Advances in Evolutionary Algorithms

Edited by Witold Kosinski

Advances in Evolutionary Algorithms

Edited by Witold Kosinski

Advances in Evolutionary Algorithms

Edited by Witold Kosinski

DOI: 10.5772/73

Publishing Process Manager
Technical Editor
Cover Designer
AvE4EvA MuViMix Records

ISBN-10 953-7619-11-7
ISBN-13 978-953-7619-11-4

Contents

Preface

With the recent trends towards massive data sets and significant computational power, combined with evolutionary algorithmic advances evolutionary computation is becoming much more relevant to practice.

Aim of the book is to present recent improvements, innovative ideas and concepts in a part of a huge EA field.

Part I:

Foundations and New Methods

Limit Properties of Evolutionary Algorithms

Witold Kosiński[1,3] and Stefan Kotowski[1,2]
[1] *Faculty of Computer Science Polish-Japanese Institute of Information Technology,*
[2] *Institute of Fundamental Technological Research IPPT PAN,*
[3] *Institute of Environmental Mechanics and Applied Computer Science Kazimierz Wielki University*
Poland

1. Introduction

In the chapter limit properties of genetic algorithms and theproblem of their classification are elaborated. Recently one can observe an increasing interest in properties of genetic algorithms modelled by Markov chains (Vose, Rowe). However, the known results are mainly limited to existence theorems. They say that there exists a limit distribution for a Markov chain describing a simple genetic algorithm. In the chapter we perform the next step on this way and present a formula for this limit distribution for a Markov chain. Moreover, we claim that our convergence theorems can be extended to algorithms which admit the change in the mutation rate and others parameters.

The formula for a limit distribution requires some knowledge about the distribution of the fitness function on the whole solution space. However, it suggests the methods to control the algorithm parameters to get better convergence rate. The formula can play an important role in deriving new classification tools for genetic algorithms that use methods of the theory of dynamical systems. That tools will exploit real dynamics of the search and be independent of the taxonomic methods of classification that are used nowadays.

On the base of the knowledge of the limit distribution we construct an optimal genetic algorithm in the probabilistic sense. Generally this algorithm is impossible to describe. This is an open problem at the moment, however, its existence and its form suggest an improvement of the original algorithm by changing its parameters. Constructed in this way the optimal genetic algorithm is an answer to one of the questions stayed by famous No Free Lunch Theorem. Moreover, it is a complementary result to this theorem. On the base of this theoretical result we perform a classification of algorithms and show empirical (computational) results in getting which the entropy, fractal dimension, or its approximations: the box-counting dimension or information dimension, are used.

One of the most difficult, however, of practical importance, problems is the choice of an algorithm to given optimisation problem.

The distinguishing between an optimisation problem and the algorithm and its choice creates to the main difficulty. Consequently, the distinguishing is an artificial operation because it abstains from the idea of genetic algorithm (GA), since the fitness function, arises from the cost function (i.e. the function to be optimised) is the main object of the genetic algorithm and it emerges from the formulation of the optimisation problem and it is difficult to speak about genetic algorithm as an operator without the fitness function. However, in our consideration we will simultaneously use both notions of the genetic algorithms. The first notion as an operator acting on the cost (fitness) function, the second - as a specific (real) algorithm for which the fitness is the main component being the algorithm's parameter.

This dual meaning of the genetic algorithm is crucial for ou consideration, because our main aim is to try to classify genetic algorithms. The classification should lead to a specific choice of methodology of genetic algorithms understood as operators. It is expected that in terms of this methodology one will be able to choose the appropriate algorithm to given optimisation problem. We claim that using this classification one could improve existing heuristic methods of assortment of genetic algorithms that are based mainly on experiences and programmer intuition.

There is the so-called "No-free lunch theorem" [12] according to which it does not exist a best evolutionary algorithm and moreover, one cannot find most suitable operator between all possible mechanisms of crossover, mutation and selection without referring to the particular class of optimisation problems under investigation. Evolutionary algorithms are the methods of optimizations which use a limited knowledge about investigated problem. On the other hand, our knowledge about the algorithm in use is often limited as well [13, 14].

The "no free lunch" results indicate that matching algorithms to problems give higher average performance than those applying a fixed algorithm to all problems. In the view of these facts, the choice of the best algorithm may be correctly stated only in the context of the optimisation problem.

These facts imply the necessity of searching particular genetic algorithms suitable to the problem at hand.

The present paper is an attempt to introduce an enlarged investigation method to the theory of genetic (evolutionary) algorithms. We aim at

1. the investigation of convergence properties of genetic algorithms,
2. the formulation of a new method of analysis of evolutionary algorithms regarded as dynamical processes, and
3. the development of some tools suitable for characterization of evolutionary algorithms based on the notions of the symbolic dynamics.

Genetic algorithm (GA) performs a multi-directional search by maintaining a population of potential solutions and encourages information formation and exchange between these directions. A population undergoes a simulated evolution due to the iterative action with some probability distributions of a composition of mutation, crossover and selection operators. The action of that composition is a random operation on populations.

If we imagine that a population is a point in the space Z of (encoded) potential solutions then the efect of one iteration of this composition is to move that population to another point. In this way the action of GA is a discrete (stochastic) dynamical system. We claim that by implementing the methods and the results of the theory of dynamical systems, especially those known from the analysis of dynamics of 1D mappings, one can move towards the goal of the theory of GA, which is the explanation of the foundations of genetic algorithm's operations and their features.

In GA with the known fitness function the proportional selection can be treated as a multiplication of each component of the frequency vector by the quotient of the fitness of the corresponding element to the average fitness of the population. This allows to write the probability distribution for the next population in the form of the multiplication of the diagonal matrix times the population (frequency) vector. Moreover, results of the mutation can also be written as a product of another matrix with the population (probability) vector. Finally the composition of both operations is a matrix, which leads to the general form of the transition operator (cf.(17)) acting on a new probability vector representing a probability distribution of appearance of all populations of the same *PopSize*. The matrix appearing there turns to be Markovian and each subsequent application of SGA is the same as the subsequent composition of that matrix with itself. (cf.(19)). Thanks to the well-developed theory of Markov operators ([18, 22, 26, 27]) new conditions for the asymptotic stability of the transition operator are formulated.

2. Genetic algorithms

In the paper we use the term *population* in two meanings; in the first it is a finite multi-set (a set with elements that can repeat) of solutions, in the second it is a frequency vector composed of fractions, i.e. the ratio of the number of copies of each element $z_k \in Z$ to the total population size *PopSize*.
In our analysis we are concerned with probability distributions of each population for a particular case of the simple genetic algorithm (SGA) in which the crossover follows the mutation and the proportional selection. In the case of a binary genetic algorithm (BGA) the mutation can be characterized by the bitwise mutation rate μ - the probability of the mutation of one bit of a chromosome. In the paper, however, we are not confined to binary operators; the present discussion and results are valid under very week assumptions concerning the mutation and selection operators.

2.1 Population and frequency vector

Let

$$Z = \{z_0, ..., z_{s-1}\},$$

be the set of individuals called *chromosomes*. [1]By a *population* we understand any multi-set of r chromosomes from Z, then r is the population size: *PopSize*.

Definition 1. *By a frequency vector of population we understand the vector*

$$\boldsymbol{p} = (p_0, ..., p_{s-1}) \ , \ \ where \ p_k = \frac{a_k}{r} \ , \tag{1}$$

where a_k is a number of copies of the element z_k.

The set of all possible populations (frequency vectors) is

$$\Lambda = \{\boldsymbol{p} \in \mathbf{R}^s : \ p_k \geq 0, \ p_k = \frac{d}{r}, \ d \in \mathbf{N}, \ \sum_{k=0}^{s-1} p_k = 1 \ \}. \qquad ()$$

[1] If one considers all binary l-element sequences then after ordering them one can compose the set Z with $s = 2^l$ elements.

When a genetic algorithm is realized, then we act on populations, and new populations are generated. The transition between two subsequent populations is random and is realized by a probabilistic operator. Hence, if one starts with a frequency vector, a probabilistic vector can be obtained. It means that in some cases p_i cannot be rational any more. Hence the closure of the set Λ, namely

$$\overline{\Lambda} = \{\boldsymbol{x} \in \mathbf{R}^s : \forall k,\ x_k \geq 0,\ \text{and } \sum_{k=0}^{s-1} x_k = 1\ \}, \tag{3}$$

is more suitable for our analysis of such random processes acting on probabilistic vectors; they are in the set $\overline{\Lambda}$.

2.2 Selection operator

Let a fitness function $f : Z \rightarrow \mathbf{R}^+$ and population $\boldsymbol{p}$ be given. If we assume the main genetic operator is the *fitness proportional selection*, then the probability that the element z_k will appear in the next population equals

$$\frac{f(z_k)p_k}{\overline{f}(\boldsymbol{p})}, \tag{4}$$

where $\overline{f}$ $(\boldsymbol{p})$ is the *average population fitness* denoted by

$$\overline{f}(\boldsymbol{p}) = \sum_{k=0}^{s-1} f(z_k)p_k. \tag{5}$$

We can create the matrix $\boldsymbol{S}$ of the size s, where its values on the main diagonal are

$$S_{kk} = f(z_k). \tag{6}$$

Then the transition from the population p into the new one, say q is given by

$$\boldsymbol{q} = \frac{1}{\overline{f}(\boldsymbol{p})}\boldsymbol{S}\boldsymbol{p}, \tag{7}$$

and the matrix $\boldsymbol{S}$ describes the *selection operator* [21, 23, 24].

2.3 Mutation operator

Let us define a matrix

$$\boldsymbol{U} = [\,U_{ij}\,]\,,$$

with U_{ij} as the probability of mutation of the element z_j into the element z_i, and U_{ii} - the probability of the surviving of the element (individual) z_i. One requires that

1. $$U_{ij} \geq 0 \,;$$

2. $$\sum_{i=0}^{s-1} U_{ij} = 1 \ , \text{ for } \text{ all } j. \tag{8}$$

In the case of the binary uniform mutation with parameter μ as the probability of changing bits 0 into 1 or vice versa, if the chromosome z_i differs from z_j at c positions then

$$U_{ij} = \mu^c (1-\mu)^{l-c} \tag{9}$$

describes the probability of mutation of the element z_j into the element z_i.

