents(i)* **then**
      continue;
    **for** *j=1:$S_{nr}$* **do**
      $par^*(A_i) \leftarrow DecodeParents(i,j)$;
      $par(A_i) \leftarrow EliminateCycles(par^*(A_i))$;
      $BN^* \leftarrow ExtendNetwork(BN, A_i, par(A_i))$;
      **if** *Score(BN*) > Score(BN)* **then**
        $BN \leftarrow BN^*$;
**until** *No improvement was found* ;
**return** $BN$;

## 4.2 Test Samples

To assess the performance of the proposed search method, we build some artificially generated test samples that contain various types of multivariate interactions. We consider 10 variables $X_1, \ldots, X_{10}$, sampled 5000 times.

The *first* data set contains two highly noisy features:

1. A highly noisy $\sum_{i=1}^{4} X_i = 3) \mapsto (X_5 = 1)$ conditioning, where whenever three out of the four first variables are one, $X_5$ is also set to 1 with a probability of 0.5.
2. A noisy feature based on a parity function conditioning

$$parity([X_6, X_7, X_9, X_{10}]) \mapsto (X_8 = 0)$$

if variables $X_6, X_7, X_9, X_{10}$ have an even number of ones, $X_8$ is set to 0 with a 0.8 probability.

In the *second* dataset we reduce the amount of explicit noise but introduce an overlap between the two features, which are:

1. We have the

$$\sum_{i\in\{1,2,3,4,9,10\}} X_i = 3 \quad \mapsto (X_5 = 1)$$

noisy conditioning, where whenever half of the variables $X_1, X_2, X_3, X_4, X_9, X_{10}$ are 1, $X_5$ is also set to 1 with a probability of 0.95.

2. Again a noisy feature based on a parity function conditioning

$$parity([X_1, X_2, X_6, X_7, X_9, X_{10}]) \mapsto (X_8 = 0)$$

if variables $X_1, X_2, X_6, X_7, X_9, X_{10}$ have an even number of 1's, $X_8$ is set to 0 with a 0.9 probability.

In the *third* dataset we introduce an interplay between the features, where the realization of the first feature may inhibit the realization of the second one:

1. A noisy $\sum_{i=1}^{4} X_i = 3) \mapsto (X_5 = 1)$ conditioning, where whenever three out of the four first variables are one, $X_5$ is also set to 1 with a probability of 0.5.
2. A feature based on a sum function conditioning alike of the first feature

$$\sum_{i\in\{6,7,9,10\}} X_i = 3 \quad \mapsto (X_8 = 0)$$

but which may be short circuited by the realization of the first feature: if $X_5 = 1$, $X_8$ is not expressed.

The generation of these datasets is detailed in Listings 1-3.

**Listing 1: MATLAB code for generating the first set of samples.**

```
% generate uniformly distributed random samples
n = 10; N = 5000;
data = round(rand(N,n));

% for each sample
for i = 1:N
% define a noisy feature on the first 4 variables:
% if 3 out of the 4 bits are set to 1,
% set the 5th variable to 1 with a probability of 50%
    if ((sum(data(i,1:4)) == 3) && (rand < 0.5))
        data(i,5) = 1;
    end

% define a parity feature on variables 6, 7, 9, 10
% afecting variable 8th with a probability of 80%
    if ((mod(sum(data(i,[6 7 9 10])),2) == 0) && (rand < 0.8))
        data(i,8) = 0;
    end
end
```

**Listing 2: MATLAB code for generating the second set of samples.**

```
% generate uniformly distributed random samples
n = 10; N = 5000;
data = round(rand(N,n));
% for each sample
for i = 1:N
% define a noisy feature on the first 4 and variables 9, 10:
% if 3 out of the 6 bits are set to 1,
% set the 5th variable to 1 with a probability of 95%
    if ((sum(data(i,[1 2 3 4 9 10])) == 3) && (rand < 0.95))
        data(i,5) = 1;
    end
% define a parity feature on variables 1, 2, 6, 7, 9, 10
% afecting variable 8th with a probability of 90%
    if ((mod(sum(data(i,[1 2 6 7 9 10])),2) == 0) && (rand <
        0.9))
        data(i,8) = 1;
    end
end
```

**Listing 3: MATLAB code for generating the third set of samples.**

```
% generate uniformly distributed random samples
n = 10; N = 5000;
data = round(rand(N,n));
% for each sample
for i = 1:N
% define a noisy feature on the first 4 variables:
% if 3 out of the 4 bits are set to 1,
% set the 5th variable to 1 with a probability of 75%
    if ((sum(data(i,1:4)) == 3) && (rand < 0.75))
        data(i,5) = 1;
    end
 if 3 out of the bits from variables 6, 7, 9, 10 are 1
% and the value of the 5th variable is 0, then variable 8 is 0
    if ((sum(data(i,[6 7 9 10])) == 2) && (t(i,5) == 0))
        data(i,8) = 1;
    end
end
```

### *4.3 Model Building Performance*

For each test case, we generated 50 instances and tested the proposed method against a classical Bayes network model building, which extends a current model by performing one arc operation at the time, and using the Bayesian information criterion [23] for scoring and protection against over fitting. The allowed in degree in the classic search and the $k$ parameter in the proposed method was set to 6, thus both methods could consider up to 6 parents. The number of analyzed possible parent sets $S_{nr}$ was set to 5.

For each batch of 50 runs, we recorded the best network found, its score, the worst and the average score. Because the data is stochastically generated and incorporates noise, the exact quantity of this values is of a little importance. The same network structure will score differently when evaluated on different noisy samples. Nevertheless, these values may be used to make qualitative assessments, in the cases where the worst result of one method surpasses the best network score or the average score found by the other method.

More important aspect regards the methods ability to extract the same structure from different samples of noisy data. We measure this robustness by comparing the best and worst scoring network out of each batch of 50 runs. If the adjacency matrix of the two networks is not similar (one can not be transformed into the other one by using only row and column swapping), implies that the search method may find different network topologies on different runs.

The numerical scores are presented in Table 3. The plot of the best networks found for each of the three cases are presented in Figures 2, 3, 4.

In the first case, where there is a high amount of noise, the classical approach can not detect the real structure, the network is filled with spurious connections where often an attribute is accounted as the parent of all other attributes following after. Observe for example in Figure 2, that Node 1 is attributed as parent for all other nodes. On the other hand, even with such a high amount of noise, the extended multi-parent search is able to detect the correct topology of the network.

For the second dataset it is expected that the classical approach is not able to detect the parity, multivariate interaction as it would need to add at least six arcs at once to reveal this interaction. Furthermore, as this feature overlaps with the other feature which also spans across six variables, the method is unable to account for useful relations and returns the empty network, without edges, in all cases. Please note by looking at the best and worst score in Table 3 for test suite 2, how the same empty network may score differently when presented with different test data. The proposed method is again able to find the correct structure, as we allowed the feature space exploration up to six combined variables, which is also the length of the highest multivariate relation.

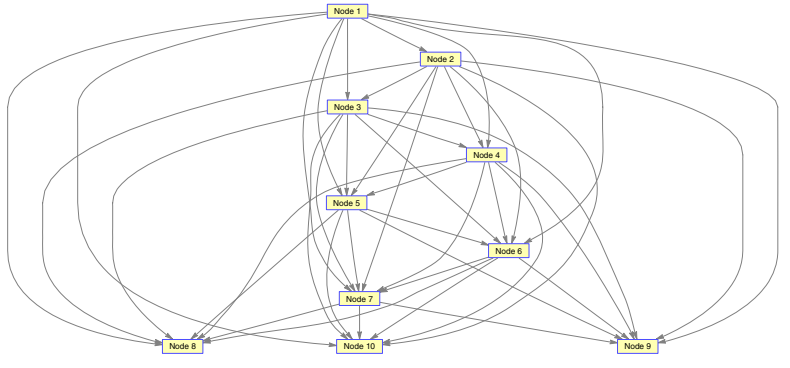
On the third case, the classical method is able detect the interactions influencing attribute 5 and its relation to attribute 8, while failing to model the synergic interaction of the other variables. Sometimes, as depicted in Figure 4 A, it reports attribute 5 as linked to other variable different from attribute 8, but this result is rarely achieved. By modeling all interactions up to size six in the feature space, the multi-parent search is able to correctly decipher the interplay between the two features.

For all cases, as shown above, the extended search found qualitatively better networks; the worst scoring results of the proposed method were always better than the best results returned by the classical method. As it does not contain stochastic components, the proposed showed robustness, finding the same topology on different runs.

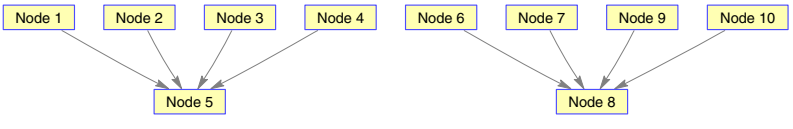
Model building in EDAs starts by finding and modeling pairwise dependencies. When no such relations are available a successful approach must do $k$-wise

**Table 3** The performance of the proposed and classical methods on the three test suites. The multi-parent extended search worse results are better than the best scores obtained by the classical search method in all cases

| | Best | Worst | Average | Std. | Robust |
|---|---|---|---|---|---|
| *Test suite 1* | | | | | |
| Classic | -34128.06 | -34269.11 | -34204.73 | 32.89 | No |
| Extended | -33662.24 | -33826.25431 | -33748.88 | 37.67 | Yes |
| *Test suite 2* | | | | | |
| Classic | -33876.23 | -34040.17 | -33950.13 | 36.63 | Yes |
| Extended | -33063.69 | -33308.58 | -33192.06 | 52.87 | Yes |
| *Test suite 3* | | | | | |
| Classic | -34172.68 | -34298.18 | -34228.73 | 27.03 | No |
| Extended | -33915.10 | -34065.95 | -33982.97 | 29.87 | Yes |




(A)




(B)

**Fig. 2** Best networks found by the classical method (A) and the multi-parent extended search (B) on the first test suite

multivariate interaction search. The proposed method has demonstrated a great ability to identify simpler and synergic multivariate interactions even in the case of noisy feature interplay, where considering one edge addition at the time is fruitless. While it uses a small number of model evaluations and it is much more effective than doing greedy search using a $k$-wise stochastic edge search operator, the extended multi-parent search is still very costly in terms of building and evaluating the feature space. Due to the combinatorial explosion of the multivariate interaction

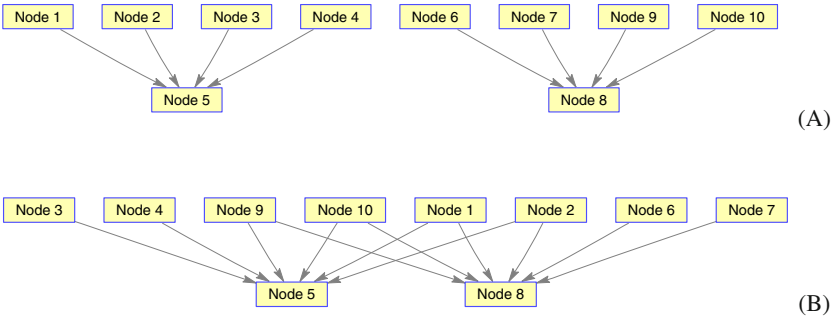



**Fig. 3** Best networks found by the classical method (A) and the multi-parent extended search (B) on the second test suite

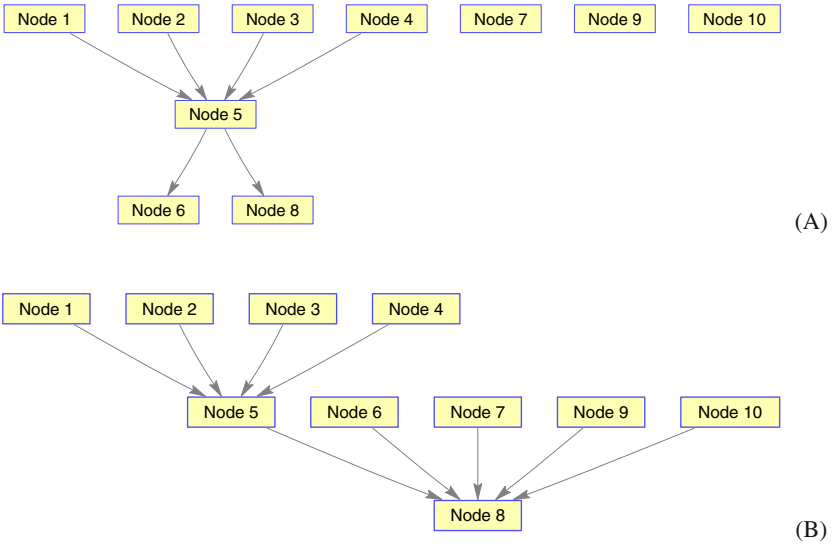



**Fig. 4** Best networks found by the classical method (A) and the multi-parent extended search (B) on the third test suite

possibilities as problem sizes increase, its scalability is not sustainable, but nor is the scalability of other approaches.

For limited problem sizes it can be used successfully as it is computationally more efficient than other methods, not requiring model evaluations against the data once the feature space has been built. Furthermore, the feature space building is highly parallelizable.

Future experiments may regard the analysis of methods that build only a small part of the feature space.

## 5 Conclusions

EDAs infer and exploit structure from the problems under consideration by building a probabilistic model and sample it. A good model induces an efficient search bias that can render hard problems feasible but such a model does not come cheap. Model-building in competent EDAs often require the consideration and evaluation of a very large number of potential models.

This chapter had proposed the usage of pairwise interactions measures like mutual information or various correlation coefficients to assist model-building. When branching from the current model in search for a better one, pairwise interaction analysis is employed to quickly identify the most promising extension, avoiding the need for testing all the other extension possibilities. This can alleviate the model search cost by orders of magnitudes and even facilitate the finding of qualitatively better models. As we had shown, the usage of potential model selection by means of pairwise interaction analysis, reduces the model-building complexity of the eCGA from $O(n^3)$ to linear, while still inferring the correct problem structures.

In another case study, this technique was used for efficient $k$-wise multivariate interaction search. Namely, a search algorithm for constructing Bayesian networks from data was developed, which showed remarkable ability to infer the difficult relationships between variables.

For extension to richer alphabets one must use mutual information for quantifying the pairwise interactions between attributes, as zero correlation between two variables implies statistical independence only in the dichotomous case.

Future work will consider applying pairwise interaction guidance to enhance and qualitatively improve model-building in other competent EDAs.

## References

1. Baluja, S.: Incorporating a priori knowledge in probabilistic-model based optimization. Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications, 205–219 (2006)
2. Bouckaert, R.: Properties of Bayesian belief network learning algorithms. In: Uncertainty in artificial intelligence: proceedings of the Tenth Conference, p. 102. Morgan Kaufmann, San Francisco (1994)
3. Chickering, D., Geiger, D., Heckerman, D.: Learning Bayesian networks is NP-hard. Tech. Rep. MSR-TR-94-17, Microsoft Research (1994)
4. Coffin, D.J., Smith, R.E.: Why is parity hard for estimation of distribution algorithms? In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, p. 624. ACM, New York (2007), http://doi.acm.org/10.1145/1276958.1277084
5. Duque, T.S., Goldberg, D.E., Sastry, K.: Enhancing the efficiency of the ECGA. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 165–174. Springer, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-87700-4_17
6. Eshelman, L.: On crossover as an evolutionarily viable strategy. In: Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 61–68. Morgan Kaufmann Publishers, San Francisco (1991)

7. Goldberg, D.E., Sastry, K., Latoza, T.: On the supply of building blocks. In: Spector, L., Goodman, E.D., Wu, A., Langdon, W.B., 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 (GECCO-2001), pp. 336–342. Morgan Kaufmann, San Francisco (2001), http://www.cs.bham.ac.uk/~wbl/biblio/gecco2001/d03a.pdf
8. Harik, G.: Linkage learning via probabilistic modeling in the ECGA. Tech. Rep. IlliGAL Report no. 99010, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign (1999)
9. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. IEEE-EC 3(4), 287 (1999)
10. Hauschild, M.W., Pelikan, M., Sastry, K., Goldberg, D.E.: Using previous models to bias structural learning in the hierarchical BOA. In: GECCO 2008: Proceedings of the 10th annual conference on Genetic and evolutionary computation, pp. 415–422. ACM, New York (2008), http://doi.acm.org/10.1145/1389095.1389172
11. Heckerman, D., Geiger, D., Chickering, D.: Learning Bayesian networks: The combination of knowledge and statistical data. Machine learning 20(3), 197–243 (1995)
12. Larrañaga, P., Etxeberria, R., Lozano, J., Peña, J.: Combinatonal Optimization by Learning and Simulation of Bayesian Networks. In: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, p. 352. Morgan Kaufmann Publishers Inc., San Francisco (2000)
13. Lima, C.F., Sastry, K., Goldberg, D.E., Lobo, F.G.: Combining competent crossover and mutation operators: a probabilistic model building approach. In: GECCO 2005, pp. 735–742. ACM, NY (2005), http://doi.acm.org/10.1145/1068009.1068131
14. Ocenásek, J., Schwarz, J.: The parallel bayesian optimization algorithm. In: Proceedings of the European Symposium on Computational Inteligence, pp. 61–67. Springer, Heidelberg (2000), http://www.fit.vutbr.cz/research/view_pub.php?id=6434
15. Pelikan, M.: Hierarchical Bayesian optimization algorithm: Toward a new generation of evolutionary algorithms. Springer, Heidelberg (2005)
16. Pelikan, M., Goldberg, D.E.: Escaping hierarchical traps with competent genetic algorithms. In: Spector, L., et al. (eds.) GECCO 2001, pp. 511–518. Morgan Kaufmann, San Francisco (2001), http://citeseer.ist.psu.edu/440187.html
17. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. In: Langdon, W.B., et al. (eds.) GECCO 1999, vol. I, pp. 525–532. Morgan Kaufmann Publishers, San Fransisco (1999)
18. Pelikan, M., Hartmann, A.K., Sastry, K.: Hierarchical BOA, cluster exact approximation, and ising spin glasses. 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. 122–131. Springer, Heidelberg (2006)
19. Pelikan, M., Sastry, K., Goldberg, D.: Sporadic model building for efficiency enhancement of the hierarchical BOA. Genetic Programming and Evolvable Machines 9(1), 53–84 (2008)
20. Pelikan, M., Sastry, K., Goldberg, D.E.: iBOA: the incremental bayesian optimization algorithm. In: GECCO 2008: Proceedings of the 10th annual conference on Genetic and evolutionary computation, pp. 455–462. ACM, New York (2008), http://doi.acm.org/10.1145/1389095.1389177
21. Rissanen, J.: Modelling by the shortest data description. Automatica 14, 465–471 (1978)

22. Sastry, K., Goldberg, D.E., Llora, X.: Towards billion-bit optimization via a parallel estimation of distribution algorithm. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 577–584. ACM, New York (2007), http://doi.acm.org/10.1145/1276958.1277077
23. Schwarz, G.: Estimating the dimension of a model. The annals of statistics 6(2), 461–464 (1978)
24. Watson, R.A., Pollack, J.B.: Hierarchically consistent test problems for genetic algorithms: Summary and additional results. In: Brave, S. (ed.) GECCO 1999: Late Breaking Papers, Orlando, Florida, USA, pp. 292–297 (1999)
25. Yu, T.L., Sastry, K., Goldberg, D.E., Pelikan, M.: Population sizing for entropy-based model building in discrete estimation of distribution algorithms. In: Lipson, H. (ed.) GECCO, pp. 601–608. ACM, New York (2007), http://doi.acm.org/10.1145/1276958.1277080

# ClusterMI: Building Probabilistic Models Using Hierarchical Clustering and Mutual Information

Thyago S.P.C. Duque and David E. Goldberg

**Abstract.** Genetic Algorithms are a class of metaheuristics with applications in several fields including biology, engineering and even arts. However, simple Genetic Algorithms may suffer from exponential scalability on hard problems. Estimation of Distribution Algorithms, a special class of Genetic Algorithms, can build complex models of the iterations among variables in the problem, solving several intractable problems in tractable polynomial time. However, the model building process can be computationally expensive and efficiency enhancements are oftentimes necessary to make tractable problems practical. This paper presents a new model building approach, called ClusterMI, inspired both by the Extended Compact Genetic Algorithm and the Dependency Structure Matrix Genetic Algorithm. The new approach has a more efficient model building process, resulting in speed ups of 10 times for moderate size problems and potentially hundreds of times for large problems. Moreover, the new approach may be easily extended to perform incremental evolution, eliminating the burden of representing the population explicitly.

## 1 Introduction

Evolutionary Algorithms (EA) [4] [9] have been successfully used in several different applications involving search, optimization and machine learning problems. Goldberg [5] presents a design-decomposition methodology for successfully designing scalable Genetic Algorithms (GAs). A GA that can solve hard problems accurately, efficiently and reliably is called a competent GA. These GAs can solve problems that are intractable for traditional methods in a tractable polynomial time.

Thyago S.P.C. Duque · David E. Goldberg
Illinois Genetic Algorithms Laboratory,
University of Illinois at Urbana Champaign,
104 S. Mathews Ave, 117 Transportation Bldg, Urbana, IL
e-mail: thyago,deg@illigal.ge.uiuc.edu

Y.-p. Chen (Ed.): Exploitation of Linkage Learning, ALO 3, pp. 123–137.
springerlink.com 

Estimation of Distribution Algorithms (EDA) [10] have been particularly successful on solving hard problems competently. These algorithms build probabilistic models that summarize the information of the current population, and then sample these models to generate a new population. By using complex models like Bayesian Networks or Marginal Product Models (MPM), these algorithms can learn the structure of the problem and use this information to guide the search.

The Extended Compact Genetic Algorithm (ECGA) [7] is one of such EDAs, and it uses the MPM to model non-overlapping iteration among variables that form a Building Block (BB) [4] [9].

Although EDAs scale polynomially, they are still computationally expensive. Therefore, in order to solve large problems it is oftentimes necessary to enhance the efficiency [16] of the algorithm using techniques like parallelization [1], hybridization [6] [19] [18], time continuation [15], evaluation relaxation [14] and incremental evolution [8], [13].

Particularly, the model building process of the ECGA is time consuming. This issue can be partially addressed using a cache structure [11], relaxing the model building process [2] or performing sporadic model building [12]. In this paper we propose a new model building algorithm, called ClusterMI, that aims on reducing the computational complexity of the model building process. Moreover, the new model building mechanism may be used to perform incremental model building, which can further improve the performance of the algorithm and will be explored on future work.

This paper is organized as follows: Section 2 presents a introduction to the ECGA and the Dependency Structure Matrix Genetic Algorithm (DSMGA), section 3 presents ClusterMI, a new approach for model building in EDAs that combine ideas from both ECGA and DSMGA. Section 4 presents the results of the new approach and compares it with the ECGA. Section 5 discusses future work, including initial ideas for incremental evolution. Section 6 presents final considerations and concludes the paper.

## 2 Background

This section briefly reviews the ECGA and the DSMGA, discussing computational aspects of the model building process on these algorithms.

As an EDA, the ECGA [7] follows the template presented on algorithm 1. Particularly, the ECGA uses the MPM as a model of the current population. The MPM has two components: (I) a partition over the variables, defining which variables are independent and which variables are linked, and (II) a probability distribution over each partition. Table 1 presents one example of a MPM that defines the partition $[x_0]$, $[x_1, x_3]$, $[x_2]$ over a string of four bits ($[x_1, x_2, x_3, x_4]$).

The model building process of the ECGA finds a partition that appropriately represents the population by greedily optimizing the Combined Complexity Criterion (CCC) [7] using algorithm 2. The CCC reflects the minimum description length (MDL) bias of the ECGA and it is presented in equation (1), where $n$ represents

**Algorithm 1.** A template for EDAs

```
1. Start with random population.
4. Do:
5.     Evaluate the population
6.   Select a group of good individuals
7.   Build a model to summarize the population
8.   Sample the built model
5. while (stop criteria not met)
```

**Table 1** A Marginal Product Model defines a joint distribution over a set of four variables $[x_1, x_2, x_3, x_4]$. In this example, $x_1$ and $x_3$ are linked together and $x_0$ and $x_2$ are independent

| $[x_0]$ | $[x_1, x_3]$ | $[x_2]$ |
|---|---|---|
| $P(x_0 = 0)$=0.3 | $P(x_1 = 0, x_3 = 0) = 0.5$ | $P(x_2 = 0) = 0.5$ |
| $P(x_0 = 1)$=0.7 | $P(x_1 = 0, x_3 = 1) = 0.0$ | $P(x_2 = 1) = 0.5$ |
| | $P(x_1 = 1, x_3 = 0) = 0.0$ | |
| | $P(x_1 = 1, x_3 = 1) = 0.5$ | |

**Algorithm 2.** Greedy search for an appropriated model in the ECGA

```
1. Start with all independent variables.
2. improvement = IMPOSSIBLE
3. best = CCC(current model)
4. Do:
5.     For each partition [i]
6.     For each partition [j], j not equals i
7.         i+j = Merge i and j
8.       ccc = CCC(i+j)
9.             if (ccc < best)
10.                (bi, bj) = save(i, j)
11.                best = ccc
12.                improvement = POSSIBLE
13.    if (improvement = POSSIBLE)
14.        Merge the saved (bi, bj)
15.    Update current model
16. while (improvement = POSSIBLE)
```

the population size, $m$ the number of partitions, $M_i$ the $i$-th partition and $E(M_i)$ the entropy of the partition $M_i$. Observe that calculating the entropy involves estimating the probability distribution over the partition, an O($n$) step.

$$CCC = n \cdot log(n) \cdot \sum_{I=0}^{m} E(M_i) \cdot 2^{Size[M_i]} \tag{1}$$

**Algorithm 3.** The use of a cache structure in the model building for ECGA

```
 1. Start with all independent variables.
 2. For each partition [i]
 3.   For each partition [j], j not equals i
 4.     i+j = Merge i and j
 5.     ccc = CCC(i+j)
 6.     cache.store(i, j, ccc)
 7. improvement = IMPOSSIBLE
 8. Do:
 9.   best = CCC(current model)
10.   For every entry (i, j, ccc) in the cache
11.     if (ccc < best)
12.       improvement = POSSIBLE
13.       best = ccc
14.       (bi, bj) = save (i, j)
15.   if (improvement = POSSIBLE)
16.     Merge the saved (bi, bj)
17.     Update the current model
19.     For every entry (i, j, ccc) in the cache
10.       if (i = bi or i = bj or j = bi or j = bj)
20.         cache.remove(i, j, ccc)
21.     For each partition [k]
22.       bi+bj+k = Merge bi+bj and k
23.       ccc = CCC(bi+bj+k)
24.       cache.store(bi+bj, k, ccc)
25. while (improvement = POSSIBLE)
```

The greedy search for the best partition requires $O(\ell^3)$ evaluations of the CCC criterion ($\ell$ indicates the problem size, see figure 3 for empirical evidence or [2] for a complexity analysis). For deceptive functions [5] the algorithm can be relaxed to require $O(\ell^2)$ [2] without losing accuracy, but it is unknown whether this result holds for other problems.

Another common way to speed-up the model building is the use of a cache structure (as implemented in [11]). Algorithm 3 implements the greedy search using a cache structure. This algorithm is guaranteed to produce the same results as the algorithm 2, since it only replaces the calculation of the CCC by a cache lookup.

Algorithm 3 requires $O(\ell^2)$ evaluations of the CCC criterion (see figure 3). These come from line 5, which is executed $O(\ell^2)$ times and line 23, also executed $O(\ell^2)$ times. However, there is the extra cost of managing the cache. Assuming a problem with size $\ell$ and building blocks of size $k$, when the search for the best partition begins at line 8, the cache will have $\ell * (\ell - 1)/2$ entries. When the algorithm finishes its execution at line 25, the cache will have $\ell/k * (\ell/k - 1)/2$ entries of size $O(k)$. Consequently, the size of the cache ranges from $O(\ell^2)$ to $O(\ell^2/k)$. That implies an additional memory requirement almost the size of the population itself.

This work also draws inspiration from the DSMGA [20] [21], a competent GA that uses a Dependency Structure Matrix (DSM) to model the relationship among variables. A clustering method is used to determine groups of variables that strongly interact among each other, forming a BB. To do so, the DSMGA uses a search method and a MDL bias: minimizing the cost of describing a given DSM. Afterwards, the information from the clustering is used to perform BB-Wise crossover.

In the DSMGA, a binary string with length $M_{max} \cdot n_c$ is used to represent a clustering arrangement, with $n_c$ as the number of variables and $M_{max}$ as the maximum number of modules. In this representation, the $(x + yn_c)$-th bit indicates whether or not the $x$-th bit belongs to the $y$-th cluster. Initially a simple GA was used to learn the BB structure, using the described representation as chromosome and the MDL metric as objective function. Later on a binary hillclimber was used to speed up the BB discovery process.

Finally, it is important to notice that both the ECGA and the DSMGA build the models using the Shannon's entropy [17]. As shown by [22] these methods require a population of size $O(2^k \ell log(\ell))$ to accurately build the model. Also, it is known that the DSMGA requires larger populations than the ECGA.

## 3 ClusterMI: A New Approach to Model Building in EDAs

This section describes our approach for model building in EDAs and how it relates with ECGA and DSMGA. We also discuss computational aspects of the new algorithm, showing how it can be superior or inferior to its competitors.

This paper proposes the use of hierarchical clustering and Mutual Information (MI) to perform model building in the ECGA. The new model building algorithm, called ClusterMI (from clustering over mutual information) can be divided in two steps.

First, a matrix is created to store the degree of dependency between every two pairs of variables. The MI between two variables is used for this purpose and it is defined in equation (2). When two variables are completely unrelated (independent), the MI assumes its minimal value of 0. On the other extreme, the MI between a variable and itself is the entropy of that variable.

$$MI(X,Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) log \quad \frac{p(x,y)}{p(x)p(y)} \tag{2}$$

After the MI between every pair of variables is calculated, a clustering method (presented on algorithm 4) is used to determine the best partition of variables. Algorithm 4 is a modified hierarchical clustering algorithm that uses the CCC from ECGA to decide the optimal number of clusters. This algorithm uses the MI (as opposed to the CCC in the ECGA) as a metric to decide which pair of partitions/clusters to merge, and relies on the CCC only to determine whether or not the chosen pair will actually be merged.

We already defined the MI between a pair of variables (equation (2)), however, to properly perform clustering we need to define a similarity measure between groups

**Algorithm 4.** ClusterMI: a simple clustering algorithm to search for the best partition of variables in the ECGA

```
 1. Start with all independent variables.
 2. improvement = IMPOSSIBLE
 3. Do:
 4.   best_mi = -1
 5.     For each cluster [i]
 6.     For each cluster [j], j not equals i
 7.         mi = MI_Cluster(i, j)
 8.             if (mi > best_mi)
 9.                 best_i = i
10.         best_j = j
11.                 best_mi = mi
12.     if (best_mi not equals -1)
13.         bi+bj = Merge best_i and best_j
14.     cccM = CCC(model with bi+bj merged)
15.     cccC = CCC(current model)
16.     if (cccM < cccC)
17.       Update the model accepting bi+bj merge
18.       improvement = POSSIBLE
19. while (improvement = POSSIBLE)
```

of variables. Line 7 of algorithm 4 extrapolates the notion of MI to represent the similarity for clusters of variables. The "MI" between two clusters is defined as the average of the MI between every variable of one cluster and every variable of the other, as presented in equation (3).

$$MI\_Cluster(I,J) = \frac{1}{|I||J|} \sum_{i \in J} \sum_{j \in J} MI(i,j) \tag{3}$$

The model building approach defined by algorithm 4 can be used in step 7 of algorithm 1. The clustering of variables resulting from algorithm 4 is equivalent to a partition of variables in the MPM. The same procedure used to estimate the distribution over partitions can be used to estimate the distribution over the clusters, and the same sampling procedure used for ECGA can be used for ClusterMI. In this sense, ClusterMI can be viewed as an ECGA with a modified greedy search for the best partition.

As for the relation between DSMGA and ClusterMI, both of them use a matrix storing the MI between variables to perform linkage learning. The main difference is that the DSMGA converts the MI matrix into a binary relation (dependent or non-dependent) and uses an MDL metric based on how well the model represents the binary matrix. On the other hand, ClusterMI uses the MI matrix to establish the order in which variables and clusters should be merged, using ECGA's MDL bias (CCC) to decide when to stop the process. Also, in DSMGA clustering is a

blind optimization problem over a binary string while ClusterMI uses an explicit hierarchical clustering algorithm to perform the model building.

Concerning computational complexity of algorithm 4, three aspects will be taken into account. First, algorithm 4 has complexity of $O(\ell^3)$ with respect to the number of evaluations of the similarity among clusters (operation performed on line 7). Observe that lines 1 through 12 of algorithm 4 implement a functionality similar to the one of lines 1 through 13 of algorithm 2 and the same cache structure can be used to reduce it to $O(\ell^2)$[1].

Second, algorithm 4 has complexity $O(\ell)$ with respect to the number of evaluations of the CCC (empirical evidence provided on figure 3). It is clear that concerning evaluations of CCC, ClusterMI is superior to both ECGA and ECGA with cache ($O(\ell)$ v.s. $O(\ell^3)$ and $O(\ell^2)$).

Finally, the memory complexity of algorithm 4 is $O(\ell^2)$, a result from the need to store the MI among every pair of variables (the MI could be calculated inside the algorithm, but that would render it inefficient). A cache structure to speed up the search for the most similar pair of clusters would require additional memory of order $O(\ell^2)$, still resulting in an overall memory complexity of $O(\ell^2)$.

To show that the complexity of ClusterMI is smaller than the complexity of the model building in ECGA (with cache) we still need to show that $O(\ell^2)$ evaluations of the similarity among clusters is better than $O(\ell^2)$ evaluations of the CCC. As mentioned earlier, calculating the CCC involves calculating the entropy of a partition. Assuming a problem of size $\ell$, composed of $m$ BBs of size $k$ and a population size $n = O(2^k \cdot \ell \cdot log(\ell))$, this step has complexity $O(k \cdot 2^k \cdot \ell \cdot log(\ell))$. On the other hand, evaluating the similarity among clusters involves averaging over the pairwise distance for every pair of variables. Since the MI between the variables is stored, this step has complexity $O(k^2)$ (assuming BBs of size k and population properly sized, allowing ClusterMI to build the correct model).

In conclusion, ClusterMI has computational complexity of $O(k \cdot 2^k \cdot \ell^2 \cdot log(\ell))$ and memory complexity of $O(\ell^2)$. The model building in the ECGA (with cache) has computational complexity of $O(k \cdot 2^k \cdot \ell^3 \cdot log(\ell))$ and memory complexity of $O(\ell^2)$. Overall, ClusterMI is one order of complexity faster while holding the same memory complexity of ECGA.

We still need to determine how ClusterMI scales in relation to population size and number of generations. DSMGA is known to have the same scalability of ECGA, requiring larger population (by a constant factor). In the next section we present empirical evidence that indicates that ClusterMI also scales as well as ECGA, requiring larger population. This is an expected phenomenon, since ClusterMI uses the same information as DSMGA: the MI between variables.

## 4 Results

In this section we present the results obtained using ClusterMI. In all results presented in this paper the benchmark function used is the *mk*-trap [15] with or without

[1] The algorithm will be very similar to algorithm 3, and was omitted.

Gaussian noise. The *mk*-trap is an additively separable deceptive function [3] composed of *m* trap functions of *k* variables. Equation (4) presents the used trap function, where *u* is the number of bits with value 1 in the BB. For noisy objective functions, a Gaussian noise with mean 0 and variance equals 5% of the maximun fitness was randomly sampled and added to the fitness of each individual.

$$trap(u) = \begin{matrix} 1, & if\, u = k \\ 0.8 * \frac{(k-1-u)}{(k-1)}, & otherwise, \end{matrix} \tag{4}$$

The proposed method will be evaluated for scalability using four performance measures: population size, number of function evaluations, number of evaluations of the CCC and overall computational time required for convergence. We compare the ClusterMI with the ECGA with and without cache. In the experiments reported, we used the implementation of the ECGA by Fernando Lobo [11] as the ECGA with cache. This implementation was modified to incorporate a version of the ECGA without cache and a version of ClusterMI. Since the three codes are built over the same framework, code optimization issues can be ignored in comparative results. All three methods used tournament selection with tournament size of 16.

In the experiments, the bisection method [14] was used to determine the minimal population size required to correctly solve at least $m - 1$ subproblems 29 out of 30 times. The bisection method is an empirical method for population sizing, and usually produces a good approximation to the theory.

In order to provide a clear presentation on the results, ECGAwc will refer to the ECGA with cache, ECGAwo will refer to the ECGA without cache. ECGA will be used to represent both ECGAwc and ECGAwo when no distinction is necessary.

Three sets of experiments will be reported in this section. The first set of experiments compares the performance of ECGAwc, ECGAwo and ClusterMI on a 4 bit trap problems ($k = 4$) of varying number of BBs. Fifteen runs of the bisection method were used to determine the minimum population size necessary to solve problems of sizes up to 160 with ECGAwo, 256 with ECGAwc and 512 with ClusterMI. These results were used to predict the population size used on the remaining of this set of experiments.

Figure 1 compares the population sizes required by ECGA and ClusterMI, showing the scalability of both methods, figure 2 compares the number of function evaluations required by ECGA and ClusterMI, figure 3 shows the cost of model building in ECGA with cache, ECGA without cache and ClusterMI and figure 4 compares the overall running time in seconds for all three methods.

As can be observed, both ClusterMI and ECGA have the same scalability concerning population size (Figure 1). ClusterMI, however, requires slightly larger populations, an expected phenomenon due to its relation with DSMGA. The population size reflects directly in the number of function evaluations and once again both ECGA and ClusterMI have the same scalability (Figure 2). When the model building is considered ClusterMI strongly outperform ECGA both with and without cache (Figure 3).