2.4 Crossover operation

In order to define the operator of crossover $\boldsymbol{C}$ one needs to introduce additional denotation. Let matrices $\boldsymbol{C}_0,\dots,\boldsymbol{C}_{s-1}$ be such that the element $(i,\ j)$ of the matrix $\boldsymbol{C}_k$ denotes the probablity that an element z_i crossovered with an element z_j will generate an element z_k.
For the presentation simplicity let us consider the case of chromosoms of the lenght l = 2. Then elements of the space B will be of the form

$$z_0 = 00,\ z_1 = 01,\ z_2 = 10,\ z_3 = 11. \tag{10}$$

For the uniform crossover operation when all elements may take part, the matrix $\boldsymbol{C}_0$ has the form

$$\boldsymbol{C}_0 = \begin{pmatrix} 1.0 & 0.5 & 0.5 & 0.25 \\ 0.5 & 0.0 & 0.25 & 0.0 \\ 0.5 & 0.25 & 0.0 & 0.0 \\ 0.25 & 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{11}$$

One can define the remaining matrices; all matrices $\boldsymbol{C}_k$ are symmetric. Finally, the operator $\boldsymbol{C}$ in the action on a population $\boldsymbol{p}$ gives

$$\boldsymbol{C}(\boldsymbol{p}) = (\boldsymbol{p}\cdot\boldsymbol{C}_0\boldsymbol{p}, ..., \boldsymbol{p}\cdot\boldsymbol{C}_{s-1}\boldsymbol{p}), \tag{12}$$

where the dot $\cdot$ denotes the formal scalar product of two vectors from s-dimentional space. Hence, from a given population (say, $\boldsymbol{p}$) to the next population (say, $\boldsymbol{q}$) the action of the simple genetic algorithm (SGA) [21, 23, 24] is described by the operator $\mathcal{G}$ being a composition of three operators: selection, mutation and crossover:

$$\mathcal{G} = \boldsymbol{C}\circ\boldsymbol{U}\circ\mathcal{F}. \tag{13}$$

The reader interested in the detailed descrition of the operators is referred to the positions [21, 23]. In what follows the crossover is not present. However, most of the results of subsequent sections hold if the crossover is present.

3. Transition operator

Let $\boldsymbol{p} = (p_0,\ldots,p_{s-1})$ be a probabilistic vector. If we consider $\boldsymbol{p} \in \overline{\Lambda}$, then transition operators should transform set $\overline{\Lambda}$ into itself. The action of the genetic algorithm at the first and at all subsequent steps is the following: if we have a given population $\boldsymbol{p}$ then we sample with returning r-elements from the set Z, and the probability of sampling the elements $z_0,\ldots, z_{s-1}$ is described by the vector $\mathcal{G}(\boldsymbol{p})$, where

$$\mathcal{G}(\boldsymbol{p}) = \frac{1}{\overline{f}(\boldsymbol{p})}\boldsymbol{USp} \,. \tag{14}$$

This r-element vector is our new population $\boldsymbol{q}$.
Let us denote by W the set of all possible r-element populations composed of elements selected from the set Z, where elements in the population could be repeated. This set is finite and let its cardinality be M: It can be proven that the number M is given by some combinatoric formula

$$M = \binom{s+r-1}{s-1} = \binom{s+r-1}{r} \,. \tag{15}$$

Let us order all populations, then we identify the set W with the list $W = \{w^1,\ldots,w^M\}$. Every w^k, $k = 1, 2,\ldots,M$, is some population for which we used the notation $\boldsymbol{p}$ in the previous section. According to what we wrote, the population will be identified with its frequency vector or probabilistic vector. This means that for the population $\boldsymbol{p} = w^k = (w_0^k,\ldots,w_{s-1}^k)$, the number w_i^k, for $i \in \{0,\ldots,s-1\}$, denotes the probability of sampling from the population w^k the individual z_i (or the fraction of the individual z_i in the population w^k).
Let us assume that we begin our implementation of SGA from an arbitrary population $\boldsymbol{p} = w^k$. In the next stage each population $w^1,\ldots,w^M$ can appear with the probability $\beta_{1k}, \beta_{lk},\ldots, \beta_{Mk}$ which can be determined from our analysis. In particular, if in the next stage the population has to be $\boldsymbol{q}$, with the position l on our list W, then this probability [23, 28, 31] is equal

$$\beta_{lk} = r! \prod_{j=0}^{s-1} \frac{(\mathcal{G}(\boldsymbol{p})_j)^{rq_j}}{(rq_j)!} \,, \text{ with } \boldsymbol{q} = w^l \text{ and } \boldsymbol{p} = w^k \,. \tag{16}$$

Notice that $\sum_{j=1}^{M} \beta_{jk} = 1$ for every $k = 1, 2,\ldots,M$. After two steps, every population $w^1,\ldots,w^M$ will appear with some probability, which is a double composition of this formula[2]. It will be analogously in the third step and so on. Then it is well founded to analyze the

[2] With our choice of denotations for the populations $\boldsymbol{p}$ and $\boldsymbol{q}$ in (16), the element β_{lk} of the matrix will give transition probability from the population with the number k into the population with the number l.

probability distribution of the population's realization in the next steps. This formula gives a possibility of determining all elements of a matrix $\boldsymbol{T}$ which defines the probability distribution of appearance of populations in the next steps, if we have current probability distribution of the populations.
It is important that elements of the matrix are determined once forever, independently of the number of steps. The transition between elements of different pairs of populations is described by different probabilities (16) represented by different elements of the matrix.
Let us denote by

$$\Gamma = \{\boldsymbol{y} \in \mathbf{R}^M : \forall k \ y_k \geq 0 \text{ oraz } ||\boldsymbol{y}|| = 1\},$$

where $\|\boldsymbol{y}\| = y_1 + \ldots + y_M$, the set of new M-dimensional probabilistic vectors. A particular component of the vector $\boldsymbol{y}$ represents the probability of the appearance of this population from the list W of all M populations. The set Γ is composed of all the possible probability distributions for M populations. Described implementation transforms at every step the set Γ into the same.
On the set Γ the basic, fundamental *transition operator,*

$$T(\cdot) : \mathbf{N} \times \Gamma \to \Gamma. \tag{17}$$

is defined. If $u \in \Gamma$, then $T(t)u = ((T(t)u)_1, \ldots, (T(t)u)_M)$ is the probability distribution for M populations in the step number t, if we have begun our implementation of SGA given by $\mathcal{G}$ ((14)) from the probability distribution $u = (u_1, \ldots, u_M) \in \Gamma$, by t – application of this method. The number $(T(t)u)_k$ for $k \in \{1 \ldots, M\}$denotes the probability of appearance of the population w_k in the step of number t. By the definition $\mathcal{G}(\boldsymbol{p})$ in (14),(16) and the remarks made at the end of the previous section the transition operator $T(t)$ is linear for all natural t.
Let us compose a nonnegative, square matrix $\boldsymbol{T}$ of dimension M, with elements β_{lk}, l, k = 1, 2,…,M, i.e

$$\boldsymbol{T} = [\beta_{lk}]. \tag{18}$$

We will call it *the transition matrix*. Then the probability distribution of all M populations in the step t is given by the formula

$$\boldsymbol{T}^t u, \; t = 0, 1, 2, \ldots$$

Elements are independent from the number of steps of the algorithm. The above introduced transition operator $T(t)$ is linked with the transition matrix by the dependence

$$T(t) = \boldsymbol{T}^t. \tag{19}$$

Notice that though the formula (16) determining individual entries (components) of the matrix $\boldsymbol{T}$ are population dependent, and hence nonlinear, the transition operator $T(t)$ is linear thanks to the order relation introduced in the set W of all M populations. The multi-index (l, k) of the component β_{lk} kills, in some sense, this nonlinearity, since it is responsible for a pair of populations between which the transition takes place. The matrix $\boldsymbol{T}$ in (18) is a Markovian matrix. This fact permits us to apply the theory of Markov operators to analyze the convergence of genetic algorithms [18, 22, 26, 27].

Let $e_k \in \Gamma$ be a vector which at the k-th position has one and zeroes at the other positions. Then e_k describes the probability distribution in which the population w^k is attained with the probability 1.

By the notation $T(t)w^k$ we will understand

$$T(t)w^k := T(t)e_k \tag{20}$$

which means that we begin the GA at the specific population w^k. Further on we will assume $U_{jj} > 0$ for $j \in \{0,\dots,s-1\}$.

For a given probability distribution $u = (u_1,\dots,u_M) \in \Gamma$ it is easy to compute that the probability of sampling the individual z_i, for $i \in \{0,\dots,s-1\}$, is equal to

$$\sum_{k=1}^{M} w_i^k u_k \ , \tag{21}$$

where w_i^k is the probability of sampling from k-th population the chromosome z^i, and u_k - the probability of appearance of the k-th population. By an *expected population* we call the vector from $\mathbf{R}^s$ of which i-th coordinate is given by (21). Since $u_k \geq 0,\ w_i^k \geq 0$ for $k \in \{1,\dots,M\}$, $i \in \{0,\dots,s-1\}$ and

$$\sum_{i=0}^{s-1}\left(\sum_{k=1}^{M} u_k w_i^k\right) = \sum_{k=1}^{M} u_k \left(\sum_{i=0}^{s-1} w_i^k\right) = \sum_{k=1}^{M} u_k = 1 \ ,$$

the vector belongs to $\overline{\Lambda}$. From (21) we obtain that the expected population is given by

$$\sum_{k=1}^{M} w^k u_k \tag{22}$$

Obviously, it is possible that the expected population could not be any possible population with r-elements.

For every $u \in \Gamma$ and for every t certain probability distribution for M populations $T(t)u$ is given. Consequently the expected population in this step is known. By $R(t)u = ((R(t)u)_0,\dots,(R(t)u)_{s-1})$ we denote the expected population at the step t, if we begun our experiment from the distribution $u \in \Gamma$; of course we have $R(t)u \in \overline{\Lambda}$.

3.1 Asymptotic stability

Definition 2. *We will say that the model is asymptotically stable if there exist $u^* \in \Gamma$ such that:*

$$T(t)u^* = u^* \quad \text{for} \quad t = 0, 1, \dots \tag{23}$$

$$\lim_{t\to\infty} ||T(t)u - u^*|| = 0 \quad \text{for all } u \in \Gamma \ . \tag{24}$$

Since for $k \in \{1,\dots, M\}$ we have

$$|(T(t)u)_k - u_k^*| \leq ||T(t)u - u^*|| \ , \tag{25}$$

then (24) will give

$$\lim_{t\to\infty} (T(t)u)_k = u_k^* \ . \tag{26}$$

It means that probability of appearance of the population w^k in the step number t converges to a certain fixed number u_k^* independently of the initial distribution u. It is realized in some special case, when our implementation began at one specific population $\boldsymbol{p} = w^j$.