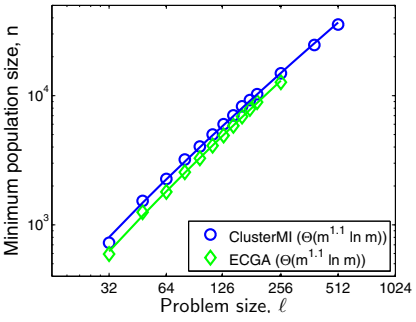



**Fig. 1** A comparison between the ECGA and ClusterMI shows that both algorithms have the same scalability, with ClusterMI using slightly larger population

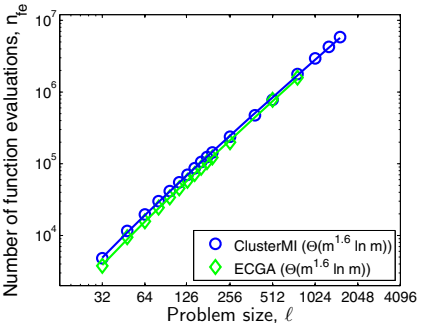



**Fig. 2** The comparison between the number of function evaluations required by the ECGA and ClusterMI shows that both methods have the same scalability

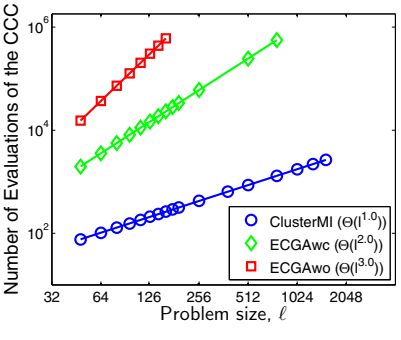



**Fig. 3** Considering the number of evaluations of the CCC necessary to accurately build the model ClusterMI outperforms its competitors both in absolute values and in scalability. ClusterMI scales one order of complexity faster than ECGAwc and two orders of complexity faster than ECGAwo

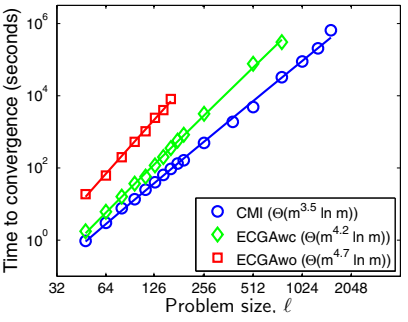



**Fig. 4** When absolute running time is used as the performance measure, ClusterMI clearly outperforms both ECGAwc and ECGAwo. ClusterMI is not only faster in absolute values, but also scales better by almost one order of complexity when compared ECGAwc and more than one order of complexity when compared to ECGAwo

The relatively small difference in population size is counterbalanced with the large difference in model building efficiency, resulting in a significantly better performance for ClusterMI when compared with ECGA (Figure 4). The improvement is dependent on the relative computational cost of the function evaluation. For computationally expensive function evaluations the ECGA is expected to outperform ClusterMI in relatively small problems. However, the extra population size reflects a constant disadvantage while the model building efficiency reflects a complexity reduction. Consequently, for any function evaluation there is always a problem large enough so that ClusterMI's speedup in model building will surpass the overhead caused by the larger population size.

The second set of experiments uses 5 bits trap to compare ECGA and ClusterMI on a harder problem. On this experiment we only used ECGA with cache since it is the most efficient implementation. Fifteen runs of the bisection method were used to determine the minimum population size required to solve problems of size up to 150. Figure 5 presents a comparison between the minimum population size required by the ECGA and ClusterMI and figure 6 presents the running time for each method. A comparison of the number of function evaluations was omitted since it can be derived from figure 5. The number of evaluations of CCC was also omitted since it is equivalent of what was presented on figure 3.

The third set of experiments is based on 4 bits trap functions with additive Gaussian noise of 5% of the maximum fitness. Fifteen runs of the bisection method were used to determine the minimum population size required by ECGA and ClusterMI to solve problems of size up to 160. Figure 7 compares both methods concerning population size and figure 8 compares them using running time as the performance measure. The number of function evaluations and evaluations of CCC were omitted for the same reasons as in the second set of experiments.

The results observed on the second and the third sets of experiments are similar to those presented on the first experiment. ECGA and ClusterMI still present the

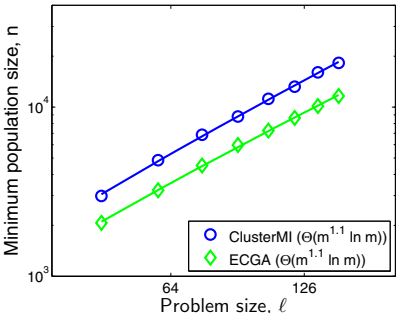



**Fig. 5** The scalability of ECGA and ClusterMI concerning population size is held constant when the size of the trap function is increased. Once again ClusterMI requires a larger population size than ECGA

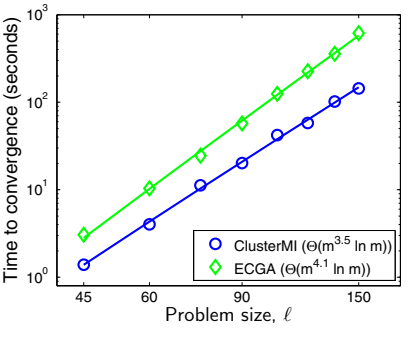



**Fig. 6** When absolute running time is considered, ClusterMI still outperforms ECGA both in absolute values and in scalability

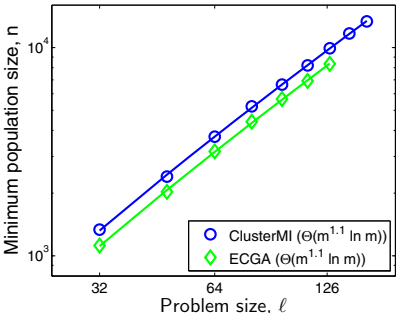



**Fig. 7** The scalability of ECGA and ClusterMI also remains constant for functions with additive Gaussian noise. Once again ClusterMI requires a larger population size than ECGA

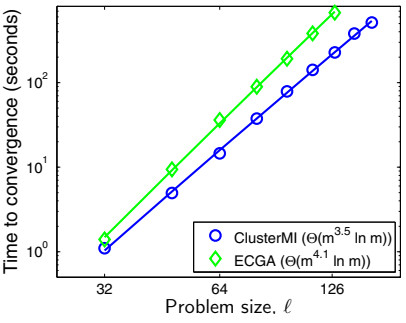



**Fig. 8** When absolute running time is considered, ClusterMI still outperforms ECGA both in absolute values and in scalability

same scalability concerning population size, with ClusterMI requiring larger population size. However, ClusterMI is still more efficient in the model building process resulting in faster running times. Once again there is a trade-off between the use of ECGA and ClusterMI, and once again ECGA will only outperform ClusterMI for relatively small problems with computationally expensive function evaluations.

This section showed that ClusterMI has the same scalability of ECGA concerning population size and number of function evaluations. However, ClusterMI is one order of complexity more efficient in the model building process resulting in an improvement of at least half an order of complexity for overall running time. The next section presents perspectives of future work, including possible extensions to allow for incremental evolution.

## 5 Future Work

This paper presents ClusterMI, a new approach to perform model building on EDAs. ClusterMI uses the MI between pairs of variables to choose which partitions to merge, based on a greedy hierarchical clustering. Although ClusterMI may produce speed ups of thousands of times, the perspective of future work is even more promising. Particularly, ClusterMI can be used to perform incremental evolution, excluding the need to explicitly represent the population and thereby reducing memory usage or even communication cost for parallel implementations. This section presents initial ideas for incremental evolution. Other topics that will be addressed in future works are also outlined.

To build a model of the population we need the MI between every pair of variables, which requires the knowledge of the marginal probability distribution for each variable, as well as the joint probability distribution for every pair of variables. It is easy to update both distributions incrementally, using a procedure similar to the one used on [8] or [13].

$$P(x_1,..,x_n,y) = \frac{P(x_1,..,x_n) * J(y)}{\sum_{\overline{y} \in Y} J(\overline{y})} \tag{5}$$

where,

$$J(y) = \prod_{i=1}^{n} P(X_i = x_i, Y = y)$$

We also need to calculate joint probability distributions of sets of variables that represent a partition or cluster. These distributions are used for sampling and on the calculation of the CCC. Estimating multivariate distributions is a common problem for incremental EDAs. iBoa solves this problem by adding one variable at a time to the distribution and by assuming the variables are independent [13]. Similarly, we propose the use of induction to construct multivariate probability distributions by adding one variable at a time to an already estimated distribution. The induction starts with a pairwise distribution and incrementally adds variables until the desired distribution is estimated. Instead of assuming independence among variables, given a joint distribution $P(X_1,...,X_n)$, we can estimate the joint distribution $P(X_1,...,X_n,Y)$ using equation (5).

Other topics worth of investigation include:

- The hybridization of ClusterMI with a preprocessing local search to reduce the required population size [2];
- Development of a parallel model building algorithm;
- Development of an incremental model building algorithm, which would consist of small incremental changes on previous models reducing the model building cost even further.
- Application of ClusterMI to real world problems and to large scale problems.

## 6 Conclusion

Some EDAs are among the most versatile problem solvers available, being able to solve problems that are intractable for traditional methods in a tractable polynomial time. However, EDAs may still be too time consuming to be practical. Several efficiency enhancement techniques have been proposed in the literature to turn EDAs from tractable to practical. This work focus on the model building, a time consuming but necessary process for EDAs. We propose the use of hierarchical clustering to build an MPM representing the current population.

The proposed algorithm is called ClusterMI and it is related both to the ECGA and to the DSMGA. Consequently, ClusterMI have advantages and disadvantages drawn from both algorithms. Specifically, ClusterMI produces a comprehensive and human interpretable model that can be used to identify BBs on additively separable functions. The scalability of ClusterMI is equivalent to the scalability of ECGA and DSMGA, however, both ClusterMI and DSMGA require larger population sizes, mostly due to the use of MI to guide the model building decisions.

The advantage of ClusterMI relies on the efficient model building process, which is one order of complexity faster than the ECGA with cache, resulting in faster overall running times. The ECGA is expected to outperform ClusterMI on relatively small problems with computationally expensive function evaluations due to the smaller population size. However, the difference in population size is constant, while the improvement in the model building reduces the complexity of the algorithm. Consequently, for any function evaluation there is a problem large enough so that the improvement in model building compensates the extra evaluations and ClusterMI outperforms ECGA.

ClusterMI can produce speed ups of 10 times in moderate size problems and potentially hundreds of times for large scale problems (projected speed up for a problem with $2^{20}$ variables is $\approx 1500$). Moreover, ClusterMI may be used to perform incremental model building or even incremental evolution, eliminating the need to explicitly represent the population and possibly improving the performance even further.

## References

1. Cantu-Paz, E.: Designing Efficient and Accurate Parallel Genetic Algorithms. Ph.D. thesis (1999)
2. Duque, T., Goldberg, D.E., Sastry, K.: Enhancing the Efficiency of The ECGA. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, p. 165. Springer, Heidelberg (2008)
3. Goldberg, D.E.: Genetic algorithms and Walsh functions: Part II, deception and its analysis. Complex Systems 3(2), 153–171 (1989)
4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
5. Goldberg, D.E.: The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Kluwer Academic Publishers, Dordrecht (2002)
6. Goldberg, D.E., Voessner, S.: Optimizing Global-Local Search Hybrids. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), vol. 1, pp. 220–228. Morgan Kaufmann, San Francisco (1999)
7. Harik, G.: Linkage Learning via probabilistic modeling in the ECGA. Tech. rep., University of Illinois at Urbana Chapaign, Urbana, IL (1999)
8. Harik, G., Lobo, F.G., Goldberg, D.E.: The Compact Genetic Algorithm. In: Proceedings of 1998 IEEE Iternational Conference on Evolutionary Computation, pp. 523–528 (1998)
9. Holland, J.H.: Adaptation in Natural and Artificial Systems. The MIT Press, Cambridge (1975)
10. Larraaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2001)
11. De la Ossa, L., Sastry, K., Lobo, F.G.: $\chi$–ary Extended Compact Genetic Algorithm in C++. Tech. rep., Illigal Report 2006013, Illinois Genetic Algorithms Lab, University of Illinois at Urbana-Champaign, 2006, Urbana, IL (2006), `http://www.illigal.uiuc.edu/web/source-code/2006/03/27/ary-extended-compact-genetic-algorithm-for-matlab-in-c/`

12. Pelikan, M., Sastry, K., Goldberg, D.E.: Sporadic model building for efficiency enhancement of hierarchical BOA. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 405–412. ACM Press, New York (2006)
13. Pelikan, M., Sastry, K., Goldberg, D.E.: iBOA: The Incremental Bayesian Optimization Algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008), Atlanta, GA, USA (2008)
14. Sastry, K.: Evaluation-relaxation Schemes for Genetic and Evolutionary Algorithms. Master's Thesis, University of Illinois at Urbana Champaign, Urbana, IL (2001)
15. Sastry, K., Goldberg, D.E.: Let's Get Ready to Rumble: Crossover Versus Mutation Head to Head. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103. Springer, Heidelberg (2004)
16. Sastry, K., Goldberg, D.E., Pelikan, M.: Efficiency Enhancment of Probabilistic Model Building Genetic Algorithm. Tech. rep., Illinois Genetic Algorithms Laboratory, Univeristy of Illinois at Urbana Champaign, Urbana, IL (2004)
17. Shannon, C.E.: A mathematical theory of communication Bell Syst. Tech. J. 27(3), 379–423 (1948)
18. Sinha, A.: Designing Efficient Genetic and Evolutionary Algorithm Hybrids (2003)
19. Sinha, A., Goldberg, D.E.: A survey of hybrid genetic and evolutionary algorithms. Tech. rep., University of Illinois at Urbana Chapaign, Urbana, IL (2003)
20. Yu, T.L.: A Matrix Approach for Finding Extreme: Problems with Modularity, Hierarchy, and Overlap. Ph.D. thesis (2006)
21. Yu, T.L., Goldberg, D.E.: Conquering hierarchical difficulty by explicit chunking: substructural chromosome compression. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 1385–1392 (2006)
22. Yu, T.L., Sastry, K., Goldberg, D.E., Pelikan, M.: Population sizing for entropy-based model building in discrete estimation of distribution algorithms. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 601–608 (2007)

# Estimation of Distribution Algorithm Based on Copula Theory

Li-Fang Wang and Jian-Chao Zeng

**Abstract.** Estimation of Distribution Algorithms (EDAs) is the hot topic of evolutionary computation currently. EDAs model the selected population using a distribution model, which is latter sampled to generate the population for the next generation. This chapter introduces a new way to estimate the distribution model and sample from it according to copula theory. The multivariate joint is decomposed into the univariate margins and a function called copula. In the EDAs based on copula theory (copula-EDAs), only the margins are estimated, and the next generation is sampled from the copula and the inverse function of the margins. The framework of the copula-EDAs is discussed in the chapter. Two 2-dimensional copula-EDAs and a high-dimensional copula-EDA are described in detail as the examples.

## 1 Introduction

Estimation of Distribution Algorithms (EDAs) are originated from Genetic Algorithms (GAs) [1]. While GAs generate a new population using crossover and mutation operators, EDAs estimate a distribution model of the selected population and sample from the estimated model. The comparison of the two algorithms is shown as Fig. 1.

As an example, the integer minimization problem $f(x)=x^2$, $x\in[0,15]$ is optimized with a kind of EDA. The value of $x$ is denoted as 4-bit binary number, *i.e.* 5 is denoted as 0101, 8 is denoted as 1000. According to the flow chat of EDAs, the first step is to generate some random integers in [0,15]. Supposing, the following 5 binary numbers are generated. Their fitness are listed followed the numbers.

| No. | $x$ | fitness |
|---|---|---|
| 1 | 1 0 1 0 | 100 |
| 2 | 1 1 0 0 | 144 |
| 3 | 0 1 0 0 | 16 |
| 4 | 0 0 1 1 | 9 |
| 5 | 0 1 0 1 | 25 |

Li-Fang Wang · Jian-Chao Zeng
Complex System and Computational Intelligence Laboratory,
Taiyuan University of Science and Technology, China P.R.
e-mail: wlf1001@163.com

Y.-p. Chen (Ed.): Exploitation of Linkage Learning, ALO 3, pp. 139–162.
springerlink.com 

Secondly, a promising population is selected according to the individual's fitness. If the truncation selection is used and the selection rate is 0.6, then the 3rd-5th individuals are selected as the promising population.

Thirdly, the distribution model of the selected population is estimated. Denote $p_i$ as the probability of the $i^{th}$ bit being 1. Thus the probability of each bit is the following: $p_1$=0, $p_2$=2/3, $p_3$=1/3, $p_4$=2/3.

Fourthly, the next generation is sampled according to the estimated model. The following individuals are generated supposing.

| No. | $x$ | fitness |
|---|---|---|
| 1 | 0 0 1 0 | 4 |
| 2 | 0 1 0 1 | 25 |
| 3 | 0 1 0 0 | 16 |
| 4 | 0 0 0 1 | 1 |
| 5 | 0 1 0 1 | 25 |

The loop is continued by going to the second step until the termination condition is met.

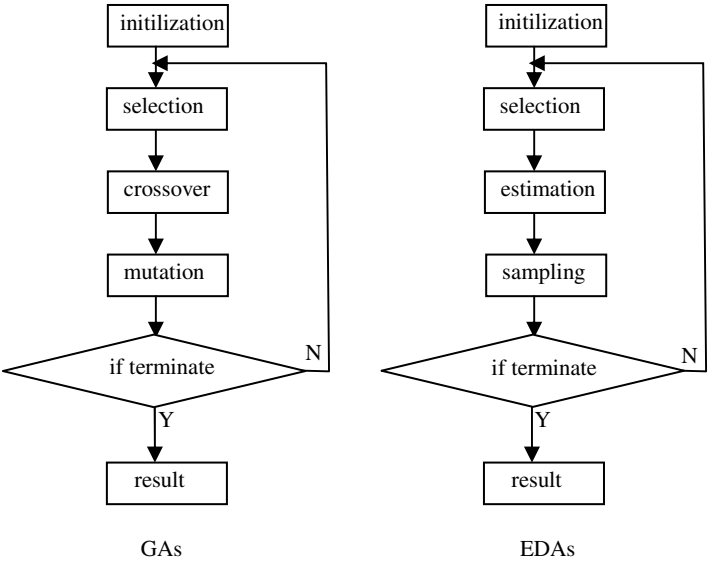



**Fig. 1** The different flowchart of EDAs and GAs

Different kinds of EDAs have been developed since the idea of EDA is introduced in 1994 [1,2].

The early EDAs regard the relationship of the random variables as independent. PBIL(Population Based Incremental Learning) [2-5], UMDA(Unirvariate Marginal Distribution Algorithm) [6-8] and cGA(compact Genetic Algorithm) [9] are all the classical algorithms.

However, in practical problems variables of the optimization problems are not independent. Some algorithms consider the linear relationship of variables. The random variables are numbered according to a certain rule, and the random variable depends only on the previous random variable. The classical algorithms of this type include MIMIC(Mutual Information Maximization of Input Clustering) [10], COMIT(Combining Optimizers with Mutual Information Trees) [11] and BMDA(Binary Marginal Distribution Algorithm) [12,13].

Many EDAs considering the multivariate dependence are proposed in the last decade. For example, ECGA(Extended Compact Genetic Algorithm) [14], FDA(Factorized Distribution Algorithms) [15-19] and BOA(Bayesian Optimization Algorithm) [20-25].

Continuous EDAs were developed later than discrete EDAs. The classical algorithms include PBILc [26], UMDAc [27], SHCLVND(Stochastic Hill Climbing with Learning by Vectors of Normal Distributions) [28], EMNA(Estimation of Multivariate Normal Algorithm)[29], ENGA(Estimation of Gaussian Networks Algorithm) [30], IDEA [31], etc. The Gaussian distribution is used in most of these algorithms.

PBILc and UMDAc are extended from PBIL and UMDA respectively to the continuous form, and the Gaussian distributions are used in both of the two algorithms. The Gaussian distribution is also used in SHCLVND and the parameter mean is adjusted by Hebbian Learning.

The multivariate Gaussian distribution is used to estimate the distribution of the population in EMNA. The parameters mean and the covariance matrix are estimated by EML. The next generation is sampled from the joint distribution. It is a problem in EMNA that the runtime to estimate the covariance matrix increased exponentially if the number of random variables has increased.

EGNA is an algorithm based on the Gaussian network. The directed edges represent the relationship of the random variables and the distribution of each random variable is described with the Gaussian distribution. The Gaussian network is adjusted according to the current selected population.

IDEA is a kind of EDA based on the hybrid Gaussian distribution and the Gaussian kernel function. The shortage of EMNA and EGNA is conquered in IDEA. But the relationship of random variables is not reflected thoroughly in IDEA.

The chapter describes how copula theory can be used in an Estimation of Distribution Algorithm. The benefit of the copula approach is that only marginal distributions need to be estimated from the population of solutions, thus promising efficiency savings.

The remaining sections are organized as follows. Section 2 talks about the basic theorem and properties in copula theory. Section 3 explains why copula theory could be used in EDA and then gives the framework of copula-EDAs. Section 4 and section 5 illustrate the copula EDAs with two 2-dimensional algorithms and one high-dimensional algorithm respectively. Finally, section 6 concludes the paper, summarizing the main results and pointing further directions.

## 2 A Brief Introduction to Copula Theory

Copula theory [32] is one of the hot topics in statistics. Its main idea is how to decompose the multivariate joint distribution into the univariate marginal distributions. A kind of function called copula ties the joint and the margins together. Thus the difficult work to estimate the joint distribution could be replaced by estimating the margins and constructing a copula. It also draws many scholars' attention to sample from copula. These are the two key steps in EDAs, i.e., to construct a distribution model to represent the selected population and to sample the constructed model, creating the next generation. The distribution of the population is reflected precisely and the runtime is shortened if the copula theory is introduced into the framework of EDAs. Copula theory is briefly introduced in the following section.

### *2.1 Definitions and Basic Properties*

The definition of copula and the basic theorem are listed in this section. The symbol **I** denotes the domain [0, 1] in this chapter. The definitions and theorems are quoted from [32].

Definition 1. *A two-dimensional subcopula (or 2-subcopula, or briefly, a subcopula) is a function C' with the following properties:*

*1) Dom$C'=S_1\times S_2$, where $S_1$ and $S_2$ are subsets of* **I** *containing 0 and 1;*
*2) C' is grounded and 2-increasing;*
3) *For every u in $S_1$ and every v in $S_2$,*

$$C'(u,1)=u \text{ and } C'(1,v)=v. \tag{1}$$

Definition 2. *A two-dimensional copula( or 2-copula, or briefly, a copula) is a 2-subcopula C whose domain is* $\mathbf{I}^2$.

*Equivalently, a copula is function C from* $\mathbf{I}^2$ *to* **I** *with the following properties:*

*1) For every u,v in* **I**,

$$C(u,0)=0=C(0,v) \tag{2}$$

*and*

$$C(u,1)=u \text{ and } C(1,v)=v; \tag{3}$$

*2) For every $u_1,u_2,v_1,v_2$ in* **I** *such that $u_1\le u_2$ and $v_1\le v_2$,*

$$C(u_2,v_2)-C(u_2,v_1)-C(u_1,v_2)+C(u_1,v_1)\ge 0. \tag{4}$$

The definition of *n*-copula is extended from the above definitions.

Definition 3. *An n-copula is a function C from* $\boldsymbol{I}^n$ *to* $\boldsymbol{I}$ *with the following properties:*

*1) For every* $\boldsymbol{u}$ *in* $\boldsymbol{I}^n$,

$$C(\boldsymbol{u})=0 \text{ if at least one coordinate of } \boldsymbol{u} \text{ is 0,} \tag{5}$$

*and*

$$\text{if all coordinates of } \boldsymbol{u} \text{ are 1 except } u_k, \text{ then } C(\boldsymbol{u})=u_k; \tag{6}$$

2) *For every* $\boldsymbol{u}$ *and* $\boldsymbol{v}$ *in* $\boldsymbol{I}^n$ *such that* $\boldsymbol{u} \le \boldsymbol{v}$*, denote*

$$V_C\left([\boldsymbol{u},\boldsymbol{v}]\right) = \sum_{\varepsilon 1,\ldots,\varepsilon n \in \{0,1\}} (-1)^{(\varepsilon 1+\ldots+\varepsilon n)} C((\varepsilon_1 u_1 + (1-\varepsilon_1)v_1) + \ldots + (\varepsilon_n u_n + (1-\varepsilon_n)v_n))$$

*then*

$$V_C([\boldsymbol{u},\boldsymbol{v}]) \ge 0. \tag{7}$$

It can be derived from the definition of copula that the condition of copula is very easy to be satisfied. So many functions can be used as copulas. The copulas are divided into two classes that are elliptical copulas and Archimedean copulas. For example, normal copula and t-copula are all elliptical copulas. Archimedean copulas are generated from different generators such as Clayton copula, Gumbel copula and others. Each Archimedean copula has one or more parameters. More details see [32].

Sklar's theorem plays an important role in copula theory. It gives the theory basis to connect the multivariate distribution with one-dimensional marginal distribution.

**Sklar's Theorem in *n*-dimensions:** *Let H be an n-dimensional distribution function with margins* $F_1,F_2,\ldots,F_n$*. Then there exists an n-copula C such that for all* $\boldsymbol{x}$ *in* $\boldsymbol{R}^n$*,*

$$H(x_1,\ldots,x_n)=C(F_1(x_1),\ldots,F_n(x_n)). \tag{8}$$

*If* $F_1,F_2,\ldots,F_n$ *are all continuous, then C is unique; otherwise, C is uniquely determined on* $RanF_1\times\ldots\times RanF_n$*. Conversely, if C is an n-copula and* $F_1,F_2,\ldots,F_n$ *are distribution functions, then the function H defined by (8) is an n-dimensional distribution function with margins* $F_1,F_2,\ldots,F_n$*.*

According to Sklar's theorem, the joint distribution of a random vector could be constructed by the one-dimensional distribution of each random variable through a copula. Therefore, the hard work to estimate the joint distribution is simplified to estimate the margins and to produce a copula.

## *2.2 Random Variable Generation*

As it will be demonstrated in section 4 and section 5, it is easy in EDAs to estimate the distribution model based on Sklar's theorem. And the next work is to sample from the distribution model that is composed by a copula and some margins. An example of sampling from 2- copula is shown. Let $F(x)$ and $G(y)$ be the margins of the random vector $(X,Y)$, and the joint is $C(F(x),G(y))$, *i.e.* $C$ is the copula. Denote $U=F(x)$, $V=G(y)$, then $U\in[0,1]$, $V\in[0,1]$. According to Sklar's theorem, the sample of $(X,Y)$ can be generated by the following two steps. First, a sample value $(u,v)$ of the random vector $(U,V)\in[0,1]^2$ that obeys the joint $C$ is generated. Second, the sample value $(x,y)$ is generated according to the inverse of margins and $(u,v)$, *i.e.* $x=F^{-1}(u)$ and $y=G^{-1}(v)$.

Obviously, the first step need to be studied and it is actually one of the current hot topics for research. A method based on the conditional distribution of $C(u,v)$ is introduced as an example. The condition distribution $c_u(v)$ when

$U=u$ is $c_u(v)=P[V\le v|U=u]=\lim_{\Delta u\to 0}\frac{C(u+\Delta u,v)-C(u,v)}{\Delta u}=\frac{\partial C(u,v)}{\partial u}$. So the steps are the following.

1) Generate two independent random numbers $u\sim U(0,1)$ and $t\sim U(0,1)$;
2) Let $v=c_u^{(-1)}(t)$, where, $c_u^{(-1)}(t)$ is the restricted inverse function of $c_u(v)$ in [0,1];
3) $(u,v)$ is the sample that obeys $C$.

This is a way to sample from the 2-copula based on the conditional distribution and it can be extended to $n$-copula. Furthermore, many ways to sample from the $n$-copula are studied by the scholars [33,34].

## 3 Motivation

The main idea of EDAs is to analyze the distribution model of the promising population and to affect the next generation by use of the distribution model. Selection, modeling and sampling are the three main steps of EDAs. Selection is used to increase the average fitness of the population by eliminating low fitness individuals while increasing the number of copies of high fitness ones. Modeling is used to quantify the distribution of the selected population, *i.e.* by probability density function (pdf), cumulative distribution function (cdf) or some other ways. Sampling is used to generate the next generation according to the modeled distribution.

To reflect the entire relation of the random variables and the distribution model of selected population, the joint of all random variables is the best way. However, the computation cost to model the joint is very large when the problem involves a large number of variables. For instance, EMNA supposes that the joint is multivariate normal distribution, and the parameters (*i.e.* mathematical expectation and covariance matrix) are estimated by EML. Obviously, the size of covariance matrix is the square of the number of variables in the problem. The larger the problem we are trying to solve, the higher is the computational cost to estimate the necessary parameters.

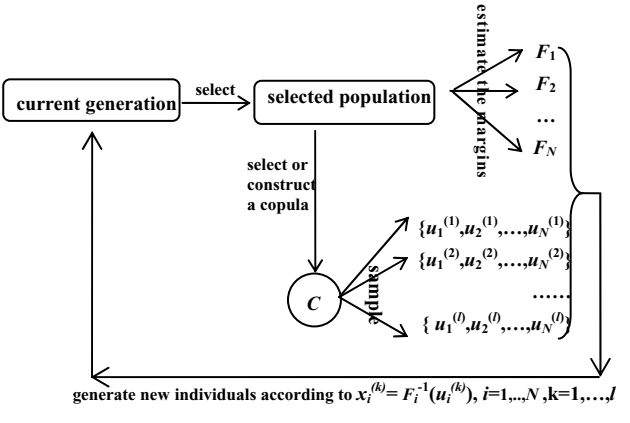



**Fig. 2** The schema of copula-EDA

On the grounds of Sklar's theorem, the joint of random variables could be represented by a copula and the margins of each random variable. Thus the relation of random variables could be denoted by a selected copula and the margins are also estimated as a certain distribution, and then the joint is obtained according to equation (8). Actually, the joint $H$ does not need to be calculated in copula-EDAs. The only two works are to select or to construct a copula $C$ and to estimate each one-dimensional margins $F_i$, $i$=1,..,$N$.

Sampling from the estimated distribution model is the next work after the distribution model of selected population is estimated, that is the new individuals generated should obey the estimated distribution. To generate an individual, the first work is to generate a vector $\{u_1,u_2,\ldots,u_N\} \in [0,1]^N$ who obeys the joint $C$ according to copula theory. Subsequently, the values $x_i = F_i^{-1}(u_i)$ are calculated according to the inverse function of each margins, and the vector $\{x_1,x_2,\ldots,x_N\}$ is the sample who obeys the joint $H$, *i.e.* $\{x_1,x_2,\ldots,x_N\}$ is one generated individual. To sum up the above arguments, the schema of EDAs based on copula theory (copula-EDA) is shown in Fig. 2. Copula-EDA generates random initial population firstly, then repeat the following 4 steps until certain terminate condition is met.

1. Select $m$ individuals from the current population as the promising population.
2. Estimate the margins. The $m$ individuals are the $m$ samples of the $N$-dimensional random vector. Each one-dimensional margin $F_i(i=1,..,N)$ is estimated according to the $m$ samples. The margins could be empirical distribution, normal distribution or others.
3. Sample from the copula $C$. Generate $l$ vectors $\{u_1^{(j)},u_2^{(j)},\ldots,u_N^{(j)}\}(\ j=1,\ldots,l)$ who obey the joint $C$.
4. Compose the next generation by the following 3 parts. 1) Reserve the best $k$ individuals of the current generation to the next generation. 2) Get the new $l$ individuals by calculate $x_i^{(j)} = F_i^{-1}(u_i^{(j)})$ ($i$=1,..,$N$, $j$=1,…,$l$). 3) to generate some random individuals in the search space depending on certain mutation rate.

## 4 Two-Dimensional Copula-EDAs

Suppose that the optimization problem is

$$\min f(X) = f(x_1,x_2,\ldots,x_n),\ x_i \in [a_i,b_i]\ \ (i=1,2,\ldots,n)$$

Let $s$ denote the population size. Let $\mathbf{x} = \{x^i = (x^i_1, x^i_2, ..., x^i_n), i = 1, 2, ..., s\}$ denote the population. Then the population $\mathbf{x}$ is composed of $s$ observed samples from a random vector $(X_1,X_2,\ldots,X_n)$. The marginal distribution of each random variable $X_i(i=1,2,\ldots,n)$ can be Gaussian distribution, t-distribution or some other distribution. On the ground of Sklar's theorem, the joint distribution function can be constructed with a selected copula and the marginal distributions. The modeling for the distribution of the selected population is finished. Next step is to generate new population by the way introduced in Sect. 2.2.

### 4.1 Gaussion Copula-EDAs

A two-dimensional optimization problem is considered. According to copula theory, the marginal distribution of each random variable and a copula are required. The marginal distributions of $(X_1,X_2)$ is easy to estimate because the observed values are given. The only thing is to evaluate sample mean and sample variance. 2-D Gauss-copula is selected which is

$$C(u,v;\rho)=\phi_\rho(\phi^{-1}(u),\phi^{-1}(v)),u,v\in[0,1]\,, \tag{9}$$

where, $\phi_\rho$ is a binary standard Gaussian distribution with correlation coefficient $\rho$. Determining $\rho$ is the key to construct copula.

If the Maximum Likelihood Estimation is used, then the log-likelihood function is

$$l(\rho)=\sum_{i=1}^{s}\{-\ln 2\pi-\frac{1}{2}\ln(1-\rho^2)-\frac{z_{1i}^2+z_{2i}^2-2\rho z_{1i}z_{2i}}{2(1-\rho^2)}\}+\sum_{i=1}^{s}[\ln f(x_1^i)+\ln g(x_2^i)] \tag{10}$$

where,

$$z_{1i}=\phi^{-1}(F(x_1^i)),z_{2i}=\phi^{-1}(G(x_2^i))\,. \tag{11}$$

If Moment Estimation is used, then

$$\rho=\frac{\frac{1}{s}\sum_{i=1}^{s}x_1^ix_2^i-\bar{X}_1\bar{X}_2}{S_{X_1}^*S_{X_2}^*}\,, \tag{12}$$

where,

$$\bar{X}_k=\frac{1}{s}\sum_{i=1}^{s}x_k^i\,,\ S_{X_k}^*=\sqrt{\frac{1}{s-1}\sum_{i=1}^{s}(x_k^i)^2}\,,(k=1,2)\,. \tag{13}$$

Next, sampling from the distribution function is discussed. Since

$$\begin{aligned}&C(u,v;\rho)=\phi_\rho(\phi^{-1}(u),\phi^{-1}(v))\\&=\int_{-\infty}^{u}\int_{-\infty}^{v}\frac{1}{2\pi\sqrt{1-\rho^2}}\exp\{-\frac{[\phi^{-1}(s)]^2+[\phi^{-1}(t)]^2-2\rho\phi^{-1}(s)\phi^{-1}(t)}{2(1-\rho^2)}\}dtds\end{aligned} \tag{14}$$

then

$$\begin{aligned}&\omega=C_u(v)=\partial C(u,v)/\partial u\\&=e^{\frac{[\phi^{-1}(u)]^2}{2}}\int_{-\infty}^{v}\frac{1}{2\pi\sqrt{1-\rho^2}}\exp\{-\frac{[\phi^{-1}(t)-\rho\phi^{-1}(u)]^2}{2(1-\rho^2)}\}dt\end{aligned} \tag{15}$$

Therefore,

$$v=\sqrt{1-\rho^2}\phi^{-1}(e^{\frac{[\phi^{-1}(u)]^2}{2}}\omega)+\rho\phi^{-1}(u)=\sqrt{1-\rho^2}\phi^{-1}(\omega')+\rho\phi^{-1}(u) \tag{16}$$

So randomly generate vector $(u,\omega') \sim U[0,1]^2$, and then the other random number $v$ is calculated by using equation (16). The vector $(x_1, x_2)$ can be calculated by using equation (17).

$$x_1 = \sigma_1\phi^{-1}(u) + \mu_1, x_2 = \sigma_2\phi^{-1}(u) + \mu_2 \tag{17}$$

where,

$$\mu_k = \bar{X}_k, \sigma_k = S^*_{X_k}, (k = 1,2). \tag{18}$$

Conclusively, the algorithm for implementing 2-D copula-EDA is as follows [35]:

1. Initialize (***pop***, *N*). Randomly generate initial population ***pop*** with size *N*. set generation count $g \leftarrow 0$.
2. Selection (***pop***, ***spop***, *select-rate*). Select the best *select-rate*×*N* agents from ***pop*** to ***spop*** according to the agents' fitness.
3. copula-generator (***pop***, ***spop***, *mutate-rate*).
   - 3.1. Construct the distribution model of ***spop***:
     1) calculate the sample average and the sample variance for each random variable according to (13), then the marginal distributions are $F = N(\bar{X}_1, S^*_{X_1})$ and $G = N(\bar{X}_2, S^*_{X_2})$;
     2) calculate the estimation value of parameter $\rho$ according to equation (11) or (12), then the copula $C$ is the same as (14);
   - 3.2. Generate a new population by iterative using procedure generation(*C, F, G*), where
   - $v = C_u^{(-1)}(\omega) = \sqrt{1-\rho^2}\phi^{-1}(\omega) + \rho\phi^{-1}(u)$.
   - 3.3. Randomly generate some agents by the *mutate-rate*.
4. Stop if the termination criterion is met.
5. Set $g \leftarrow g+1$and go to Step 2.

The following 9 test functions are used to show the behavior of the proposed algorithm cEDA and to compare the cEDA with PBILc and other EDAs. The test functions $F_1$~$F_3$ and $F_6$~$F_8$ are also used in [26].

- $F_1(x) = -\dfrac{100}{10^{-5} + \sum_i |y_i|}$, where $y_1$=$x_1$, $y_i$=$x_i$+$y_{i-1}$($i$≥2), $x_i \in [-3,3]$, the optimal result is $F_1^*(0,0,...,0) = -10^7$.
- $F_2(x) = -\dfrac{100}{10^{-5} + \sum_i |y_i|}$, where $y_1$=$x_1$, $y_i$=$x_i$+sin$y_{i-1}$($i$≥2), $x_i \in [-3,3]$, the optimal result is $F_2^*(0,0,...,0) = -10^7$.
- $F_3(x) = -\dfrac{100}{10^{-5} + \sum_i |y_i|}$, where $y_i = 0.024 \times (i-1) - x_i$, $x_i \in [-3,3]$, the optimal result is $F_3^*(0.024 \times 2, 0.024 \times 3, ..., 0.024 \times (n+1)) = -10^7$.

- $F_4(x) = \sum_i x_i^2$ , where $x_i \in [-500, 500]$ , the optimal result is $F_4^*(0,0,...,0) = 0$ .
- $F_5(x) = 1 + \sum_i (\sin x_i)^2 - 0.1\exp(-\sum_i x_i^2)$ , where $x_i \in [-10, 10]$ , the optimal result is $F_5^*(0,0,...,0) = 0.9$ .
- $F_6(x) = \sum_i (x_i^2 - A\cos(2\pi x_i) + A)$ , where $x_i \in [-5, 5]$ , the optimal result is $F_6^*(0,0,...,0) = 0$ .
- $F_7(x) = \sum_i (418.9829 + x_i \sin\sqrt{|x_i|})$ , where $x_i \in [-500, 500]$ , the optimal result is $F_7^*(-420.9687, -420.9687, ..., -420.9687) = 0$ .
- $F_8(x) = \sum_i x_i^2 - \prod_i \cos(\frac{x_i}{\sqrt{i+1}})$ , where $x_i \in [-100, 100]$ , the optimal result is $F_8^*(0,0,...,0) = -1$ .
- $F_9(x) = [1 + (x_1 + x_2 + 1)^2 \times (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$ , where $x_1, x_2 \in [-2, 2]$ , the optimal result is $F_9^*(0, -1) = 3$ .

All test functions are optimized in 2-dimensional spaces, the maximal generation *g* is set to 1000. The search terminates if the distance between the best solution found so far and the optimum is less than the predefined precision. Table 1 and Table 2 display the experimental results.

**Table 1** The convergence of copula-EDA and PBILc, the maximal generation is 1000

| Functions | Algorithm | Population size | Select rate | Mutate rate | precision | convergence rate | convergence generations |
|---|---|---|---|---|---|---|---|
| $F_1$ | cEDA | 500 | 0.2 | 0.05 | $10^{-5}$ | 50/50 | 50.700 |
| | PBILc | 500 | 0.2 | 0 | $10^{-5}$ | 50/50 | 405.660 |
| $F_2$ | cEDA | 500 | 0.2 | 0.05 | $10^{-5}$ | 46/50 | 43.369 |
| | PBILc | 500 | 0.2 | 0 | $10^{-5}$ | 50/50 | 421.120 |
| $F_3$ | cEDA | 500 | 0.2 | 0.05 | $10^{-5}$ | 50/50 | 17.600 |
| | PBILc | 500 | 0.2 | 0 | $10^{-5}$ | 50/50 | 396.420 |
| $F_4$ | cEDA | 100 | 0.2 | 0.05 | $10^{-5}$ | 50/50 | 28.791 |
| | PBILc | 100 | 0.2 | 0 | $10^{-5}$ | 50/50 | 841.400 |
| $F_5$ | cEDA | 100 | 0.2 | 0.05 | $10^{-5}$ | 50/50 | 222.760 |
| | PBILc | 100 | 0.2 | 0 | $10^{-5}$ | 50/50 | 560.860 |
| $F_6$ | cEDA | 100 | 0.2 | 0.05 | $10^{-5}$ | 50/50 | 76.800 |
| | PBILc | 100 | 0.2 | 0 | $10^{-5}$ | 50/50 | 493.600 |
| $F_7$ | cEDA | 100 | 0.2 | 0.05 | $10^{-3}$ | 44/50 | 69.386 |
| | PBILc | 100 | 0.2 | 0 | $10^{-3}$ | 32/50 | 905.593 |
| $F_8$ | cEDA | 100 | 0.2 | 0.05 | $10^{-5}$ | 50/50 | 25.250 |
| | PBILc | 100 | 0.2 | 0 | $10^{-5}$ | 50/50 | 664.300 |
| $F_9$ | cEDA | 100 | 0.2 | 0.05 | $10^{-5}$ | 50/50 | 21.306 |
| | PBILc | 100 | 0.2 | 0 | $10^{-5}$ | 41/50 | 644.195 |

**Table 2** The state of different algorithm after 50 runs

| | copula_EDA | PBILc | PBILcM |
|---|---|---|---|
| | Mean±Std | Mean±Std | Mean±Std |
| $F_1$ | -9.7229e+6±1.5966e+4 | -9.9473e+6±3.1487e+4 | -4.4996e+5±4.4091e+5 |
| $F_2$ | -9.4549e+6±1.3826e+3 | -9.9403e+6±3.1134e+4 | -3.8927e+5±3.7505e+5 |
| $F_3$ | -9.8119e+6±1.1698e+3 | -9.9519e+6±2.0524e+4 | -9.4466e+4±6.6105e+4 |
| $F_4$ | 3.6721e-12 ±2.6923e-12 | 7.8393e-12 ±8.4951e-12 | 0.2069 ± 0.8377 |
| $F_5$ | 0.9000±3.0600e-11 | 0.9000 ±3.3826e-14 | 0.9000 ±2.0910e-5 |
| $F_6$ | 8.3590e-14±5.7045e-14 | 2.2464e-12 ±2.6022e-12 | 5.8463e-4 ±6.0017e-4 |
| $F_7$ | 0.1596±0.7787 | 35.1368 ±58.5155 | 0.3105 ± 0.2903 |
| $F_8$ | -1.0000±4.4546e-14 | -1.0000 ±3.6992e-13 | -0.9907 ± 0.0388 |
| $F_9$ | 3.0000±9.9700e-9 | 7.8600 ±10.4784 | 3.0116 ± 0.0206 |

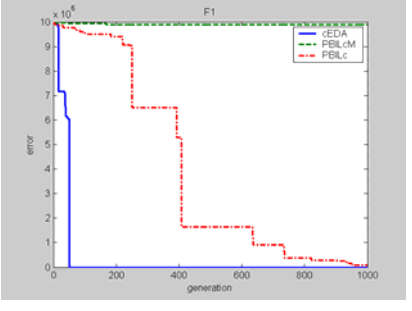



**Fig. 3** The performance on test function $F_1$

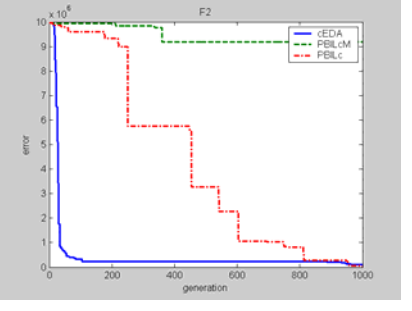



**Fig. 4** The performance on test function $F_2$

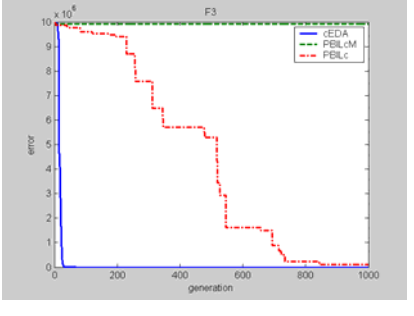



**Fig. 5** The performance on test function $F_3$

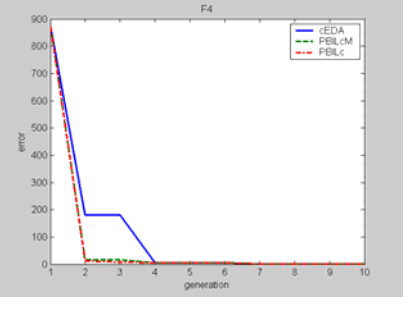



**Fig. 6** The performance on test function $F_4$

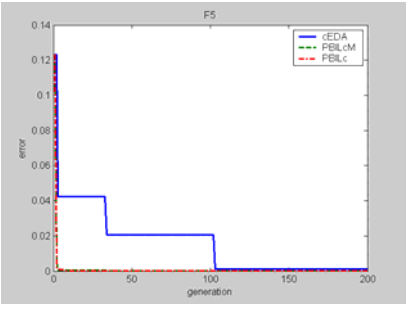



**Fig. 7** The performance on test function $F_5$

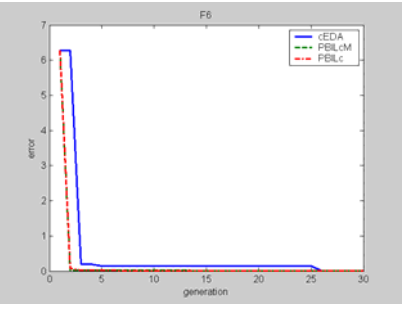



**Fig. 8** The performance on test function $F_6$

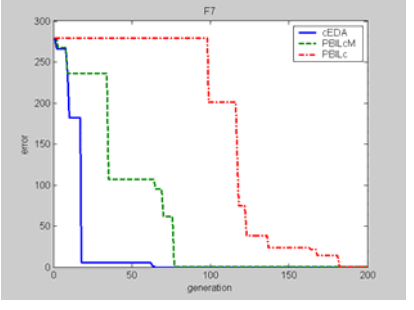



**Fig. 9** The performance on test function $F_7$

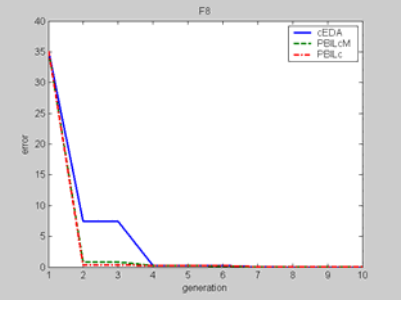



**Fig. 10** The performance on test function $F_8$

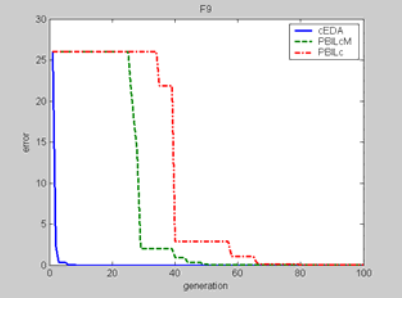



**Fig. 11** The performance on test function $F_9$

Copula-EDA is abbreviated to cEDA in Table 1 and Fig.3-11. PBILcM is performed based on PBILc with the mutation rate as 0.05 for the sake of comparing the performance of copula-EDA and PBILc with the same parameters. But the experimental results show that the convergence rate of PBILcM is 0 in each function with the parameters in Table 1. The convergence rate and the convergence

generations are the average results of 50 runs. The experimental results show that copula-EDA converges to the global optimum quickly in the test functions.

The mean fitness and the standard variance of each algorithm after 50 runs are showed in Table 2. Obviously, copula-EDA performs better than the other two algorithms. It is found through further experiments that Copula-EDA converges faster than PBILc, but it needs more agents in the population than PBILc.

The evolution processes of the above three algorithms is compared with the same population in Fig.3-11. The population in the first generation is same for each algorithm. Error in Fig.3-11 is the fitness difference. Copula-EDA converges to the best solution quickly in almost all tested functions especially for the function $F_1$-$F_3$ whose random variables are strongly correlative with each other. It is also can be seen from Fig.7-9 that copula-EDA is premature in some cases.

## *4.2 Archimedean Copula-EDAs*

According to Sklar's theorem, two steps are performed in order to construct the joint probability distribution function of a random vector. The first step is constructing the margins of each random variable separately. The second step is selecting a proper copula to construct the joint distribution. Therefore, the distribution character of each random variable and their relationship can be studied by themselves. This way can be used in EDAs to model the joint probability distribution function. And then samples are generated from the specified joint distribution by use of the copula.

The optimization problem is

$$\min f(X) = f(x_1, x_2), \quad x_i \in [a_i, b_i] \quad (i = 1, 2). \tag{19}$$

Denote the selected population with size $s$ as

$$\mathbf{x} = \{x^i = (x^i{}_1, x^i{}_2), i = 1, 2, ..., s)\} \tag{20}$$

In other words, $\mathbf{x}$ are the $s$ observations of the random vector ($X_1$, $X_2$). The marginal distribution function of each random variable $X_i$ can be estimated by normal distribution, t-distribution or empirical distribution, etc. Denote the marginal distribution function of $X_i$ as $u=F(x_1)$ and $v=G(x_2)$. The joint probability distribution function is constructed with a selected copula $C$ and the estimated margins in the light of Sklar's theorem.

The next step is to generate samples from the joint distribution using the copula as a tool. By virtue of Sklar's theorem, it is necessary only to generate a pair $(u,v)$ of observations of uniform (0,1) random variables $(U,V)$ whose joint distribution function is $C$, and then transform those uniform random variables via the quasi-inverse of the marginal distribution functions. One procedure for generating such of a pair $(u,v)$ of uniform (0,1) random variables is the conditional distribution method. For this method, the conditional distribution function for $V$ given $U = u$ is need, which is denoted as $C_u(v)$:

$$C_u(v) = P(V \le v \mid U = u) = \lim_{\Delta u \to 0} \frac{C(u+\Delta u, v) - C(u,v)}{\Delta u} = \frac{\partial C(u,v)}{\partial u}. \quad (21)$$

$C_u(v)$ exists and is non-decreasing almost everywhere in **I**.

Conclusively, the generation of sample is performed as the following steps:

1. Generate two independent uniform (0, 1) random variables $u$ and $t$;
2. Set $v = C_u^{(-1)}(t)$, where $C_u^{(-1)}(t)$ denotes a quasi-inverse of $C_u(v)$.
3. The desired pair is $(u,v)$.
4. Set $x_1=F^{(-1)}(u)$, $x_2=G^{(-1)}(v)$, then $(x_1,x_2)$ is a sample of the specified joint distribution.

To sum up, the process for implementing 2-D Copula-EDA is as follows [36]:

1. Initialize (***pop***, *N*). Randomly generate initial population pop with size *N*. set generation count $g \leftarrow 0$.
2. Selection (***pop***, ***spop***, *select-rate*). Select the best *select-rate*×*N* agents from pop to ***spop*** according to the agents' fitness.
3. Copula-generator (***pop***, ***spop***, *mutate-rate*).
   - 3.1. Construct the distribution model of ***spop***;
   - 3.2. Generate a new population based on the specified joint distribution, and randomly generate some agents by the rate *mutate-rate*.
4. Stop if the termination criterion is met.
5. Set $g \leftarrow g+1$, and then go to step 2.

The functions in Sect. 4.1 are used to test the effectiveness of the proposed algorithm.The following two Archimedean copulas are chosen.

- Clayton: $C_1(u,v) = (u^{-\theta} + v^{-\theta} - 1)^{-1/\theta}, \theta \ge -1, \theta \ne 0$
- Ali-Mikhail-Haq: $C_2(u,v) = \dfrac{uv}{1-\theta(1-u)(1-v)}, -1 \le \theta < 1$

All the one-dimensional marginal distributions are normal distributions. Table 3 displays the experimental results.

All test functions are optimized in 2-dimensional spaces, the maximal generation $g$ is set to 1000. The search terminates if the distance between the best solution found so far and the optimum is less than the predefined precision ($10^{-5}$ for other test functions in spite of $10^{-3}$ for $F_7$). Parameters are set to (*select-rate*=0.2, *mutate-rate*=0.05, population size *N*=100) for all experiments. The convergence rate and the convergence generations are the average results of 50 runs. The experimental results show that Copula-EDA converges to the global optimum quickly in the test functions. There is not much difference in performance between two copulas for other test functions despite $F_3$ and $F_6$. Both the algorithms proposed in this section perform better than the copula-EDA based on Gaussian copula (see Sect. 4.1) and PBILc [26].

**Table 3** Experimental results of Archimedean copula-EDAs

| Test Function | Copula | Convergence Rate | Convergence Generation |
|---|---|---|---|
| $F_1$ | $C_1$ | 50/50 | 13.7400 |
| | $C_2$ | 50/50 | 13.0600 |
| $F_2$ | $C_1$ | 50/50 | 13.6800 |
| | $C_2$ | 50/50 | 13.1400 |
| $F_3$ | $C_1$ | 50/50 | 4.8000 |
| | $C_2$ | 50/50 | 11.7200 |
| $F_4$ | $C_1$ | 50/50 | 16.2800 |
| | $C_2$ | 50/50 | 16.7000 |
| $F_5$ | $C_1$ | 50/50 | 75.7800 |
| | $C_2$ | 50/50 | 76.7800 |
| $F_6$ | $C_1$ | 50/50 | 43.6800 |
| | $C_2$ | 50/50 | 28.7400 |
| $F_7$ | $C_1$ | 47/50 | 55.5957 |
| | $C_2$ | 50/50 | 32.8600 |
| $F_8$ | $C_1$ | 50/50 | 14.9600 |
| | $C_2$ | 50/50 | 15.1200 |
| $F_9$ | $C_1$ | 48/50 | 12.2083 |
| | $C_2$ | 50/50 | 12.2600 |

## 5 High-Dimensional Copula-EDAs

In fact, many optimization problems are high-dimensional. The 2-dimensional copula-EDAs only show the feasibility to apply copula theory to EDAs. The copula-EDA based on the empirical copula is discussed in this section.

### *5.1 High-Dimensional Copula Constructions*

Let $F(\mathbf{x})$ be the joint of the $D$-dimensional random vector $\mathbf{x}$, the margins of each random variable are $F_i(x)(i=1,2,\ldots,D)$, and the copula, then the equation $C(u_1,u_2,\ldots,u_D)=F(F_1^{-1}(u_1), F_2^{-1}(u_2)\ldots, F_D^{-1}(u_D))$ is gotten according to Sklar's theorem. A random vector $(u_1,u_2,\ldots,u_D)$ obeying $C(\mathbf{u})$ is generated firstly in order to generate a random vector obeying $F(\mathbf{x})$. Subsequently, the vector $(x_1,x_2,\ldots,x_D)$ is

calculated according to the inverse function of the margins $F_i(x)(i=1,2,\ldots,D)$, *i.e.* $x_i= F_i^{-1}(x)(i=1,2,\ldots,D)$.

If the density of the copula $C(\mathbf{u})$ is $c(\mathbf{u})$, then the marginal probability density functions are

$$c_d(u) = c_d(u_1,u_2,...,u_d) = \int_{u_{d+1}=0}^{1}...\int_{u_D=0}^{1} c(u)du_{d+1}...du_D, d=1,2,...,D-1 \quad (22)$$

Obviously, $c(\mathbf{u})=c_D(\mathbf{u})$. The condition distribution functions are

$$C_d(u_d \mid u_1,...,u_{d-1}) = P\{U_d \le u_d \mid U_1=u_1,...,U_{d-1}=u_{d-1}\} = \frac{\int_{u=0}^{u_d} c_d(u_1,...,u_{d-1},u)du}{c_{d-1}(u_1,...,u_{d-1})}, \ d=1,2,\ldots D \quad (23)$$

A method to construct a copula based on the empirical distribution and to sample from the empirical copula is proposed in [34] and it is briefly introduced in the following paragraphs.

Let the samples of the $D$-dimensional random vector $\mathbf{z}$ be $\mathbf{z}_i=(z_{i1},z_{i2},\ldots z_{iD})(i=1,2,\ldots,n)$. The empirical distribution function of each dimension is constructed firstly according to the samples and then each sample $\mathbf{z}_i$ is mapped to a value $u_i$ belonging to $\mathbf{I}^D$. The values $\mathbf{u}_i$ $(i=1,2,\ldots,n)$ are actually the samples of the random vector $\mathbf{u}$ obeying cdf $C(\mathbf{u})$. In order to sample from the estimated cdf $C(\mathbf{u})$, the conditional distribution functions $c_d(\mathbf{u})$ are necessary to be estimated according to the samples $u_i$ $(i=1,2,\ldots,n)$. the details are shown as follows.

The samples $\mathbf{z}_i=(z_{i1},z_{i2},\ldots z_{iD})(i=1,2,\ldots,n)$ are sorted in each dimension. For example, the sorted sequence of the $d^{\text{th}}$ dimension is $z_{(1)d}$, $z_{(2)d}$,…, $z_{(n)d}$, then the empirical distribution of the $d^{\text{th}}$ dimension is

$$F_d(z) = \begin{cases} 0 & z < z_{(1)d} \\ i/n & z_{(i)d} \le z \le z_{(i+1)d}, i=1,2,...,n-1 \\ 1 & z_{(n)d} \le z \end{cases} \quad (24)$$

According to equation (24), the samples $\mathbf{z}_i=(z_{i1},z_{i2},\ldots z_{iD})(i=1,2,\ldots,n)$ are mapped to the vectors $\mathbf{u}_i=(u_{i1},u_{i2},\ldots u_{iD}) \in \{1/n,2/n,\ldots,1\}^D$ $(i=1,2,\ldots,n)$.

Denote $S_1,S_2,\ldots,S_K$ as the partition of the interval $\mathbf{I}=[0,1]$, where $S_i=((i-1)\delta,i\delta]$ (i=1,2,…,K), $\delta=1/K$, K is a positive integer. Thus the D-cube $\mathbf{I}^D$ is divided into $K^D$ subcubes $\Delta_{\boldsymbol{i}}= S_{i_1}\times S_{i_2}\times\ldots\times S_{i_D}$, $\boldsymbol{i}=(i_1,i_2,\ldots,i_D)\in\{1,2,\ldots,K\}^D$. denote $N_{\boldsymbol{i}}$ as the number of points in ,*i.e.* $N_{\boldsymbol{i}}=|\{\mathbf{u}_j|\ \mathbf{u}_j\in\Delta_{\boldsymbol{i}}\}|$. The density of the copula is

$$c(\mathbf{u}) = f_{\mathbf{i}} = N_{\mathbf{i}} / n / \delta^D \quad (25)$$

Let $\uparrow u = \max\{1,\lceil uK\rceil\}$, then $\mathbf{i}=(\uparrow u_1,\uparrow u_2,...,\uparrow u_D)\in\{1,2,...,K\}^D$, and

$$c_d(u_1,...,u_d) = \delta^{D-d} f^{(d)}_{\uparrow u_1,...,\uparrow u_d}, d=1,...,D-1 \quad (26)$$

where,

$$f^{(d)}_{i_1,\ldots,i_d} = \sum_{(i_{d+1},\ldots,i_D)=(1,\ldots,1)}^{(K,\ldots,K)} f_{\mathbf{i}}, \quad (i_1,\ldots,i_d) = \{1,2,\ldots,K\}^d \tag{27}$$

The conditional distribution is

$$C_2(u_2 \mid u_1) = \delta^{D-1} \sum_{i_2=1}^{\downarrow u_2} f^{(2)}_{\uparrow u_1, i_2} + (u_2 - \downarrow u_2 \delta)\delta^{D-2} f^{(2)}_{\uparrow u_1, \uparrow u_2} \tag{28}$$

$$C_d(u_d \mid u_1,\ldots,u_{d-1}) = \frac{\delta \sum_{i_d=1}^{\downarrow u_d} f^{(d)}_{\uparrow u_1,\ldots,\uparrow u_{d-1}, i_d} + (u_d - \downarrow u_d \delta) f^{(d)}_{\uparrow u_1,\ldots,\uparrow u_d}}{\delta f^{(d-1)}_{\uparrow u_1,\ldots,\uparrow u_{d-1}}} \tag{29}$$

where, $\downarrow u = \uparrow u - 1$.

The following algorithm (Algorithm 1) can be applied to calculate the arrays $f^{(d)}$ which we use for the calculation of the conditional distribution functions of the copula.

1. Calculate the empirical distribution functions of the marginal distributions of the sample with equation (24) and the image points $\mathbf{u}_i$ of the sample points $\mathbf{z}_i(i=1,2,\ldots,n)$.
2. Set all elements of the arrays $f^{(d)}$ to 0.
3. for $i$:=1 to $n$ do

```
    for d:=1 to D do
       j_d:=↑u_{d,i};
    end
    for d:=2 to D do
       f^(d)_{j_1,...,j_d} := f^(d)_{j_1,...,j_d} + 1/n/δ^D
    end
end
```

Let $\mathbf{f}=\{f_i^d\}$,( $\boldsymbol{i}=(i_1,i_2,\ldots,i_D) \in \{1,2,\ldots,K\}^D$, $d$=1,2,…,$D$-1).The procedure *generation(***f***)* can be used to generate random numbers $u_1,u_2,\ldots,u_D$ obeying the empirical copula.

1. Generate two independent random numbers $u_1$ and $u$;
2. Calculate $u_2 = \downarrow u_2 \delta + \dfrac{u - \delta^{D-1} \sum_{i_2=1}^{\downarrow u_2} f^{(2)}_{\uparrow u_1, i_2}}{\delta^{D-2} f^{(2)}_{\uparrow u_1, \uparrow u_2}}$, where $\downarrow u_2$ is the minimal number in $\{0,1,\ldots,K\text{-}1\}$ satisfied with the condition $uK^{D-1} \le \sum_{i_2=1}^{\downarrow u_2+1} f^{(2)}_{\uparrow u_1, i_2}$;

3. for $d$:=3 to $D$

   – 3.1. Generate a random number $u$ in $\mathbf{I}$;

   – 3.2. Calculate $u_d = \downarrow u_d \delta + \delta \dfrac{uf^{(d-1)}_{\uparrow u_1,\ldots,\uparrow u_{d-1}} - \sum_{i_d=1}^{\downarrow u_d} f^{(d)}_{\uparrow u_1,\ldots,\uparrow u_{d-1},i_d}}{f^{(d)}_{\uparrow u_1,\ldots,\uparrow u_d}}$, where $\downarrow u_d$ is the minimal number in $\{0,1,\ldots,K\text{-}1\}$ satisfied with the condition $uf^{(d-1)}_{\uparrow u_1,\ldots,\uparrow u_{d-1}} \le \sum_{i_d=1}^{\downarrow u_d+1} f^{(d)}_{\uparrow u_1,\ldots,\uparrow u_{d-1},i_d}$

   end

## 5.2 Copula-EDA Based on Empirical Copula

For the optimization problem min $f(x_1,x_2,\ldots,x_D)$, $x_i \in [a_i,b_i],(i=1,2,\ldots,D)$, the copula-EDA based on the above empirical copula is in the following [37].

1. Generate $N$ individuals in the search space randomly. Decide the selection rate $s$ and the mutation rate $t$;
2. Select $n=s\times N$ individuals $\{\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_n\}$ among the $N$ individuals as the promising population, and reserve them in the next generation;
3. Calculate $\mathbf{f}=F(\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_n)$ by doing Algorithm 1
4. Repeat the following steps $m=(1\text{-}s\text{-}t)\times N$ times, add the new $m$ individuals into the population.

   – 4.1 $(u_1,u_2,\ldots,u_D)$=*generation*($\mathbf{f}$);

   – 4.2 for $i$:=1 to $D$

   $x_i=F_i^{-1}(u_i)$, where $F_i$ is the empirical distribution (24) or the normal distribution

   end

   $(x_1,x_2,\ldots,x_D)$ is the new individual.

5. Generate $t\times N$ individuals in the search space randomly as the new individuals;
6. Stop if the terminate condition is met, and the best individual of the population is the optimal result.

The following 3 functions are used to test the proposed algorithm.

- Sumcan function: $f_1(x) = -\{10^{-5} + \sum_{i=1}^{D} | y_i |\}^{-1}$, where, $y_1=x_1$, $y_i=y_{i-1}+x_i$, $i=2,\ldots,d$, $-0.16 \le x_i \le 0.16$
- Schwefel function: $f_2(x) = \sum_{i=1}^{D} [(x_1 - x_i^2)^2 + (x_i - 1)^2]$, where, $-10 \le x_i \le 10$

- Griewank function: $f_3(x) = 1 + \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}})$ , where, $-600 \le x_i \le 600$

The above three functions are all minimal optimization problem. The empirical copula and the normal margins are used in the tested copula-EDA, and is denoted as $cEDA_{PG}$. The parameters are the same as in [30], *i.e.* the dimension of the problem is D=10, the population size of the three problem are 2000, 2000 and 750 respectively. The maximal fitness evaluation is 300,000. The truncation selection is used. The mutation is used and the mutation rate is 0.05. The search space is not divided when the empirical copula is constructed, *i.e.* K=1. The variance for sampling is one of the following three strategies.

A. The variance is fixed.

B. The variance is set to the sample variance, *i.e.* $\sigma_i = \sqrt{\frac{\sum_{j=1}^{S} (x_{ij} - \overline{x_i})^2}{S-1}}$, where, $\overline{x_i} = \frac{1}{S} \sum_{j=1}^{S} x_{ij}$ .

C. The variance is linearly changed, *i.e.*

$$\sigma_i^{t+1} = (1-\alpha)\sigma_i^t + \alpha \sqrt{\frac{\sum_{j=1}^{S} (x_{ij} - \overline{x_i})^2}{S-1}}$$ , α=0.2

**Table 4** The comparison among cEDA and other algorithms on Sumcan function

| Algorithm | Selection rate | σ | F1 | | | |
|---|---|---|---|---|---|---|
| | | | mean | StdVar | min | max |
| $UMDA_c^G$ | | | -53460 | | | |
| $MIMIC_c^G$ | | | -58775 | | | |
| $EGNA_{ee}$ | | | -100000 | | | |
| $EGNA_{BGe}$ | | | -100000 | | | |
| ES | | | -5910 | | | |
| $cEDA_{PG}$ | 0.5 | A:0.02 | -30.2775 | 3.4345 | -37.9727 | -26.2.83 |
| | 0.5 | A:0.05 | -12.5174 | 1.4837 | -15.7628 | -11.0210 |
| | 0.2B | B | **-60770** | **27564** | **-99866** | **-6390.9** |
| | 0.2 | B | -1074.6 | 482.2301 | -2168.2 | -513.8331 |
| | 0.5 | C | -184.0976 | 52.9640 | -235.0931 | -46.0561 |
| | 1/3 | C | -561.7312 | 178.8254 | -764.9960 | -114.3033 |
| | 0.2B | C | -9910.3 | 2692.9 | -12511 | -2786.9 |
| | 0.2 | C | -2089.4 | 587.4801 | -2784.2 | -547.5289 |

**Table 5** The comparison among cEDA and other algorithms on Schwefel function

| Algorithm | Selection rate | σ | F2 | | | |
|---|---|---|---|---|---|---|
| | | | mean | StdVar | min | max |
| $UMDA_c^G$ | | | 0.13754 | | | |
| $MIMIC_c^G$ | | | 0.13397 | | | |
| $EGNA_{ee}$ | | | 0.09914 | | | |
| $EGNA_{BGe}$ | | | 0.0250 | | | |
| ES | | | 0 | | | |
| $cEDA_{PG}$ | 0.5 | A:.02 | 4.0743 | 0.8572 | 2.8008 | 5.3325 |
| | 0.5 | A:.05 | 0.0097 | 0.0032 | 0.0061 | 0.0167 |
| | 0.2B | B | $2.3833\times10^{-4}$ | $5.2429\times10^{-5}$ | $1.2190\times10^{-4}$ | $3.6281\times10^{-4}$ |
| | 0.2 | B | 0.1478 | 0.0259 | 0.1066 | 0.1954 |
| | 0.5 | C | $8.2443\times10^{-4}$ | 0.0019 | $1.0471\times10^{-4}$ | 0.0063 |
| | 1/3 | C | $1.8707\times0^{-5}$ | $4.6817\times10^{-5}$ | $2.8442\times10^{-6}$ | $1.5194\times10^{-4}$ |
| | 0.2 | C | **$3.0034\times10^{-7}$** | **$7.3179\times10^{-7}$** | **$4.0579\times10^{-8}$** | **$2.3825\times10^{-6}$** |

**Table 6** The comparison among cEDA and other algorithms on Griewank function

| Algorithm | Selection rate | σ | F3 | | | |
|---|---|---|---|---|---|---|
| | | | mean | StdVar | min | max |
| $UMDA_c^G$ | | | 0.011076 | | | |
| $MIMIC_c^G$ | | | 0.007794 | | | |
| $EGNA_{ee}$ | | | 0.008175 | | | |
| $EGNA_{BGe}$ | | | 0.012605 | | | |
| ES | | | 0.034477 | | | |
| $cEDA_{PG}$ | 0.5 | A:.02 | 1.0796 | 0.1977 | 0.6106 | 1.2775 |
| | 0.5 | A:.05 | 0.5489 | 0.3694 | 0.2215 | 1.4507 |
| | 0.2B | B | 0.0046 | 0.0151 | 0 | 0.0857 |
| | 0.2 | B | **0** | **0** | **0** | **0** |
| | 0.5 | C | $1.3192\times10^{-9}$ | $3.1529\times10^{-9}$ | $1.2085\times10^{-10}$ | $1.0288\times10^{-8}$ |
| | 1/3 | C | $4.2966\times10^{-14}$ | $1.0565\times10^{-13}$ | $7.2164\times10^{-15}$ | $3.4361\times10^{-13}$ |
| | 0.2 | C | **0** | **0** | **0** | **0** |

The experimental results in Table 4-6 indicate that the performance of $cEDA_{PG}$ is affected by the parameters. cEDA finds the better results than other compared algorithms in certain parameters. Especially for Griewank function, $cEDA_{PG}$ finds the optimal result after 68.7 generations when selection rate is 0.2 and the variance is changed in strategy B. It can be indicated from Fig. 12-13 that $cEDA_{PG}$ is good at exploration and is not good at exploitation.

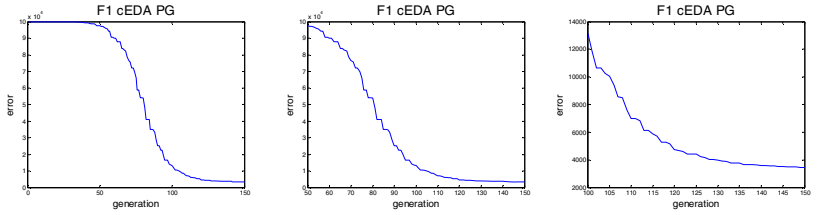



**Fig. 12** The performance of $cEDA_{PG}$ in Sumcan function

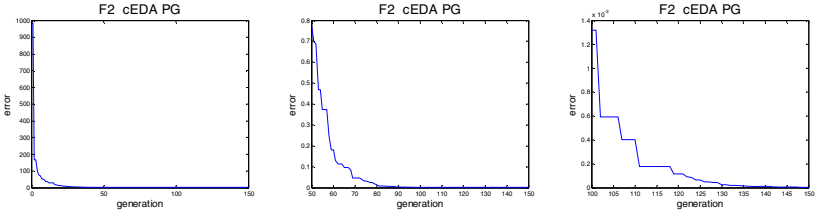



**Fig. 13** The performance of $cEDA_{PG}$ in Schwefel function

## 6 Conclusion

Estimation of Distribution Algorithms utilize the distribution characters of the selected population to guide the creation of the next generation. Therefore, the population evolves towards the optimal direction and finds the optimal solution quickly. Two key steps in EDAs are to estimate the distribution of the selected population and to sample from the distribution model.

Copula theory contributes to the estimation of the distribution model and sampling from the estimated model because copula theory provides a way to present the multivariate joint distribution with the univariate marginal distributions and a function called copula. How to construct copula and to sample from the copula are also the research topics of copula theory.

Copula theory is applied into the research of EDAs and three EDAs based on different copulas are introduced in this chapter. They are respectively copula-EDA based on Gaussian copula, copula-EDA based on Archimedean copula and copula-EDA based on empirical copula. All of them can find the optimal results quick than the other algorithms used for comparison. But there are also shortages in them. For example, the copula-EDA based on empirical copula is not good at exploitation and the convergence speed in last generations is slow and the algorithm is affected by the selection of parameters.

Different copula is corresponded to different relationship of the random variables. So the performance of copula-EDAs based on different copula is different on optimization problems. The following questions need to be studied further.

- The expansion of copula-EDA based on Gaussian copula and Archimedean copula on high-dimensional optimization problems.
- How to select and construct copula.

- The comparison among the copula-EDAs based on different copulas.
- The relationship between the characters of the optimization problem and the optimal copula.

## References

[1] Larranaga, P., Lozano, J.A.: Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Boston (2002)
[2] Baluja, S.: Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Citeseer (1994), doi: 10.1.1.9.8084
[3] Baluja, S.: An Empirical Comparison of Seven Iterative and Evolutionary Function Optimization Heuristics. Technical Report CMU-CS-95-193, Computer Science Department, Carnegie Mellon University (1995)
[4] Hohfeld, M., Rudolph, G.: Towards a Theory of Population Based Incremental Learning. In: Proc. of the 4th International Conference on Evolutionary Computation, pp. 1–5 (1997)
[5] Cristina, G., Lozano, J.A., Larranaga, P.: Analyzing the PBIL Algorithm by means of Discrete Dynamical Systems. Complex Syst. 12(4), 465–479 (2001)
[6] Muhlenbein, H., Paa$\beta$, G.: From recombination of genes to the estimation of distributions I. Binary Parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
[7] Muhlenbein, H.: The Equation for Response to Selection and its use for Prediction. Evol. Comput. 5(3), 303–346 (1997)
[8] Shapiro, J.L.: Drift and Scaling in Estimation of Distribution Algorithms. Evol. Comput. 13(1), 99–123 (2005)
[9] Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. In: Proc. IEEE Conf. Evol Computation, Indianapolis, USA, pp. 523–528 (1998)
[10] De Bonet, J.S., Isbell, C.L., Viola, P.: MIMIC: Finding optima by estimation probability densities. In: Becker, S. (ed.) Advances in Neural Information Processing Systems, pp. 424–430. MIT Press, Cambridge (1997)
[11] Baluja, S., Davies, S.: Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space. In: Proc. of the 14th international conference on machine learning, pp. 30–38. Morgan Kaufmann, San Francisco (1977)
[12] Pelican, M., Muhlenbein, H.: The birandom variable marginal distribution algorithm. In: Furuhashi, T. (ed.) Advances in Soft Computing-Engineering Design and Manufacturing, pp. 521–535. Springer, London (1999)
[13] Pelikan, M., Muhlenbein, H.: Marginal Distributions in Evolutionary Algorithms. In: Proc. of the International Conference on Genetic Algorithms, pp. 90–95. Technical University of Brno, Brno (1998)
[14] Harik, G.: Linkage Learning via Probabilistic Modeling in the ECGA, Illigal Rep. No.99010, Illinois Genetic Algorithms Lab., University of Illinois, Urbana-Champaign, Illinois (1999)
[15] Muhlenbein, H., Mahnig, T.: FDA- a scalable evolutionary algorithm for the optimization of additively decomposed functions. Evolu. Comput. 7(4), 353–376 (1999)

[16] Muhlenbein, H., Mahnig, T.: Convergence Theory and Applications of the Factorized Distribution Algorithm. Journal of Comput and Information Technology 7(1), 19–32 (1999)
[17] Muhlenbein, H., Mahnig, T.: The Factorized Distribution Algorithm for Additively Decomposable Functions. In: Second Symposium on Artificial Intelligence. Adaptive Systems. CIMAF 1999, La Habana, pp. 301–313 (1999)
[18] Muhlenbein, H., Mahnig, T.: Evolutionary algorithms: From recombination to search distributions. In: Kallel, L., Naudts, B., Rogers, A. (eds.) Theoretical Aspects of Evolutionary Computating., pp. 137–176. Springer, London (2001)
[19] Muhlenbein, H., Mahnig, T., Ochoa, A. R.: Distributions and Graphical Models in Evolutionary Optimization. J. Heuristics 5, 215–247 (1999)
[20] Pelikan, M., Goldberg, D.E., Cantu-Paz, E.: BOA: the Bayesian optimization algorithm. In: Proc. Genetic and Evolutionary Computation Conference (GECCO 1999), Orlando, FL, pp. 525–532 (1999)
[21] Pelikan M.: A Simple Implementation of Bayesian Optimization Algorithms in C++(Version 1.0). Illegal Report No. 99011, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, Illinois (1999)
[22] Pelikan, M., Goldberg, D.E., Cantu-Paz, E.: Bayesian Optimization Algorithm, Population Sizing, and Time to Convergence. IlleGAL Report No. 2000001, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana Champaign, Urbanan, Illinois (2000)
[23] Pelikan, M.: Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms. Springer, New York (2005)
[24] Gao, Y., Culberson, J.: Space Complexity of Estimation of Distribution Algorithms. Evol. Comput. 13(1), 125–143 (2005)
[25] Pelikan, M., Sastry, K., Goldberg, D. E.: Scalability of the Bayesian Optimization Algorithm. International Journal of Approximate Reasoning 31(3), 221–258 (2002)
[26] 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)
[27] Larranaga, P., Etxeberria, R., Lozano, J.A., Pena, J.M.: Optimization in continuous domains by learning and simulation of Gaussian networks. In: Proc. 2000 Genetic and Evolutionary Computation Conf. Workshop Program, Las Vegas, Nevada, pp. 201–204 (2000)
[28] Rudlof, S., Koppen, M.: Stochastic Hill Climbing by Vectors of Normal Distributions. In: Proc. of the first online workshop on soft computing(WSC1), Nagoya, Japan (1996)
[29] Larrañaga, P., Lozano, J.A., Bengoetxea, E.: Estimation of Distribution Algorithms based on multivariate normal and Gaussian networks. Technical Report KZZA-IK-1-01 of the Department of Computer Science and Artificial Intelligence, University of the Basque Country, Spain (2001)
[30] Larranaga, P., Etxeberria, R., Lozano, J.A., Pena, J.M.: Optimization in continuous domains by learning and simulation of Gaussian networks. In: Proc. of the Genetic and Evolutionary Computation Conference, GECCO 2000, Las Vegas, Nevada, USA, July 8-12, pp. 201–204. Morgan Kaufmann, San Francisco (2000)
[31] Bosman, P.A.N., Thierena, 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)

[32] Nelsen, R.B.: An introduction to copulas, 2nd edn. Springer, New York (2006)
[33] Whelan, N.: Sampling from Archimedean Copulas (2004), http://home.golden.net/~annaw/Niall/papers/Archimedean.pdf (Accessed January 1, 2009)
[34] Strelen, J.C., Nassaj, F.: Analysis and generation of random vectors with copulas. In: Henderson, S.G., Biller, B., Hsieh, M.H., Shortle, J., Tew, J.D., Barton, R.R. (eds.) Proc. of the 2007 Winter Simulation Conference (WSC 2007), Washington, DC, USA, pp. 488–496 (2007)
[35] Wang, L.F., Zeng, J.C., Hong, Y.: Estimation of Distribution Based on Copula Theory. In: IEEE CEC 2009, Trondheim, Norway, May 18-21, pp. 1057–1063 (2009)
[36] Wang, L.F., Zeng, J.C., Hong, Y.: Estimation of Distribution Based on Archimedean Copulas. In: ACM SIGEVO GEC 2009, Shaihai, China, June 12-14, pp. 993–996 (2009)
[37] Wang, L.F., Zeng, J.C., Hong, Y.: Estimation of Distribution Algorithm Modeling and Sampling by means of Copula. Submitted to: Chinese journal of computers (in Chinese) (2009)

# Analyzing the $k$ Most Probable Solutions in EDAs Based on Bayesian Networks

Carlos Echegoyen, Alexander Mendiburu, Roberto Santana, and Jose A. Lozano

**Abstract.** Estimation of distribution algorithms (EDAs) have been successfully applied to a wide variety of problems but, for the most complex approaches, there is no clear understanding of the way these algorithms complete the search. For that reason, in this work we exploit the probabilistic models that EDAs based on Bayesian networks are able to learn in order to provide new information about their behavior. Particularly, we analyze the $k$ solutions with the highest probability in the distributions estimated during the search. In order to study the relationship between the probabilistic model and the fitness function, we focus on calculating, for the $k$ most probable solutions (MPSs), the probability values, the function values and the correlation between both sets of values at each step of the algorithm. Furthermore, the objective functions of the $k$ MPSs are contrasted with the $k$ best individuals in the population. We complete the analysis by calculating the position of the optimum in the $k$ MPSs during the search and the genotypic diversity of these solutions. We carry out the analysis by optimizing functions of different natures such as Trap5, two variants of Ising spin glass and Max-SAT. The results not only show information about the relationship between the probabilistic model and the fitness function, but also allow us to observe characteristics of the search space, the quality of the setup of the parameters and even distinguish between successful and unsuccessful runs.