Theorem 1. *If the model is asymptotically stable, then*

$$\lim_{t\to\infty} ||R(t)u - \boldsymbol{p}^*|| = 0 \quad \text{for} \quad u \in \Gamma \ , \tag{27}$$

where $\boldsymbol{p}^* \in \overline{\Lambda}$ is *the expected population adequate to the distribution* u^*. *Particularly, we have also*

$$\lim_{t\to\infty} ||R(t)\boldsymbol{p} - \boldsymbol{p}^*|| = 0 \quad \text{for} \quad \boldsymbol{p} \in W. \tag{28}$$

Proof. From (22) we have

$$R(t)u = \sum_{k=1}^{M} w^k (T(t)u)_k$$

and

$$\boldsymbol{p}^* = \sum_{k=1}^{M} w^k u_k^* \ .$$

Then

$$||R(t)u - \boldsymbol{p}^*|| = \sum_{j=0}^{s-1} |\sum_{k=1}^{M} w_j^k (T(t)u)_k - \sum_{k=1}^{M} w_j^k u_k^*|$$

$$\leq \sum_{j=0}^{s-1}\sum_{k=1}^{M} w_j^k |(T(t)u)_k - u_k^*| = ||T(t)u - u^*||.$$

On the basis of (24) the equality follows (27). Taking into account our notation, given in (20), the formula (28) is the particular case of (27). △

Theorem 1 states that for the asymptotically stable case the expected population stabilizes, converging to $\boldsymbol{p}^* \in \overline{\Lambda}$ independently of initial conditions. This result has a fundamental meaning for the analysis of the convergence of genetic algorithms. This generalization will be the subject of our next paper. Moreover, this theorem is an extension of *Th.*4.2.2.4 4 from

[24] for the case when it is possible to attain any population in a finite number of steps, (not only in one step). It means that the transition operator does not need to be positively defined, but there exists such *k*, that the *k*-th power of the transition matrix possesses a column which is strongly positive. The same concerns *Th*.4.2.2 1 of [24, 25] which is true only for a positively defined transition matrix.
We shall say that from the chromosome z_a it is possible to obtain z_b in one mutation step with a positive probability if $U_{ba} > 0$. We shall say that from the chromosome z_a it is possible to get the chromosome z_b with positive probability in n-step mutation if there exists a sequence of chromosomes $z_{l_0},\dots, z_{l_n}$, such that $z_{l_0} = z_a$, $z_{l_n} = z_b$, and for any $k = 1,\dots, n$ it is possible to attain the chromosome z_{l_k} from $z_{l_{k-1}}$ in one step with a positive probability.

Definition 3. *Model is pointwise asymptotically stable if there exists such a population w^j that*

$$\lim_{t\to\infty}(T(t)u)_j = 1 \text{ for } u \in \Gamma \ . \tag{29}$$

Condition (29) denotes that in successive steps the probability of appearance of a population other than w^j tends to zero. It is a special case of the asymptotic stability for which

$$u^* = e_j .$$

Theorem 2. *Model is pointwise asymptotically stable if and only if there exists exactly one chromosome z_a with such a property that it is possible to attain it from any chromosome in a finite number of steps with a positive probability. In this situation the population w^j is exclusively composed of the chromosomes z_a and*

$$T(t)w^j = w^j \tag{30}$$

holds. Moreover, the probability of appearance of population other than w^j tends to zero in the step number *t* with a geometrical rate, i.e. there exists $\lambda \in (0, 1)$, $D \in \boldsymbol{R}_+$ such that

$$\sum_{\substack{i=1\\ i\neq j}}^{M}(T(t)u)_i \leq D\cdot\lambda^t \ . \tag{31}$$

△

The proofs of our theorems and auxiliary lemmas are stated in other articles [29-31, 33].
From the formula (30) it follows, that from a population w^j we receive w^j with the probability equal 1. Moreover, if w^j becomes once, then from this moment on we shall permanently have populations w^j . Numbers λ and *D* could be determined for a specific model. It will be the subject of the next articles.
Theorem 2 states that the convergence to one population could occur only under specific assumptions. This justifies the investigation of the asymptotic stability which is different from that in Definition 3.

Definition 4. *By an attainable chromosome we denote $z_a \in Z$ such that it is possible to attain it from any other chromosome in a finite number of steps with a positive probability. Let us denote by Z^* the set of all z_a with this property.*

Theorem 3. *Model is asymptotically stable if and only if* $Z^* \neq 0$.

△

Theorem 4. *Let us assume that the model is asymptotically stable. Then the next relationship holds*:

(*war*) $u_k^* > 0$ *if and only if the population* w^k *is exclusively composed of chromosomes belonging to the set* Z^*. △

Corollary 1. *If* $Z^* = Z$ *then* $u_k^* > 0$ *for all* $k \in \{1,\ldots,M\}$. △

Here we set the summary of our results:

1. $Z^* = 0 \Rightarrow$ lack of asymptotic stability;
2. $Z^* \neq 0 \Rightarrow$asymptotic stability but:
3. cardinality $(Z^*) = 1 \Rightarrow$ pointwise asymptotic stability (in some sense convergence to one population);
4. cardinality $(Z^*) > 1 \Rightarrow$ asymptotic stability, but there is no pointwise asymptotic stability.

If one restricts to a binary simple genetic algorithm with a positive mutation probability, then it is possible to attain any individual (chromosome) from any other individual. Then there is more than one binary chromosome which is possible to attain from any other in a finite number of steps with a positive probability, and by Corollary 1, it is impossible to get the population composed exclusively of one type of chromosome. It could be interesting to consider non-binary cases for which the above observation does not hold.

3.2 Genetic algorithms with parameters adaptation

Genetic algorithm is realized as an adaptation process, hence it is natural to expect, that during its action its parameters are adapted on the base of some internal dynamics of the algorithm. It follows from the conjecture, that at different states, i.e. at different steps of the algorithm, values of algorithm parameters could be changed in the optimal way to accelerate the process convergence.

Till now the problem of algorithm parameters fitting is complex and not well defined, and it has an undefined structure. However, there exist many arguments for parameters adaptations that can improve action of actual genetic algorithm. There exists an opinion that by adding individual algorithm or metha-algorithm related to the actual one one can improve the solution of the problem. Such situation may be realized by an adaptation of genetic algorithms parameters on the base of the present state of the process (i.e. the actual population). It is conducted, for example, by introducing the methodology of parameters changing, which uses information on populations and values of the fitness function. The same can be proposed by a modification of the fitness function only.

In most case such adaptation is realised by increasing not only the dimension of chromosoms but also the search space, and consequently the population vector. Then, there appears an extra meta-algorithm, which runs parallel to the actual genetic one.

Even in such situations our algorithm model is conserved (16), and then the search space is enlarged (the arguments set) and in consequence the number of possible populations grow. The dimension of the Markovian matrix describing new, composed algorithm 18 grows. However, the transition operator (19) has the same properties as in the classical simple genetic algorithm. Consequently, all theorems on convergence of genetic algorithms from the previous sections are conserved, as well as the results concerning the limit algorithm of the next Section 4.2 and the form of the optimal algorithm in probabilistic sense.

4. Classification of algorithms and its invariants

The convergence of GAs is one of the main issues of the theoretical foundations of GAs, and has been investigated by means of Markov's chains. The model of GA as a Markov's chain is relatively close to the methods known in the theory of dynamical systems.
In the analysis of GAs regarded as (stochastic) dynamical systems one can use the fact, (proven by Ornstein and Friedman [4, 10]) which states that mixing Markov's chains are Bernoulli's systems and consequently, the entropy of the systems is a complete metric invariant.
Those facts enable us to classify GAs using the entropy. The systems for which the entropies have the same value are isomorphic. Hence the entropy makes it possible to classify GAs by splitting them into equivalence classes.

4.1 Isomorphism of algorithms

The domain of research of the ergodic theory is a space with measure and mappings which preserve it. The measure space is the point set X with a measure m (when normalised to one, it is called the probability) defined on σ - algebra of its subsets $\mathcal{B}$, called measureable. To use results of the theory some defintions [16, 15] must be introduced.
Definition 5. *Let* $(X_1, \mathcal{B}_1, m_1)$, $(X_2, \mathcal{B}_2, m_2)$ *be measure spaces. We say that a mapping* $\phi : X_1 \to X_2$ *is measure preserving if: i) it is measurable, i.e.* $\phi^{-1}(A) \in \mathcal{B}_1$ *for every* $A \in \mathcal{B}_2$*, and ii)* $m_1(\phi^{-1}(A)) = m_2(A)$. *If* $X_1 = X_2$ *and* $m_1 = m_2 =: m$ *and* ϕ *preserves a measure* m *then we say that* m *is* ϕ*-invariant (or invariant under* ϕ*).*
In the example below we will say that so-called 1D *backer's transformation*[3]preserves Lebesgue measure of the line. Let $X = [0; 1)$ and consider $\phi_1(x) = 2x$ (mod 1). Notice that even though the mapping doubles the length of an interval I, its inverse image has two pieces in general, each of which has the length of I, and when we add them, the sum equals the original lenght of I. So ϕ_1 preserves Lebesgue measure.
The generalization of the above mapping to 2D is the backer' transformation defined[4] on the square $X = [0, 1] \times [0, 1]$ as

$$\phi_2(x,y) = \left\{ \begin{array}{ll} (2x, \frac{1}{2}y) \;, & 0 \le x \le \frac{1}{2} \\ (2x-1, \frac{1}{2}y + \frac{1}{2}) \;, & \frac{1}{2} \le x \le 1 \end{array} \right\} \tag{32}$$

which presereves the 2D Lebesgue measure on the unit square.
Definition 6. Probability spaces $(X_1, \mathcal{B}_1, m_1)$, $(X_2, \mathcal{B}_2, m_2)$ are said to be *isomorphic* if there exist $M_1 \in \mathcal{B}_1$, $M_2 \in \mathcal{B}_2$ with $m_1(M_1) = 1 = m_2(M_2)$ and an invertible measure preservimg transformation $\phi : M_1 \to M_2$.

[3] It is also called 1D Bernoulli shift.

[4] The transformation is the composition of three transformations of the unit square first, press down the square, cut in the midle and move the right half to the top of the left half.