Carlos Echegoyen · Alexander Mendiburu · Jose A. Lozano
Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country, Paseo Manuel de Lardizábal 1, 20080 San Sebastián - Donostia, Spain
e-mail: {carlos.echegoyen,alexander.mendiburu,ja.lozano}@ehu.es

Roberto Santana
Universidad Politécnica de Madrid, Campus de Montegacedo sn. 28660. Boadilla del Monte, Madrid, Spain
e-mail: roberto.santana@upm.es

Y.-p. Chen (Ed.): Exploitation of Linkage Learning, ALO 3, pp. 163–189.
springerlink.com 

## 1 Introduction

Estimation of distribution algorithms (EDAs) [23, 32] are an evolutionary computation branch which have been successfully applied in problems of different domains [2, 27, 40]. However, despite their successful results there are a wide variety of open questions [43] regarding the behavior of this type of algorithms. Therefore, it is necessary to continue studying EDAs in order to better understand how they solve problems and advance their development.

The main characteristic of EDAs is the use of probabilistic models instead of the typical crossover and mutation operators employed by genetic algorithms [17]. Thus, linkage learning, understood as the ability to capture the relationships between the variables of the optimization problem, is accomplished in EDAs by detecting and representing probabilistic dependencies using probability models. This type of algorithms uses machine learning methods to extract relevant features of the search space through the selected individuals of the population. The collected information is represented using a probabilistic model which is later employed to generate new solutions. In this way, a learning and sampling iterative process is used to lead the search to promising areas of the search space.

The models employed to encode the probability distributions are a key point in the performance of EDAs. In this regard, probabilistic graphical models [7] are a powerful tool, and in particular, Bayesian networks have been extensively applied in this field [15, 29, 38]. One of the benefits of EDAs that use these kind of models is that the complexity of the learned structure depends on the characteristics of the selected individuals. Additionally, the Bayesian networks learned during the search are suitable for human interpretation, helping to discover unknown information about the problem structure. In fact, a straightforward form of analyzing EDAs based on Bayesian networks is through the explicit interactions among the variables they provide. In this sense, it has been shown how different parameters of the algorithm influence the structural model accuracy [25], how the dependencies of the probabilistic models change during the search [19] and how the networks learned can provide information about the problem structure [11, 14, 19].

In previous works [12, 13], we took a different path in the study of EDAs by recording probabilities and function values of distinguished solutions of the search space during the run. In this chapter, we extend that work focusing on the $k$ most probable solutions in the probability distribution encoded by the probabilistic models at each generation of an estimation of Bayesian network algorithm (EBNA) [15]. By collecting different measurements, we analyze these solutions using four classes of test problems which have been widely used in EDAs: Trap5, two-dimensional Gaussian Ising spin glasses, two-dimensional $\pm J$ Ising spin glasses and Max-SAT. We argue that the analysis proposed supports a different perspective that is able to reveal patterns of behavior inside this type of algorithms. Furthermore, we show that the probabilistic models, besides structural information about the problem, can contain useful information about the function landscape and also about the quality of the search. Thus, the results provided in this work promote the exploitation

of linkage learning in order to learn more about the algorithms and advance their development.

The rest of the chapter is organized as follows. Section 2 introduces Bayesian networks, the calculation of the $k$ most probable solutions and presents estimation of Bayesian network algorithms. Section 3 explains the experimental design. Sections 4, 5, 6 and 7 discuss Trap5, Gaussian Ising, $\pm J$ Ising and Max-SAT problems respectively, analyzing the behavior of the $k$ most probable solutions during the optimization process of each function. Section 8 discusses relevant previous works and finally, Section 9 draws the conclusions obtained during the study.

## 2 Background

### 2.1 Bayesian Networks

Formally, a Bayesian network [7] is a pair $(S, \theta)$ representing a graphical factorization of a probability distribution. The structure $S$ is a directed acyclic graph which reflects the set of conditional (in)dependencies among $n$ random variables $\mathbf{X} = (X_1, \ldots, X_n)$. On the other hand, $\theta$ is a set of parameters for the local probability distributions associated with each variable.

The factorization of the probability distribution is codified by $S$:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|\mathbf{pa}_i) \tag{1}$$

where $\mathbf{pa}_i$ denotes a value of the variables $\mathbf{Pa}_i$, the parent set of $X_i$ in the graph $S$.

With reference to the set of parameters $\theta$, if the variable $X_i$ has $r_i$ possible values, the local distribution $p(x_i|\mathbf{pa}_i^j, \theta_i)$ is an unrestricted discrete distribution:

$$p(x_i^k|\mathbf{pa}_i^j, \theta_i) \equiv \theta_{ijk} \tag{2}$$

where $\mathbf{pa}_i^1, \ldots, \mathbf{pa}_i^{q_i}$ denote the $q_i$ possible values of the parent set $\mathbf{Pa}_i$. In other words, the parameter $\theta_{ijk}$ represents the probability of variable $X_i$ being in its $k$-th value, knowing that the set of its parents' variables is in its $j$-th value. Therefore, the local parameters are given by $\theta_i = ((\theta_{ijk})_{k=1}^{r_i})_{j=1}^{q_i}$.

### 2.2 Learning Bayesian Networks from Data

In order to complete the Bayesian network learning, it is necessary to obtain a structure $S$ and a set of parameters $\theta$ from a data set. Firstly, although there are different strategies to learn the structure, we focus on a method called "score + search" which is the one used in the experiments presented in this paper. This approach is based on a procedure to search high quality structures according to a determined metric or score. This score tries to measure how well a given structure $S$ represents the underlying probability distribution of a dataset $D$. Particularly, in this work we use the Bayesian Information Criterion score ($BIC$) [44] based on penalized maximum

likelihood. In order to make the search in a space of structures, we use Algorithm B [6] because it is able to return good results efficiently. Algorithm B is a greedy search procedure which starts with an arcless structure and, at each step, adds the arc with the maximum improvement in the score. The algorithm finishes when there is no arc whose addition improves the score.

Once the structure has been learned, the parameters of the Bayesian network are calculated using the Laplace correction:

$$\hat{\theta}_{ijk} = \frac{N_{ijk} + 1}{N_{ij} + r_i} \tag{3}$$

where $N_{ijk}$ denotes the number of cases in $D$ in which the variable $X_i$ has the value $x_i^k$ and $\mathbf{Pa}_i$ has its $j^{th}$ value, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$.

## 2.3 Estimation of Distribution Algorithms Based on Bayesian Networks

Following the main scheme of EDAs, EBNA [15] works with populations of $N$ individuals that constitute sets of $N$ candidate solutions. The initial population is generated according to a uniform distribution, and hence, all the solutions have the same probability to be sampled. Each iteration starts by selecting a subset of promising individuals from the population. Although there are different selection methods, in this case we use truncation selection with threshold 50%. Thus, the $N/2$ individuals with the best fitness value are selected. The next step is to learn a Bayesian network from the subset of selected individuals. Once the Bayesian network is built, the new

**Algorithm 1. EBNA**

1 $BN_0 \leftarrow (S_0, \theta^0)$ where $S_0$ is an arc-less structure, and $\theta^0$ is uniform
2 $D_0 \leftarrow$ Sample $N$ individuals from $BN_0$
3 t $\leftarrow 1$
4 **do** {
5 $D_{t-1} \leftarrow$ Evaluate individuals
6 $D_{t-1}^{Se} \leftarrow$ Select $N/2$ individuals from $D_{t-1}$
7 $S_t^* \leftarrow$ Obtain a network structure
8 $\theta^t \leftarrow$ Calculate $\theta_{ijk}^t$ using $D_{t-1}^{Se}$ as the data set
9 $BN_t \leftarrow (S_t^*, \theta^t)$
10 $D_t \leftarrow$ Sample $N-1$ individuals from $BN_t$ and create the new population
11 } **until** Stop criterion is met

population can be generated. At this point there are different possibilities. We use an elitist criterion. From the Bayesian network, $N-1$ new solutions are sampled and then mixed with the $N$ individuals of the current population. The $N$ best individuals, among the $2N-1$ available, constitute the new population. The procedure of selection, learning and sampling is repeated until a stop condition is fulfilled. A pseudocode of EBNA is shown in Algorithm 1.

### *2.4 Abductive Inference and Most Probable Configurations*

Given a probabilistic graphical model, the problem of finding the most probable configuration (MPC), consists of finding a complete assignment with maximum probability that is consistent with the evidence. Similarly, the problem of finding the $k$ most probable configurations consists of finding the $k$ configurations with maximum probability in the distribution encoded by the graphical model. The name given to the most probable configurations changes between the different domains of application and according to the type of graphical models used. It is also known as the most probable explanation problem, and the $n$-best list [35] or $n$-best hypothesis problem [1]. In Bayesian networks, the process of generating the MPC is usually called abductive inference. In our context of EDAs, these points will be called the most probable solutions (MPSs).

Computing the MPC can be easily done on a junction tree by a simple message-passing algorithm [9]. However, the computation of the $k$ MPCs requires the application of more complex schemes [34, 45]. In any case, when the inference is done by using junction trees the results are exact.

In this work we use the algorithm presented in [34]. It has been conceived for finding the most probable configurations in a junction tree and therefore gives us exact solutions. It is based on the use of a simple message-passing scheme and a dynamic programming procedure to partition the space of solutions. The algorithm starts by finding the most probable configuration, and from this configuration, the space of solutions is partitioned in disjunct sets for which the respective MPC is found. The second MPC is the one with the highest probabilities among all of the partitions. More details about the algorithm can be found in [34].

The algorithm used in this work has been implemented using the Bayes Net Toolbox [33] and can be found in [42].

## 3 Experimental Framework

In order to analyze the $k$ most probable solutions at each step of an EBNA, we deal with four test problems and take different measurements. Next, we explain in detail the components of the experiments. In [42], the necessary tools to reproduce the experiments or to carry out a similar analysis are implemented.

### 3.1 Problems

The whole set of problems is based on additively decomposable functions (ADFs) defined as,

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{s_i}) \tag{4}$$

where $\mathbf{S_i} \subseteq \mathbf{X}$. Trying to cover a wide spectrum of applications and observe the behavior of EDAs in different scenarios, we chose these four test problems: Trap5, Gaussian Ising, $\pm J$ Ising and Max-SAT. The details of each one are introduced in the following sections. These problems are selected for several reasons. Firstly, in order to investigate the influence of multimodality in the behavior of EDAs, we deal with problems that have different number of optimal solutions. The first two problems have a unique optimum and the last two problems have several optima. Secondly, all of them are optimization problems which have been widely used to analyze EDAs [4, 19, 37]. And finally, all the problems have a different nature. Trap5 [10] is a deceptive function designed in the context of genetic algorithms [17] aimed at finding their limitations. It is a separable function and in practice can be easily optimized if the structure is known. Gaussian Ising and $\pm J$ Ising come from statistical physics domains and are instances of the Ising model proposed to analyze ferromagnetism [22]. The variables are disposed on a grid and the interactions do not allow dividing the problem into independent subproblems of bounded order efficiently [31]. The Ising problem is a challenge in optimization [19, 37] and in its general form is NP-complete [3]. Max-SAT is a variation for optimization of a classic benchmark problem in computational complexity, the propositional satisfiability or SAT. In fact, SAT was the first problem proven to be NP-complete in its general form [8]. An instance of this problem can contain a very high number of interactions among variables and, in general, it can not be efficiently divided into subproblems of bounded size in order to reach the optimum. Except for the function Trap5, we have dealt with two instances for each type of problem. We only show the results for the instance with the most relevant information.

### 3.2 Measurements

Our first goal is to study the relationship between the probabilistic model and the function during the search. To that end, we obtain the probabilities and function values for the $k$ MPSs at each step of EBNA and then, we calculate the Pearson correlation coefficient $\rho$ for each pair of samples (functions and probabilities). The correlation coefficient $\rho$ can be expressed as,

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{5}$$

where $X$ and $Y$ are random variables, $cov$ means covariance and $\sigma$ is the standard deviation.

We also calculate, at each step of the algorithm, the average function values for the $k$ MPSs and the $k$ best individuals of the population. Moreover, we are interested in knowing the position of the optimum in the ordered set of the $k$ MPSs at each generation. Finally, trying to observe, from a genotypic point of view, how different the $k$ MPSs are, we calculate at each EBNA step the entropy of the $k$ MPSs by means of adding the entropy of each variable belonging to the function,

$$H(\mathbf{X}) = -\sum_{i=1}^{n}\sum_{j=1}^{r_i} p(x_i^j)\cdot \log_2 p(x_i^j) \tag{6}$$

where $n$ is the number of variables and $r_i$ is the number of states of each variable.

### 3.3 Parameter Configuration

Sample size is very important in order to learn Bayesian networks [16] and, hence, an important parameter in EDAs based on this type of models. Thus, we use two different population sizes in order to analyze their influence in the algorithm. Firstly, we have used the bisection method [36] to determine an adequate population size to reach the optimum (with high probability). This size is denoted by $m$. The stopping criterion for bisection is to obtain the optimum in 5 out of 5 independent runs. The final population size is the average over 20 successful bisection runs. The second population size is half of the bisection, $m/2$. With this size we try to create a more realistic scenario in which achieving the optimum is less likely. Thus, we can analyze the algorithm when the optimum is not reached.

The calculation of the $k$ most probable solutions has a high computational cost. Therefore, it is necessary to limit both $k$ and the number of function variables. In this work we deal with $k = 50$ and with two different problem sizes: dimension $n = 50$ for Trap5 and Max-SAT and dimension $n = 64$ (grid $8 \times 8$) for Gaussian Ising and $\pm J$ Ising. The stopping criterion for EBNA is a fixed number of iterations, in particular 30. It it is independent of obtaining the optimum. This number is enough to observe the convergence of the algorithm.

Finally, for each experiment type i.e. EBNA solving a problem with a given population size, 20 independent runs have been carried out. Each set of 20 executions is divided into successful (the optimum is reached) and unsuccessful (the optimum is not reached) runs which will be analyzed separately.

## 4 Analyzing the $k$ MPSs in Trap5

### 4.1 Trap5 Description

Our first function, Trap5 [10], is an additively separable (non overlapping) function with a unique optimum. It divides the set $\mathbf{X}$ of $n$ variables, into disjoint subsets $\mathbf{X}_I$ of 5 variables. It can be defined using a unitation function $u(\mathbf{y}) = \sum_{i=1}^{p} y_i$ where $\mathbf{y} \in \{0,1\}^p$ as,

$$\text{Trap5}(\mathbf{x}) = \sum_{I=1}^{\frac{n}{5}} trap_5(\mathbf{x}_I) \tag{7}$$

where $trap_5$ is defined as,

$$trap_5(\mathbf{x}_I) = \begin{array}{ll} 5 & \text{if } u(\mathbf{x}_I) = 5 \\ 4 - u(\mathbf{x}_I) & \text{otherwise} \end{array} \tag{8}$$

and $\mathbf{x}_I = (x_{5I-4}, x_{5I-3}, x_{5I-2}, x_{5I-1}, x_{5I})$ is an assignment to each trap partition $\mathbf{X}_I$. This function has one global optimum in the assignment of all ones for $\mathbf{X}$ and a large number of local optima, $2^{n/5} - 1$.

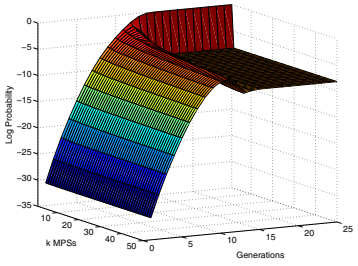
Trap5 function has been used in previous works [19] to study the structure of the probabilistic models in EDAs based on Bayesian networks as well as the influence of different parameters [25]. It is important to note that this function is difficult to optimize if the probabilistic model is not able to identify interactions between variables [12].

## *4.2 Experimental Results*

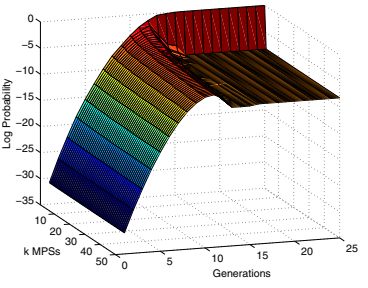
In this section, we present and discuss the results obtained when EBNA tries to optimize the Trap5 function. Firstly, in Fig. 1 and Fig. 2 we show the probabilities and function values for the $k$ MPSs during the search and also their Pearson correlation using different population sizes respectively. We can observe that the probability distribution tends to concentrate on a unique solution at the end of the run assigning the same probability to the remainder $k-1$ MPSs. This behavior is analogous for the function values, where the MPS has the highest function value and the rest of $k-1$ MPSs have lower values. In general, the correlation rapidly increases and decreases in the first generations and grows during the rest of the search.

Particularly, a very interesting observation from these results is the difference not only between population sizes but also between successful and unsuccessful runs. Firstly, with population size $m/2$ (Fig. 2), the correlation suffers a clearer decrease in the middle of the run than with population size $m$. The function values for the $k$ MPSs also reflect a difference between population sizes. Thus, with size $m/2$ (Fig. 2) the function values grow more slowly during the search. Secondly, we have different behaviors depending on the success of the search. When EBNA reaches the optimum with both population sizes, we can see that the correlation tends to 1 at the end of the run. This is independent of the number of $k$ MPSs that we take into account. However, in unsuccessful runs the correlation at the end of the run is lower with higher values of $k$. Moreover, in unsuccessful runs, the correlation reaches lower values with population size $m/2$ than with $m$.

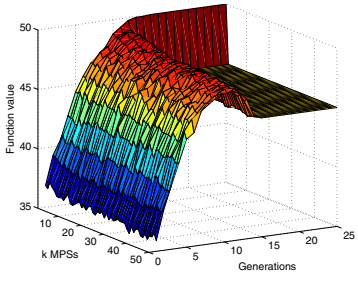
The charts of probability values in logarithmic scale have a constant pattern in all cases (Figs. 1 and 2). However, there are clear differences in the behavior of the function values between successful and unsuccessful runs in both scenarios (population sizes $m$ and $m/2$). This difference is particularly notable in the last generations when EBNA converges to a unique solution [12]. This indicates that these function



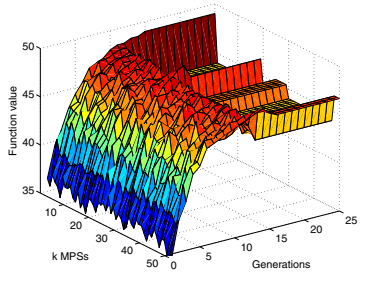

(a) Log. of the probability. Successful



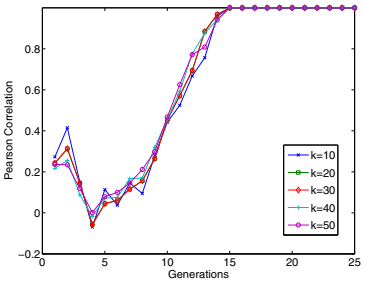

(b) Log. of the probability. Unsuccessful



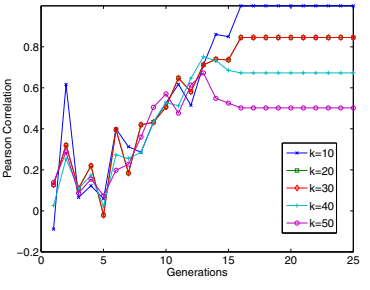

(c) Function values. Successful




(d) Function values. Unsuccessful



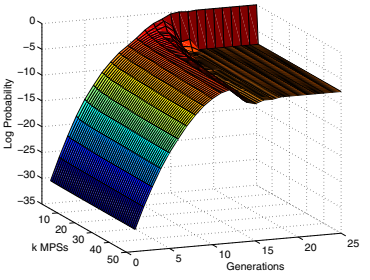

(e) Correlation. Successful



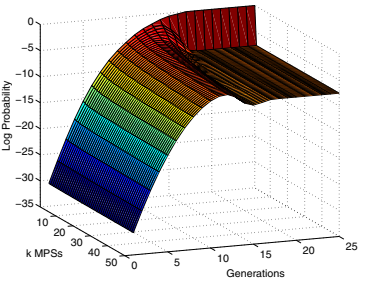

(f) Correlation. Unsuccessful

**Fig. 1** Probability values, function values and Pearson correlation of the 50 most probable solutions at each generation of EBNA when it is applied to Trap5 with population size $m$. In (a), (c), (e) we provide the results for 17 successful runs out of 20. In (b), (d), (f) we provide the results for 3 unsuccessful runs

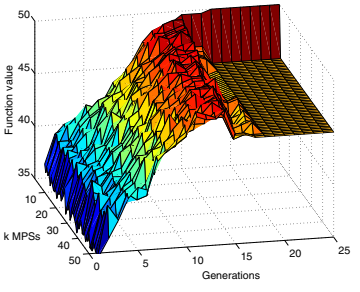
values are mainly responsible for the different behavior in the correlation between successful and unsuccessful runs. The function values for the $k$ MPSs and therefore the correlation, are an important source of information to distinguish both types of runs in Trap5.



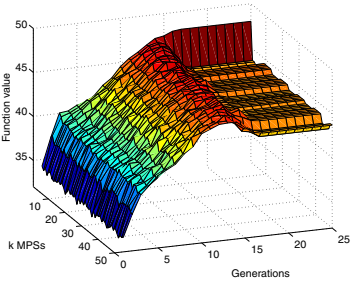

(a) Log. of the probability. Successful



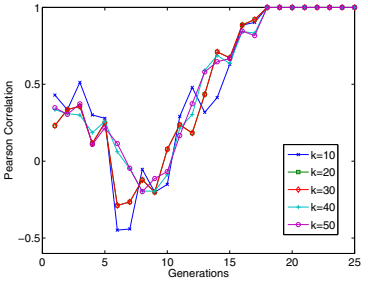

(b) Log. of the probability. Unsuccessful



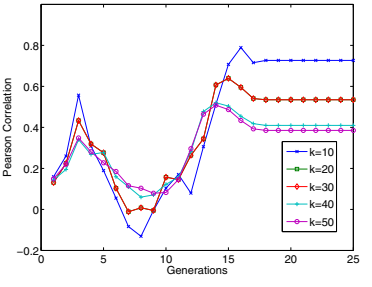

(c) Function values. Successful




(d) Function values. Unsuccessful



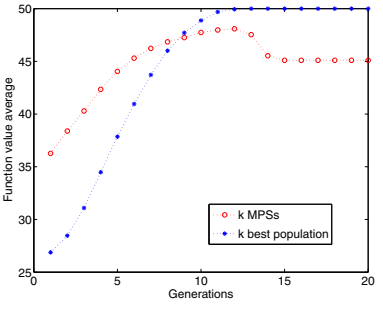

(e) Correlation. Successful



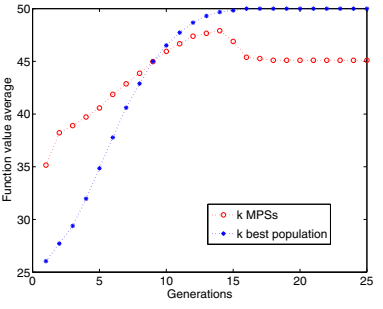

(f) Correlation. Unsuccessful

**Fig. 2** Probability values, function values and Pearson correlation of the 50 most probable solutions at each generation of EBNA when it is applied to Trap5 with population size $m$. In (a), (c), (e) we provide the results for 3 successful runs out of 20. In (b), (d), (f) we provide the results for 17 unsuccessful runs

In Fig. 3 we report the average function values for the 50 MPSs and the 50 best individuals in the population. The results agree with [12] where the function value for the MPS and the best individual of the population are analyzed. The function values for the $k$ MPSs are better than the $k$ best values in the population at the beginning of the run. This difference is higher with population size $m$. At the end of the run, the MPSs have lower values because the EBNA population converges to a

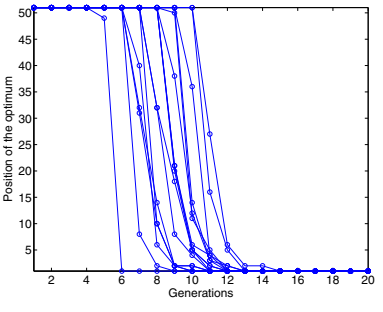



(a) Population size $m$

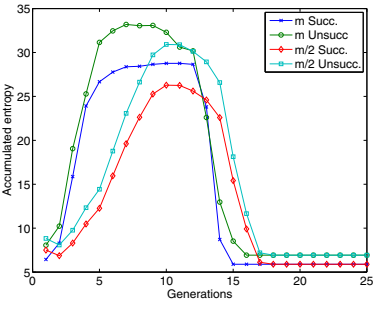



(b) Population size $m/2$

**Fig. 3** Function values average for the 50 most probable solutions and the 50 best individuals of the population at each generation of EBNA when it is applied to Trap5 and the optimum is reached




(a) Position of the optimum




(b) $k$ MPSs entropies

**Fig. 4** EBNA is applied to Trap5. (a) Position of the optimum in the $k$ most probable solutions at each generation with population size $m$. (b) Accumulated entropy of the whole set of variables in the $k$ most probable solutions

solution, while in the MPSs there are different solutions. For this type of analysis, the results are similar in both successful and unsuccessful runs as was shown in [12, 13].

The charts in Fig. 4 correspond to two different analysis. Firstly, Fig. 4(a) reports the position of the optimum in the ordered set of $k$ MPSs for each run (different curves in the chart) during the search. We only show the results for successful runs because when the optimum is not reached it does not enter the $k$ MPSs. We can see that the optimum quickly goes up in the ranking before reaching the first position. Secondly, Fig. 4(b) reports the entropy accumulated for the whole set of variables in the $k$ MPSs at each generation. We can observe a constant pattern in the four cases reported. The entropy increases at the beginning of the search and decreases in the last generations when EBNA converges to a solution. In the first and last generations, when the Bayesian networks have the lowest complexity, the $k$ MPSs

have very similar genotypes. Nevertheless, in the middle of the run the entropy increases with the complexity of the Bayesian network. It is also important to note that it is possible to distinguish between successful and unsuccessful runs in these results. The $k$ MPSs are more entropic in the middle of the run when the optimum is not reached. This occurs for both population sizes $m$ and $m/2$.

## 5 Analyzing the $k$ MPSs in Gaussian Ising

### 5.1 2D Ising Spin Glass Description

Ising spin glass model [22] is often used as optimization problem in benchmarking EDAs [19, 37, 39]. A classic 2D Ising spin glass can be formulated in a simple way. The set of variables $\mathbf{X}$ is seen as a set of $n$ spins disposed on a regular 2D grid $L$ with $n = l \times l$ sites and periodic boundaries (see Fig. 5). Each node of $L$ corresponds to a spin $X_i$ and each edge $(i, j)$ corresponds to a coupling between $X_i$ and $X_j$. Thus, each spin variable interacts with its four nearest neighbors in the toroidal structure $L$. Moreover, each edge of $L$ has an associated coupling strength $J_{ij}$ between the related spins. For the classical Ising model each spin takes the value 1 or $-1$. The target is, given couplings $J_{ij}$, to find the spin configuration that minimizes the energy of the system computed as,

$$E(\mathbf{x}) = -\sum_{(i,j)\in L} x_i J_{ij} x_j - \sum_{i\in L} h_i x_i \tag{9}$$

where the sum runs over all coupled spins. In our experiments we take $h_i = 0\ \forall i \in L$. The states with minimum energy are called *ground states*.

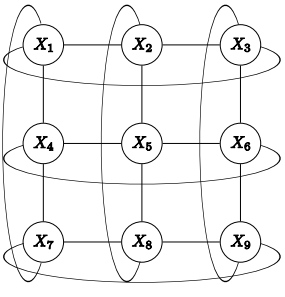



**Fig. 5** A $3 \times 3$ grid structure $L$ showing the interactions between spins for a 2D Ising spin glass with periodic boundaries. Each edge has an associated strength $J_{ij}$

Depending on the range chosen for the couplings $J_{ij}$ we have different versions of the problem. Thus, the problem is called *Gaussian Ising* when the couplings $J_{ij}$ are real numbers generated following a Gaussian distribution. For this type of couplings, the problem has only one optimal solution. A specified set $J_{ij}$ of coupling defines a

spin glass instance. We generated the Gaussian Ising instances using the Spin Glass Ground State server[1] for the experiments. The minimum energy of the system is also provided in this server.

### 5.2 Experimental Results

In this section, we present and discuss the results obtained when EBNA tries to solve the Gaussian Ising problem. In Fig. 6, we report the probabilities, function values and correlations for the $k$ MPSs with the population size given by bisection. In Gaussian Ising, the results for population size $m/2$ are not shown because they do not provide new relevant information. In contrast with Trap5, the first type of analysis for this problem does not show a clear difference between either population sizes or between successful and unsuccessful runs. Nonetheless, for this last matter we can observe little differences in the correlations (Fig. 6(c) and 6(f)) where the curves for all $k$ reach very close values in unsuccessful runs. Moreover, in the charts of function values (Fig. 6(b) and 6(e)), when EBNA does not reach the optimum, the MPSs with the lowest probability (they are always ordered from 1 to 50) have worse function values in the last generations.

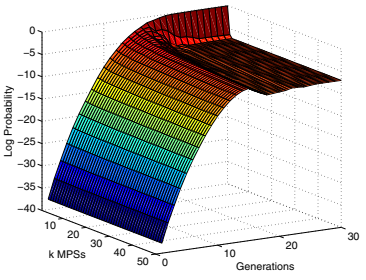
In any case, the different behavior of the $k$ MPSs between these two problems is evident. Firstly, in Gaussian Ising, although the probability distribution has an analogous behavior, it concentrates on a unique solution more slowly than in Trap5 in the last generations. Secondly, there is a clear difference in the behavior of the function values for the $k$ MPSs. In Trap5, the MPS always has a better function value than the rest of $k-1$ solutions at the end of the run. In Gaussian Ising, however, there is a clear diversity in the function values for the $k$ MPSs and their form is barely related with the probability values. Lastly, these facts are reflected in the charts of correlation. In Gaussian Ising, the curve of the correlation has a lower slope and reaches lower values than in Trap5 at the end of the run.

Although both problems have a unique optimum, we believe that the difference in the behavior of the $k$ MPSs is due to the properties of the landscape for each problem. In Gaussian Ising few solutions share the same function value, while in Trap5 there are different sets of solutions with the same function value.

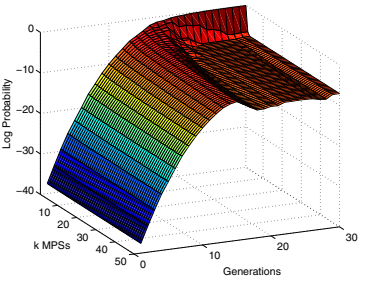
In Fig. 7 we report the average function value for the 50 MPSs and the 50 best individuals in the population. Once again, the results agree with [12]. In this problem, there is a shorter distance than in Trap5 between both curves during the search. Moreover, there is a lower difference between both population sizes. Nevertheless, the average function value for the $k$ MPSs is better than for the $k$ best individuals in the population during the first generations.

As in the previous section, Fig. 8(a) reports the position of the optimum in the $k$ MPSs during the search. In general, this analysis shows a behavior similar to Trap5. However, in this case, the optimum can decrease in position from one generation to the next. This can be seen because some curves have upward lines. Regarding the entropies of the $k$ MPSs, we present Fig. 8(b). In this problem the behavior of the

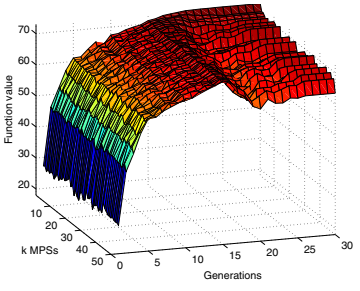
[1] `http://www.informatik.uni-koeln.de/ls_juenger/index.html`



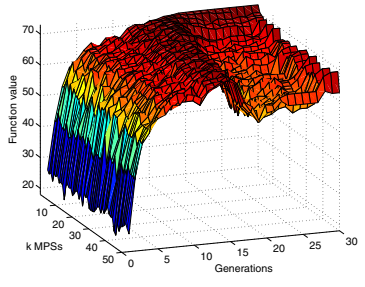

(a) Log. of the probability. Successful



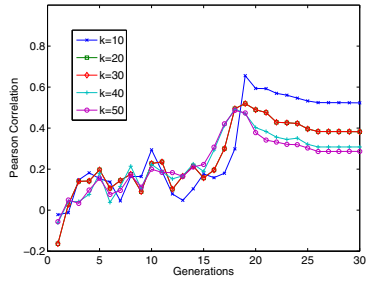

(b) Log. of the probability. Unsuccessful



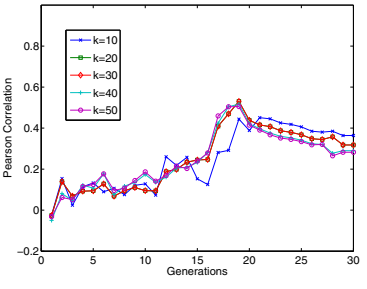

(c) Function values. Successful




(d) Function values. Unsuccessful



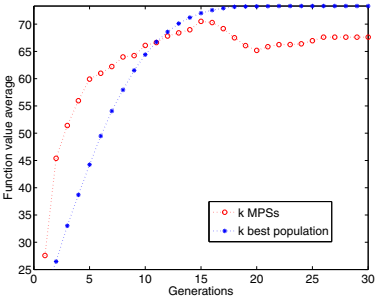

(e) Correlation. Successful



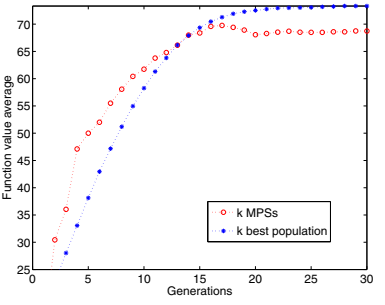

(f) Correlation. Unsuccessful

**Fig. 6** Probability values, function values and Pearson correlation of the 50 most probable solutions at each generation of EBNA when it is applied to Gaussian Ising with population size $m$. In (a), (c), (e) we provide the results for 15 successful runs out of 20. In (b), (d), (f) we provide the results for 5 unsuccessful runs

different curves is more variable and the pattern in general is less clear than in Trap5. Nevertheless, this chart allows us to distinguish between different scenarios. First of all, with population size $m$ we can distinguish between successful and unsuccessful runs. This is because the entropy of the $k$ MPSs in unsuccessful runs reaches higher values. This behavior was also observed for Trap5. Secondly, we can observe a different behavior with population size $m$ and $m/2$. The curves for $m$ clearly increase

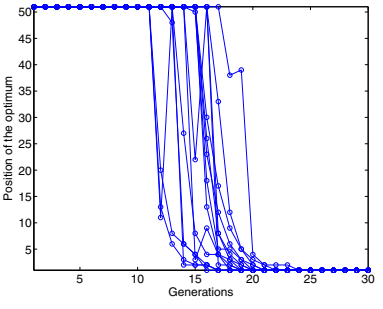



(a) Population size $m$

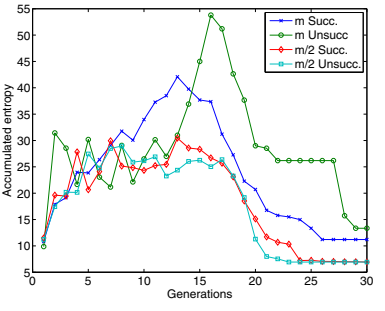



(b) Population size $m/2$

**Fig. 7** Function values average for the 50 most probable solutions and the 50 best individuals of the population at each generation of EBNA when it is applied to Gaussian Ising and the optimum is reached




(a) Position of the optimum




(b) $k$ MPSs entropies

**Fig. 8** EBNA is applied to Gaussian Ising. (a) Position of the optimum in the $k$ most probable solutions at each generation with population size $m$. (b) Accumulated entropies of the whole set of variables in the $k$ most probable solutions

from the beginning of the run to the middle and decrease in the last generations, while for population size $m/2$ the curves have a lower growth.

## 6 Analyzing $k$ MPSs in $\pm J$ Ising

### *6.1* $\pm$ *J Ising Description*

As explained in Section 5, the main difference between both versions of $2D$ Ising spin glass is the range of values chosen for the couplings $J_{ij}$. In this second Ising problem, the couplings $J_{ij}$ are set randomly to either $+1$ or $-1$ with equal probability. This version, that will be called $\pm J$ *Ising*, could have different configurations of the spins that reach the ground state (lowest energy) and therefore many optimal

solutions may arise. As for the previous case, the $\pm J$ Ising instances were generated using the Spin Glass Ground State server. This server also provided the value of the minimum energy of the system.

## 6.2 Experimental Results

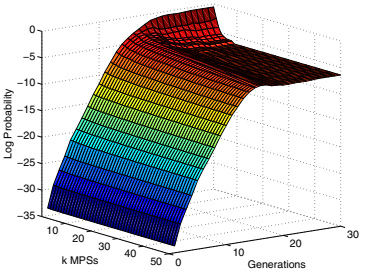
In this section, we present and discuss the results obtained when EBNA tries to solve the $\pm J$ Ising problem. The analyisis of the probabilities, function values and correlation for the $k$ MPSs are shown in Fig. 9. In this problem the results with different population sizes are very similar. We only report those with population size $m/2$ in order to have a wider variety of successful and unsuccessful runs. As we previously explained, $\pm J$ Ising problem has several optimal solutions and this fact is reflected in these results.

First for all, we must explain a particular behavior which can arise in the problems with several optima. If we look at Fig. 9(f), we can see that there is only one curve ($k = 50$) in the last generations and it is not continuous. The reason for such behavior is that in certain generations of some runs, all the function values for the $k$ MPSs are equal. In this case, the standard deviation is 0 and therefore Equation 5 makes no sense. This situation will be more evident in the Max-SAT problem which is analyzed in the following section. Although we do not directly analyze this matter, it occurs more frequently in unsuccessful runs. This can also be intuited through the charts of function values. Thus, if we compare Figs. 9(b) and 9(e), we can see that in unsuccessful runs the function values have very little diversity in the last generations. However, for successful runs we can appreciate different function values from the first $k$ MPSs to the last. This is reflected in the correlation charts where all curves are complete for successful runs. In this case, we have a low correlation during the search and it just increases in the last generations when the algorithm converges.

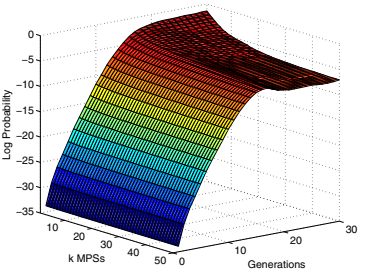
In this problem with several optima, we can observe, for the first time, differences between successful and unsuccesful runs by looking at the probability values of the $k$ MPSs. Depending on the type of run, the probability is distributed in a different manner in the last generations. Thus, in successful runs the probability is more concentrated in the first MPSs.

For this problem, Fig. 10 shows a similar behavior than for Gaussian Ising. Once again the function values for the $k$ MPSs outperform the $k$ best individuals in the population at the beginning of the run. We can also see a difference between population sizes. The curves have a greater distance in the first generations with a population size $m$.

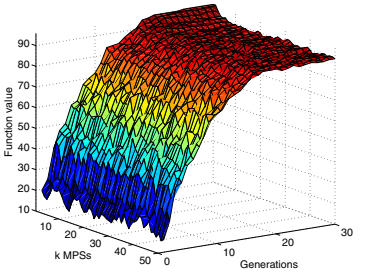
Finally, in Fig. 11 we report the position of the optimum in the $k$ MPSs and their entropy during the search. In Fig. 11(a), where EBNA solves $\pm J$ Ising with population size $m$, the behavior is similar to that in Gaussian Ising. The curves have similar slopes and the optimum can decrease in the ranking from one generation to the next. We have selected this population size in order to have more successful runs (more curves) and therefore more variety. In Fig. 11(b), the entropy curves are also similar to those for Gaussian Ising. We can only distinguish between successful and



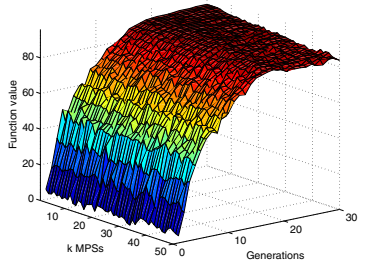

(a) Log. of the probability. Successful



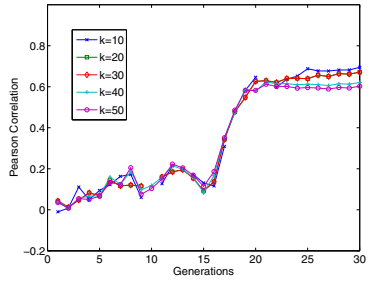

(b) Log. of the probability. Unsuccessful



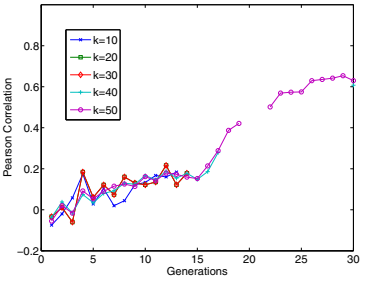

(c) Function values. Successful




(d) Function values. Unsuccessful



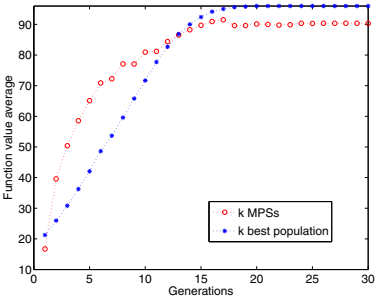

(e) Correlation. Successful



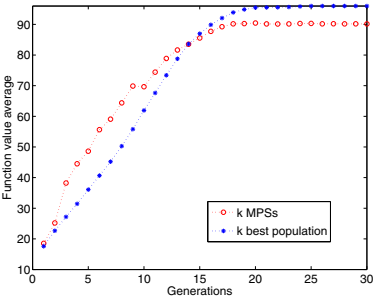

(f) Correlation. Unsuccessful

**Fig. 9** Probability values, function values and Pearson correlation of the 50 most probable solutions at each generation of EBNA when it is applied to $\pm J$ Ising with population size $m/2$. In (a), (c), (e) we provide the results for 8 successful runs out of 20. In (b), (d), (f) we provide the results for 12 unsuccessful runs

unsuccessful runs when the population size given by bisection is used. In this case, the $k$ MPSs are clearly more entropic in unsuccessful runs at the end of the run. Regarding the population size, we can see that with size $m$, there is more genotypic diversity during the search.

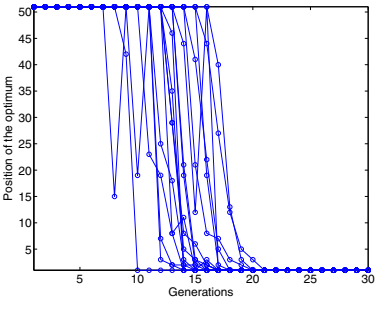



(a) Population size $m$

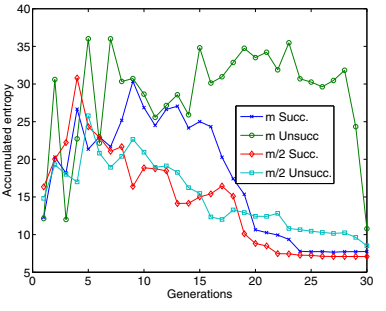



(b) Population size $m/2$

**Fig. 10** Function values average for the 50 most probable solutions and the 50 best individuals of the population at each generation of EBNA when it is applied to $\pm J$ Ising and the optimum is reached




(a) Position of the optimum




(b) $k$ MPSs entropies

**Fig. 11** EBNA is applied to $\pm J$ Ising. (a) Position of the optimum in the $k$ most probable solutions at each generation with population size $m$. (b) Accumulated entropies of the whole set of variables in the $k$ most probable solutions

# 7 Analyzing $k$ MPSs in Max-SAT

## *7.1 Max-SAT Description*

The last problem in our analysis is the maximum satisfiability or Max-SAT problem, which has often been used in different works about EDAs [4, 37]. Without going into details, given a set of Boolean variables $\mathbf{X}$ and a Boolean expression $\phi$, SAT problem asks if there is an assignment $\mathbf{x}$ of the variables such that the expression $\phi$ is satisfied. In a Boolean expression we can combine the variables using Boolean connectives such as $\wedge$ (logical and), $\vee$ (logical or) and $\neg$ (negation). An expression of the form $x_i$ or $\neg x_i$ is called a literal.

Every Boolean expression can be rewritten into an equivalent expression in a convenient specialized style. In particular, we use the *conjunctive normal form* (CNF) $\phi = \bigwedge_{i=1}^{q} C_i$. Each of the $q$ $C_i$s is the disjunction of two or more literals which are called clauses of the expression $\phi$. We work with clauses of length $k = 3$. When $k \geq 3$, the SAT problem becomes NP-Complete [8]. An example of a CNF expression with 5 Boolean variables $X_1$, $X_2$, $X_3$, $X_4$, $X_5$ and 3 clauses could be, $\phi = (x_1 \vee \neg x_3 \vee x_5) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_4 \vee \neg x_2)$.

The Max-SAT problem has the same structure as SAT, but the result, for an assignment $\mathbf{x}$, is the number of satisfied clauses instead of a truth value. In order to solve Max-SAT, the assignment for $\mathbf{X}$ that maximizes the number of satisfied clauses must be found. Thus, the function can be written as,

$$f_{Max-SAT}(\mathbf{x}) = \sum_{i=1}^{q} \phi(C_i) \tag{10}$$

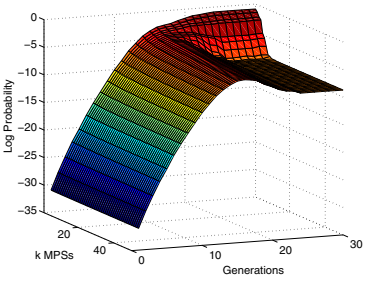
where each clause $C_i$ of three literals is evaluated as a Boolean expression that returns 1 if the expression is $true$ and 0 if it is $false$. Since $C_i$ is a disjunction, it is satisfied if at least one of its literals is $true$. The variables of $\mathbf{X}$ can overlap arbitrarily in the clauses.

Particularly, we work with 3-CNF SAT problems obtained from the SATLIB [21] repository which provides a large number of SAT instances. The instances used are satisfiable. They have 50 variables and 218 clauses. It is important to note that there could be several assignments for $\mathbf{X}$ that satisfy all clauses and therefore this problem could have different optimal solutions.

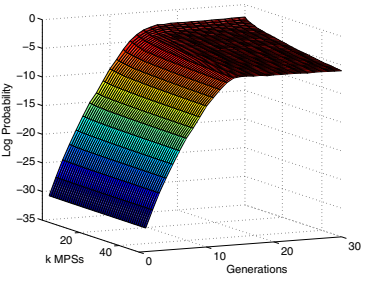
## 7.2 Experimental Results

In this section, we present and discuss the results obtained when EBNA tries to solve the Max-SAT problem. As with $\pm J$ Ising, this problem has several optimal solutions and it is reflected in the analysis. In Fig. 12 we present the probabilities in logarithmic scale, the function values and their correlation when population size $m/2$ is used. Max-SAT has a similar behavior for both population sizes in this type of analysis. So only results for the most representative size ($m/2$) are shown.

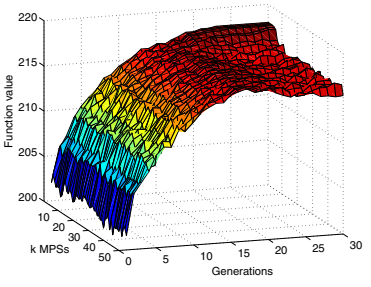
We can find clear differences between successful and unsuccessful runs in all the charts of Fig. 12. Firstly, regarding the probability values, at the end of the run it is evident that the probability is concentrated on the first MPSs when EBNA reaches the optimum. By contrast, the probability is distributed more equitably among the $k$ MPSs in unsuccessful runs. Secondly, if we look at the charts of function values, we can see that almost all $k$ MPSs have the same value in the last generations for unsuccessful runs. On the contrary, when the optimum is reached, there is diversity in the function values for this set of solutions. Lastly, in the charts of correlation we have the same problem as in $\pm J$ Ising because in some generations the calculation of the correlation is not possible. This problem is more evident in unsuccessful runs where we can see that the function values reach equal values in most of the cases. Once again, there is a low correlation between probabilities and functions during the



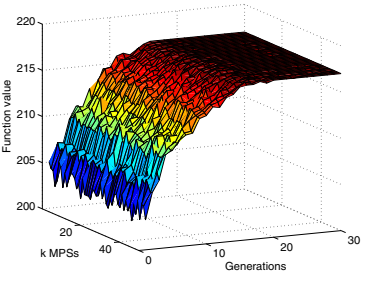

(a) Log. of the probability. Successful



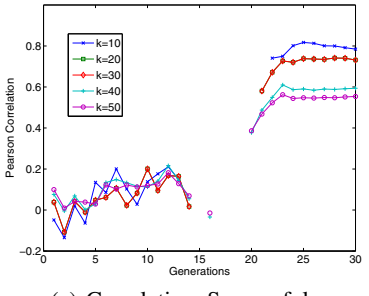

(b) Log. of the probability. Unsuccessful



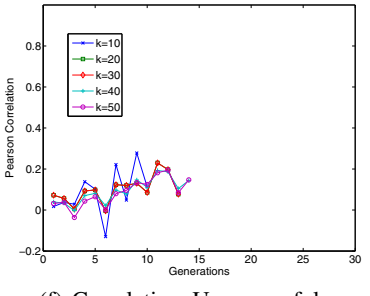

(c) Function values. Successful




(d) Function values. Unsuccessful




(e) Correlation. Successful



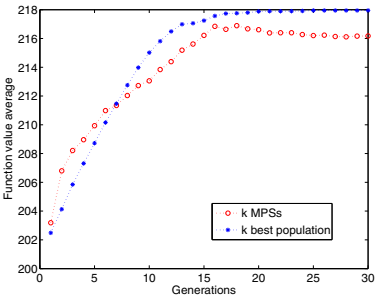

(f) Correlation. Unsuccessful

**Fig. 12** Probability values, function values and Pearson correlation of the 50 most probable solutions at each generation of EBNA when it is applied to Max-SAT with population size $m/2$. In (a), (c), (e) we provide the results for 12 successful runs out of 20. In (b), (d), (f) we provide the results for 8 unsuccessful runs

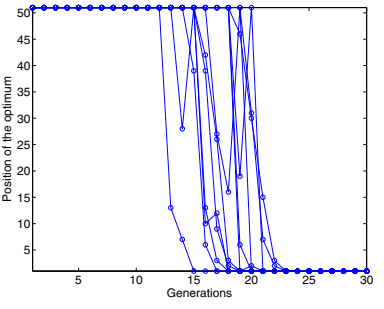
search. Only at the end of the run, when the optimum is reached, does the correlation increase (see Fig. 12(c)).

In Fig. 13 we show the rest of the analysis. A little difference between population sizes in the analysis of the average function value can be seen. Therefore, we only show Fig. 13(a) as an example of the behavior. We can see that the $k$ MPSs are only better than the $k$ best individuals of the population in few generations at the beginning of the run. In the rest of the run there are better individuals in the population. It

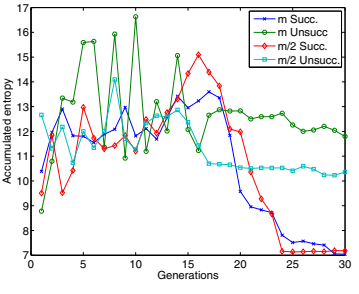
is possible that Bayesian networks do not manage to accurately represent the structure of the problem. Fig. 13 shows the position of the optimum in the $k$ MPSs during the search and the behavior is the same as in the rest of the problems. Finally, the entropies of the $k$ MPSs in Fig. 13(c) reveal a clear difference between successful and unsuccessful runs. Thus, when the optimum is not reached by EBNA our analyzed solutions are more entropic at the end of the run. However, in this chart it is difficult to distinguish between population sizes.




(a) Function values




(b) Position of the optimum




(c) $k$ MPSs entropies

**Fig. 13** EBNA is applied to Max-SAT. (a) Function values average for the 50 MPSs and the 50 best individuals of the population for successful runs with population size $m$. (b) Position of the optimum into the 50 MPSs at each generation with population size $m/2$. (c) Accumulated entropy of the whole set of variables in the 50 MPSs

## 8 Related Works

Most of the research done in the scope of the models learned by EDAs that use Bayesian networks [15, 30, 38] has focused on structural descriptors of the networks, in particular on the type (i.e. correct or spurious) and number of the network edges [11, 14, 18, 19, 24, 25]. The analysis of the Bayesian network edges learned by EDAs has allowed to study the effect of the selection and replacement [19, 25] as well as the learning method [11, 14, 24] in the accuracy of the models learned by EDAs and the efficiency of these algorithms. A more recent work [24] considers the

likelihood given to the selected set during the model learning step as another source of information about the algorithm's behavior.

We are in the line of works which analyze EDAs from a quantitative point of view. For EDAs that use Markov models, in [5] the product moment correlation coefficient between the Markov model learned by DEUM and the fitness function is used to measure the quality of the model as a fitness function predictor. For a given solution, the prediction is the value given by the Markov model to the solution. The quality of the model is measured using the correlation computed from samples of the search space.

For EDAs based on Bayesian networks, in [12, 13] the probability of the optimum is analyzed. By using the most probable solution given by the probabilistic model, the optimum of the function and the best individual of the population, basic issues about the behavior of EDAs are studied. In this chapter, we extend this type of quantitative analysis, taking into consideration the $k$ MPSs as descriptors of the probabilistic models. The computation of the MPS and the $k$ MPSs has also been used as a way to improve the sampling step in EDAs [20, 28]. In this context, different questions arise such as which is a good value for $k$, at which time of the evolution it is more convenient to introduce the k best MPSs, how to avoid premature convergence, etc. The kind of analysis we present may be useful to answer these questions and advance in this line of research.

## 9 Conclusions

In this work, we have exploited the quantitative component of the probabilistic models learned by EDAs during the search in order to better understand their behavior and aid in their development. Particularly, we have analyzed the $k$ solutions with the highest probability in the distributions estimated at each step of an EBNA. We have conducted systematic experiments for several functions.

Firstly, we have studied the relation between the probabilistic model and the function. For the $k$ MPSs, we have recorded the probabilities, the function values and the correlation between both at each step of the algorithm. We have observed that the results change markedly depending on the problem. This allows us to discover some properties of the search space. Firstly, it is possible to distinguish multimodal functions such as $\pm J$ Ising and Max-SAT. The main source to identify this characteristic in the function are the probability values. Thus, when the function has a unique optimum, the probability is concentrated in the MPS at the end of the run. However, when the function is multimodal, the probability is shared among several $k$ MPSs in the last generations. This supports the conclusions drawn in [13]. Secondly, the function values for the $k$ MPSs are a rich source of information in the problems with a unique optimum (Trap5 and Gaussian Ising). Although the probability values have a similar behavior in both problems, the function values provide information about the properties of the search space. Thus, we have seen that Trap5 has many local optima with the same function value. However, in Gaussian Ising, the function values for the $k$ MPSs suffer important variations even at the end of the run. Therefore,

we can deduce that this function assigns more different function values than Trap5 to the search space. The correlation between probabilities and function values has a different curve depending on the problem. Only in Trap5 does it reach the value of 1 in the last generations.

It is very important to note that this analysis reveals a different behavior between successful and unsuccessful runs in most of the cases. For this goal, the function values and the correlations are the main source of information in the problems with a unique optimum. When the optimum is not reached, the function values suffer variations and the correlation is lower at the end of the run. Although in Gaussian Ising there is only a little difference between both types of runs, in Trap5 the successful runs are easily identifiable. In problems with several optima, both the probability and function values, and hence the correlations, allow us to identify the success of the search. Firstly, the probability values are more spread among the $k$ MPSs when the optimum is not reached. This fact is more evident in Max-SAT. Secondly, the function values for these solutions are more variable in successful runs. By contrast, when the optimum is not reached, it is more frequent that all $k$ MPSs have equal function values in a given generation. Therefore, this affects the calculation of the correlation. Thus, in unsuccessful runs, there are more generations for which the correlation has no solution. The distinction between types of runs is a very important matter. It could open a line of work in order to create methods to predict or estimate the success of a particular run in real problems where the optimum is not known.

Regarding the analysis of the average function value for the $k$ MPSs and the $k$ best individuals of the population, we have observed that they agree with the results obtained in [12]. Thus, the function values for the $k$ MPSs always have a higher quality than the best individuals of the population at the beginning of the run. In general, this difference is greater when a population size given by bisection is used in EBNA. However, in the middle and at the end of the run the behavior depends on the problem. In general, the behavior observed in this type of analysis confirms the benefit of exploiting the information that the Bayesian network contains about the function by using inference techniques [28], at least at the beginning of the run.

The position of the optimum inside the $k$ MPSs during the search has a similar behavior in different problems. Thus, the optimum usually reaches the first position in few generations when it is reached. Except for Trap5, it is possible for the optimum to go down in the ranking from one generation to the next. Moreover, although these results have not been presented, it is also possible for the optimum to enter the $k$ MPSs and to leave in the next generation. Once again, this supports the idea of using approximate inference techniques of the MPSs in the sampling step of an EDA.

Finally, the analysis of the entropies for the $k$ MPSs allows us to make important distinctions not only between problems but also between successful and unsuccessful runs and population sizes. A common pattern in all problems is that the entropy of the $k$ MPSs is lower in successful runs, especially in the middle of the run. It indicates that the probability distribution assigns the highest probabilities to more

homogeneous solutions and it is beneficial for the search. Regarding the difference between population sizes, the entropy tends to be lower with a reduced size.

We can assert that the $k$ MPSs are a rich source of information about the behavior of an EDA. However, their computation requires a high computational cost. We believe that carrying out an approximate calculation of the $k$ MPSs could be useful to obtain relevant information about the optimization process. This type of measurements could be taken on-line during the search allowing certain automatic decision making in the algorithm which could be really useful to develop adaptive EDAs.

It is important to highlight that our method may be extended to EDAs that use other classes of probabilistic models by modifying the class of message passing algorithm used in the computation of the MPSs (e.g. using loopy belief propagation). Furthermore, it is also possible to use information about the fitness function (structure and fitness values) [20, 26, 41] to conveniently modify the definition of the abductive inference step and the computation of the MPSs.

## Acknowledgments

This work has been partially supported by the Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN2008-06815-C02-01 and Consolider Ingenio 2010 - CSD2007-00018 projects (Spanish Ministry of Science and Innovation) and COMBIOMED network in computational biomedicine (Carlos III Health Institute). Carlos Echegoyen has a grant from UPV-EHU.

## References

1. Anumanchipalli, G.K., Ravishankar, M., Reddy, R.: Improving pronunciation inference using n-best list, acoustics and orthography. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP 2007, Honolulu, HI, vol. IV, pp. 925–928 (2007)
2. Armañanzas, R., Inza, I., Santana, R., Saeys, Y., Flores, J.L., Lozano, J.A., Van de Peer, Y., Blanco, R., Robles, V., Bielza, C., Larrañaga, P.: A review of estimation of distribution algorithms in bioinformatics. BioData Mining 1(6) (2008)
3. Barahona, F.: On the computational complexity of Ising spin glass model. Journal of Physics A: Mathematical and General 15(10) (1982)
4. Brownlee, S., McCall, J., Brown, D.: Solving the MAXSAT problem using a multivariate EDA based on Markov networks. In: Proceedings of the 2007 conference on Genetic and Evolutionary Computation (GECCO-2007), London, England, pp. 2423–2428. ACM, New York (2007)
5. Brownlee, S., McCall, J., Zhang, Q., Brown, D.: Approaches to selection and their effect on fitness modelling in an estimation of distribution algorithm. In: Proceedings of the 2008 Congress on Evolutionary Computation (CEC 2008), Hong Kong, pp. 2621–2628. IEEE Press, Los Alamitos (2008)
6. Buntine, W.: Theory refinement on Bayesian networks. In: Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence, pp. 52–60 (1991)
7. Castillo, E., Gutierrez, J.M., Hadi, A.S.: Expert Systems and Probabilistic Network Models. Springer, Heidelberg (1997)

8. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the Third Annual ACM Symposium on Theory of Computing, pp. 151–158. Shaker Heights, Ohio (1971)
9. Cowell, R.: Sampling without replacement in junction trees. Technical Report Statistical Research Paper 15, City University, London (1997)
10. Deb, K., Goldberg, D.E.: Sufficient conditions for deceptive and easy binary functions. Annals of Mathematics and Artificial Intelligence 10, 385–408 (1994)
11. Echegoyen, C., Lozano, J.A., Santana, R., Larrañaga, P.: Exact Bayesian network learning in estimation of distribution algorithms. In: Proceedings of the 2007 Congress on Evolutionary Computation (CEC 2007), pp. 1051–1058. IEEE Press, Los Alamitos (2007)
12. Echegoyen, C., Mendiburu, A., Santana, R., Lozano, J.: Analyzing the probability of the optimum in EDAs based on Bayesian networks. In: Proceedings of the 2009 Congress on Evolutionary Computation (CEC 2009), Trondheim, Norway, pp. 1652–1659. IEEE Press, Los Alamitos (2009)
13. Echegoyen, C., Mendiburu, A., Santana, R., Lozano, J.: A quantitative analysis of estimation of distribution algorithms based on Bayesian networks. Technical Report EHU-KZAA-TR-2-2009, Department of Computer Science and Artificial Intelligence (2009)
14. Echegoyen, C., Santana, R., Lozano, J., Larrañaga, P.: The Impact of Exact Probabilistic Learning Algorithms in EDAs Based on Bayesian Networks. In: Linkage in Evolutionary Computation, pp. 109–139. Springer, Heidelberg (2008)
15. Etxeberria, R., Larrañaga, P.: Global optimization using Bayesian networks. In: Ochoa, A., Soto, M.R., Santana, R. (eds.) Proceedings of the Second Symposium on Artificial Intelligence (CIMAF 1999), Havana, Cuba, pp. 151–173 (1999)
16. Friedman, N., Yakhini, Z.: On the sample complexity of learning Bayesian networks. In: Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI 1996), pp. 274–282. Morgan Kaufmann, San Francisco (1996)
17. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
18. Hauschild, M., Pelikan, M.: Enhancing efficiency of hierarchical BOA via distance-based model restrictions. MEDAL Report No. 2008007, Missouri Estimation of Distribution Algorithms Laboratory (MEDAL) (April 2008)
19. Hauschild, M., Pelikan, M., Sastry, K., Lima, C.: Analyzing Probabilistic Models in Hierarchical BOA. IEEE Transactions on Evolutionary Computation (to appear)
20. Höns, R., Santana, R., Larrañaga, P., Lozano, J.A.: Optimization by max-propagation using Kikuchi approximations. Technical Report EHU-KZAA-IK-2/07, Department of Computer Science and Artificial Intelligence, University of the Basque Country (November 2007)
21. Hoos, H., Stutzle, T.: SATLIB: An online resource for research on SAT. In: van Maaren, H., Gent, I.P., Walsh, T. (eds.) SAT 2000, pp. 283–292. IOS Press, Amsterdam (2000)
22. Ising, E.: The theory of ferromagnetism. Zeitschrift fuer Physik 31, 253–258 (1925)
23. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2002)
24. Lima, C.F., Lobo, F.G., Pelikan, M.: From mating pool distributions to model overfitting. In: Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO 2008), pp. 431–438. IEEE Press, Los Alamitos (2008)

25. Lima, C.F., Pelikan, M., Goldberg, D.E., Lobo, F.G., Sastry, K., Hauschild, M.: Influence of selection and replacement strategies on linkage learning in BOA. In: Proceedings of the 2007 Congress on Evolutionary Computation (CEC 2007), pp. 1083–1090. IEEE Press, Los Alamitos (2007)
26. Lima, C.F., Pelikan, M., Lobo, F.G., Goldberg, D.E.: Loopy Substructural Local Search for the Bayesian Optimization Algorithm. In: Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics, pp. 61–75. Springer, Heidelberg (2009)
27. Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.): Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms. Springer, Heidelberg (2006)
28. Mendiburu, A., Santana, R., Lozano, J.A.: Introducing belief propagation in estimation of distribution algorithms: A parallel framework. Technical Report EHU-KAT-IK-11/07, Department of Computer Science and Artificial Intelligence, University of the Basque Country (October 2007)
29. Mühlenbein, H., Mahnig, T.: FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. Evolutionary Computation 7(4), 353–376 (1999)
30. Mühlenbein, H., Mahnig, T.: Evolutionary synthesis of Bayesian networks for optimization. In: Patel, M., Honavar, V., Balakrishnan, K. (eds.) Advances in Evolutionary Synthesis of Intelligent Agents, pp. 429–455. MIT Press, Cambridge (2001)
31. Mühlenbein, H., Mahnig, T., Ochoa, A.: Schemata, distributions and graphical models in evolutionary optimization. Journal of Heuristics 5(2), 213–247 (1999)
32. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
33. Murphy, K.: The Bayes Net Toolbox for Matlab. In: Computer science and Statistics: Proceedings of Interface, vol. 33 (2001)
34. Nilsson, D.: An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. Statistics and Computing 2, 159–173 (1998)
35. Nilsson, D., Goldberger, J.: Sequentially finding the n-best list in Hidden Markov Models. In: Proceedings of he Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001 (2001)
36. Pelikan, M.: Hierarchical Bayesian Optimization Algorithm. Toward a New Generation of Evolutionary Algorithms. Studies in Fuzziness and Soft Computing. Springer, Heidelberg (2005)
37. Pelikan, M., Goldberg, D.E.: Hierarchical BOA solves Ising Spin Glasses and MAXSAT. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1271–1282. Springer, Heidelberg (2003)
38. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO1999), Orlando, FL, vol. I, pp. 525–532. Morgan Kaufmann Publishers, San Francisco (1999)

39. Pelikan, M., Hartmann, A.K.: Searching for ground states of Ising spin glasses with hierarchical BOA and cluster exact approximation. In: Pelikan, M., Sastry, K., Cantú-Paz, E. (eds.) Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications. Studies in Computational Intelligence, pp. 333–349. Springer, Heidelberg (2006)
40. Pelikan, M., Sastry, K., Cantú-Paz, E. (eds.): Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications. Studies in Computational Intelligence. Springer, Heidelberg (2006)
41. Santana, R.: Advances in Probabilistic Graphical Models for Optimization and Learning. Applications in Protein Modelling. PhD thesis, University of the Basque Country (2006)
42. Santana, R., Echegoyen, C., Mendiburu, A., Bielza, C., Lozano, J.A., Larrañaga, P., Armañanzas, R., Shakya, S.: MATEDA: A suite of EDA programs in matlab. Technical Report EHU-KZAA-IK-2/09, Department of Computer Science and Artificial Intelligence (2009)
43. Santana, R., Larrañaga, P., Lozano, J.A.: Research topics on discrete estimation of distribution algorithms. Memetic Computing 1(1), 35–54 (2009)
44. Schwarz, G.: Estimating the dimension of a model. Annals of Statistics 7(2), 461–464 (1978)
45. Yanover, C., Weiss, Y.: Approximate inference and protein-folding. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems, vol. 15, pp. 1457–1464. MIT Press, Cambridge (2003)

# Part III
# Applications

# Protein Structure Prediction Based on HP Model Using an Improved Hybrid EDA

Benhui Chen and Jinglu Hu

**Abstract.** Protein structure prediction (PSP) is one of the most important problems in computational biology. This chapter introduces a novel hybrid Estimation of Distribution Algorithm (EDA) to solve the PSP problem on HP model. Firstly, a composite fitness function containing the information of folding structure core (H-Core) is introduced to replace the traditional fitness function of HP model. The new fitness function is expected to select better individuals for probabilistic model of EDA. Secondly, local search with guided operators is utilized to refine found solutions for improving efficiency of EDA. Thirdly, an improved backtracking-based repairing method is introduced to repair invalid individuals sampled by the probabilistic model of EDA. It can significantly reduce the number of backtracking searching operation and the computational cost for long sequence protein. Experimental results demonstrate that the new method outperforms the basic EDAs method. At the same time, it is very competitive with other existing algorithms for the PSP problem on lattice HP models.

## 1 Introduction

Protein structure prediction (PSP) is one of the most important problems in computational biology. A protein is a chain of amino acids (also called as residues) that folds into a specific native tertiary structure under certain physiological conditions. Understanding protein structures is vital to determining the function of a protein

Benhui Chen
Graduate School of Information, Production and Systems, Waseda University,
Hibikino 2-7, Wakamatsu-ku, Kitakyushu-shi, Fukuoka 808-0135, Japan
e-mail: bhchen@fuji.waseda.jp

Jinglu Hu
Graduate School of Information, Production and Systems, Waseda University,
Hibikino 2-7, Wakamatsu-ku, Kitakyushu-shi, Fukuoka 808-0135, Japan
e-mail: jinglu@waseda.jp

Y.-p. Chen (Ed.): Exploitation of Linkage Learning, ALO 3, pp. 193–214.
springerlink.com 

and its interaction with DNA, RNA and enzyme. The information about its conformation can provide essential information for drug design and protein engineering. While there are over a million known protein sequences, only a limited number of protein structures are experimentally determined. Hence, prediction of protein structures from protein sequences using computer programs is an important step to unveil proteins' three dimensional conformation and functions.

Because of the complexity of the PSP problem, simplified models like Dill's HP-lattice [17] model have become the major tools for investigating general properties of protein folding. In HP model, 20-letter alphabet of residues is simplified to a two-letter alphabet, namely $H$ (hydrophobic) and $P$ (polar). Experiments on small protein suggest that the native state of a protein corresponds to a free energy minimum. This hypothesis is widely accepted, and forms the basis for computational prediction of a protein's conformation from its residue sequence. The problem of finding such a minimum energy configuration has been proved to be NP-complete for the bi-dimensional (2-D) [8] and tri-dimensional (3-D) lattices [4]. Therefore, a deterministic approaches is always not practical for this problem.

Many genetic algorithm (GA) based methods have been proposed to solve the PSP problem in the HP model in recent years [27, 11, 26, 7]. However, it has been acknowledged that the crossover operators, particularly one-point crossover and uniform crossover, in a conventional GA do not perform well for this problem [15, 9]. On the other hand, R. Santana et al. (2008) pointed out that the evolutionary algorithms able to learn and use the relevant interactions that may arise between the variables of the problem can perform well for this kind of problems. Estimation of distribution algorithm (EDA) is known as one of such kind of evolutionary algorithms.

In the EDAs [16, 18], instead of using conventional crossover and mutation operations, probabilistic models are used to sample the genetic information in the next population. The use of probabilistic models, especially, models taking into account bivariate or multivariate dependencies between variables, allows EDAs to capture genetic tendencies in the current population effectively. In brief, these algorithms construct, in each generation, a probabilistic model that estimates the probability distribution of the selected solutions. Dependency regulars are then used to generate next generation solutions during a simulation step. It is expected that the generated solutions share a number of characteristics with the selected ones. In this way, the search leads to promising areas of the search space.

In Ref. [24], the EDAs that use Markov probabilistic model or other probabilistic models outperform other population-based methods when solving the HP model folding problem, especially for the long sequence protein instances. But those methods have three obvious disadvantages as follow. 1) For most long sequence protein instances, the chance of finding the global optimum is very low, and the algorithm often need be set by very large generation number and population size for finding the global optimum. 2) For some deceptive sequences, those methods can only find the suboptimum solutions. 3) In those methods, a backtracking method is used to repair invalid individuals sampled by the probabilistic model of EDAs. For a traditional backtracking algorithm, the computational cost of repairing procedure is very heavy for those long sequence instances.

This chapter introduces a hybrid method to solve above problems of the EDAs based method for HP model protein folding. Firstly, a composite fitness function containing the information of folding structure core (H-Core) is introduced to replace the traditional fitness function of HP model. The new fitness function is expected to select better individuals for probabilistic model of EDAs algorithm. It can help to increase the chance of finding the global optimum and reduce the complexity of EDA (population size and the number of generation needed). Secondly, local search with guided operators is utilized to refine the found solutions for improving efficiency of EDA. The local information of solutions found so far can be helpful for exploitation, while the global information can guide the search for exploring promising areas. Local search with guided operators generates offspring through combination of global statistical information and the location information of solutions found so far. Thirdly, an improved backtracking-based repairing method is introduced to repair invalid individuals sampled by the probabilistic model of EDAs for the long sequence protein instances. The traditional backtracking repairing procedure will produce heavy computational cost for searching invalid closed-areas of folding structure. In the improved method, to avoid entering invalid closed-areas, a detection procedure for feasibility is introduced when selecting directions for the residues in backtracking searching procedure. It can significantly reduce the number of backtracking searching operation and the computational cost for the long protein sequences. The presented work extends the previous papers [6, 5] including further empirical investigation and extending explanations about critical aspects of the algorithm's behavior.

The rest of the chapter is organized as follows. In Section 2 we give a brief overview of protein HP model and the EDAs. In Section 3 we describe the new hybrid EDA for HP model protein folding. It includes the proposed composite fitness function and local search with guided operators. In Section 4 we formulate the improved backtracking repairing algorithm for invalid solutions. Section 5 presents the experiment results of the introduced method. Finally, the conclusions and further work directions are given.

## 2 Protein HP Model and EDAs

### *2.1 Protein Folding and HP Model*

Proteins are macromolecules made out of 20 different residues. A residue has a peptide backbone and a distinctive side chain group. The peptide bond is defined by an amino group and a carboxyl group connected to an alpha carbon to which a hydrogen and side chain group is attached. Residues are combined to form sequences which are considered as the primary structure of proteins. The secondary structure is the locally ordered structure brought about via hydrogen bounding mainly within the peptide backbone. The most common secondary structure elements in proteins are the alpha helix and the beta sheet. The tertiary structure is the global folding of a single polypeptide chain.

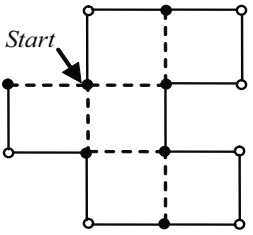



**Fig. 1** One possible configuration of the sequence $HPHPPHHPPHPHPH$ in 2-D HP model. There are six HH topological neighbors (represented by broken lines)

Under specific conditions, a protein sequence folds into a unique native 3-D structure. Each possible protein fold has an associated energy. The thermodynamic hypothesis states that the native structure of a protein is the one for which the free energy achieves the minimum. Based on this hypothesis, many methods are proposed to search for the protein native structure by defining an approximation of the protein energy and utilizing the optimization methods. These approaches mainly differ in the type of energy approximation employed and in the characteristics of the protein modeling.

In this chapter, we focus on lattice models, in particular, we use the well-known Dills HP model. The HP model takes into account the hydrophobic interaction as the main driving force in protein folding. In HP model, each amino acid is represented as a bead, and connecting bonds are represented as lines. In this approach, a protein is considered as a sequence $S \in \{H,P\}^+$ , where $H$ represents a hydrophobic residue and $P$ represents a hydrophilic or polar residue. The HP model restricts the space of conformations to self-avoiding paths on a lattice in which vertices are labeled by the residues.