In [16] the defintion is more general and requires the mapping ϕ to be defined on whole X_1 and be almost everywhere bijective from X_1 onto X_2, i.e. it must be bijective except for the sets of measure zero. However, in view of Definition 6 the sets $X_1 \backslash M_1$ and $X_2 \backslash M_2$ have zero measure.
In order to investigate genetic algorithms and their similarity (or even more - isomorphism) we need to consider mappings defined on probability space.
Definition 7. *Suppose probability spaces* $(X_1, \mathcal{B}_1, m_1)$, $(X_2, \mathcal{B}_2, m_2)$ *together with measure preserving transformations* $T_1 : X_1 \rightarrow X_1$; $T_2 : X_2 \rightarrow X_2$. *We say that* T_1 *is* ***isomorphic to*** T_2 *if there exist* $M_1 \in \mathcal{B}_1$, $M_2 \in \mathcal{B}_2$ *with* $m_1(M_1) = m_2(M_2) = 1$ *such that: i)* $T_1(M_1) \subseteq M_1$, $T_2(M_2) \subseteq M_2$, *and ii) there is an invertible measure-preserving transformation*

$$\phi : M_1 \rightarrow M_2 \quad with\ \phi(T_1(x)) = T_2(\phi((x)) \quad for\ all\ x \in M_1.$$

Consider infinite strings made of k symbols from $[1,\ldots, k]$. Put $X = \prod_1^\infty \{1, ..., k\}$. An element x of X is denoted by $(x_1\, x_2\, x_3 \ldots)$.[5] Let a finite sequence $p_1, p_2, \ldots, p_k$, where for each i the number $p_i \in [0, 1]$ be such that $\sum_{i=1}^{k} p_i = 1$. For $t \geq 1$ define a cylinder set (or a block) of length n by

$$[a_1, a_2, ..., a_n]_{t,...,t+n-1} = \{x \in X : x_{t+1} = a_1, ..., x_{t+n} = a_n\} \tag{33}$$

With this denotation let us introduce the main definition of the Bernoulli shift which plays the main role in our approach [15, 16].
Definition 8. *Define a measure* μ *on cylinder sets by*

$$\mu([a_1, a_2, ..., a_n]_{t,...,t+n-1}) = p_{a_1} \cdots p_{a_n} \ . \tag{34}$$

A probability measure on X, again denoted by μ, *is uniquely defined on the* σ *- algebra generated by cylinder sets. We call* μ *the* $(p_1,\ldots,p_k)$*-Bernoulli measure and X is the Bernoulli shift space. The* ***one-sided Bernoulli shift transformation*** *T on X defined by*

$$(x_1 x_2 x_3 \ldots) \longmapsto (x_2 x_3 x_4 \ldots) \ , \tag{35}$$

$i.e.\ T(x_1 x_2 x_3 \ldots) = (x_2 x_3 x_4 \ldots).$
Similarly, we may define the *two-sided Bernoulli shift transformation* by

$$(\ldots \overset{*}{x}_0 x_1 x_2 x_3 \ldots) \longmapsto (\ldots \overset{*}{x}_1 x_2 x_3 x_4 \ldots)$$

on $\prod_{-\infty}^{\infty} \{1, ..., k\}$ where * denotes the 0-th coordinate in a sequence. Let us notice that the shift preserves the measure μ.
In the case of a binary sequence when we have two symbols only and if each symbol has probablity $\frac{1}{2}$ the space X identified with $X = \prod_1^\infty \{0, 1\}$ is $(\frac{1}{2}, \frac{1}{2})$-Bernoulli shift space. Moreover, the space X is somorphic to [0, 1] with Lebesgue measure if each element $x = (b_1, b_2, \ldots) \in X$ and the transformation is defined by

[5] If $k = 2$ then x is said to be a **binary** sequence.

$$\phi(x) = \sum_{n=1}^{\infty} b_n 2^{-n} \ . \tag{36}$$

To see why, notice that not every $y \in [0, 1]$ has unique binary expansion, but the set of such points has measure zero, and we ignore them. Hence the transformation (36) is almost everywhere bijective (cf. remark below Def. 6) and measure preserving.

The next notions are related to Markov measure and Markov shift. As previously consider the space $X = \prod_1^{\infty}\{1, ..., k\}$. and let $\boldsymbol{P} = (P_{ij})$ be a $k \times k$ stochastic matrix with the right hand operation[6]. Suppose that $\boldsymbol{\pi} = (\pi_i)$ be the right probability eigenvector of $\boldsymbol{P}$, i.e. it satisfies $\sum_{i=1}^{k} \pi_i = 1$ and $\boldsymbol{P\pi} = \boldsymbol{\pi}$. Define ν on the cylinder sets by

$$\nu([a_1, \ldots, a_n])_{t,\ldots,t+n-1} = P_{a_n a_{n-1}} \ldots P_{a_2 a_1} \pi_{a_1} \ . \tag{37}$$

Notice that the sequence of appearance is $a_1, a_2, \ldots, a_n$.

Definition 9. *A unique shift invariant probability measure, again denoted by* ν, *on the* σ*-algebra generated by the cylinder sets, we call the Markov measure and then X is called the Markov shift space.*

Notice that the matrix $\boldsymbol{P}$ defines the transition probabilty

$$Pr(x_{n+1} = j | x_n = i) = P_{ji}$$

which is the conditional probabilty (of an event $x_{n+1} = j$ given that an event $x_n = i$ has occured). Notice that Markov shifts are Bernoulli shifts if the columns of the matrix $\boldsymbol{B}$ are identical. Moreover, the numbers $Pr(x_{n+1} = j | x_n = i)$ satisfy

$$\sum_{b=1}^{k} Pr(b|a) = 1$$

for any $a \in \{1, 2, \ldots, k\}$.

We can identify a Bernoulli measure or a Markov measure with a measure on the interval [0, 1] through the binary expansion (36) (i.e. each binary sequence $x = (b_1, b_2, \ldots)$ is identified with the sum of R.H.S. of (36)). If the probability $p \notin \{0, 1/2, 1\}$, then the $(p, 1\text{-}p)$- Bernoulli measure represented on [0, 1] is singular continuous [16].

4.2 Limit distribution

Now, after [16] we are ready to formualate main facts concerning the limit distribution of the Markov matrix.

Theorem 5. *Let* $\boldsymbol{T} = (T_{ij})$ *be a M×M stochastic matrix. Suppose that* $\boldsymbol{\pi} = (\pi_i)$ *be a right probability eigenvector of* $\boldsymbol{T}$ *, i.e. it satisfies* $\sum_{i=1}^{M} \pi_i = 1$ *and*

$$\boldsymbol{T\pi} = \boldsymbol{\pi}. \tag{38}$$

Then the following relationship hold:

[6] Choe in [16] considers the left hand operation.

i. *there exists* $\boldsymbol{Q} = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} \boldsymbol{T}^n$.

ii. *$\boldsymbol{Q}$ is stochastic (i.e. Markovian) matrix ,*

iii. *$\boldsymbol{QT} = \boldsymbol{TQ} = \boldsymbol{Q}$,*

iv. *If $\boldsymbol{Tv} = \boldsymbol{v}$ then $\boldsymbol{Qv} = \boldsymbol{v}$.*

△

Theorem 6. *All columns of $\boldsymbol{Q}$ are identical and equal to the column vector $\boldsymbol{\pi}$.* △

Since each Markov shift is a Bernoulli shift if columns of the Markov matrix are identical, the limit distribution may be regarded as a Bernoulli shift. Hence the isomorphism of the limit distribution may be treated in the same way as for Bernoulli shifts, i.e. with the help of the entropy, cf. Theorem 9.

Theorem 7. *The convergence $\|\boldsymbol{T}^n - \boldsymbol{Q}\|$ is of exponential type, when $n\to\infty$.* △

One may ask whether it is possible to find a convergence bound in terms of the second eigenvalue of the matrix $\boldsymbol{T}$ and how it is related to the eigenvalues of the matrix $\boldsymbol{Q}$? Moreover, the limit operator $\boldsymbol{Q}$ is a projection operator $\boldsymbol{QQ} = \boldsymbol{Q}$. Its eigenspace is composed of one eigenvector $\boldsymbol{\pi}$ and its properties will help in finding relations to NFL. It will be the subject of the next publication [32].

Theorem 8. *If a genetic algorithm (14) is described by a transition matrix (18) that possesses the eigenvector $\boldsymbol{\pi}$ as a probability vector corresponding to the unit eigenvalue, i.e. the matrix satisfies Eq. (38), then there exists an optimal algorithm in the probabilistic sense. It means that the algorithm starting at an arbitrary initial distribution of populations in one step generates the limit distribution. This limit distribution is desrcibed by the matrix $\boldsymbol{Q}$ appearing in Theorem 5.*

Proof. Let a vector $\boldsymbol{c} = (c_i)$ describe the initial distribution of populations, with $\sum_{i=1}^{M} c_i = 1$.

Let us take an arbitrary row of the matrix $\boldsymbol{Q}$, say j. Then in view of Theorem 6 all elements of this row are the same and equal to π_j. Then making the product $\boldsymbol{Qc}$ we will get for this row

$$c_1\pi_j + c_2\pi_j + \cdots + c_M\pi_j = \pi_j(c_1 + c_2 + \cdots + c_M) = \pi_j.$$

This means that $\boldsymbol{Qc} = \boldsymbol{\pi}$.

The recent theorem is in some sense complementary to the No Free Lunch Theorem. NFL Theorem describes the whole universe of optimization problems and algorithms used to solve them. The present theorem, on the other side, concernes on an individual algorithm dedicated to an individual optimization problem. The former theorem tells that in the mean all algorithms behave in similar way as far as all problems are concerned. The latter theorem, however, states that for allmost every genetic (evolutionary) algorithm and every single optimization problem there exists not only the better algorithm but also the best (optimal) in the probabilistic sense. This algorithm cannot be, in general, deterministic, since the assumptions concerning the pointwise asymptotic stability may not hold (cf. Definition 3 and Theorem 2). The problem of determining, even in the approximate form, the best algoritm is still open. It is hope that the pointwise asymptotic stability can be helpful here.

There is of course the question of uniquenss: two different genetic algoritms may lead to two different limit distributions. Moreover, to two different algorithms may correspond one optimal algorithm. This remark may be used in formulation new methods of classification of genetic algorithms, additional to the entropy and the fractal dimension.

5. Trajectory of BGA

Let X be a space of solutions of an optimisation problem characterized by a fitness function $f : X \rightarrow \mathbf{R}$;$X \subset \mathbf{R}^m$ for which a binary genetic algorithm (BGA) will be invented. Each element $x \in X$ will be encoded in the form of a binary chromosome of the length l (cf. Section 2.1). The coding function φ: $X \rightarrow \{0, 1\}^l = B$ maps elements of X into chromosome from the B space.

Let us assume that the genetic algorithm acts on r-element populations. Each population forms a multiset $[P^r]$ in the product space B^r. For the i-th generation we will use the denotation $[P_i^r]$, for the population and each element of this multiset can be identified with a vector

$$P_i^r = [x_1^i, x_2^i, \ldots, x_r^i] \tag{39}$$

rembering that a population is an equivalent class of points from the vector space B_r. The equivalent relation is defined by the class of all possible permutations of the set of r-th numbers {1, 2,…, r}. Notice that in view of our denotation from Sec.2.1 each x_j^i, j = 1, 2,…, r is one of elements of the set Z.

Let us notice that we can identify points from X with their encoded targets in B under the action of space X^r. By a trajectory of the genetic algorithm of the duration N we mean a set

$$\mathcal{T} = \bigcup_{i=1}^{N} [P_i^r] \ , \tag{40}$$

where N is the number of steps (generations) of the genetic algorithm which is realized.

Let p_m and p_c be the probabilities of the mutation and crossover, respectively, while p_s is the probability of selection, all independent from the generation.

Then, for such a genetic algorithm the probability of the appearance of the population $[P_{i+1}^r]$ at the generation i + 1 after the population $[P_i^r]$ at the generation i, is the conditional probability

$$P(P_{i+1}^r | P_i^r \ , f(P_i^r), p_m, p_c, p_s) \ . \tag{41}$$

Here by $f(P_i^r)$ we understand the vector{valued function of the form $[f(x_1^i), f(x_2^i), \ldots, f(x_r^i)]$.