Given a pair of residues, they are considered neighbors if they are adjacent either in the chain (connected neighbors) or in the lattice but not connected in the chain (topological neighbors). Let $\varepsilon_{HH}$ denote the interaction energy between topological neighbor of two $H$ residues, $\varepsilon_{PP}$ for two $P$ residues, $\varepsilon_{HP}$ for a $H$ residue and a $P$ residue. An energy function is defined as the total energy of topological neighbors with $\varepsilon_{HH} = -1$ and $\varepsilon_{PP} = \varepsilon_{HP} = 0$. The HP problem is to find the folding conformation that minimizes the total energy $E(x)$. Figure 1 shows the graphical representation of a possible configuration for sequence $HPHPPHHPPHPHPH$ in 2-D HP model, hydrophobic residuals are represented by black beads and polar residuals by white beads. The energy that the HP model associates with this configuration is -6.

Although more complex models have been proposed, the HP model remains a focus of research in computational biology, chemical and statistical physics. By varying the energy function and the bead sequence of the chain (the primary structure), effects on the native state structure and the kinetics (rate) of folding can be explored, and this may provide insights into the folding of real proteins. In particular, the HP model has been used to investigate the energy landscapes of proteins, i.e. the variation of their internal free energy as a function of conformation. In

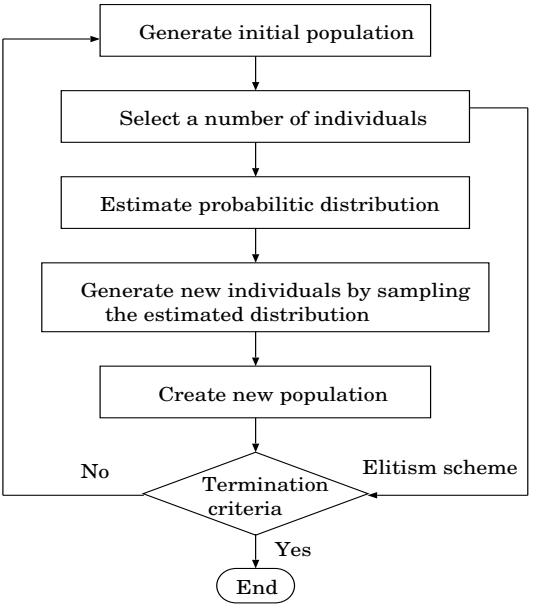



**Fig. 2** The flowchart of Estimation of Distribution Algorithms (EDAs)

evolutionary computation, the model is still employed because of its simplicity and its usefulness as a test-bed for new evolutionary optimization approaches [24].

## *2.2 Estimation of Distribution Algorithms*

In EDAs [16], there are neither crossover nor mutation operators. Instead, the new population is sampled from a probability distribution, which is estimated from a database that contains the selected individuals from the previous generation. Thus, the interrelations between the different variables that represent the individuals are explicitly expressed through the joint probability distribution associated with the individuals selected at each generation.

Figure 2 illustrates the flow chart for an EDA approach. Initially, a random solutions is generated. These solutions are evaluated using an objective function. An objective function evaluates how accurate each solution is for the problem. Based on this evaluation, a subset of solutions is selected. Hence, solutions with better function values have a bigger chance of being selected. Then, a probabilistic model of the selected solutions is built, and a new set of solutions is sampled from the model. The process is iterated until the optimum has been found or another termination criterion is fulfilled [1].

In order to explain the behavior of this heuristic, a common outline for EDAs is listed as follow.

1) Generate the first population of $M$ individuals and evaluate each of them. Usually, this generation is made assuming a uniform distribution on each variable.
2) $N$ individuals are selected from the set of $M$, following a given selection method.
3) An $l$ (size of the individual) dimensional probabilistic model that shows the interdependencies among the variables is induced from the $N$ selected individuals.
4) Finally, a new population of individuals is generated based on the sampling of the probability distribution learnt in the previous step.
5) Steps of 2 to 4 are repeated until some stop criterion is met (e.g. a maximum number of generations, a homogeneous population, or no improvement after a certain number of generations).

Essentially EDAs assume that it is possible to build a model of the promising areas of the search space, and use this model to guide the search for the optimum. In EDAs, modeling is achieved by building a probabilistic graphical model that represents a condensed representation of the features shared by the selected solutions. Such a model can capture different patterns of interactions between subsets of the problem variables, and can conveniently use this knowledge to sample new solutions.

Probabilistic modeling gives EDAs an advantage over other evolutionary algorithms that do not employ models, such as GAs. These algorithms are generally unable to deal with problems where there are important interactions among the problems' components. This, together with EDAs' capacity to solve different types of problems in a robust and scalable manner, has led to EDAs sometimes also being referred to as competent GAs [10, 22]. EDAs can be seen as a development of GAs. By recombining a subset of selected solutions, GAs are able to process the information learned during the search, and to orient the exploration to promising areas of the search space. Nevertheless, it has been proved that GAs experience limitations in their capacity to deal with problems where there are complex interactions between different components of the solutions. In these scenarios, EDAs can exhibit a better performance [9, 19].

EDAs can be broadly divided according to the complexity of the probabilistic models used to capture the interdependencies between the variables: univariate, bivariate and multivariate approaches [1]. Univariate EDAs, such as PBIL [2], cGA [12] and UMDA [21], assume that all variables are independent and factorize the joint probability of the selected solutions as a product of univariate marginal probabilities. Consequently, these algorithms are the simplest EDAs and have also been applied to problems with continuous representation.

The bivariate models can represent low order dependencies between the variables and be learnt using fast algorithms. MIMIC [14], the bivariate marginal distribution algorithm BMDA [23], dependency tree-based EDAs [3] and the tree-based estimation of distribution algorithm (Tree-EDA) [25] are all members of this subclass. The latter two use tree and forest-based factorizations, respectively. They are recommended for problems with a high cardinality of the variables and where interactions are known to play an important role. Trees and forests can also be combined to represent higher-order interactions using models based on mixtures of distributions.

Multivariate EDAs factorize the joint probability distribution using statistics of order greater than two. As the number of dependencies among the variables is higher

than in the above categories, the complexity of the probabilistic structure, as well as the computational effort required to find the structure that best suits the selected solutions, is greater. Therefore, these approaches require a more complex learning process. Some of the popular Multivariate EDAs are the Factorized Distribution Algorithm (FDA) [20], the Bayesian optimization algorithm (BOA) [23] and the extended compact Genetic Algorithm (EcGA) [12].

Since several EDAs have been proposed with a variety of models and learning algorithms, the selection of the best EDA to deal with a given optimization problem is not always straightforward. One criterion that could be followed in this choice is to trade off the complexity of the probabilistic model against the computational cost of storing and learning the selected model. Both issues are also related to the problem dimensionality (i.e. number of variables) and to the type of representation (e.g. discrete, continuous, mixed).

The simple models generally have minimal storage requirements, and are easy to learn. However, they have a limited capacity to represent higher order interactions. On the other hand, more complex models, which are able to represent more involved relationships, may require sophisticated data structures and costly learning algorithms. The impact that the choice between simple and more complex models has in the search efficiency will depend on the addressed optimization problem. In some cases, a simple model can help to reach non-optimal but acceptable solutions in a short time. In other situations, e.g. deceptive problems, an EDA that uses a simple model could move the search away from the area of promising solutions. Another criterion that should be taken into consideration to choose an EDA is whether there is any previous knowledge about the problem structure, and which kind of probabilistic model is best suited to represent this knowledge [1].

## 3 New Hybrid EDA for Protein Folding Based on HP Model

### *3.1 Problem Representation for EDA*

In the algorithm of protein folding optimum, one of the important problems is how to present a specific conformation. To embed a hydrophobic pattern $S \in \{H,P\}^+$ into a lattice, we have three methods: Cartesian Coordinate, Internal Coordinate and Distance Matrix [15].

1) Cartesian Coordinate: The position of residues is specified independently from other residues.
2) Internal Coordinate: The position of each residue depends upon its predecessor residues in the sequence. There are two types of internal coordinate: absolute directions where the residue directions are relative to the axes defined by the lattice, and relative directions where the residue directions are relative to the direction of the previous move.
3) Distance Matrix: The location of a given residue is computed by means of its distance matrix.

Krasnogor et al. (1999) [15] performed an exhaustive comparative study using evolutionary algorithms (EAs) with relative and absolute directions. The experimental results show that relative directions almost always outperform absolute directions over square and cubic lattice, while absolute directions have better performances when facing triangular lattices. Experimental evidence suggests internal coordinates with relative directions should be used. However, in general, it is difficult to assess the effectiveness of direction encoding on an EAs performance.

In this chapter, we use the representation of internal coordinates with relative direction, the position of each residue depends upon the previous move. Relative direction representation presents the direction of each residue relative to the main chain next turn direction. This representation can reduce the direction number of each position. For 2-D HP model, the set of direction is left, right and forward (L, R, F). And it is left, right, forward, up and down, (L, R, F, U, D) for the 3-D HP model. For example, by relative direction representation, the representation of the protein structure shown in Fig.1 is $s = (RFRRLLRRFRLR)$.

It can be noted that the backward direction is not used, because the backward direction will cause overlap in this representation. Thus, this representation can reduce the position collision in a certain degree to guarantee the self-avoiding walk folding procedure. There are other advantages of the relative direction representation. One is that the sequence conformation can be presented as one dimension array. The most important is that the change of a start direction will not influence the structure of other part in sequence.

## *3.2 The Probabilistic Model of EDA*

It is very important for EDAs to select an appropriate probabilistic model according to a given application problem. The probabilistic model is represented by conditional probability distributions for each variable and estimated from the genetic information of selected individuals in the current generation. Therefore, the type of probabilistic model also influences the number and strength of the interactions learned by the model.

In Ref. [24], three probabilistic models for EDAs are proposed to solve the HP model problem: $k$-order Markov model, tree model and mixtures of trees model. In our practice, we find that the $k$-order Markov model is an appropriate probabilistic model for the HP model problem, where $k \geq 0$ is a parameter of the model. It can effectively embody the self-avoiding folding characteristics of the HP model problem, because it is assumed that positions of adjacent residues are related in the protein folding procedure.

The $k$-order Markov model can encode the dependencies between the move of a residue and the moves of the previous residues in the sequence, and this information can be used in the generation of solutions. It is described as follow. The joint probability mass function of $X$ is denoted as $p(\mathrm{X} = X)$ or $p(X)$. And use $p(\mathrm{X}_i = x_i | \mathrm{X}_j = x_j)$ or the simplified form $p(x_i | x_j)$ to denote the conditional probability distribution of $\mathrm{X}_i = x_i$ given $\mathrm{X}_j = x_j$ .

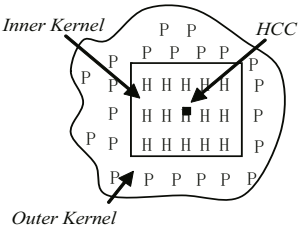



**Fig. 3** The H-Core of protein folding structure in 2-D HP model

In the $k$-order Markov model, the value of variable $X_i$ depends on the values of the previous $k$ variables. The joint probability distribution can be factorized as follows:

$$p_{MK}(X) = p(x_1, \cdots, x_{k+1}) \prod_{i=k+2}^{n} p(x_i | x_{i-1}, \cdots, x_{i-k}) \tag{1}$$

Since the structure of the Markov model is given, it can be used to construct the probabilistic model through computing the marginal and conditional probabilities of the set of selected individuals and to sample the new generation. To sample a new solution, first variables in the factor $(x_1, \cdots, x_{k+1})$ are generated and the rest of variables are sampled according to the order specified by the Markov factorization.

### *3.3 The Composite Fitness Function*

In order to increase the chance of finding the global optimum and reduce the complexity of EDA (population size and the number of generation needed), a composite fitness function containing the information of folding structure core is introduced to replace the traditional fitness function of HP model.

It is well known that the energy potential in the HP model reflects the fact that hydrophobic residues have a propensity to form a hydrophobic core. The Hs (hydrophobic residues) form the protein core and the Ps (hydrophilic or polar residues) tend to remain in the outer surface. As shown in Fig.3, the inner kernel, called the H-Core [13], is compact and mainly formed of Hs while the outer kernel consists mostly of Ps. The H-Core Center is called HCC. The H-Core is a rectangle-like area in 2-D lattice and cube-like space in 3-D lattice. The coordinates of HCC can be calculated by follows equations.

$$x_{HCC} = \frac{1}{n_H} \sum_{i=1}^{n_H} x_i, \quad y_{HCC} = \frac{1}{n_H} \sum_{i=1}^{n_H} y_i$$
$$z_{HCC} = \frac{1}{n_H} \sum_{i=1}^{n_H} z_i \tag{2}$$

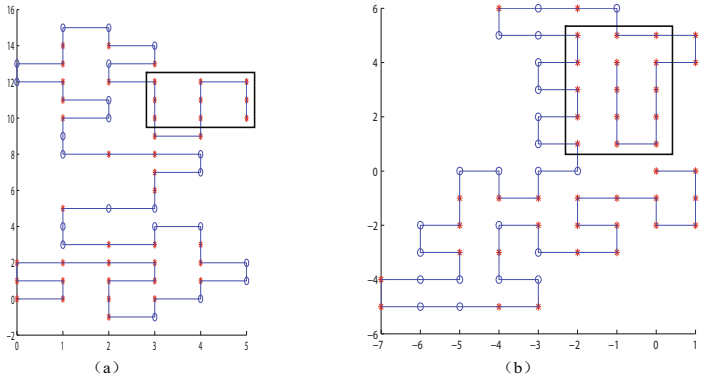



**Fig. 4** Two example solutions with same energy (Instance *S*8 in Tab.1, length: 64, energy: -25)

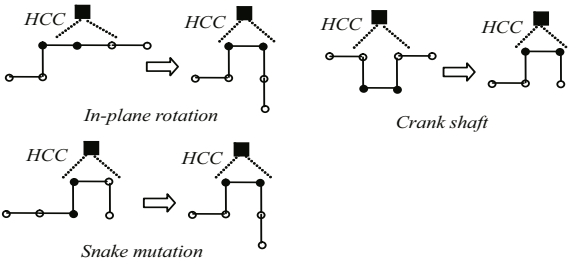



**Fig. 5** The guided operators for local search

where $n_H$ is the sum of hydrophobic residues in solution, $x_i$, $y_i$ and $z_i$ (for 3-D HP model) are the coordinates of hydrophobic residues position in lattice. We can calculate the number of Hs in inner kernel H-Core (denoted as $N_{HC}(x)$) through search surrounding rectangle area (cube space for 3-D HP model) of HCC.

The number of Hs in inner kernel H-Core is an important characteristic for the folding solution. It also reflects the optimum degree of solution. In the practice we find that, as showed in Fig.4, for two solutions with same basic HP model energy $E(x)$ (defined by the number of topological neighbor residues in lattice), the solution with bigger H-Core has more similar to the optimum solution, and it also has more biology significance. The two possible solutions of the Instance *S*8 (length is 64) have same basic HP model energy (-25), but they have different $N_{HC}(x)$ values (9 and 15 respectively). Obviously, the solution Fig.4(b) has more similar to the optimum solution.

In the new method, we introduce a novel composite fitness function containing the information of H-Core for the *k*-order Markov EDA:

$$Fit_{cp} = \omega(-E(x)) + (1-\omega)N_{HC}(x) \tag{3}$$

where, $E(x)$ is the total energy of the interaction between topological neighbor residues of HP model ($\varepsilon_{HH} = -1$, $\varepsilon_{PP} = \varepsilon_{HP} = 0$). $N_{HC}(x)$ is the number of Hs in inner kernel H-Core. $\omega$ is weight parameter of the fitness function, and we always take the $\omega > 0.5$, because the interaction energy $E(x)$ is the dominant characteristic of protein folding solution.

### *3.4 Local Search with Guided Operators*

An efficient evolutionary algorithm should make use of both the local information of solutions found so far and the global information about the search space. The local information of solutions found so far can be helpful for exploitation, while the global information can guide the search for exploring promising areas. The search in EDAs is mainly based on the global information, but local search is an exploitation method based on local information. Therefore, it is worthwhile investigating whether combining local search with EDA could improve the performance of the EDA.

Local search with a set of guided operators is implemented in the new hybrid EDA. Some of these operations have been utilized as mutations in the previous GA and ant colony optimizations studies of protein folding [26]. But in this chapter, we call them as "guided operators" meaning that those operations are implemented only under some special conditions.

Take 2-D HP model as example, the special conditions defined as follow. 1) Guided operation should guarantee the validity of individual, i.e. it can not produce position collision in lattice. If we want to change some residues to other positions in lattice, the object positions must be empty. 2) Guided operation should follow a basic principle that make Hs as near as possible to the HCC and Ps far away from the HCC according to the relative position in lattice, as shown in Fig.5.

The way of choosing individuals to implement local search is described as follow. In each iteration procedure of EDAs, use the composite fitness function (described by Eq.(3)) to sort the selection individuals. According to the distribution of individuals' fitness, randomly select some individuals (the number is a certain percentage of the population) in each fitness domain to implement the local search with guided operators.

EDAs extract globally statistical information from the previous search and then build a probabilistic model for modeling the distribution of best solutions visited in the search space. However, the information of the locations of best individual solutions found so far is not directly used for guiding the further search. Local search with guided operator generates offspring through combination of global statistical information and the location information of solutions found so far. The resultant solution can (hopefully) fall in or close to a promising area which is characterized by the probabilistic model.

# 4 Improved Backtracking-Based Repairing Method

## 4.1 Backtracking Method

In HP model, A *collision* is the embedding of two different peptides onto the same vertex of the lattice. As each member of the initial population of EA based method is randomly generated, it may represent an illegal conformation resulting one or more collisions when embedded. Similarly, *crossover* and *mutation* operations of GAs, *sampling* from probabilistic model operation of EDAs and other genetic operations may produce additional collisions.

In Ref. [7], a backtracking algorithm was introduced to repair the positional collisions. It utilizes backtracking strategy to search feasible positions for collision residues in folding procedure. This particular algorithm constitutes a simple yet efficient approach for the purposed task. Its pseudocode is showed in Fig.6.

The algorithm receives three parameters. The first one is $\lambda$, a table containing the allowed moves for each residue in the protein; thus, $\lambda_k$ is a list of allowed moves for $(k+1)$-th residue and $\lambda_{k,r}$ is the $r$-th move. Although $\lambda$ may contain in principle the full set of moves, in general $|\lambda|$ will not be the same for every $k$. The second parameter $s$ is a partial conformation involving $|s|$ residues. As to the third parameter, it is a Boolean flag used to finalize the execution of the algorithm as soon as a feasible conformation is found. Notice finally that the operator :: represents the sequence concatenation operator.

## 4.2 Disadvantage of Traditional Backtracking-Based Method

The basic backtracking method mentioned in the previous section has been shown to be a simple and efficient means of positional collision repairing for protein folding. But in our practice, we found that the repairing computational cost is very heavy for long sequence instances of more than 50 residues.

In the generic search procedure of protein folding, especially for the long protein sequences, the EAs based algorithm will produce a lots of valid and invalid individuals that contain closed-areas (or closed-spaces in 3-D circumstance). The Fig.7 shows a valid 2-D individual's conformation contains two closed-areas. When the basic backtracking method is used to repair invalid individuals contain closed-areas, some invalid closed-areas made by backtracking searching folding procedure will produce computational cost wastes.

Take the 2-D circumstance as example, as showed in Fig.8(a), we assume that there is a closed-area formed by residues from 1 to $n$. If the folding procedure select right (R) as the next direction for $n+1$ residue, it will enter the closed-area. Thus, even if the size of this closed-area can not satisfy the length of remain residues (called it as invalid closed-area), the traditional backtracking method will still search all empty position in closed-area by backtracking operation. This will produce a large number of computational cost wastes. According to our experiment,

```
1.  R-Backtracking (↓ λ:MOVE[], ↓↑ s:MOVE[],
    ↑ SolutionFound:bool).
2.    if Feasible(s) then
3.      if |s| = n − 1 then
4.        SolutionFound ← TRUE
5.      else
6.        SolutionFound ← FALSE
7.        i ← 1
8.        while ¬SolutionFound ∧ (i ≤ |λ_|s||) do
9.          if Detect − fea ( λ_|s|,i , s) then
10.           s′ ← s :: λ_|s|,i
11.           R-Backtracking (λ, s′, SolutionFound)
12.         endif
13.         i ← i + 1
14.         if SolutionFound then
15.           s ← s′
16.         endif
17.       endwhile
18.     endif
19.   else
20.     SolutionFound ← FALSE
21.   endif
```

**Fig. 6** The pseudocode of the backtracking repairing algorithm

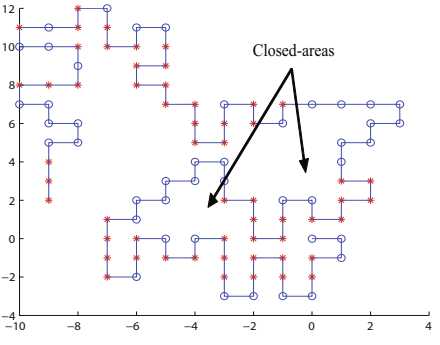



**Fig. 7** Illustration of the closed-areas in 2-D HP model

this phenomenon takes place with a high probability in repairing procedures for long sequence proteins.

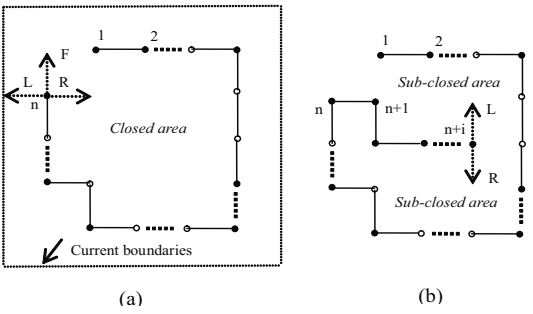



**Fig. 8** (a) Illustration of closed-area detection. (b) The situation that need to implement backtracking

## *4.3 The Improved Method*

To solve the above problem, in the improved method, a detection for feasibility is introduced. The detection procedure is implemented before selecting direction for next residue to avoid entering an invalid closed-area (The procedure $Detect - fea$ in Line 9 of Fig.6).

The pseudocode of the introduced detection algorithm is shown in Fig.9. The main idea of the detection procedure is described as follow.

1) The current boundaries in lattice is defined as shown in Fig.8(a), the scale of boundary coordinates is larger one position than current filled area and will be changed with current folding procedure. For example, four current boundaries of the 2-D solution shown in Fig.7 are $x = -11$ (left), $x = 4$ (right), $y = 13$ (up) and $y = -4$ (down). They can be used to check the folding procedure whether enter a closed-area. If the detection meet the current boundaries, the folding procedure will not enter a closed-area.

2) A search approach, similar to Floodfill strategy, is utilized to count possible empty positions connected to the detected direction $\langle \lambda_{|s|,i} \rangle$, i.e. those empty positions which could be arrived through this direction. The Floodfill-like search approach count and label the possible empty positions based on a queue $Q$. If the queue $Q$ becomes empty, it means that all possible empty positions are labeled.

3) In the operation of detection procedure, if the current boundaries are met or the number of counted empty positions is larger than the length of remain residues, it means that the folding will not enter a closed-area or entered closed-area is not invalid. Under such circumstance, the detected direction $\langle \lambda_{|s|,i} \rangle$ could be chosen for the next residue.

For long length protein sequences, there are many invalid closed-area in folding procedure. The improved method can significantly reduce the computational cost. Although the detection procedure has some computational cost, it is far less than the cost of backtracking searching operations for invalid closed-areas.

The main reason of the improvement is that the improved method can significantly reduce the number of backtracking operation. The folding procedure

```
1. ↑ Detect-fea (↓ ⟨λ⟩, ↓ s:MOVE[]):bool.
2.   Calculate current boundaries according to s
3.   Set label for every positions in s (i.e. not empty).
4.   if Feasible(s :: ⟨λ⟩) then
5.     counter = 0 and set an empty queue Q
6.     Add ⟨λ⟩ to the end of Q
7.     while Q is not empty do
8.       x=first element of Q
9.       if position x is unlabeled
10.        Set label for position x
11.        counter = counter + 1
12.      endif
13.      if (position x meet the current boundaries) or (counter is larger than the length of
     remain residues)
14.        Return TRUE
15.      endif
16.      Remove the first element of Q
17.      if west neighbor of x is unlabeled
18.        Set label for west − x
19.        Add west − x to the end of Q
20.      endif
21.      Check and process other three (five for 3D) neighbor positions of x using similar
     strategies Step (17)-(20)
22.    endwhile
23.  endif
24.  Return FALSE
```

**Fig. 9** The pseudocode of the detection procedure

implements backtracking operation only under few special circumstances. As shown in Fig.8(b), if the folding procedure has selected the right (R) direction for the $n+1$ residues. But at $n+i$ position, the folding procedure produce two sub-closed-areas and all of two are invalid closed-area for remain residues. The folding procedure should implement a backtracking operation under this situation. It will back to the $n+i-1$ residue and search other possible directions.

## 5 Experiments

### *5.1 Problem Benchmark*

For our experiments, we use the first nine instances of the *Tortilla* 2-D HP Benchmarks[1], and the last two instances are taken from Ref. [24] to test the searching capability of the new method. In Tab.1, $E^*$ is the optimal or best-known energy

[1] `http://www.cs.sandia.gov/tech-reports/compbio/tortilla-hp-benchmarks.html`

**Table 1** HP instances used in the experiments

| No. | Size | $E^*$ | Sequence |
|---|---|---|---|
| s1 | 20 | -9 | $HPHP_2H_2PHP_2HPH_2P_2HPH$ |
| s2 | 24 | -9 | $H_2P_2(HP_2)_6H_2$ |
| s3 | 25 | -8 | $P_2HP_2(H_2P_4)_3H_2$ |
| s4 | 36 | -14 | $P_3H_2P_2H_2P_5H_7P_2H_2P_4H_2P_2HP_2$ |
| s5 | 48 | -23 | $P_2H(P_2H_2)_2P_5H_{10}P_6(H_2P_2)_2HP_2H_5$ |
| s6 | 50 | -21 | $H_2(PH)_3PH_4P(HP_3)_2HPH_4(PH)_4H$ |
| s7 | 60 | -36 | $P_2H_3PH_8P_3H_{10}PHP_3H_12P_4H_6PH_2PHP$ |
| s8 | 64 | -42 | $H_{12}(PH)_2(P_2H_2)_2P_2H(P_2H_2)_2P_2H(P_2H_2)_2P_2(HP)_2H_{12}$ |
| s9 | 85 | -53 | $H_4P_4H_{12}P_6(H_{12}P_3)_3H(P_2H_2)_2P_2HPH$ |
| s10 | 100 | -48 | $P_6HPH_2P_5H_3PH_5PH_2P_4H_2P_2H_2PH_5PH_{10}PH_2PH_7P_{11}H_7$ $P_2HPH_3P_6HPHH$ |
| s11 | 100 | -50 | $P_3H_2P_2H_4P_2H_3(PH_2)_3H_2P_8H_6P_2H_6P_9HPH_2PH_{11}P_2H_3P$ $H_2PHP_2HPH_3P_6H_3$ |

value, $H_i$, $P_i$ and $(\cdots)_i$ indicate $i$ repetitions of the relative symbol or subsequence. It is important to highlight that most randomly generated amino acid sequences do not behave like natural proteins, because the latter are products of natural selection. Likewise, most randomly generated sequences of $H$ and $P$ residues in the HP model do not fold to a single conformation [24].

## *5.2 Results of the Hybrid EDA for HP Model*

In order to test the effects of the composite fitness function and the local search with guided mutation, we implemented different experiments by using one of them independently. The composite fitness function can help to reduce the complexity of EDA, it can obtain same results with basic $k$-order Markov EDA (MK-EDA) by using less population size and generation number. The local search with guided mutation can help to obtain the global optimum for some instances. But it seems that combination of two strategies can get much better results in practice. We investigated the performance of MK-EDA for $k \in \{2,3,4\}$ and find that the algorithm perform very well when $k = 3$.

In the experiments of the hybrid EDA, all algorithms use a population size of 2000 individuals. Truncation selection is used as selection strategy. In this strategy, individuals are ordered by fitness, and the best $T * PopSize$ are selected where $T$ is the truncation coefficient. The parameter $T = 0.15$ is used in our algorithms. The best elitism scheme is also implemented in algorithms, the set of selected solutions in the current generation are passed to the next generation. The stop criteria considered are a maximum number of generation $G = 1000$ or that the number of different individuals in the population falls below 5. For the protein instances of $s6$ to $s11$, we use the improved backtracking-based method to repair the invalid solutions.

The results of the new method comparing with the MK-EDA for the 2-D HP Model shown in Tab.2. It includes the best solution and the percentage of times the

**Table 2** Results of comparing with MK-EDA for 2-D HP model

| No. | $E^*$ | New Method | | MK-EDA | |
|---|---|---|---|---|---|
| | | $H(X)$ | *Percentage* | $H(X)$ | *Percentage* |
| s1 | -9 | -9 | 100 | -9 | 100 |
| s2 | -9 | -9 | 100 | -9 | 100 |
| s3 | -8 | -8 | 100 | -8 | 100 |
| s4 | -14 | -14 | 16 | -14 | 5 |
| s5 | -23 | -23 | 22 | -23 | 7 |
| s6 | -21 | -21 | 92 | -21 | 57 |
| s7 | -36 | -36 | 24 | -35 | 12 |
| s8 | -42 | -42 | 16 | -42 | 4 |
| s9 | -53 | -53 | 8 | -52 | 3 |
| s10 | -48 | -48 | 12 | -47 | 4 |
| s11 | -50 | -49 | 6 | -48 | 2 |

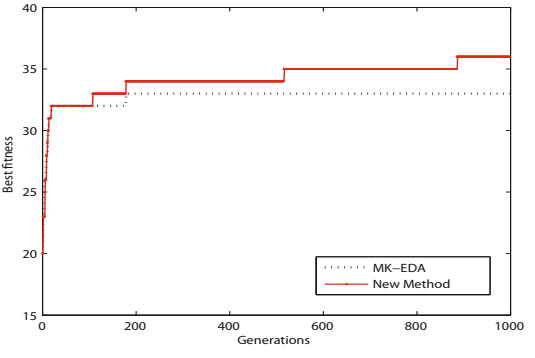



**Fig. 10** The best fitness for one representative run of instance $S7$

best solution has been found in 100 experiments. The results of MK-EDA are also obtained by our experiments with same EDA parameters ($Pop = 2000$, $G = 1000$ and $T = 0.15$) as the new method. From the experiment results we can find that the new method has more chance to find global optimum or suboptimum solution for long sequences. The MK-EDA cannot find the global optimum of the deceptive sequences and long sequences $s7$, $s9$, $s10$ and $s11$, but the new method can find the global optimum of the sequences $s7$, $s9$ and $s10$, and can find the second best solution for sequence $s11$. Figure 10 shows the best fitness for one representative run of the instance $S7$.

The performance of the new method comparing with the best results achieved with other evolutionary and Monte Carlo optimization algorithms is shown in Tab.3 (2-D HP model) and Tab.4 (3-D HP model). The results of other method are cited from Ref. [24, 13].From the experiment results we can find that none of the algorithms are able to outperform the rest of algorithms for all the instances. The PERM is one of the best contenders in all cases except $s8$ in which its result is very poor.

**Table 3** Results achieved by different search methods for 2-D HP model

| No. | New Method $H(X)$ | MK-EDA $H(X)$ | GA $H(X)$ | NewACO $H(X)$ | PERM $H(X)$ |
|---|---|---|---|---|---|
| s1 | -9 | -9 | -9 | -9 | -9 |
| s2 | -9 | -9 | -9 | -9 | -9 |
| s3 | -8 | -8 | -8 | -8 | -8 |
| s4 | -14 | -14 | -14 | -14 | -14 |
| s5 | -23 | -23 | -22 | -23 | -23 |
| s6 | -21 | -21 | -21 | -21 | -21 |
| s7 | -36 | -35 | -34 | -36 | -36 |
| s8 | -42 | -42 | -37 | -42 | -38 |
| s9 | -53 | -52 | | -51 | -53 |
| s10 | -48 | -47 | | -47 | -48 |
| s11 | -49 | -48 | | -47 | -50 |

**Table 4** Results achieved by different search methods for 3-D HP model

| No. | New Method $H(X)$ | MK-EDA $H(X)$ | Hybrid GA $H(X)$ | IA $H(X)$ |
|---|---|---|---|---|
| s1 | -11 | -11 | -11 | -11 |
| s2 | -13 | -13 | -11 | -13 |
| s3 | -9 | -9 | -9 | -9 |
| s4 | -18 | -18 | -18 | -18 |
| s5 | -29 | -29 | -28 | -28 |
| s6 | -30 | -29 | -22 | -23 |
| s7 | -49 | -48 | -48 | -41 |
| s8 | -51 | -50 | -46 | -42 |

It shows that the new method is very competitive with the other existing algorithms for the PSP on lattice HP models. It should be noted that all fitness values of the new method in the comparing results are calculated by basic HP-model fitness definition. The composite fitness function is only used in optimization procedure of EDA.

## *5.3 Results of Comparing Computational Cost*

The hybrid EDA is an improved method based on the MK-EDA. The detailed computational cost analysis of the MK-EDA method can be found in the Ref. [24]. Comparing with the MK-EDA, there are three modifications in the new method: 1) the composite fitness function; 2) the local search with guided operations; 3) the improved backtracking-based repairing method for the long protein instances. As far as the computational cost is concerned, modifications of 1) and 2) will produce some additional computational cost. The modification 3) can significantly reduce the repairing costs for EDA invalid individuals.

**Table 5** Comparing Results of Improved Backtracking Repairing Method in 2-D HP model

| No. | Size | AVG-Backtracking operation | | AVG-CPUTime (Hour) | |
|---|---|---|---|---|---|
| | | MK-EDA | New Method | MK-EDA | New Method |
| s1 | 20 | 2.1492E+4 | 2.1496E+4 | 0.2181 | 0.2309 |
| s2 | 24 | 2.5715E+4 | 2.5715E+4 | 0.2659 | 0.2761 |
| s3 | 25 | 2.6834E+4 | 2.7044E+4 | 0.2774 | 0.2805 |
| s4 | 36 | 3.8898E+4 | 3.8797E+4 | 0.3059 | 0.3203 |
| s5 | 48 | 5.2577E+5 | 5.2617E+5 | 0.4341 | 0.4659 |
| s6 | 50 | 5.7842E+6 | *5.4887E+6 | 0.6249 | 0.5768 |
| s7 | 60 | 7.4545E+6 | *6.7478E+6 | 0.9276 | 0.7516 |
| s8 | 64 | 9.6731E+6 | *7.2236E+6 | 1.1276 | 0.8661 |
| s9 | 85 | 2.0829E+8 | *1.1196E+7 | 12.3077 | 1.6086 |
| s10 | 100 | 2.6531E+8 | *1.9765E+7 | 15.7125 | 2.0007 |

To demonstrate the computational cost of the hybrid EDA comparing with MK-EDA, some practical experiments in 2-D are implemented. Two comparing methods are implemented with same parameters (population:1000, generation:100, the truncation selection of parameter $T = 0.15$) and same computational environment[2]. Because there are few closed-areas existing in short protein folding, the improved backtracking-based repairing method can not improve the EDA efficiency for short instances. In the comparing experiments for the short instances of $s1$ to $s5$, same basic backtracking repairing methods are used in two comparing methods. For the long instances of $s6$ to $s10$, the improved backtracking-based repairing method is used in the hybrid EDA.

The number of backtracking searching operation and computer CPU-Time are recorded. The average backtracking searching operations and the CPU-Times of 10 runs are shown in Tab.5. According to the results of the short instances of $s1$ to $s5$, we can find that the local search operations and the composite fitness calculation in the hybrid EDA produce some additional computational costs. But it is not very serious. The results of the long instances of $s6$ to $s10$ show that the proposed repairing method can significantly reduce the repairing costs. It not only covers the additional computational costs caused by local search and composite fitness calculation, but also improves the algorithm efficiency remarkably.

The backtracking searching operations of each generation for sequence $s8$ (the length is 64), $s9$ (the length is 85) and $s10$ (the length is 100) are also be counted and shown in Fig.11, Fig.12 and Fig.13. We can find that the improved backtracking-based repairing method can significantly reduce the number of backtracking searching operation. And the longer the protein sequence length is, the more remarkable the improvement achieves.

[2] All experiments are performed on the computers with Intel Xeon 2.20 GHz processor, and 1 GB of RAM.

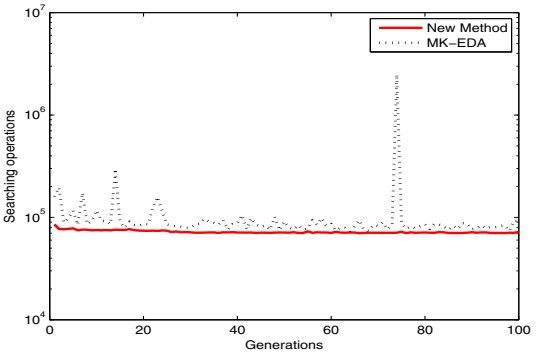



**Fig. 11** The number of backtracking searching operations for instance *s*8

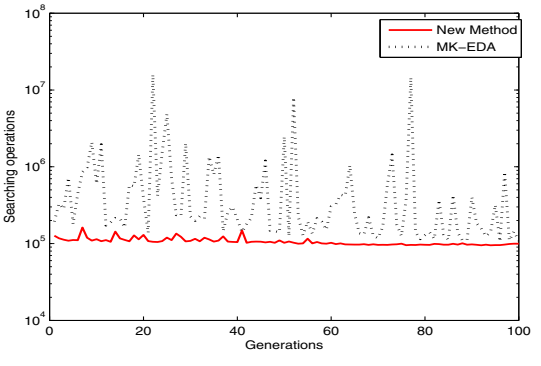



**Fig. 12** The number of backtracking searching operations for instance *s*9

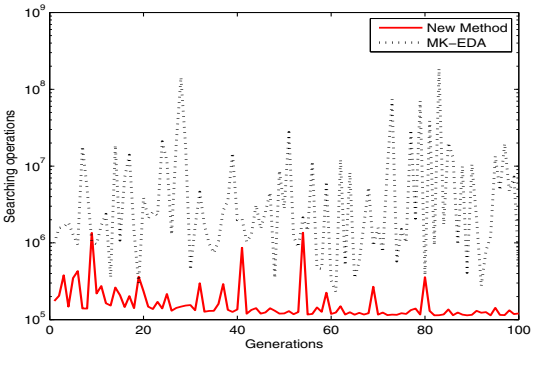



**Fig. 13** The number of backtracking searching operations for instance *s*10

## 6 Conclusions and Further Work

In this chapter, we introduce a novel hybrid EDA method to solve the HP model problem. For the basic *k*-order Markov EDA, it has very low chance to find the general optimum for those long sequence and deceptive protein instances. A composite fitness function containing information of folding structure core is introduced to replace the traditional fitness function of HP model. It can help to select better individuals for probabilistic model of EDA algorithm. In addition, local search with guided operators is utilized to refine found solutions for improving efficiency of EDA.