The initial population $[P_1^r]$ is generated by the use of a uniform probability distribution over the set B, i.e. each point from B has the same probability of being selected as a member

(component) of [P_1^r]. Next populations following that one, i.e. chosen in next generations, are results of the action of the GA and, hence, may have a non-uniform probability distribution.
Let us notice that in view of our assumptions it follows from (41) that the probability of the appearance of each population depends on the previous population and does not depend on the history (i.e. on earlier population; the probabilities p_m, p_c and p_s can be regarded as parameters of the function *P*).
Now, if we look at the trajectory of the GA defined by (40), we can see that its generation is an ergodic (mixing) process and Markov's one. Subsequent populations (i.e. points of the trajectory) are states of the process about which we can say that each state is accessible with the probability 1.

6. Entropy

Let us denote by T_i the operator which maps *i*-th generation (point of the trajectory) into the next one. Having the probability distribution (41) characterizing the mapping T_i from one population to another, we can define the entropy of the mapping

$$H(\mathtt{T}_i) = -\sum_{j=1}^{N} P(P^r_{i+1,j}|P^r_i, f(P^r_i), p_m, p_c, p_s) \cdot \log P(P^r_{i+1,j}|P^r_i, f(P^r_i), p_m, p_c, p_s) \tag{42}$$

where [$P^r_{i+1,j}$] is a possible population from the coding space *B*, $j = 1, 2,\ldots, 2^{N_r},\ldots,M$:

According to our previous proposition the initial population is generated by the use of a uniform probability, and the entropy may attain the maximal value generated by the GA. In the next step the probabilities of populations are not uniform and differ at each generation; this is the essence of the action of GA. Consequently the entropy of the mapping T_i decreases. In the limit case when the number of steps tends to infinity one could expect that the terminal population will be composed of *r* copies (exactly speaking, according to (39) { a cartesian product) of the same element (an optimal solution). However, this case will be possible only in the case of the pointwise asymptotic stability of GA. In general, the entropy will tend to minimum.
Entropy as a function of the probability of mutation and selectio grows with the growing mutation probability and decreases when the selection pressure grows. Then the entropy could realize a measure of interactions between mutations and selection operators. Entropy also depends on the number of elements in population and it is decreasing when the population grows. The entropy value of the trajectory could be linked with computational complexity of the evolutionary algorithms.
Now several questions arise. Does an optimal form of the entrop change exist? What is its limit value, if it is different from zero for the optimisation process performed by GA ? Does an optimal process of the entropy change exist along which an optimal value of the solution can be reached?
Since the determination of the probability of the mapping T_i, as well as the entropy H_i, in an analytical way is rather difficult to be performed, we are proposing to substitute them with a fractal dimension which is related to the entropy [10] and can characterize non-

deterministic features of GA. It should be mentioned that in [8] general statistical and topological methods of analysis of GAs have been introduced from another viewpoint.

Theorem 9. *(Ornstein [10]) Every two Bernoulli shifts with the same entropy are isomorphic.* △

Lemma 1. *(Choe [16])) Let* $X = \prod_1^\infty \{0,1\}$ *be the (p, 1- p) Bernoulli shift space that is regarded as the unit interval* [0, 1) *endowed with the Euclidean metric. Let* X_p *denote the set of all binary sequences* $x \in X$ *such that*

$$X_p = \{x \in X : \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} x_i = p\}$$

then Hausdorff dimension of the set X_p *is equal to the entropy* $-p \log_2 p - (1 - p) \log_2(1 - p)$ *of the Bernoulli shift transformation. Similar results can be obtained for a Markov shift space.* △

Moreover one can use the Hausdorff dimension or its approximation as an invariant of equivalence of algorithms.

7. Fractal dimensions

To be more evident, let us recall the notion of the s-dimensional[7] Hausdorff measure ([5]) of the subset $E \subset \mathbf{R}^l$, where $s \geq 0$. If $E \subset \bigcup_i U_i$ and the diameter of U_i, denoted by $\delta(U_i)$, is less than ϵ for each i, we say that $\{U_i\}$ is an ϵ - **cover of** E. For $\epsilon > 0$, let us define

$$\mathcal{H}^s_\epsilon(E) = \inf \sum_{i=1}^{\infty} [\epsilon(U_i]^s \tag{43}$$

where the in_mum is over all ϵ-covers $\{\boldsymbol{U}_i\}$ of $\boldsymbol{E}$. The limit of $\mathcal{H}^s_\epsilon$ as $\epsilon \to 0$ denoted by $\mathcal{H}^s$(E), is the s-dimensional Hausdorff measure of E.

Let us notice that in the space $\boldsymbol{R}^l$ one can prove that $\mathcal{H}^l(\boldsymbol{E}) = k^l \mathcal{L}^l(\boldsymbol{E})$, where $\mathcal{L}^l$ is the $\boldsymbol{l}$-dimensional Lebesgue measure and $\boldsymbol{k}^l$ is a ratio of volume of the $\boldsymbol{l}$ - dimensional cube to $\boldsymbol{l}$ - dimensional ball inscribed in the cube.

It is evident that $\mathcal{H}^s_\epsilon(E)$ increases as the maximal diameter ϵ of the sets U_i tends to zero, therefore, it requires to take finer and finer details, that might not be apparent in the larger scale into account. On the other hand for the Hausdorff measure the value $\mathcal{H}^s(E)$ decreases as s increases, and for large s this value becomes 0. Then the **Hausdorff dimension** of E is defined by

$$\dim_H(E) = \inf\{s : \mathcal{H}^s(E) = 0\}, \tag{44}$$

and it can be verified that $\dim_H(E) = \sup\{s : \mathcal{H}^s(E) = \infty\}$.

Working with compact subsets of a metric space (X, d) new dimension is introduced. This dimension is also less accurate than the Hausdorff dimension. To calculate this dimension

[7] This s has nothing to do with s introduced in Section 2.

for a set $S \subset X$ imagine this set lying on an evenly-spaced grid. Let us count how many boxes are required to cover the set. The **box-counting** dimension is calculated by observing how this number changes as we make the grid finer. Suppose that $N(\epsilon)$ is the number of boxes of the side length ϵ required to cover the set. Then the box-counting dimension is defined as:

$$\dim_{box}(S) = \lim_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log(1/\epsilon)} \tag{45}$$

In Appendix more detailed presentation of properties of the Hausdorff and box-counting dimensions is included. Harrison in [5] recommends the box-counting dimension to be used only for closed sets, although even for compact sets it can differ from Hausdorff dimension and, moreover, the box dimension gives the most natural result than the measure $\mathcal{H}^s$.

8. Dimension of trajectory

By inventing the fractal (Hausdorff) dimension the trajectory of GA's or its attractor can be investigated. Algorithms could be regarded as equivalent if they have the same computational complexity while solving the same problem. As the measure of computational complexity of genetic algorithm, we propose a product of population's size and the number of steps after which an optimal solution is reached. This measure of computational complexity of genetic algorithms joins the memory and the temporal complexity.

During the execution of genetic algorithms, a trajectory is realized and should "converge" to some attraction set. It is expected that an ideal genetic algorithm produces an optimal solution which, in the term of its trajectory, leads to an attractor which is one{ element set. On the other hand, for an algorithm without selection the attractor is the whole space. Then, we could say that algorithms are equivalent when they produce similar attractors [6].

Our proposal is to use fractal dimensions to measure the similarity of attractors on the base of Lemma 1.

Definition 10. *Two genetic algorithms are* ***equivalent*** *if they realize trajectories with the same fractal dimension.*

Hence, instead of the entropy, the fractal dimension will be use as an indicator, or better to say - a measure of the classifications of GAs.

The transfer from the entropy to the new indicator can be made with the help of particular gauges. The first gauge could be the so-called ρ-entropy based dimension introduced by Pontrjagin and Schnirelman in 1932 (and repeated by Kolmogorov and Tihomirov in 1959), in the following way: among all collections of balls of radius ρ that cover a set E in $\boldsymbol{R}^l$ (or in more general case, in some metric space) is by definition one that requires the smallest number of balls. When E is bounded, this smallest number is finite and can be denoted by $\boldsymbol{N}(\rho)$ and called ρ - entropy. Their dimension, called the ***lower entropy dimension***, was defined by

$$\lim_{\rho \to 0} \inf \frac{\log \mathsf{N}(\rho)}{\log(\frac{1}{\rho})}. \tag{46}$$

The second gauge is the so-called *information dimension* of the trajectory defined by:

$$D_I(\mathcal{T}) = -\lim_{\epsilon \to 0} \frac{\sum_{i=1}^{W(\epsilon)} p_i \ln p_i}{\ln(\frac{1}{\epsilon})} \tag{47}$$

where $W(\epsilon)$ is the number of elements of the trajectory which are contained in a l - dimensional cube with the edge length equal to ϵ, and $p_i = \frac{N_i}{N}$ is the probability of finding of *i* - *th* element, and N_i - number of points in *i*-th hypercube, N - number of trajectory points. In further analysis we are going to replace (47) and (45) with its approximation, namely the *box* or *capacity dimension*.
In [6] the box counting dimension de_ned in [3] has been introduced with its approximated formula (cf. (2) in [6]).
Here we use another approach to the approximation. Let $N(\mathcal{T}, \epsilon)$ be the minimum number of r-dimensional cubes with the edge length equal to ϵ, that covers the trajectory $\mathcal{T} \subset X$, and X is a l- dimensional search space. To be more evident let us consider the case when $\epsilon = 2^{-k}$ and diminish the length of cube edges by half. Then the following ratio will approximate the box counting dimension of trajectory $\mathcal{T}$

$$D_c(\mathcal{T}) \approx \frac{\log_2 N(T, 2^{-(k+1)}) - \log_2 N(T, 2^{-k})}{\log_2 2^{k+1} - \log 2^k} = \log_2 \frac{N(T, 2^{-(k+1)}}{N(T, 2^{-k})} \tag{48}$$

due to the fact that $\log_2 x = \log_2 e \ \ln x$. The approximated expression (48) of the box dimension counts the increase in the number of cubes when the length of their edges is diminished by half.

8.1 Compression ratio

It is our conjecture that some characteristic feature of the trajectory of GA can be obtained by analysing the ration of the compressed trajectory to itself. We decided to investigate Lempel-Ziv compression algorithm [17] applied to populations executed by various genetic algorithms. We implemented five De Jong's functions with 10 different parameters sets. Each experiment was run 10 times. All together we obtained 500 different trajectories. The following settings of algorithms were considered

EXP	CROS	PC	PM	SEL
1	1	0.25	0.001	t
2	2	0.6	0.01	r
3	u	0.95	0.05	p
4	1	0.6	0.05	p
5	2	0.25	0.001	r
6	u	0.6	0.01	t
7	1	0.95	0.01	r
8	2	0.95	0.001	p
9	u	0.25	0.05	t
10	1	0.95	0.99	r

where EXP is the experiment number; CROS is type of crossover operator (one point, two point, uniform); PC and PM are probabilities of crossover and mutation, respectively; and SEL is type of selection operator (tournament, rank, and proportional). In each experiment the population consisted of 25 points and the genetic algorithm was realized on 100 generations (points).
We have performed numerous experiments on compressing particula generations with Lempel-Ziv algorithm of various bit resolution. We have measured number of prefixes resulting from compression process and corresponding compression ratio in scenarios of two types. The first one has considered single generations, and for each trajectory we have obtained corresponding trajectory of number of prefixes used. In the second scenario, each next generation was added to all past generations forming an ascending family of sets of generations. Compressing elements of such family gave an overall picture how number of prefixes used in the compression stabilizes over time.

8.2 Experiments with dimensions

The first experiments with attractors generated by GAs and the expression (48) have been performed by our co-worker in [6]. His results allow us to claim that the present approach can be useful in the GA's dynamics research.
In our paper we include new calculation results. 12 benchmark functions were used (cf. [13, 7]) in the analysis. Experiments were performed for different dimension: 10, 15, 20 bits with operator parameters and Popsize. Then the box counting dimension was used to calculate the trajectory dimension.

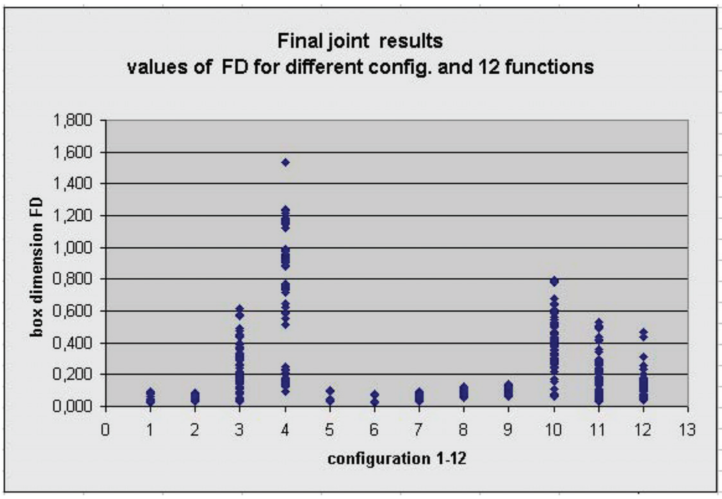


Fig. 1. Final joint results of fractal dimension

As far as the analytical approach and the formal definitions of dimensions (43) and (47) are concerned their computer implementation needs additional investigations. Computer accuracy is finite, hence all limits with e tending to zero will give unrealistic results. For example, if in (47) the calculation drops below the computing accuracy the expression value becomes zero or undefined. It means that we have to stop taking limit values in early stage.

Hence, the questions arise: to which minimal value of e the calculation should be performed and whether and how the relations with limits should be substituted with finite, non-asymptotic, expression? This, however, will be the subject of our further research.
The main idea of our experiments was the verification and confrontation of our theoretical considerations and conjectures with real genetic algorithms.

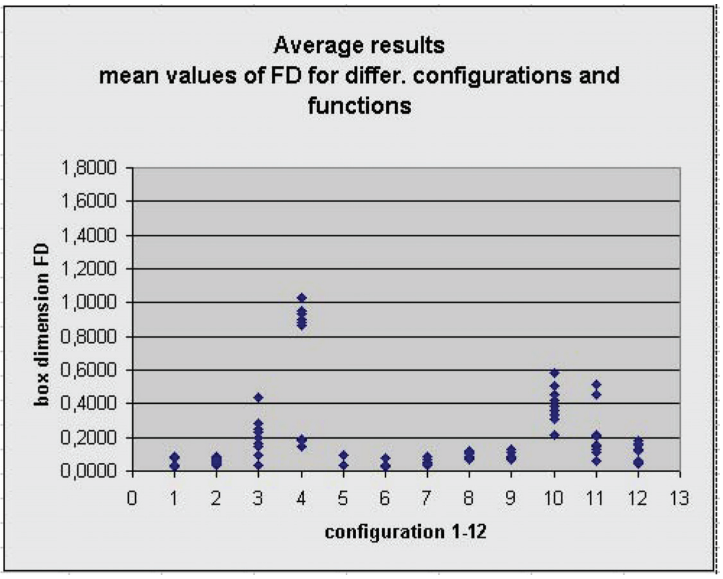


Fig. 2. Average results of fractal dimension

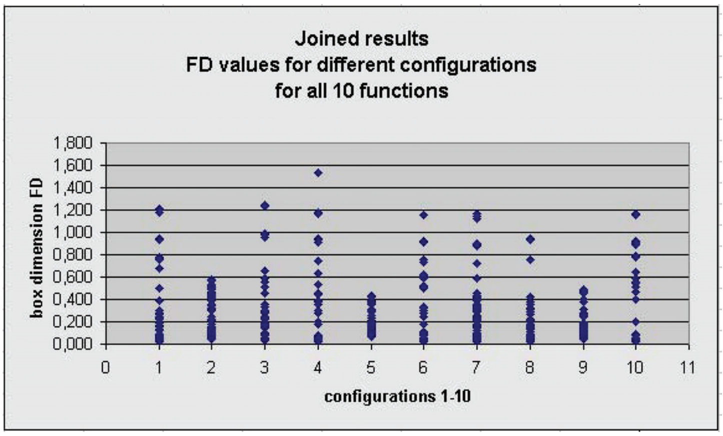


Fig. 3. Joint results of fractal dimension

On the basis of our experiments we can conclude that:
1. Selection.
Change of the selection methods while preserving the other parameters does not effect the values of fractal dimension.

2. Crossover.
When the number of crossover positions is changing the fractal dimension is growing with roulette selection method and is decreasing when selection is a tournament.
3. Populations.
Fractal dimension is growing with the number of individuals in population.
4. Mutation probability changes have small implication on the value of fractal dimension.
The analysis of the experimental result.
The value of box-counting dimension of the trajectory of genetic algorithms is not random. When we use the same fitness function and the same configurations, then the box dimensions become clustered near the same value. Whole trials of the independent running attains the same values. Moreover with the different functions but the same configuration we deal with the conservation of box-counting dimension clustering.
Average values of the box-counting dimension for the united trajectories of the algorithms from the same trial were similar to these which were calculated by averaging of the dimension of individual trajectories. This fact acknowledges the conjectures that box-counting dimension could characterize the complexity of algorithms. Box-counting dimension describes the way of evolution during search. Algorithms which attain the maximum in a wide loose set have bigger dimension than others which trajectories were narrow, with small differences between individuals.
One can say that bigger box dimension characterizes more random algorithms. The main result of the experiments states that fractal dimension is the same in the case when some boxes contains one individual as well as when these boxes contain many elements (individuals). Box dimension does not distinguish the fact that two or more elements are in the same place. They undergo counting as one element. The value of dimension should depend on the number of elements placed in each box. Our main conclusion is that good characterization is the information dimension.

9. Conclusions

One of the main results reported in this Chapter is the limiting algorithm and populations' distribution at the end of infinite steps. Theorem 5 does not tell about the form of the next population when actual population is known; it gives rather the limit distribution of all possible populations of the algorithm considered. The limiting algorithm describes globally the action of the genetic algorithm. It plays the role of the law of big numbers, known from the probability theory, however, for genetic algorithms. Knowledge the limiting algorithm could help in standard calculations: just in one step one could obtain the limit distribution. It could accelerate calculations and gives chance to omit the infinite numbers of calculation steps.
If the limiting algorithm is known an extra classification tool is for our disposal, and new hierarchial classification method can be suggested. It will base not only on entropy, fractal and dimensions of trajectory, but on transition matrix $\boldsymbol{T}$, its eigenvalues, eigenvectors and limiting matrix $\boldsymbol{Q}$. This hierarchie could be as follows:

- Two genetic algorithms are equivalent if their transition matrices are the same.
- Two genetic algorithms are equivalent if they have the same limit distribution $\boldsymbol{\pi}$.
- Two genetic algorithms are equivalent if their limiting algorithm, described by the matrix $\boldsymbol{Q}$ is the same.

- Two genetic algorithms are equivalent if the entropy of their trajectories is the same.
- Two genetic algorithms are equivalent if the fractal (box-counting, information, Hausdorff) dimensions of their trajectories are the same.
- Two genetic algorithms are equivalent if they generate the same order in populations.

We can see that the proposed scheme of classification referes to concepts known in the probability theory and the theory of dynamical systems. The open question is the role of different concepts and their importance. Is it possible to introduce the order relations in the proposed scheme? This will be investigated in the next publications.

10. Acknowledgement

The research work on the paper was partially done by W.K and S.K. in the framework of the KBN Project (State Committee for Scientific Research) No. 3 T11 C007 28. Authors thanks Professor Zbigniew Michalewicz and Dr. Jolanta Soca la for the inspiration and valuable discussions.

11. Appendix

Fractal and box - counting dimensions

To make the definitions more evident let us notice that for the graph Γ^f of a smooth, i.e. C^1, real function f of one variable we have $\dim_H(\Gamma^f) = 1$, while if the function f is C^ϵ (i.e. Hölder continuous of class ϵ) then $\dim_H(\Gamma^f) \leq 2 - \epsilon$. The Hausdorff dimension of the Peano curve has dimension 2 while the Hausdorff dimension of the Cantor middle set is log2=log3, while its topological dimension D^T is zero. In most cases Hausdorff dimension ≥ the topological one. In its classical form a **fractal** is by definition a set for which the Hausdorff dimension strictly exceeds the topological dimension.

Topological dimension takes non-negative integer values and is invariant under homeomorphism, while the Hausdorff dimension is invariant under bi-Lipschitz maps (sometimes called quasi-isometries). For self-similar sets ([5, 3]) that are built from pieces similar to the entire set but on a finer and finer scale, and can be regarded as an invariant set for a finite set of contraction maps on $\mathbf{R}^l$, the Hausdorff dimension is the same as its similarity dimension[8] It is the theory of fractal and its main object of interest, namely **iterated function systems** where fractal dimensions are commonly in use [2]. Deterministic and random algorithms are constructed for computing fractals from iterated function systems. However, such procedure are mostly implemented for 2D case, i.e. for fractals in $\mathbf{R}^2$. For genetic algorithm applications such tools are of small importance. More investigations on the similarities between genetic algorithms and iterated function systems with probabilities ([2]) are needed.