For the disadvantage of heavy computational cost of the traditional backtracking method which used to repair the invalid individuals in population. It will produce heavy computational cost for searching invalid closed-areas of folding structure. An improved method is introduced to reduce the repairing computational cost for the long protein sequences. A detection procedure for feasibility is added to avoid entering invalid closed-areas when selecting directions for the residues. Thus, it can significantly reduce the number of backtracking searching operation and the computational cost for long sequence protein. It can be noted that the improved backtracking repairing method can be used in all EA based PSP methods that need to repair invalid individuals. And the underlying mutations are implemented for individuals in repairing procedure.

Experimental results demonstrate that the new method outperform the basic EDA method. At the same time, the new method is very competitive with other existing algorithms for the PSP on lattice HP models. Further research is needed to determine more efficient local search strategies and probabilistic models of EDA for protein HP model problem.

## References

1. Armananzas, R., Inza, I., Santana, R., Saeys, Y., Flores, J.L., Lozano, J.A., Peer, Y.V., Blanco, R., Robles, V., Bielza, C., Larranaga, P.: A review of estimation of distribution algorithms in bioinformatics. BioData Mining I(6), 1–12 (2008)
2. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. CMU-CS-94C163, Carnegie Mellon University (1994)
3. Baluja, S., Davies, S.: Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In: Proc. the 14th International Conference on Machine Learning, pp. 30–38 (1997)
4. Berger, B., Leight, T.: Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. Journal of Computational Biology 5(1), 27–40 (1998)
5. Chen, B., Li, L., Hu, J.: An improved backtracking method for EDAs based protein folding. In: Proc. of ICROS-SICE International Joint Conference 2009, Fukuoka Japan, pp. 4669–4673 (2009)
6. Chen, B., Li, L., Hu, J.: A novel EDAs based method for HP model protein folding. In: Proc. 2009 IEEE Congress on Evolutionary Computation (CEC 2009), Trondheim Norway, pp. 309–315 (2009)
7. Cotta, C.: Protein structure prediction using evolutionary algorithms hybridized with backtracking. In: Mira, J. (ed.) Evolutionary Computation in Bioinformatics, pp. 321–328. Springer, Berlin (2003)

8. Crescenzi, P., Goldman, D., Papadimitriou, C., Piccolboni, A., Yannakakis, M.: On the complexity of protein folding. Journal of Computational Biology 5(3), 423–466 (1998)
9. Flores, S., Smith, J.: Study of fitness landscapes for the HP model of protein structure prediction. In: Proc. 2003 IEEE Congress on Evolutionary Computation (CEC 2003), Trondheim Norway, pp. 2338–2345 (2003)
10. Goldberg, D.E.: The Design of Innovation: Genetic Algorithms and Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2002)
11. Greenwood, G.W., Shin, J.M.: On the evolutionary search for solutions to the protein folding problem. In: Fogel, G.B. (ed.) Artificial Neural Nets Problem Methods, pp. 115–136. Elsevier Science and Technology Books, Amsterdam (2002)
12. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. IEEE Transactions on Evolutionary Computation 3(4), 287–297 (1999)
13. Hoque, T., Chetty, M., Dooley, L.S.: A guided genetic algorithm for protein folding prediction using 3D hydrophobic-hydrophilic model. In: Proc. 2006 IEEE Congress on Evolutionary Computation (CEC 2006), pp. 2339–2346 (2006)
14. Jeremy, S.D., Charles Jr., L.I., Paul, V.: Mimic: Finding optima by estimating probability densities. In: Advances in Neural Information Processing Systems, p. 424. The MIT Press, Cambridge (1996)
15. Krasnogor, N., Hart, W.E., Smith, J., Pelta, D.A.: Protein structure prediction with evolutionary algorithms. In: Proc. of Genetic Evol.Comput. Conf., Orlando, pp. 1596–1601 (1999)
16. Larranaga, P., Lozano, J.A.: Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic, Dordrecht (2002)
17. Lau, K.F., Dill, K.A.: A lattice statistical mechanics model of the conformational and sequence spaces of proteins. Marcromolecules 22(10), 3986–3997 (1989)
18. Lozano, J.A., Larranaga, P., Inza, I., Bengoetxea, E.: Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms. Springer, Berlin (2006)
19. Mendiburu, A., Lozano, J.A., Miguel-Alonso, J.: Parallel implementation of EDAs based on probabilistic graphical models. IEEE Transactions on Evolutionary Computation 9(4), 406–423 (2005)
20. Muhlenbein, H., Mahnig, T., Rodriguez, A.O.: Schemata, distributions and graphical models in evolutionary optimization. Journal of Heuristics 5(2), 213–247 (1999)
21. Muhlenbein, H., PaaB, G.: From recombination of genes to the estimation of distributions i. binary parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
22. Pelikan, M., Goldber, D., Lobo, F.: A survey of optimization by building and using probabilistic models. Computational Optimization and Applications 21, 5–20 (2002)
23. Pelikan, M., Goldberg, D.E., Cantu-Paz, E.: Boa: The bayesian optimization algorithm, pp. 525–532. Morgan Kaufmann, San Francisco (1999)
24. Santana, R., Larranaga, P., Lozano, J.A.: Protein folding in simplified models with estimation of distribution algorithms. IEEE Transactions on Evolutionary Computation 12(4), 418–438 (2008)
25. Santana, R., Ponce, L., Ochoa, A.: The edge incident model. In: Proc. the Second Symposium on Artificial Intelligence (CIMAF 1999), pp. 352–359 (1999)
26. Song, J., Cheng, J., Zheng, T., Mao, J.: A novel genetic algorithm for HP model protein folding. In: Proc. of 6th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2005), pp. 935–937 (2005)
27. Unger, R., Moult, J.: Genetic algorithms for protein folding simulations. Journal of Molecular Biology 231(1), 75–81 (1993)

# Sensible Initialization of a Computational Evolution System Using Expert Knowledge for Epistasis Analysis in Human Genetics

Joshua L. Payne, Casey S. Greene, Douglas P. Hill, and Jason H. Moore

**Abstract.** High throughput sequencing technologies now routinely measure over one million DNA sequence variations on the human genome. Analyses of these data have demonstrated that single sequence variants predictive of common human disease are rare. Instead, disease risk is thought to be the result of a confluence of many genes acting in concert, often with no statistically significant individual effects. The detection and characterization of such gene-gene interactions that predispose for human disease is a computationally daunting task, since the search space grows exponentially with the number of measured genetic variations. Traditional artificial evolution methods have offered some promise in this problem domain, but they are plagued by the lack of marginal effects of individual sequence variants. To address this problem, we have developed a computational evolution system that allows for the evolution of solutions and solution operators of arbitrary complexity. In this study, we incorporate a linkage learning technique into the population initialization method of the computational evolution system and investigate its influence on the ability to detect and characterize gene-gene interactions in synthetic data sets. These data sets are generated to exhibit characteristics of real genome-wide association studies for purely epistatic diseases with various heritabilities. Our results demonstrate that incorporating linkage learning in population initialization via expert knowledge sources improves classification accuracy, enhancing our ability to automate the discovery and characterization of the genetic causes of common human diseases.

## 1 Introduction

Recent technological advances have allowed for inexpensive and dense mappings of the human genome, making genome-wide association studies (GWAS) a standard

Joshua L. Payne · Casey S. Greene · Douglas P. Hill · Jason H. Moore
Computational Genetics Laboratory, Dartmouth Medical School, 1 Medical Center Drive, Lebanon, NH, USA
e-mail:{Joshua.L.Payne,Casey.S.Greene,Douglas.P.Hill,
Jason.H.Moore}@Dartmouth.edu

Y.-p. Chen (Ed.): Exploitation of Linkage Learning, ALO 3, pp. 215–226.
springerlink.com 

form of analysis in the detection of common human disease. The goal of GWAS is to identify genetic markers that differ significantly between diseased and healthy individuals, through a comparison of allele frequencies at specific loci. One commonly employed genetic marker is the single nucleotide polymorphism (SNP), which is a single location in the genome that varies between people. To provide sufficient coverage of the human genome for GWAS, it is estimated that over one million SNPs have to be considered [8], and samples of this size are now readily provided by high-throughput technologies. However, analyses of these data have rarely identified single sequence variants that are predictive of common human disease. Given the robustness and complex structure of metabolic and proteomic networks [18], it is reasonable to assume that such monogenic diseases are the exception, not the rule, and that many diseases are caused by two or more interacting genes. Such gene-gene interactions, or epistasis, dramatically increase the difficulty of using GWAS to uncover the genetic basis of disease [13]. For one million candidate SNPs, there are $5 \times 10^{11}$ pairwise combinations and $1.7 \times 10^{17}$ three-way combinations. For higher order interactions, the number of possible combinations is enormous. A major charge for bioinformatics is to develop efficient algorithms to navigate through these astronomical search spaces, in order to detect and characterize the genetic causes of common human disease.

Due to the combinatorial nature of this problem, algorithms designed to discover gene-gene interactions in GWAS will need to rely on heuristics. Methods that employ exhaustive search will not be feasible. Statistical and machine learning techniques, such as neural networks [10], have been applied in this problem domain, but have only proven successful for cases with a small number of SNPs. Alternative approaches, such as multifactor dimensionality reduction [17] and random chemistry [3], have also shown promise, though they are similarly limited to data sets with only a small number of SNPs. Artificial evolution techniques, such as genetic programming, have been investigated in this problem domain, but they have had limited success because individual SNPs often show little or no marginal effects, and as such, there are no building blocks for evolution to piece together. However, recent results have demonstrated that the inclusion of expert knowledge, such as information gained from feature selection methods, can be used to bias such nature-inspired classification algorithms toward SNPs that are suspected to play a role in disease predisposition [6, 7, 14, 11, 15].

One source of expert knowledge that has proven useful in this domain is a family of machine learning techniques referred to as Relief [5, 9, 16]. These algorithms are able to detect SNPs that are associated with disease via independent or main effects, although they cannot provide a model of the genetic architecture of disease. However, the information provided by Relief can be used to supply artificial evolution with the building blocks needed to successfully generate such an architectural model. For example, improvements in classification power have been obtained by using Relief variants to bias mutation operators [6] and population initialization [7] in genetic programming. Such feature selection techniques are a form of linkage learning, where potential interactions between SNPs are inferred and subsequently exploited to bias evolutionary search.

Though classical artificial evolution methods, such as genetic programming, have shown promise in this problem domain if guided by expert knowledge [6, 7, 14, 11, 15], it has been suggested that the inclusion of a greater degree of biological realism may improve algorithm performance. Specifically, Banzhaf et al. [1] have called for the development of computational evolution systems (CES) that embrace, and attempt to emulate, the complexity of natural systems. To this end, we have developed a hierarchical, spatially-extended CES that includes evolvable solution operators of arbitrary complexity, population memory via archives, feedback loops between archives and solutions, hierarchical organization, and environmental sensing. In a series of recent investigations [4, 7, 11], this system has been successfully applied to epistasis analysis in GWAS for human genetics.

Here, we investigate the inclusion of linkage learning via sensible initialization in CES for the detection of epistatic interactions in GWAS. Specifically, we develop an expert-knowledge-aware initialization method that uses the feature weights provided by a machine learning technique to bias the selection of attributes for the initial population. We compare this initialization method to both random and enumerative initialization on synthetic data sets generated to exhibit representative characteristics of GWAS.

## 2 Computational Evolution System

In order to directly infer the influence of the initialization strategy on algorithm performance in the absence of other confounding effects, we use a simplified version of the computational evolution system (CES) discussed in [11]. In this section, we describe the CES as it is employed in this study.

In Fig. 1, we provide a schematic diagram of the system. Solutions are organized on a lattice at the bottom layer of the hierarchy, where competition between solutions occurs locally among adjacent lattice sites (Fig. 1D). At the second layer of the hierarchy is a lattice of solution operators of arbitrary size and complexity, which are used to modify the solutions (Fig. 1C). At the third layer, is a lattice of mutation operators that modify the solution operators (Fig. 1B). At the fourth layer is the mutation frequency, which governs the rate at which the mutation operators are modified (Fig. 1A).

### *2.1 Solution Representation, Evaluation, and Selection*

Solutions are represented using stacks, where each element in the stack consists of a function and two input arguments (Fig. 1D). The function set contains $+, -, *, /, \%, <, \leq, >, \geq, ==, \neq$, where $\%$ is a protected modulus operator. The input arguments are SNPs.

Each solution produces a real valued output $S_i$ when applied to an individual $i$ in a SNP data set. These outputs are used to classify individuals as healthy or diseased using symbolic discriminant analysis (SDA) [12], as follows. The solution is applied to all healthy individuals in the data set and a distribution of outputs $S^{healthy}$

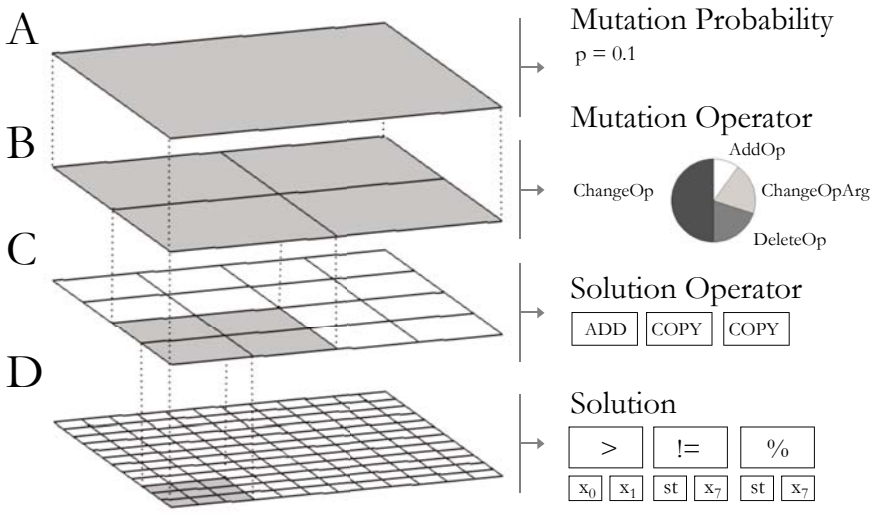



**Fig. 1** Schematic diagram of the simplified computational evolution system considered in this study. The hierarchical lattice structure is shown on the left and specific details of each layer are provided on the right. At the lowest level (D) is a two-dimensional toroidal lattice of solutions, where each lattice cell contains a single solution. Solutions are represented using stacks. In the above example, the Boolean output of $x_0 > x_1$ will be tested for inequality with $x_7$ via the stack (denoted by st) and this Boolean result will be an operand of the modulus operator (again, via st). At the second level (C) is a grid of solution operators that each consist of some combination of the building blocks ADD, ALTER, COPY, DELETE, and REPLACE. The top two levels of the hierarchy (A and B) generate variability in the solution operators. The experiments considered herein used a solution lattice of $32 \times 32$ cells. A $12 \times 12$ lattice is shown here for visual clarity

is recorded. Similarly, the solution is applied to all diseased individuals in the data set and a distribution of outputs $S^{diseased}$ is recorded. A classification threshold $S_0$ is then calculated as the arithmetic mean of the medians of the $S^{healthy}$ and $S^{diseased}$ distributions. The classification rule then assigns an individual $i$ healthy status if $S_i > S_0$ and diseased status if $S_i \leq S_0$.

The classification rule of a given solution can be used to calculate the number of true positives ($TP$), false positives ($FP$), true negatives ($TN$), and false negatives ($FN$), through a comparison of the predicted and actual clinical endpoints. This information can then be used to calculate a measure of solution accuracy

$$A = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right). \tag{1}$$

The fitness $f$ of a solution is given by its accuracy, weighted by solution length to encourage parsimony

$$f = A + \frac{\alpha}{L}, \tag{2}$$

where $L$ is the number of elements in the solution stack and $\alpha$ is a tunable parameter (for all experiments considered here, $\alpha = 0.001$).

The population is organized on a toroidal, two-dimensional lattice where each solution resides in its own cell. Selection is synchronous and occurs within spatially-localized, overlapping neighborhoods. Specifically, each solution competes with those solutions residing in the eight surrounding cells (Moore neighborhood) and the solution with the highest fitness is selected to repopulate that cell for the next generation. Reproduction occurs using the evolvable solution operators described in the next section.

## *2.2 Solution Operators*

One of the simplifying assumptions of traditional artificial evolution methods is that genetic variation is introduced via point mutations and linear recombination events. However, the variation operators of biological systems are myriad, with insertions, deletions, inversions, transpositions, and point mutations all occurring in concert. In order to better mimic these salient features of natural systems, our CES allows for the evolution of variation operators of arbitrary complexity. This is achieved by initializing the solution operator lattice (Fig. 1C) with five basic building blocks, ADD, ALTER, COPY, DELETE, and REPLACE, which can be recombined in any way to form new operators.

These operators work as follows. ADD places a new function and its arguments into the focal solution stack. ALTER randomly chooses an element of the focal solution stack, and mutates either the function or one of its input arguments. COPY inserts a random element of the focal solution stack into the stack of a randomly chosen neighboring solution. DELETE removes an element from the focal solution stack and REPLACE extracts a sequence of random length from a neighboring solution stack and overwrites a randomly chosen sequence of the focal solution stack with that information.

In the extended version of CES [11], each solution operator also has an associated vector of probabilities that determine the frequency with which functions and attributes are modified at random, via expert knowledge sources, or archives. In the simplified CES considered here, all modifications occur at random.

Similar to the solutions, the solution operators reside on a two-dimensional lattice (Fig. 1C). However, the granularity of the solution operator lattice is more coarse than the solution lattice, such that each solution operator is assigned to operate on a $3 \times 3$ sub-grid of solutions. The solution operators are also under selective pressure, and are assigned a fitness score based on how much change they evoke in the solutions they control [11]. Competition among solution operators occurs locally in a manner similar to the competition among solutions.

## *2.3 Mutation Operators*

The solution operators are modified by mutation operators that reside in the third layer of the hierarchy (Fig. 1B). The granularity of this lattice is further coarsened,

with each cell controlling one quarter of the solution operator lattice below. We consider four mutation operators. The first (DeleteOp) deletes an element of a solution operator. The second (AddOp) adds an element to a solution operator. The third (ChangeOp) mutates an existing element in a solution operator. The fourth (ChangeOpArg) alters the probability vectors associated with a solution operator. (In the simplified CES considered herein, this is a null operation.)

A four-element vector is used to store the probabilities with which each mutation operator is employed (Fig. 1B). These probabilities undergo mutation at a rate specified by the highest level in the hierarchy (Fig. 1A). The probability vectors of the four lattice cells are in competition with one another, with fitness assessment analogous to the solution operators.

## 3 Population Initialization

We consider three forms of population initialization. In each case, all initial solutions begin as a single, randomly chosen function with two input arguments, which can subsequently evolve into arbitrarily complex functional forms. The selection of the initial input arguments varies between the three methods. The first form of initialization is the approach taken in most artificial evolution systems, where the population is initialized at random. In CES, this entails choosing the attributes for each initial function with uniform probability from all available attributes, with replacement.

The second initialization method attempts to maximize diversity in the population, by ensuring that all attributes are represented at least once. This enumerative initializer works by selecting attributes at random from the pool of all attributes, without replacement, until all attributes have been selected. The attribute pool is then refreshed and the process continues until all initial solutions possess their required input arguments.

The third initialization method capitalizes on the expert knowledge gained from a member of the Relief family of machine learning algorithms. This algorithm is referred to as Spatially Uniform ReliefF (SURF) [5], an extension of Tuned ReliefF [16] that has proven effective in detecting interacting SNPs in GWAS with noisy data sets and small interaction effects. In brief, SURF provides weights to each SNP based on how likely that SNP is to be predictive of disease. Weights are adjusted by iteratively selecting individuals that are within a specified similarity threshold, and then increasing the weights of common SNPs if these individuals have different disease status or decreasing their weights by the same amount if the individuals have the same disease status. These SNP weights are then used to bias the selection of attributes in the expert-knowledge-aware initialization function. Each attribute is selected with probability proportional to its weight, with the caveat that the same attribute cannot be included twice in the same function.

## 4 Data Simulation

The artificial data sets considered in this study were generated to exhibit pure epistasis (i.e., no marginal effects) and specific heritabilities, where heritability is defined as the proportion of disease cases attributable to genetic effects. We consider heritabilities of 0.025, 0.05, 0.1, 0.2, 0.3, and 0.4. For each heritability, we created five two-locus penetrance functions according to the method described in [2], and from each penetrance function we generated 100 data sets. Each data set consists of an equal number of diseased (800 cases) and healthy (800 controls) individuals and possesses 1000 SNPs. Of the 1000 SNPs, only two are predictive of disease and the other 998 are generated at random to exhibit no correlation with clinical endpoint, other than by chance alone.

## 5 Experimental Design

To facilitate a fair comparison between the three initialization methods, we ensure that for each replicate the same functions are used to seed all three initial populations. Specifically, for each cell in the solution lattice we choose a random initial function to place in that cell. These initial functions are held constant across the three initialization methods; only the selected attributes differ.

To assess the performance of CES using each initialization method, we report (i) the evolutionary dynamics of the best training accuracy and (ii) the testing accuracy obtained using the best model found by CES. The latter is calculated by applying the best model found during training to another data set generated for that particular penetrance table. Thus, for each heritability, we have 500 independent training and testing pairs. Both training and testing accuracy are calculated using Eq. 1.

## 6 Results and Discussion

In Fig. 2 we depict the evolutionary dynamics of the best training accuracy for the CES using random, enumerative, and expert-knowledge-aware initializers, for the six heritabilities considered in this study. For all heritabilities, the best training accuracy found in the initial population was highest when expert-knowledge-aware initialization was used (in each panel of Fig. 2, compare the height of the symbol types at generation zero). The random and enumerative initializers produced initial populations with nearly identical best training accuracies.

In most cases, the CES improved the training accuracy of the models supplied in the initial population by each of the initialization methods. For example, the insets of Fig. 2 depict the distributions of improvements in training accuracy obtained by the CES using the expert-knowledge aware initializer. The distributions are always bimodal, with one peak at zero and another centered between 0.05 and 0.15. The lower mode indicates that in some cases, the CES is unable to improve upon the best solution provided in the initial population. However, the higher mode indicates that in the majority of cases, some improvement in training accuracy is observed

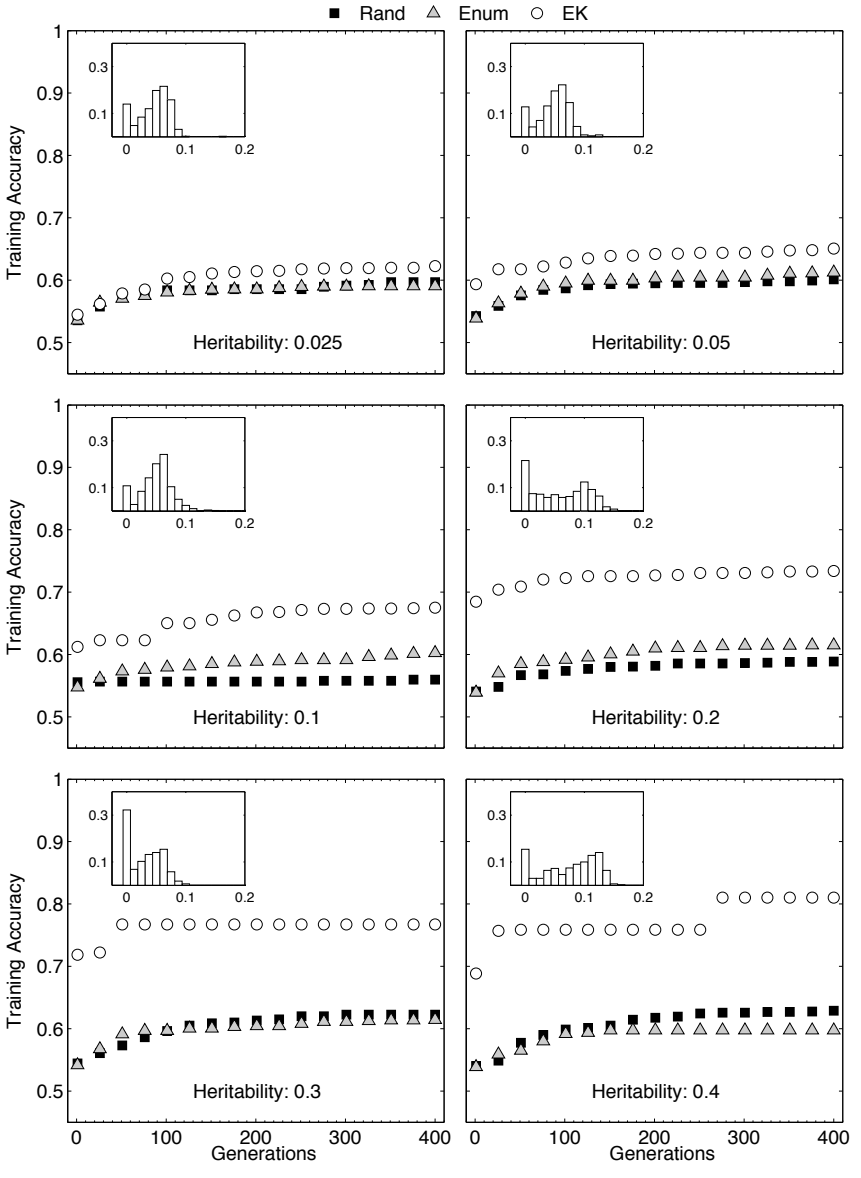



**Fig. 2** Evolutionary dynamics of best training accuracy for CES using random (black squares), enumerative (gray triangles), and expert-knowledge-aware (open circles) initializers for the six heritabilities considered in this study. The data presented in each panel correspond to a single replicate. The insets depict the distributions of improvements in training accuracy for CES with expert-knowledge-aware initialization, measured as the difference between the training accuracy at generation 0 and 400

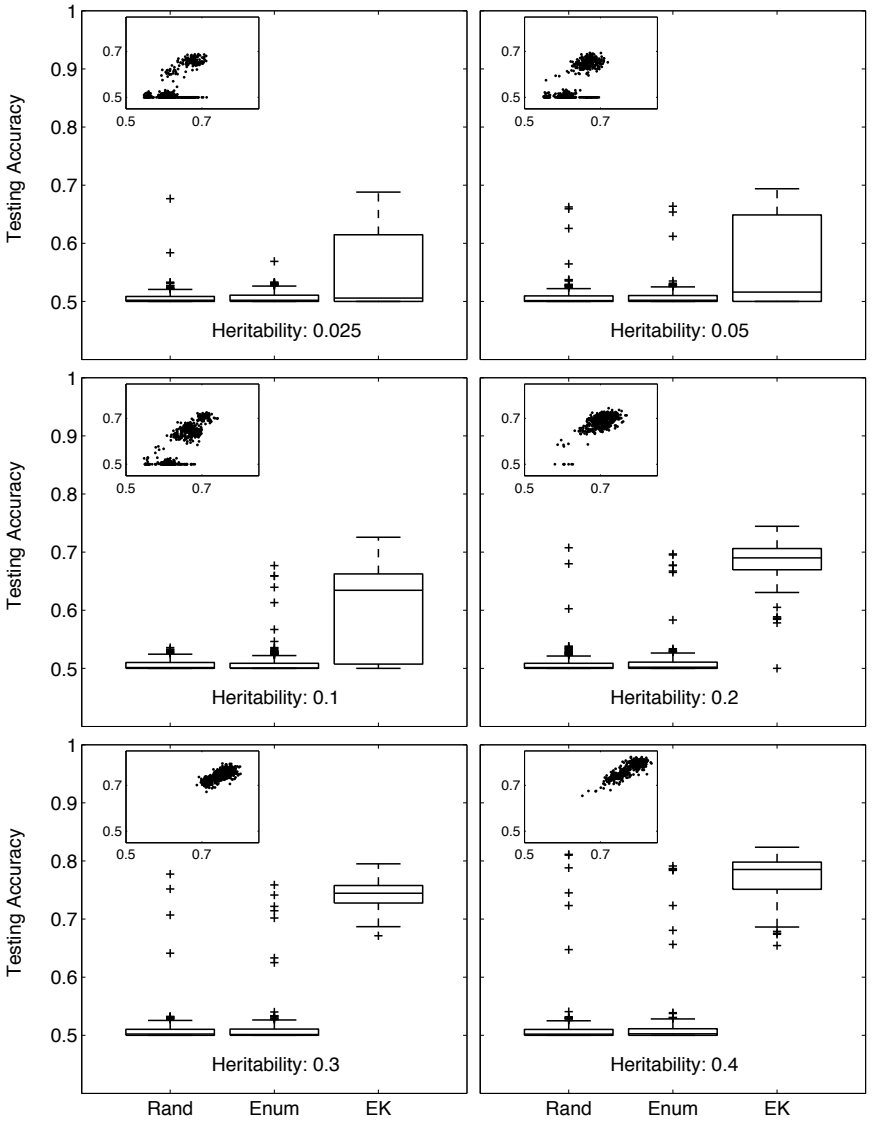



**Fig. 3** Testing accuracy of the best models found by CES using the random (Rand), enumerative (Enum), and expert-knowledge-aware (EK) initialization methods, for the six heritabilities considered in this study. The insets depict the testing accuracy (y-axis) as a function of the training accuracy (x-axis) for CES with expert-knowledge-aware initialization, across the 500 data sets considered for each heritability

using the CES. This indicates that SURF is able to correctly identify SNP linkage and that CES can exploit this information to build an architectural model of genetic predisposition to disease.

Using the expert-knowledge-aware initializer, the best training accuracy found in the initial population generally increased with increasing heritability. In contrast, using the random and enumerative initialization methods, the best training accuracy of the initial population remained approximately constant across heritabilities. These observations stem from the frequency with which the three methods supplied the two target SNPs to the initial population. Of the 500 replicates considered for each heritability, the percentage of trials in which the two interacting SNPs were correctly identified within a single solution by the expert-knowledge-aware initializer increased linearly from 41% at a heritability of 0.025 to 100% at a heritability of 0.2 ($r^2 = 0.99$). For heritabilities greater than or equal to 0.2, the two target SNPs were always identified within a single solution. Using the random and enumerative initializers, less than 1% of all trials contained the two target SNPs in a single solution, a figure that remained consistent across heritabilities.

In Fig. 3, we depict the testing accuracies of the best solutions obtained by the CES, using random, enumerative, and expert-knowledge-aware initialization. For all heritabilities, the testing accuracies of the best solutions found using the expert-knowledge-aware initializer were significantly higher than those obtained using either random or enumerative initialization. Following the trends of the training data (Fig. 2), the testing accuracies obtained with expert-knowledge-aware initialization increased as heritability increased, whereas the testing accuracy of the random and enumerative methods remained consistently low. The insets of Fig. 3 depict the testing accuracy of the best solution found by CES with expert-knowledge-aware initialization, as a function of its training accuracy. For low heritabilities, the data is clustered into two distinct groups, one in which testing accuracy is not correlated with training accuracy and another in which testing accuracy is positively correlated with training accuracy. As heritability increases, the data begin to migrate toward the cluster that exhibits positive correlation between testing and training accuracy, indicating a reduction in overfitting.

## 7 Concluding Remarks

We have investigated the influence of population initialization on the ability of a computational evolution system (CES) to detect epistatically interacting single nucleotide polymorphisms (SNP) in genome-wide association studies (GWAS). Our results demonstrate that the CES finds solutions of higher quality, both in terms of training and testing accuracy, when the population is initialized using an expert knowledge source than when it is not. Specifically, we found that biasing the selection of attributes in the initial population using a machine learning algorithm called Spatially Uniform ReliefF (SURF) [5] is superior to both random and enumerative initialization schemes.

These results complement those presented in [7], where it was shown that expert-knowledge-aware population initialization can improve the classification power of genetic programming for detecting gene-gene interactions in GWAS. Taken together, these results further highlight the critical need for expert knowledge sources in this problem domain [15]. Alternative approaches to incorporating expert knowledge sources, such as their inclusion in fitness assessment, selection, and mutation have also proven valuable [6, 14, 15]. Future work will investigate the combination of these expert-knowledge guided operators with expert-knowledge-aware initialization. Of particular interest is the utilization of alternative sources of expert knowledge, such as the causal information provided by metabolic and proteomic interaction networks. The incorporation of the many available sources of expert knowledge into artificial and computational evolution systems offers the potential to improve our ability to detect and characterize the genetic causes of human disease.

## References

1. Banzhaf, W., Beslon, G., Christensen, S., Foster, J.A., Képès, F., Lefort, V., Miller, J.F., Radman, M., Ramsden, J.J.: From artificial evolution to computational evolution: a research agenda. Nature Reviews Genetics 7, 729–735 (2006)
2. Culverhouse, R., Suarez, B.K., Lin, J., Reich, T.: A perspective on epistasis: limits of models displaying no main effect. American Journal of Human Genetics 70(2), 461–471 (2002)
3. Eppstein, M.J., Payne, J.L., White, B.C., Moore, J.H.: Genomic mining for complex disease traits with 'random chemistry'. Genetic Programming and Evolvable Machines 8, 395–411 (2007)
4. Greene, C.S., Hill, D.P., Moore, J.H.: Environmental noise improves epistasis models of genetic data discovered using a computational evolution system. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1785–1786 (2009)
5. Greene, C.S., Penrod, N.M., Kiralis, J., Moore, J.H.: Spatially uniform ReliefF (SURF) for computationally-efficient filtering of gene-gene interactions. BioData Mining 2(5) (2009)
6. Greene, C.S., White, B.C., Moore, J.H.: An expert knowledge-guided mutation operator for genome-wide genetic analysis using genetic programming. In: Rajapakse, J.C., Schmidt, B., Volkert, L.G. (eds.) PRIB 2007. LNCS (LNBI), vol. 4774, pp. 30–40. Springer, Heidelberg (2007)
7. Greene, C.S., White, B.C., Moore, J.H.: Sensible initialization using expert knowledge for genome-wide analysis of epistasis using genetic programming. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1289–1296 (2009)
8. Hirschhorn, J.N., Daly, M.J.: Genome-wide association studies for common diseases and complex traits. Nature Reviews Genetics 6, 95–108 (2005)
9. Kononenko, I.: Estimating attributes: analysis and extensions of RELIEF. In: Bergadano, F., De Raedt, L. (eds.) ECML 1994. LNCS, vol. 784, pp. 171–182. Springer, Heidelberg (1994)
10. Lucek, P.R., Ott, J.: Neural network analysis of complex traits. Genetic Epidemiology 14, 1101–1106 (1997)

11. Moore, J.H., Greene, C.S., Andrews, P.C., White, B.C.: Does complexity matter? Artificial evolution, computational evolution, and the genetic analysis of epistasis in common human diseases. In: Genetic Programming Theory and Practice VI, ch. 9. Springer, Heidelberg (2009)
12. Moore, J.H., Parker, J.S., Olsen, N.J., Aune, T.M.: Symbolic discriminant analysis of microarray data in autoimmune disease. Genetic Epidemiology 23, 57–69 (2002)
13. Moore, J.H., Ritchie, M.D.: The challenges of whole-genome approaches to common diseases. Journal of the American Medical Association 291(13), 1642–1643 (2004)
14. Moore, J.H., White, B.C.: Exploiting expert knowledge in genetic programming for genome-wide genetic analysis. 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. 969–977. Springer, Heidelberg (2006)
15. Moore, J.H., White, B.C.: Genome-wide genetic analysis using genetic programming: The critical need for expert knowledge. In: Genetic Programming Theory and Practice IV, ch. 2. Springer, Heidelberg (2007)
16. Moore, J.H., White, B.C.: Tuning ReliefF for genome-wide genetic analysis. In: Marchiori, E., Moore, J.H., Rajapakse, J.C. (eds.) EvoBIO 2007. LNCS, vol. 4447, pp. 166–175. Springer, Heidelberg (2007)
17. Ritchie, M.D., Hahn, L.W., Roodi, N., Bailey, L.R., Dupont, W.D., Parl, F.F., Moore, J.H.: Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. American Journal of Human Genetics 69(1), 138–147 (2001)
18. Wagner, A.: Robustness and evolvability in living systems. Princeton University Press, Princeton (2007)

# Estimating Optimal Stopping Rules in the Multiple Best Choice Problem with Minimal Summarized Rank via the Cross-Entropy Method

T.V. Polushina

**Abstract.** The best choice problem is an important class of the theory of optimal stopping rules. In this article, we present the Cross-Entropy method for solving the multiple best choice problem with the minimal expected ranks of selected objects. We also compare computation results by Cross-Entropy method with results by the genetic algorithm. Computational results showed that the Cross-Entropy method is producing high-quality solution.

## 1 Introduction

The best choice problem is an important class of the theory of optimal stopping rules. It has been studied by many authors: Chow, Robbins, and Siegmund [2], Dynkin and Yushkevich [4], Gilbert and Mosteller [6], Shiryaev [13].

In this chapter we consider the multiple best choice problem [9], [10]. We have a known number $N$ of objects numbered $1,2,\ldots,N$, so that, say, an object numbered 1 is classified as "the best", ..., and an object numbered $N$ is classified as "the worst". It is assumed that the objects arrive one by one in random order, i.e all $N!$ permutations are equiprobable. It is clear from comparing any two of these objects which one is better, although their actual number still remain unknown. After having known each sequential object, we either accept this object (and then a choice of one object is made), or reject it and continue observation (it is impossible to return to the rejected object). The object is to find a stopping rule which minimizes the expected absolute rank of the individual selected.

We can use this model to analyse some behavioral ecology problems such as sequential mate choice or optimal choice of the place of foraging. Indeed, in some species, active individuals (generally, females) sequentially mate with different passive individuals (usually males) within a single mating period (see, e.g., Gabor and

T.V. Polushina
Mari State University, Lenin sq. 1, Yoshkar-Ola, Russia
e-mail: tvpolushina@inbox.ru

Y.-p. Chen (Ed.): Exploitation of Linkage Learning, ALO 3, pp. 227–241.
springerlink.com 

Halliday [5], Pitcher et al. [11]). Note also that an individual can sequentially choose more than one place to forage. So we can consider the random variable as quality of item (potential mate or place of foraging) which appears at time $t$.

For one choice problem the minimal expected rank for the rank minimization problem tends to the value $\prod_{j=1}^{\infty}(1+\frac{2}{j})^{\frac{1}{1+j}} \approx 3.8695$ [3]. The model considered in [7] is similar to the one choice problem. But instead of being a fixed integer $N$, the total number of individuals is a strictly positive, integer-valued, bounded random variable. The author studied the asymptotic behavior of the minimal expected rank. Bruss and Ferguson [1] considered this problem in full information setting where the decision is based on the actual values associated with the applicants, assumed to be independent and identically distributed from a known distribution. Tamaki considered the best choice problem which allows the applicant to refuse an offer of acceptance with probability $1-p, 0<p<1$ [15].

The rest of the chapter is organized as follows. Section 2 introduces the multiple best problem with minimal summarized rank. In Section 3 we describe the cross-entropy method. Section 4 discusses cross-entropy method for the multiple best problem with minimal summarized rank. Section 5 presents the experiment results of the cross-entropy method for the problem. In Section 6 we explain the genetic algorithm. Section 7 discusses numeric results of the genetic algorithm for the problem. Finally, the conclusions are given.

## 2 The Multiple Best Choice Problem with Minimal Summarized Rank

Let we have $N$ objects, which are ordered on quality. At time $n$ we can compare current object with all previous objects, but we nothing know about quality remaining $N-n$ objects. After getting acquainted with $a_n$ we can it or accept (and then the choice of one object is made), or reject and continue observation (we can't return to rejected object).

Let $x_i$ be an absolute rank of selected object, i.e. $x_i = 1+$ number of objects from $(a_1, a_2, \ldots, a_N) < a_i$. The objective is to find optimal procedure such that the expected gain $\mathbf{E}(x_{\tau_1} + \ldots + x_{\tau_k})$, $k \geq 2$ is minimal.

Denote by $(a_1, a_2, \ldots, a_N)$ any permutation of numbers $(1, 2, \ldots, N)$, 1 corresponds to the best object, $N$ corresponds to the worst one. All $N!$ permutations being equally likely. For any $i = 1, 2, \ldots, N$ let $y_i =$ number of terms $a_1, a_2, \ldots, a_i$ which are $\leq a_i$, and $y_i$ is called the relative rank of the $i$th object. As $y_1, y_2, \ldots, y_N$ are independent, and

$$\mathbf{P}(y_i = j) = 1/i \quad (j = 1, 2, \ldots, i),$$
$$\mathbf{P}(x_i = k \mid y_1 = l_1, \ldots, y_{i-1} = l_{i-1}, y_i = j) = \mathbf{P}(x_i = k \mid y_i = j)$$
$$= \frac{C_{k-1}^{j-1} \cdot C_{N-k}^{i-j}}{C_N^i}.$$

Then

$$\mathbf{E}(x_i \mid y_i = j) = \sum_{k=1}^{N} k\mathbf{P}(x_i = k \mid y_i = j) = \frac{N+1}{i+1} j.$$

By definition, put $v = \inf_\tau \mathbf{E}(x_{\tau_1} + \ldots + x_{\tau_k}), \quad \tau = (\tau_1, \ldots, \tau_k)$. We want to find the optimal procedure $\tau^* = (\tau_1^*, \ldots, \tau_k^*)$ and the value of the game $v$.

Let $\mathscr{F}_{(m)_i}$ be the $\sigma$-algebra, generated by $(y_1, y_2, \ldots, y_{m_i})$. If we suppose

$$Z_{(m)_k} = \mathbf{E}(x_{\tau_1} + \ldots + x_{\tau_k} \mid \mathscr{F}_{(m)_k}),$$

then

$$v = \inf_\tau \mathbf{E} Z_\tau, \quad \tau = (\tau_1, \ldots, \tau_k).$$

So we reduce our problem to the problem of multiple stopping of the sequence $Z_{(m)_k}$.

As was shown in [9], [10], the solution of this problem is the following optimal strategy: there exist integer vectors

$$\begin{aligned}
&\delta^{(k)} = (\delta_1^{(k)}, \ldots, \delta_{N-k+1}^{(k)}),\\
&0 \le \delta_1^{(k)} \le \ldots \le \delta_{N-k}^{(k)} < \delta_{N-k+1}^{(k)} = N,\\
&\quad \vdots\\
&\delta^{(2)} = (\delta_{k-1}^{(2)}, \ldots, \delta_{N-1}^{(2)}),\\
&0 \le \delta_{k-1}^{(2)} \le \ldots \le \delta_{N-2}^{(2)} < \delta_{N-1}^{(2)} = N,\\
&\delta^{(1)} = (\delta_k^{(1)}, \ldots, \delta_N^{(1)}),\\
&0 \le \delta_k^{(1)} \le \ldots \le \delta_{n-1}^{(1)} < \delta_N^{(1)} = N,\\
&\delta_j^{(i_1)} \le \delta_j^{(i_2)}, \quad 1 \le i_1 < i_2 \le k,\\
&k - i_1 + 1 \le j \le N - i_2 + 1
\end{aligned} \tag{1}$$

such that

$$\begin{aligned}
&\tau_1^* = \min\{m_1 : y_{m_1} \le \delta_{m_1}^{(k)}\},\\
&\tau_i^* = \min\{m_i > m_{i-1} : y_{m_i} \le \delta_{m_i}^{(k-i+1)}\},
\end{aligned}$$

on the set $F_{i-1} = \{\omega : \tau_1^* = m_1, \ldots, \tau_{i-1}^* = m_{i-1}\}$, $i = 2, \ldots, k$, $F_0 = \Omega$.

For small $N$ we can obtain exact values. Table 1 displays the vectors $\delta^{(k)}, \ldots, \delta^{(1)}$, and the values $v$, where $(l_5, l_6), (l_7, l_8) \in \{(1,2), (1,3), (2,2), (2,3)\}$ [14].

If $N = 5$, $k = 2$ then the value of the game $v = 4.600$. We can select any optimal rule indicated in the proper cell of Table 1. Specifically, $\delta^{(2)} = (0,1,2,5), \delta^{(1)} = (0,1,2,5)$. We have the following optimal rule:

$$\tau_1^* = \min\{m_1 \ge 1 : y_{m_1} \le \delta_{m_1}^{(2)}\},$$

**Table 1** The vectors $\delta^{(k)},\dots,\delta^{(1)}$, and the values $v$

| $N\backslash k$ | 2 | 3 | 4 |
|---|---|---|---|
| 3 | $\delta^{(2)}=(1,3)$<br>$\delta^{(1)}=(1,3)$<br>$v=3.6667$ | | |
| 4 | $\delta^{(2)}=(0,1,4)$<br>$\delta^{(1)}=(0,l_1,4)$<br>or<br>$\delta^{(2)}=(1,l_2,4)$<br>$\delta^{(1)}=(1,l_3,4)$<br>$l_1,l_2,l_3\in\{1,2\}$<br>$v=4.3750$ | $\delta^{(3)}=(1,4)$<br>$\delta^{(2)}=(1,4)$<br>$\delta^{(1)}=(l_4,4)$<br>$l_4\in\{1,2\}$<br>$v=6.8750$ | |
| 5 | $\delta^{(2)}=(0,1,2,5)$<br>$\delta^{(1)}=(0,1,2,5)$<br>$v=4.6000$ | $\delta^{(3)}=(1,2,5)$<br>$\delta^{(2)}=(1,2,5)$<br>$\delta^{(1)}=(1,2,5)$<br>$v=7.6000$ | $\delta^{(4)}=(1,5)$<br>$\delta^{(3)}=(1,5)$<br>$\delta^{(2)}=(2,5)$<br>$\delta^{(1)}=(2,5)$<br>$v=11.0500$ |
| 6 | $\delta^{(2)}=(0,1,l_5,l_6,6)$<br>$\delta^{(1)}=(0,1,l_7,l_8,6)$<br>$v=4.9583$ | $\delta^{(3)}=(0,1,2,6)$<br>$\delta^{(2)}=(0,l_5,l_6,6)$<br>$\delta^{(1)}=(0,l_7,l_8,6)$<br>or<br>$\delta^{(3)}=(1,2,3,6)$<br>$\delta^{(2)}=(1,l_5,l_6,6)$<br>$\delta^{(1)}=(1,l_7,l_8,6)$<br>$v=8.4583$ | $\delta^{(4)}=(1,2,6)$<br>$\delta^{(3)}=(1,2,6)$<br>$\delta^{(2)}=(l_5,l_6,6)$<br>$\delta^{(1)}=(l_7,l_8,6)$<br>$v=11.9583$ |

$$\tau_2^*=\min\{m_2>m_1: y_{m_2}\le\delta_{m_2}^{(1)}\}$$

If we observe the following sequence $x_1,\dots,x_5$: 3, 2, 5, 1, 4; then $y_1,\dots,y_5$: 1, 1, 3, 1, 4. So we get $m_1=2$ ($y_2=1$, $\delta_2^{(2)}=1$), $m_2=4$ ($y_4=1$, $\delta_4^{(1)}=2$). Consequently we obtain two best objects with summarized rank $2+1=3$.

## 3 Cross-Entropy Method

It is difficult to obtain the set $\delta^{(1)},\dots,\delta^{(k)}$ and the value $v$ with backward induction, but we can get them by simulation. So we consider the following maximization problem

$$\max_{x\in\mathscr{X}}\mathbf{E}\widehat{S}(x,R), \tag{2}$$

where $\mathscr{X}=\{x=(x^{(1)},\dots,x^{(k)}):\text{conditions (1) are hold}\}$, $R=(R_1,\dots,R_N)$ is a random permutation of numbers $1,2,\dots,N$, $\widehat{S}(x)$ is an unbiased estimator of $\mathbf{E}\widehat{S}(x,R)$

$$\widehat{S}(x) = \frac{1}{N_1}\sum_{n=1}^{N_1}(R_{n\tau_1} + \ldots + R_{n\tau_k}),$$

where $(R_{n1},\ldots,R_{nN})$ is the $n$th copy of random permutation $R$, $N_1$ is simulation parameter, integer number.

Therefore we can apply the cross-entropy (CE) algorithm for noisy optimization [12]. The starting point in the methodology of the CE method is to associate an estimation problem with optimization problem. To this end we define a collection of indicator function $\{I_{\{S(x)\geq\gamma\}}\}$ on $\mathscr{X}$ for various levels $\gamma \in R$. Let $\{f(\cdot,u)\}$ be a family of pdfs on $\mathscr{X}$, parameterized by a real-valued parameter $u$. For a certain $u$ we associate with (2) the problem of estimating the number

$$l(\gamma) = \mathbf{P}_u(S(X) \geq \gamma) = \sum_x I_{\{S(x)\geq\gamma\}} f(x,u) = \mathbf{E}_u I_{\{S(X)\geq\gamma\}},$$

where $\mathbf{P}_u$ is the probability measure under which the random state $X$ has pdf $f(\cdot;u)$ and $\gamma$ is a known or unknown parameter. Typically estimation of $l$ is a non-trivial problem. The CE method solves this efficiently by making adaptive changes to the probability density function according to the Kullback-Leibler CE, thus creating a sequence $f(\cdot,u_0)$, $f(\cdot,u_1)$, $f(\cdot,u_2)$,...,$f(\cdot,u^*)$ corresponding to the degenerate density at an optimal point. In fact the CE method generates a sequence of tuples $\{(\gamma_t,u_t)\}$, which converges quickly to a small neighborhood of the optimal tuple $(\gamma^*,u^*)$. More specifically, we initialize by setting $u_0$, choosing a not very small quantity $\rho$, and than we proceed as follows:

1. **Adaptive updating of $\gamma_t$.** For a fixed $u_{t-1}$, let $\gamma_t$ be a $(1-\rho)$-quantile of $\widehat{S}(X)$ under $u_{t-1}$. A simple estimator $\widehat{\gamma}_t$ of $\gamma_t$ is

   $$\widehat{\gamma}_t = \widehat{S}_{(\lceil(1-\rho)N_2\rceil)},$$

   where, for a random sample $X_1,\ldots,X_{N_2}$ from $f(\cdot;u_{t-1})$, $\widehat{S}_{(i)}$ is the $i$th order statistic of the performances $\widehat{S}(X_1),\ldots,\widehat{S}(X_{N_2})$.
2. **Adaptive updating of $u_t$.** For fixed $\gamma_t$ and $u_{t-1}$, derive $u_t$ from the solution of the CE program

   $$\max_u D(u) = \max_u \mathbf{E}_{u_{t-1}} I_{\{\widehat{S}(X)\geq\gamma_t\}} \ln f(X;u). \tag{3}$$

   The stochastic counterpart of (3) is the following: for fixed $\widehat{\gamma}_t$ and $\widehat{u}_{t-1}$, derive $\widehat{u}_t$ from the program

   $$\max_u \widehat{D}(u) = \max_u \frac{1}{N_2}\sum_{n=1}^{N_2} I_{\{\widehat{S}(X_n)\geq\widehat{\gamma}_t\}} \ln f(X_n;u). \tag{4}$$

Instead of updating the parameter vector $v$ we use the following smoothed version

$$\widehat{u}_t = \alpha\widehat{u}_t + (1-\alpha)\widehat{u}_{t-1}, i = 1,\ldots,n,$$

where $\alpha$ is called the smoothing parameter, with $0.7 < \alpha \leq 1$. Clearly, for $\alpha = 1$ we have our original updating rule.

To complete specification of the algorithm, one must supply values for $N_2$ and $\rho$, initial parameters $u_0$, and a stopping criterion.

We use stopping criterion from [12]. To identify $T$, we consider the following moving average-process

$$B_t(K) = \frac{1}{K} \sum_{s=t-K+1}^{t} \widehat{\gamma}_s, t = s, s = 1, \ldots, s \geq K,$$

where $K$ is fixed;

$$C_t(K) = \frac{\frac{1}{K-1}\{\sum_{s=t-K+1}^{t}(\widehat{\gamma}_s - B_t(K))^2\}}{B_t(K)^2}.$$

Next define

$$C_t^-(K,R) = \min_{j=1,\ldots,R} C_{t+j}(K)$$

and

$$C_t^+(K,R) = \max_{j=1,\ldots,R} C_{t+j}(K),$$

respectively, where R is fixed.

We define stopping criterion as

$$T = \min\{t : \frac{C_t^+(K,R) - C_t^-(K,R)}{C_t^-(K,R)} \leq \varepsilon\}, \tag{5}$$

where $K$ and $R$ are fixed and $\varepsilon$ is a small number, say $\varepsilon \leq 0.01$.

## 4 The Cross-Entropy Method for the Problem

We solve maximization problem

$$\max_{x \in \mathscr{X}} \mathbf{E}\widehat{S}(x,R), \tag{6}$$

where $\mathscr{X} = \{x = (x^{(1)}, \ldots, x^{(k)}) : \text{conditions (1) are hold}\}$, $R = (R_1, \ldots, R_N)$ is a random permutation of numbers $1, 2, \ldots, N$, $\widehat{S}(x)$ is an unbiased estimator of $\mathbf{E}\widehat{S}(x,R)$

$$\widehat{S}(x) = \frac{1}{N_1} \sum_{n=1}^{N_1} (R_{n\tau_1} + \ldots + R_{n\tau_k}). \tag{7}$$

As in [14] we consider a 3-dimensional matrix of parameters $u = \{u_{ijl}\}$

$$u_{ijl} = \mathbf{P}\{X_j^{(i)} = l\},\ i = 1, \ldots, k;$$
$$j = k - i + 1, \ldots, N - i + 1;\ l = 0, \ldots, N - 1.$$

It follows easily that

$$f(x_j^{(i)};u) = \sum_{l=0}^{N-1} u_{ijl} I_{\{x_j^{(i)}=l\}}.$$

We can see that

$$\widehat{u}_{ijl}^{(t)} = \frac{\sum_{n=1}^{N_2} I_{\{\widehat{S}(X_n)\geq\widehat{\gamma}_t\}} W_{nij}^{(t-1)} I_{\{X_{nij}=l\}}}{\sum_{n=1}^{N_2} I_{\{\widehat{S}(X_n)\geq\widehat{\gamma}_t\}} W_{nij}^{(t-1)}},$$

$$W_{nij}^{(t-1)} = \frac{\widehat{u}_{ijX_{nij}}^{(0)}}{\widehat{u}_{ijX_{nij}}^{(t-1)}}, \tag{8}$$

where $X_n = \{X_{nij}\}$, $X_{nij}$ is a random variable from $f(x_j^{(i)};\widehat{u}_{t-1})$. Formula (4) becomes (8).

## 5 Numeric Results

In this section we present numerical results for $N = 10$ and $k = 2$. For different set of $(\delta^{(1)},\delta^{(2)})$ we calculate $v$, which is decreased from 11 to 6 (figure 1). A Monte Carlo technique ($N_1 = 50000$) is applied for finding the gain $v$ for 5000 different sets $(\delta^{(1)},\delta^{(2)})$. Then we show that the CE algorithm allows to find $v = 5.8638$, and the optimal sets $(\delta^{(1)},\delta^{(2)})$. While the minimal $v$ by Monte Carlo technique is about 6.

We use the CE method with simulation parameters $\rho = 0.1$, $\alpha = 0.7$, $N_2 = 200$, $N_1 = 100$, $N_{last} = 5000$, $K = 6$, $R = 3$, $\varepsilon = 0.01$. We run the algorithm 100 times. Figure 2 shows the histogram of solutions that were obtained by the CE method. Note that the alogithm finds the optimal sets $(\delta^{(1)},\delta^{(2)})$ and neighbouring optimal sets.

Figures 3, 4 show how the CE method works. Initially we have uniform distribution, that is, $\widehat{u}_{ijl}^{(0)} = 0.1$ for $i = 1,2$; $j = 3-i,\ldots,11-i$; $l = 0,\ldots,9$. For example, $\widehat{u}_{120}^{(t)}$ is considered. It is situated in the first subdiagram in position 0 (figure 3). Enlarged diagram $\widehat{u}_{120}^{(t)}$ is showed on figure 5. At first $\widehat{u}_{120}^{(0)} = 0.1$. Then with iterations $t$ $\widehat{u}_{120}^{(t)}$ increases and amounts to 1. This implies that $\mathbf{P}\{X_2^{(1)} = 0\} = 1$. Thus in set $\delta^{(1)}$ in the first position 0 is situated. Similarly, zeros are the second and the third elements of set $\delta^{(1)}$.

Simulation parameters are $\rho = 0.1$, $\alpha = 0.7$, $N_2 = 200$, $N_1 = 100$, $N_{last} = 500$, $K = 6$, $R = 3$, $\varepsilon = 0.01$, repeat 5 times. By simulation we obtained $\delta^{(1)} = (0,0,0,1,\ 1,2,3,4,10)$, $\delta^{(2)} = (0,0,1,1,2,2,3,5,10)$. Nikolaev M.L. [10] shows that for $N = 10$ theoretical optimal are $\delta^{(2)} = (0,0,1,2,2,3,4,5,10)$, $\delta^{(1)} = (0,0,1,1,2,2,3,5,10)$.

We also compare $v$ with theoretical $(\delta^{(1)},\delta^{(2)})$ and $v$ with modelling $(\delta^{(1)},\delta^{(2)})$ (table 2). For this table $N_1 = 50000$. We can see that $v$ with theoretical $(\delta^{(1)},\delta^{(2)})$

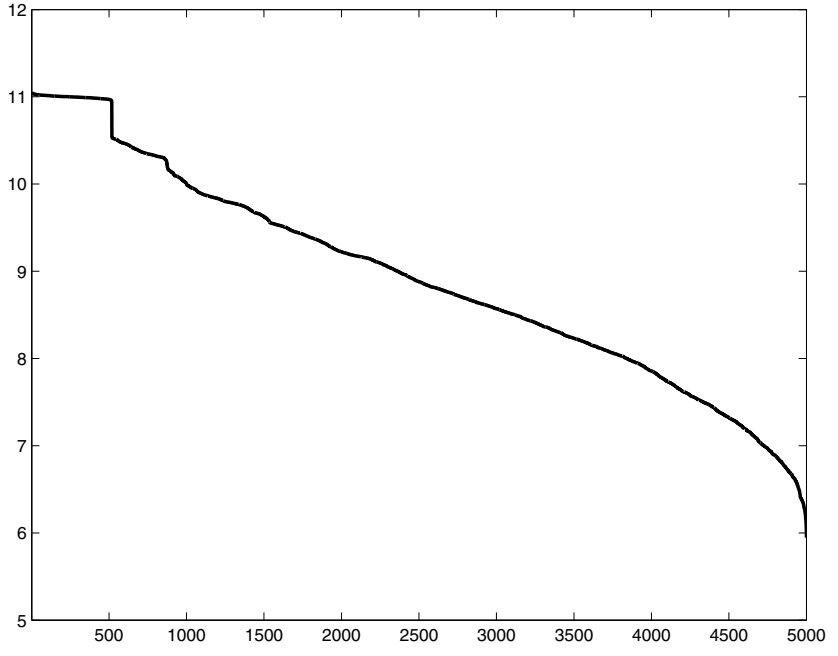



**Fig. 1** Gain $v$ for different $(\delta^{(1)}, \delta^{(2)})$

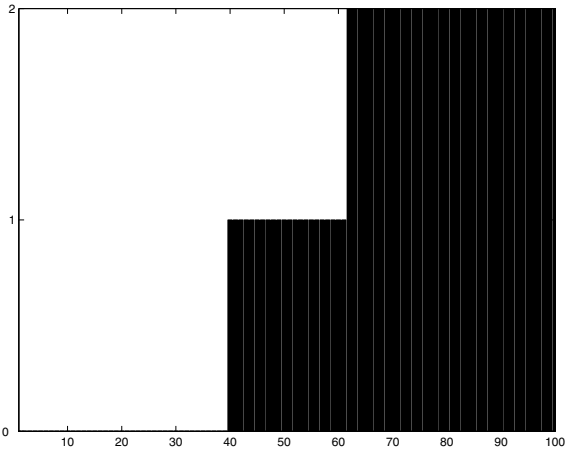



**Fig. 2** Bar graph for $N = 10$

is differ from $v$ with modelling $(\delta^{(1)}, \delta^{(2)})$ slightly. Difference connects that $N_1$ and $N_{last}$ are not big.

Then we use the CE method for different $N$, using the following parameters $N_1 = 100$, $N_2 = 200$, $N_{last} = 500$, $\rho = 0.1$, $\alpha = 0.7$, $\varepsilon = 0.1$, $K = 6$, $R = 3$. This method

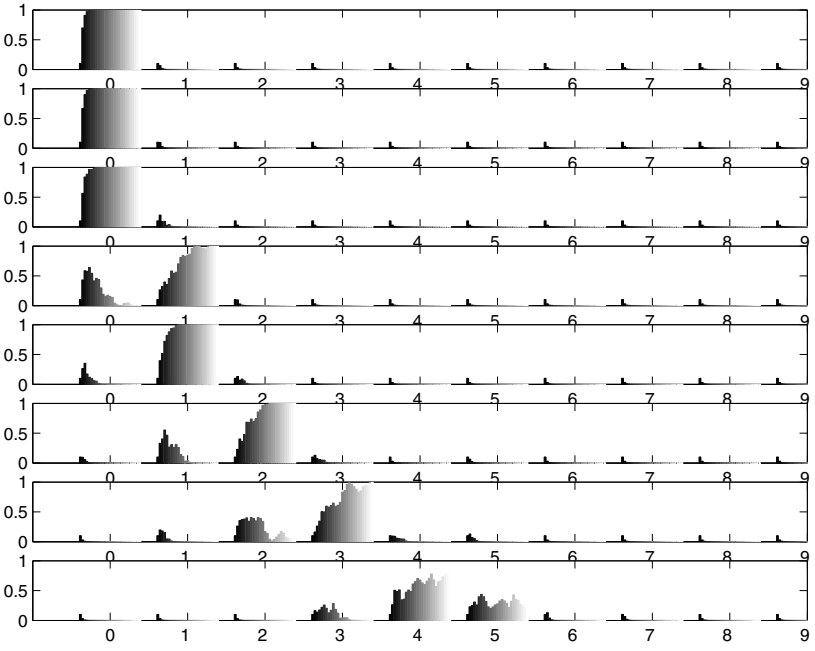


**Fig. 3** Values $\widehat{u}_{1jl}^{(t)}$ for $N = 10$

**Table 2** $v$ with theoretical $(\delta^{(1)}, \delta^{(2)})$ and with modelling $(\delta^{(1)}, \delta^{(2)})$

| | $\delta^{(1)}$ | $\delta^{(2)}$ | $v$ |
|---|---|---|---|
| theoretical | (0,0,1,1,2,2,3,5,10) | (0,0,1,2,2,3,4,5,10) | 5.9124 |
| modelling | (0,0,0,1,1,2,3,4,10) | (0,0,1,1,2,2,3,5,10) | 5.8638 |

**Table 3** Value mean, maximum and minimum $v$ for different $N$

| $N$ | mean $v$ | max $v$ | min $v$ | standard error | CPU time |
|---|---|---|---|---|---|
| 5 | 4.5908 | 4.6060 | 4.5800 | 0.0074 | 10.7470 |
| 10 | 5.8698 | 5.7880 | 5.6760 | 0.0458 | 15.6188 |
| 15 | 6.6448 | 6.7620 | 6.5320 | 0.0927 | 26.2562 |
| 20 | 7.5668 | 7.8260 | 7.3580 | 0.2793 | 35.1936 |

has been implemented in MatLab on a PC (AMD Athlon 64 2.01 GHz). Using this algorithm parameters after 5 repetitions, we obtain the results summarized in table 3.

Figure 6 shows values $v$ with standard error and for different $N$, $N = 3, \ldots, 20$. Modelling parameters are $\rho = 0.1$, $\alpha = 0.7$, $N_2 = 100$, $N_1 = 100$, $N_{last} = 500$. $\varepsilon = 0.1$, $K = 6$, $R = 3$. Method was repeated 5 times for each $N$. Nikolaev M.L. shows

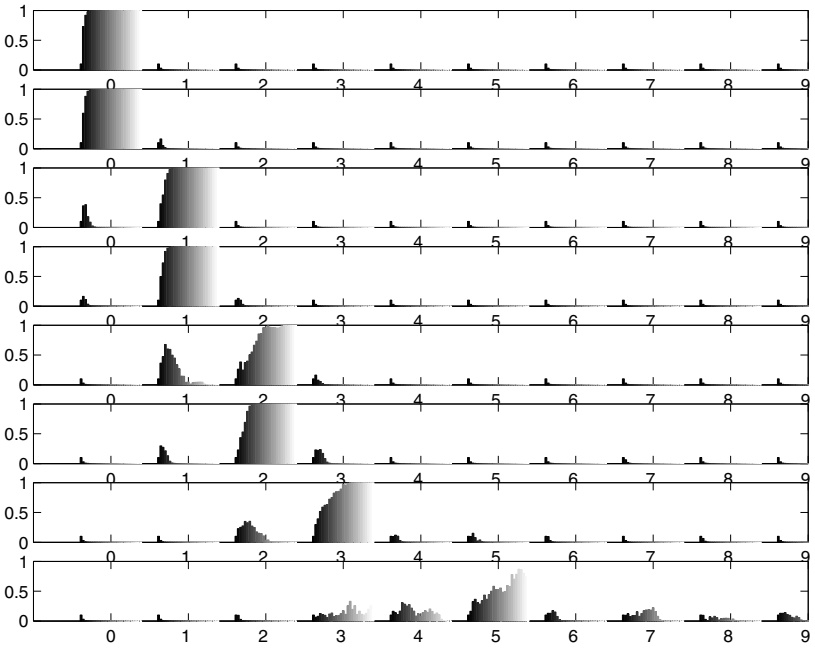


**Fig. 4** Values $\widehat{u}_{2jl}^{(t)}$ for $N = 10$

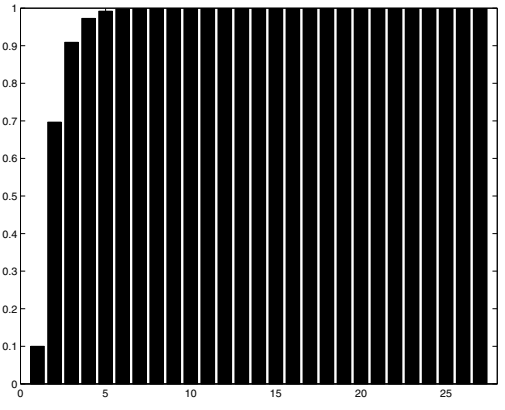


**Fig. 5** Values $\widehat{u}_{120}^{(t)}$ for $N = 10$

that asymptotic $v$ is 7.739 [10]. We can see that $v$ is bigger than the expected value for big $N$. It arises from $N! >> N_1, N_2, N_{last}$, which are used for simulation.

Notwithstanding $N_1, N_2, N_{last}$ are much smaller than $N!$ the cross-entropy method finds optimal stopping rules that are nearly optimal.

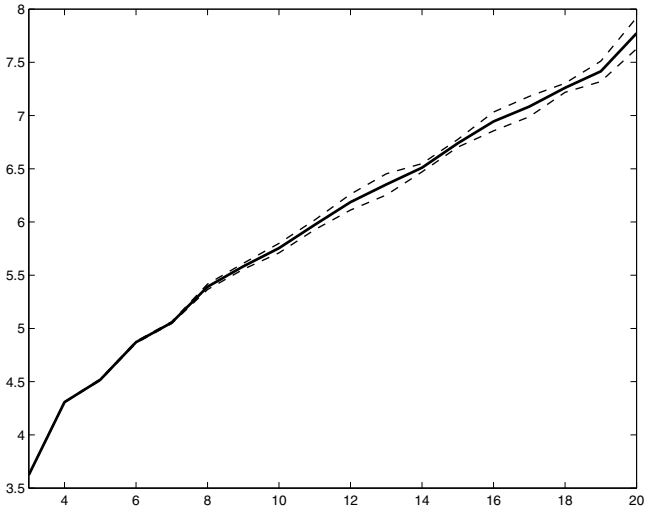



**Fig. 6** Gain $v$ with standard error for $N = 3, \ldots, 20$

## 6 Genetic Algorithm

In this section, the Genetic Algorithm (GA) is considered. The GA is a stochastic search technique. Choosing an appropriate genotype or representation to encode the important features of the problem to be solved is the first step in applying the GA to any problem. The fitness function is defined over the genetic representation and measures the quality of the represented solution.

The GA can develop a population of potential good solution by applying the genetic operators, and finally find a good solution of the problem. The individuals of the population are called "chromosomes". In genetic operators, crossover and mutation are important operators which influence the behavior of GA. The purpose of crossover consists in the combination of useful string segments from different chromosomes to form new, hopefully better performing offspring. Crossover is the process in which the chromosomes are mixed and matched in a random fashion to produce a pair of new chromosomes (offspring). The purpose of mutation is give population diversity to the offspring to selecting a gene at random with a mutation rate and perturbing its value. Mutation operator is the process used to rearrange the structure of the chromosomes to produce a new one.

Following Wang, Okazaki [16], we propose an improved GA by modifying the crossover and mutation behavior. Let $N_p$ is the population size and $N_c$ is the current number of children generated. First $N_c$ is set to zero. Then $(N_p - N_c)/2$ pairs of parent chromosomes are randomly selected, and the difference-degree for every pair of parent chromosomes are calculated. The difference-degree $d_i$ of $i$ parent pair is defined as follow: $d_i = \frac{N_d}{N_g}$, where $N_g$ is the size of chromosome, and $N_d$ is the number of different genes between the two parent chromosomes. A new parameter called setting difference-degree $D_s$ is introduced. If $d_i$ is larger than the $D_s$, then

the crossover is applied with 100% certainty on the parent pair to generate two children. After crossover, the total number of children generated is calculated. If the total number $N_c$ is found to be smaller than the population size $N_p$, than mutation is performed with 100% certainty on parent pairs chromosomes with $d_i$ less than the $D_s$. The above procedure is performed in a loop until the total number of children is equal to the population size.

The genetic search process is described as follows:

1. GA stars with an initial set of random solutions for the problem under consideration. This set of solutions is known as the population
2. Evaluate the fitness of each individual in the population
3. Repeat until termination:
   a. Select best-ranking individuals to reproduce
   b. Breed new generation through crossover and mutation and give birth to offspring
   c. Evaluate the individual fitnesses of the offspring
   d. Replace worst ranked part of population with offspring [8].

The technique of GAs requires a string representation scheme (chromosomes). In this chapter we put that each element of sets $(\delta^{(1)}, \ldots, \delta^{(k)})$ correspond to a locus of chromosome. Than we solve a combinatorial optimization problem by the GA. So we consider the problem (2). The fitness function is calculated from (7). For population creation we use vector of parameters $u = \{u_{ijl}\}$, same as in the CE method.

Than for initial population GA is applied, and we find the first approximation to problem (2). The same as the CE method instead of updating the parameter vector we use the following smoothed version

$$u_t = \alpha u_t + (1 - \alpha) u_{t-1}, i = 1, \ldots, n.$$

The selection operator is applied to select parent chromosomes from the population. A Monte Carlo selection technique is applied. A parent selection procedure functions as:

1. Calculate the fitness of all population members using (7)
2. Return the first population member whose fitness is among the best $fit \cdot N_1\%$ members of population
3. Repeat step 2 for the second population member and check that the new selected member is not the same as the first member; and so on.

Fitness value $fit$ is a given arbitrary constant.

The selected chromosomes to crossover will be crossed to produce two offspring chromosomes by using crossover operator. Crossover operator is described as follows. Let a pair of parent chromosomes $(P_1, P_2)$. Select two random number to be aligned to the parents. The genes are exchanged so that portion of genetic codes from $P_1$ is transferred to $P_2$, and conversely. If the chromosome has large size genes, the cutting section is differing from small to large, which reflects the flexibility of the approach.

The mutation operator is used to rearrange the structure of a chromosome. The swap mutation is used, which is simply selecting two genes at random and swapping their contents. Mutation helps to increase the searching power. In order to explain the need of mutation. Consider the case where reproduction or crossover may not produce a good solution to a problem. During the creation of a generation it is possible that the entire population of strings is missing a vital gene of information that is important for determining the correct or the most nearly optimum solution. In that case mutation becomes very important.

This generational process is repeated until a termination condition has been reached. We use stopping criterion (5).

## 7 Numeric Results of GA Process

We also use the GA for different $N$ and parameters for simulation are $N_2 = 100$, $N_1 = 500$, $fit = 0.25$, $\alpha = 0.95$, $\varepsilon = 0.01$. $K = 6$, $R = 3$. Using this algorithm parameters after 5 repetitions, we obtain the results summarized in Table 4.

**Table 4** Minimum, maximum, mean value $v$ and standard error for different $N$ by GA method

| $N$ | mean $v$ | max $v$ | min $v$ | standard error | CPU time |
|---|---|---|---|---|---|
| 5 | 4.5023 | 4.8470 | 4.3900 | 0.0702 | 1.0318 |
| 10 | 5.7726 | 5.8910 | 5.5130 | 0.1096 | 8.2614 |
| 15 | 6.6572 | 6.9820 | 6.1140 | 0.2084 | 12.6725 |
| 20 | 7.8912 | 8.0160 | 7.1370 | 0.7264 | 17.9326 |

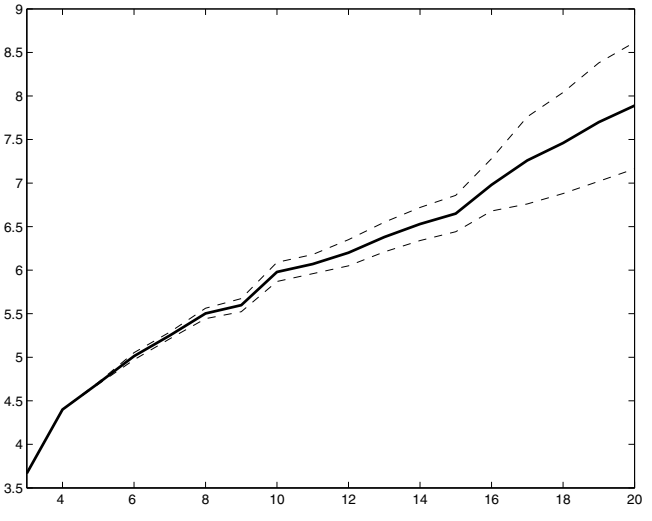



**Fig. 7** Gain $v$ with standard error for $N = 3, \ldots, 20$ by GA method

Figure 7 shows value $v$ with standard error for different $N$, $N = 3,\ldots,20$. Modelling parameters are $\alpha = 0.95$, $fit = 0.25$, $N_2 = 100$, $N_1 = 500$, $\varepsilon = 0.1$, $K = 6$, $R = 3$. Method was repeated 5 times for each $N$.

The simulation results show that the genetic algorithm can generate solutions with bigger dispersion compared with the cross-entropy method. Solutions by cross-entropy method are more closely optimum but this method need a lot more computational resource. Besides, if $k$, $N$ and simulation parameters are sufficiently great, the genetic algorithm will generate better solutions.

## 8 Conclusions

In this chapter we have proposed how CE method can be used to solve the multiple best choice problem. The CE method is better than the GA algorithm for solution this problem. But for $N$ tends to infinity, the CE method gives heavy error, and it should be modified. If $N_1, N_2, N_{last}$ increase, the cross-entropy method finds optimal stopping rules that are nearly optimal. The methodology can also be extended to more general models.

**Acknowledgements.** T.V. Polushina would like to thank G. Yu. Sofronov for his invaluable advices and helpful discussion.

## References

1. Bruss, F.t., Ferguson, T.S.: Minimizing the expected rank with full information. J.Appl. Prob. 30, 616–626 (1993)
2. Chow, Y.S., Robbins, H., Siegmund, D.: Great Expectations: The Theory of Optimal Stopping. Houghton Mifflin, Boston (1971)
3. Chow, Y.S., Moriguti, S., Robbins, H., Samuels, S.M.: Optimal selection based on relative rank. Israel J.Math. 2, 81–90 (1964)
4. Dynkin, E.B., Yushkevich, A.A.: Theorems and Problems on Markov Processes. Plenum, New York (1969)
5. Gabor, C.P., Halliday, T.R.: Sequential mate choice by multiply mating smooth newts: females become more choosy. Behavioral Ecology 8, 162–166 (1997)
6. Gilbert, J.P., Mosteller, F.: Recognizing the maximum of sequence. J. Am. Stat. Assoc. 61, 35–73 (1966)
7. Cianini-Pettitt, J.: Optimal selection based on relative ranks with a random number of individuals. Adv. Appl. Prob. 11, 720–736 (1979)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Kluwer Academic Publishers, Boston (1989)
9. Nikolaev, M.L.: On a generalization of the best choice problem. Theory Prob. Appl. 22, 187–190 (1977)
10. Nikolaev, M.L.: Optimal multi-stopping rules. Obozr. Prikl. Prom. Mat. 5, 309–348 (1998)
11. Pitcher, T.E., Neff, B.D., Rodd, F.H., Rowe, L.: Multiple mating and sequential mate choice in guppies: females trade up. Proceedings of the Royal Society B: Biological Sciences 270, 1623–1629 (2003)

12. Rubinstein, R.Y., Kroese, D.P.: The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization. In: Monte-Carlo Simulation and Machine Learning. Springer, New York (2004)
13. Shiryaev, A.N.: Optimal Stopping Rules. Springer, New York (1978)
14. Sofronov, G.Y., Kroese, D.P., Keith, J.M., Nikolaev, M.L.: Simulation of thresholds in the multiple best choice problem. Obozr. Prikl. Prom. Math. 13, 975–983 (2006)
15. Tamaki, M.: Minimal expected ranks for the secretary problems with uncertain selection. Lecture Notes-Monograph Series, vol. 35, pp. 127–139. Institute of Mathematical Statistics (2000)
16. Wang, R., Okazaki, K.: Solving facility layout problem using an improved genetic algorithm. IEICE Trans. Fundam. E88–A N2, 606–610 (2005)

# Author Index

# Index