In fractal geometry, the Minkowski dimension is a way of determining the fractal dimension of a set S in a Euclidean space $\mathbf{R}^n$, or more generally of a metric space (X, d). This dimension is also, less accurately, sometimes known as the packing dimension or the box-counting

[8] Let **frig** be the contraction ratios of the family of contraction maps ($S_1, S_2, \ldots, S_m$) and E be the invariant set for this family, then the unique positive number s such that $\sum_{i=1}^{m} r_i^s = 1$ is the similarity dimension of E ([5]).

dimension. To calculate this dimension for a fractal *S*, imagine this fractal lying on an evenly-spaced grid, and count how many boxes are required to cover the set. The **box-counting** dimension is calculated by seeing how this number changes as we make the grid finer. Suppose that $N(\epsilon)$ is the number of boxes of side length ϵ required to cover the set. Then the box-counting dimension is defined as:

$$\dim_{box}(S) = \lim_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log(1/\epsilon)} \tag{49}$$

If the limit does not exist then one must talk about the **upper box dimension** and the **lower box dimension** which correspon to the upper limit and lower limit respectively in the expression above. In other words, the box-counting dimension is well defined only if the upper and lower box dimensions are equal. The upper box dimension is sometimes called the **entropy dimension**, Kolmogorov dimension, Kolmogorov capacity or upper Minkowski dimension, while the lower box dimension is also called the **lower Minkowski dimension**. Both are strongly related to the more popular Hausdorff dimension. Only in very specialized applications is it important to distinguish between the three. See below for more details. Also, another measure of fractal dimension is the correlation dimension.
Both box dimensions are finitely additive, i.e. if a finite collection of sets $\{A_1, A_2, \ldots, A_n\}$ is given then

$$\dim_{box}(\{A_1 \ldots \cup A_n\}) = \max\{\dim_{box}(A_1), \ldots, \dim_{box}(A_n)\} \ .$$

However, they are not countably additive, i.e. this equality does not hold for an infinite sequence of sets. For example, the box dimension of a single point is 0, but the box dimension of the collection of rational numbers in the interval [0, 1] has dimension 1. The Hausdorff dimension by comparison, is countably additive. An interesting property of the upper box dimension not shared with either the lower box dimension or the Hausdorff dimension is the connection to set addition. If *A* and *B* are two sets in a Euclidean space then *A* + *B* is formed by taking all the couples of points *a*, *b* where *a* is from *A* and *b* is from *B* and adding *a* + *b*. One has

$$\dim_{upperbox}(A + B) \leq \dim_{upperbox}(A) + \dim_{upperbox}(B).$$

Relations to the Hausdorff dimension The box-counting dimension is one of a number of definitions for dimension that can be applied to fractals. For many well behaved fractals all these dimensions are equal. For example, the Hausdorff dimension, lower box dimension, and upper box dimension of the Cantor set are all equal to log(2)/ log(3). However, the definitions are not equivalent. The box dimensions and the Hausdorff dimension are related by the inequality

$$\dim_H(S) \leq \dim_{lowerbox}(S) \leq \dim_{upperbox}(S) \tag{50}$$

In general both inequalities may be strict. The upper box dimension may be bigger than the lower box dimension if the fractal has different behaviour in different scales. For example, examine the interval [0, 1], and examine the set of numbers satisfying the condition for any *n*, all the digits between the 2^{2n}-th digit and the 2^{2n+1}-1-th digit are zero. The digits in the "odd places", i.e. between $2^{2n} + 1$ and $2^{2n+2} -1$ are not restricted and may take any value. This fractal has upper box dimension 2/3 and lower box dimension 1/3, a fact which may be easily verified by calculating $N(\epsilon)$ for $\epsilon = -10^{2^n}$ and noting that their values behaves differently for n even and odd. To see that the Hausdorff dimension may be smaller than the lower box dimension, return to the example of the rational numbers in [0, 1] discussed above. The Hausdorff dimension of this set is 0.

Box counting dimension also lacks certain stability properties one would expect of a dimension. For instance, one might expect that adding a countable set would have no effect on the dimension of set. This property fails for box dimension. In fact

$$\dim_{box}(\{0, 1, 1/2, 1/3, 1/4, ...\}) = \frac{1}{2}$$

It is possible to define the box dimensions using balls, with either the covering number or the packing number. The covering number $N_{covering}(\epsilon)$ is the minimal number of open balls of radius ϵ required to cover the fractal, or in other words, such that their union contains the fractal. We can also consider the intrinsic covering number $N'_{covering}(\epsilon)$, which is defined the same way but with the additional requirement that the centers of the open balls lie inside the set S. The packing number $N_{packing}(\epsilon)$ is the maximal number of disjoint balls of radius ϵ one can situate such that their centers would be inside the fractal. While N, $N_{covering}$, $N'_{covering}$ and $N_{packing}$ are not exactly identical, they are closely related, and give rise to identical definitions of the upper and lower box dimensions. This is easy to prove once the following inequalities are proven:

$$N'_{covering}(2\epsilon) \leq N_{covering}(\epsilon), \quad \text{and } N_{packing}(\epsilon) \leq N'_{covering}(\epsilon). \tag{51}$$

The logarithm of the packing and covering numbers are sometimes referred to as **entropy numbers**, and are somewhat analogous (though not identical) to the concepts of **thermodynamic entropy** and **information-theoretic entropy**, in that they measure the amount of "disorder" in the metric space or fractal at scale ϵ, and also measure how many "bits" one would need to describe an element of the metric space or fractal to accuracy ϵ.

Sometimes it is just too hard to find the Hausdorff dimension of a set E, but possible for other definitions that have some restriction on the ϵ -covers considered in the definition. We recall here the most common alternative. It is the **box dimension**, introduced by Kolmogorov in 1961 (cf.[5]), and which is defined in the same way as Hausdorff dimension except that in the definition of measure only balls (discs) in $\mathbf{R}^l$ of the same radius ϵ are considered for covers of E. It follows that box dimension of E is always ≥ dim(E). Moreover the box dimension of the closure of E is the same as for the set E itself. Since the box-counting dimension is so often used to calculate the dimensions of fractal sets, it is sometimes referred to as “fractal dimension”. We prefer the term box dimension, however, because sometimes the term “fractal dimension” might refer to box dimension, Hausdorff dimension, or even other measures of dimension such as the information dimension or capacity dimension.

Sometimes box counting dimension is referred to as “similarity dimension” in the context of self-similar sets. If a set is self-similar, there is an expansion factor r by which one can blow

up a small copy to get the whole set. If there are exactly N such small copies that make up the entire set, the box dimension is easily seen to be $\ln N/\ln r$.
Let us consider the $\mathcal{Q} = \{p/q | p, q \text{ integers}\}$ be the set of rational numbers in the interval [0, 1], that is $p \leq q$ are relatively prime integers. Since the rationals are dense in [0, 1], any interval we choose contains some. This means for every ϵ we need $N_\epsilon = \frac{1}{\epsilon}$ boxes to cover the whole $\mathcal{Q}$. Consequently $\dim_{box}(\mathcal{Q}) = \lim_{\epsilon \to 0} \frac{\frac{1}{\epsilon}}{\frac{1}{\epsilon}} = 1$. Thus the box dimension of the rational numbers is 1.
The last example will be given by the new set $\mathcal{P}$ = $\{x \in [0, 1]\}$ x has a decimal expansion which does not contain 4 nor 5. Notice that 0.4 has the two representations, namely .4 and .39999(9). The set $\mathcal{P}$ is disconnected: it does not contain the open interval (0.4, 0.6). We shall see that the set is closed and also self-similar: any small piece of it can be scaled up to look like the whole thing just by multiplying by an appropriate power of 10. It can be proven that

$$\dim_{box} \mathcal{P} = -\lim_{n \to \infty} \frac{\ln 8^{n-1}}{\ln (4 \cdot 10^{-n})} = \frac{\ln 8}{\ln 10} =\approx 0.903089987$$

At the same time the topological dimension of $\mathcal{P}$ is zero.

12. References

Baker G.L. and Gollub J.P.: *Chaotic Dynamics: an Introduction*, Cambridge Univ. Press, Cambridge, 1992.

Barnsley M. F.: Lecture notes on iterated function systems, in *Chaos and Fractals.The Mathematics Behind the Computer Graphics, Proc.Symp. Appl. Math.*, vol. 39, R. L. Denamney and L. Keen (eds.) American Mathematical Society, Providence, Rhode Island, pp. 127-144, 1989.

Falconer K.J.: Fractal geometry, *Math. Found. Appl.* John Wiley, Chichester, pp.15 -25, 1990.

Friedman N.A., Ornstein D.S.: On isomorphisms of weak Bernoulli transformations, *Adv. in Math.* , 5, pp. 365-394, 1970.

Harrison J.: An introduction to fractals, in *Chaos and Fractals. The Mathematics Behind the Computer Graphics, Proc.Symp. Appl. Math.*, vol. 39, R. L. Denamney and L. Keen (eds.) American Mathematical Society, Providence, Rhode Island, pp. 107-126, 1989.

Kieś P.: Dimension of attractors generated by a genetic algorithm, in *Proc. of Workshop Intelligent Information Systems IX* held in Bystra, Poland, June 12-16, pp. 40-45, 2000.

Michalewicz Z.: *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd, rev. edition, Springer, Berlin, Heidelberg *et al.*, 1996.

Ossowski A.: Statistical and topological dynamics of Statistical and topological dynamics of evolutionary algorithms, in *Proc. of Workshop Intelligent Information Systems IX* held in Bystra, Poland, June 12-16, pp. 94-103, 2000.

Ott E.: *Chaos in Dynamical Systems* Cambridge Univ. Press, Cambridge, 1996.

Ornstein D.S.: *Ergodic theory, Randomness and Dynamical Systems*, Yale Univ. Press, 1974.

Vose M.D.: Modelling Simple Genetic Algorithms, *Evolutionary Computation*, 3 (4) 453-472, 1996.

Wolpert D.H. and Macready W.G.: No Free Lunch Theorems for Optimization, *IEEE Transaction on Evolutionary Computation*, 1 (1), 67-82, 1997, http://ic.arc.nasa.gov /people/dhw/papers/78.pdf

Igel, C., and Toussaint, M.: "A No-Free-Lunch Theorem for Non-Uniform Distributions of Target Functions," *Journal of Mathematical Modelling and Algorithms* 3, 313-322, 2004.

English, T.: No More Lunch: Analysis of Sequential Search, *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pp. 227-234. 2004, http://BoundedTheoretics.com /CEC04.pdf

Szlenk W., *An Introduction to the Theory of Smooth Dynamical Systems.*,PWN, Warszawa,John Wiley&Sons, Chichester, 1984 G.H.

Choe G. H., *Computational Ergodic Theory*. Springer, Heidelber, New York 2005

G. Frizelle G., Suhov Y.M.: An entropic measurement of queueing behaviour in a class of manufacturing operations. *Proc. Royal Soc. London A* (2001) 457, 1579- 1601.

A. Lasota, Asymptotic properties of semigroups of Markov operators (in Polish), *Matematyka Stosowana. Matematyka dla Społeczeństwa*,PTM, Warszawa, 3(45), 2002, 39-51.

P. Kieś and Z. Michalewicz, Foundations of genetic algorithms (in Polish), *Matematyka Stosowana. Matematyka dla Społeczeństwa* , PTM Warszawa 1(44), 2000, 68{91.

Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, 1996.

M. D. Vose, *The Simple Genetic Algorithm: Foundation and Theory*, MIT Press, Cambridge, MA, 1999.

A. Lasota, J. A. Yorke, Exact dynamical systems and the Frobenius-Perron operator, *Trans. Amer. Math. Soc.* 273 (1982), 375{384.

J. E. Rowe, The dynamical system models of the simple genetic algorithm, in *Theoretical Aspects of Evolutionary Computing*, Leila Kallel, Bart Naudts, Alex Rogers (Eds.), Springer, 2001, pp.31-57.

R. Schaefer, *Podstawy genetycznej optymalizacji globalnej* (in Polish), Wydawnictwo Uniwersytetu Jagiellońskiego, Kraków 2002.

R. Schaefer, *Foundations of Global Genetic Optimization*, Series: Studies in Computational Intelligence, Vol. 74, Springer, Berlin, Heidelberg, 2007,

R. Rudnicki, On asymptotic stability and sweeping for Markov operators, *Bull. Polish Acad. Sci. Math.*, 43 (1995), 245-262.

J. Socała, Asymptotic behaviour of the iterates of nonnegative operators on a Banach lattice, *Ann. Polon. Math.*, 68 (1), (1998), 1-16.

J. Socała, W. Kosiński, S. Kotowski, On asymptotic behaviour of a simple genetic algorithm (in Polish: O asymptotycznym zachowaniu prostego algorytmu genetycznego), *Matematyka Stosowana. Matematyka dla Społeczeństwa* , PTM, Warszawa,6 (47), 2005, 70-86.

J. Socała, W. Kosiński, Lower-bound function method in the converegence analysis of genetic algorithms, (in Polish: Zastosowanie metody funkcji dolnej do badania zbieżności algorytmów genetycznych, *Matematyka Stosowana. Matematyka dla Spo leczeństwa*, PTM, Warszawwa, 8 (49), 2007 , 33-44.

J. Socała, W. Kosiński, On convergence of a simple genetic algorithm, *ICAICS, 9-th International Conference on Artifical Intelligence and Soft Computing, 2008*, LNAI, Springer, Berlin, Heidelberg, poz.366, in print.

J. Socała, Markovian approach to genetic algorithms, under preparation.

S.Kotowski, W. Kosiński, Z. Michalewicz, J. Nowicki, B. Przepiórkiewicz, Fractal dimension of trajectory as invariant of genetic algorithms, *ICAICS, 9-th International Conference on Artifical Intelligence and Soft Computing, 2008*, LNAI, Springer, Berlin, Heidelberg, poz.401, in print.

S. Kotowski, Analysis of genetic algorithms as dynamical systems (in Polish: Analiza algorytmów genetycznych jako układów dynamicznych, under preparation, 2008.

2

Evolutionary Systems Identification: New Algorithmic Concepts and Applications

Michael Affenzeller, Stephan Winkler and Stefan Wagner
Upper Austrian University of Applied Sciences
School of Informatics, Communications and Media
Heuristic and Evolutionary Algorithms Laboratory
Softwarepark 11, 4232 Hagenberg,
Austria

1. Introduction

There are many problems in various theoretical and practical disciplines that require robust structure identification techniques with as few restricting assumptions in terms of model structure and potentially relevant input variables (features) as possible. Due to its implicit variable selection and the possibility to identify also nonlinear model structures, the basic concept of Genetic Programming (GP) has the required descriptive potential and provides results in form of easily interpretable formulae as an additional benefit. However, when using standard GP techniques, the potential of GP is still rather limited and restricted to special applications.
This chapter presents further developed algorithmic concepts which can be combined with a Genetic Algorithm (GA) as well as with Genetic Programming (GP). Especially the latter combination provides a very powerful, generic and stable algorithm for the identification of nonlinear systems, no matter if the application at hand is in the context of regression, classification or time-series analysis.
After a general introduction in heuristic optimization and Evolutionary Algorithms, the further developed algorithmic concepts are explained. Furthermore, some exemplary applications of Genetic Programming to data based system identification problems are illustrated.

2. Heuristic optimization

Many practical and theoretical optimization problems are characterized by their highly multimodal search spaces. These problems include NP-hard problems of combinatorial optimization, the identification of complex structures, or multimodal function optimization. In the area of production planning and logistics such problems occur especially frequently (as for example task allocation, routing, machine sequencing, container charging). The application of conventional methods of Operations Research (OR) like dynamic programming, the simplex method, or gradient techniques, often fails for these kinds of problems, because the computation effort grows exponentially with the problem dimension.
Therefore, heuristic methods with much lower computational costs are applied quite frequently, even if they can no longer assure the achievement of a global optimal solution.

About three decades ago, inspired by nature, literature started to discuss generic heuristic methods which often surpass problem specific heuristics and are moreover much more flexible concerning modifications in the problem definition.
These optimization techniques derived from nature include Simulated Annealing (SA) which draws an analogy between the annealing of material to its lowest energetic state and an optimization problem, or Evolutionary Algorithms (EAs) which are basically inspired by biological evolution. Further recent approaches like Tabu Search (TS), Ant-Colony Optimization (ACO), or Particle Swarm Optimization (PSO) are also mentionable in the context of bionically inspired optimization techniques. Agent theory is also on the verge of achieving greater importance in the field of heuristic optimization.

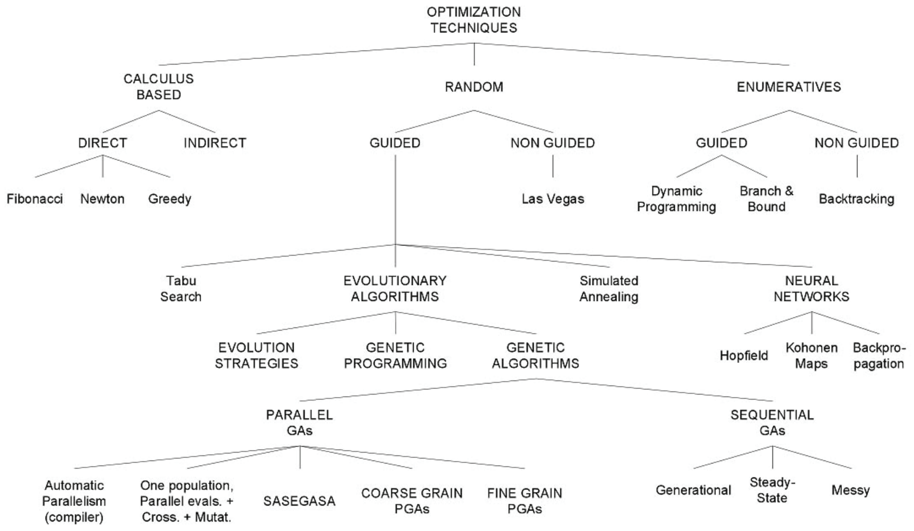


Fig. 1. Taxonomy of optimization techniques

A well-established taxonomy of optimization techniques is given in Fig. 1 whereby our classification describes those classes of methodologies in more detail which are more relevant in the context of the present contribution. This has especially been done for the class of Evolutionary Algorithms which is described in further detail in the following section. The detailed analysis of variants of Genetic Algorithms as shown in Fig. 1 can in principle also be applied to Genetic Programming since it is based on the same algorithmic and methodological concepts.

3. Evolutionary computation

3.1 Evolutionary algorithms: genetic algorithms, evolution strategies and genetic programming

Literature generally distinguishes Evolutionary Algorithms into Genetic Algorithms (GAs), Evolution Strategies (ES), and Genetic Programming (GP).
Genetic Algorithms, possibly the most prevalent representative of Evolutionary Computation, were first presented by Holland (Holland, 1975). Based upon Holland's ideas the concept of the Standard Genetic Algorithm (SGA), which is still very much influenced by the biological archetype, became accepted (described e.g. in (Tomassini, 1995). Due to the

enormous increase of computational power since 1975, the potential of GAs has been tapped more and more. Consequently the popularity of GA-concepts increased steadily and many groups around the world started to solve various problems with GAs. However, it soon became clear that for most practical tasks the binary encoding originally used by Holland was not at all sufficient. Accordingly many different encodings, and also necessary new crossover and mutation operators, were introduced which showed qualitatively very diverse behavior. An overview of different encodings and operators developed for various applications can for instance be found in (Dumitrescu et al., 2000). Since then GAs have been successfully applied to a wide range of problems including many combinatorial optimization problems, multimodal function optimization, machine learning, and the evolution of complex structures such as neural networks. An overview of GAs and their implementation in various fields is given by Goldberg (Goldberg, 1989) and Michalewicz (Michalewicz, 1996).

Evolution Strategies, the second major representative of Evolutionary Algorithms, were introduced by Rechenberg (Rechenberg, 1973) and Schwefel (Schwefel, 1994). Evolution Strategies tend to find local optima quite efficiently. Though, in the case of multimodal solution spaces, Evolution Strategies tend to detect a global optimum hardly, if none of the starting values is located in the absorbing region of such a global optimum. Nevertheless, ES have lead the way in the implementation of self-adaptive concepts in the area of Evolutionary Computation and are considered one of the most powerful and efficient concepts for the optimization of real-valued parameter vectors.

Genetic Programming (GP) has been established as an independent branch in the field of Evolutionary Computation even if this technique could also be interpreted as a special class of GAs. Based on the basic considerations of Koza (Koza, 1992) to interpret the underlying problem representation in a more general and dynamic way than a usual GA, the basic mechanisms of selection, recombination, and mutation are adapted and applied in a similar manner as found within GAs. The more general problem representation of GP allows the definition of individuals of a population as structures, formulas, or even more generally as programs. This allows the consideration of new applications of EAs like data based systems identification, for example; however, it still seems to be a very ambitious goal to generate more complex programs by means of Genetic Programming.

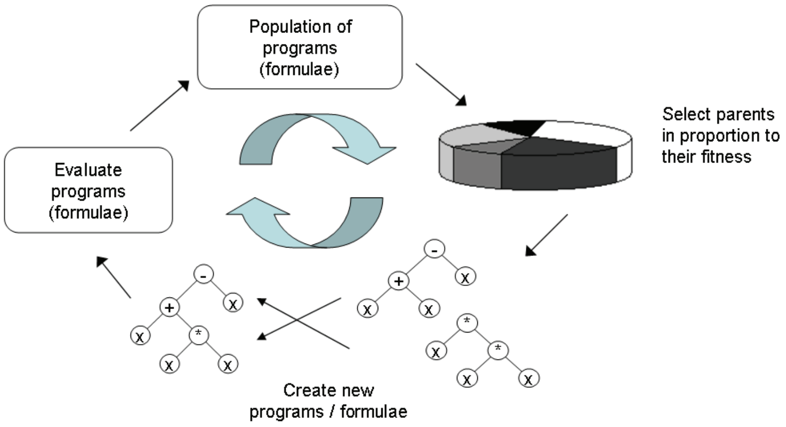

Population of programs (formulae)
Select parents in proportion to their fitness
Evaluate programs (formulae)
Create new programs / formulae
-
+
*
X

Fig. 2. The GP Lifecycle (Langdon & Poli, 2002)

In (Koza, 1992) it has been pointed out that virtually all problems in artificial intelligence, machine learning, adaptive systems, and automated learning can be recast as a search for a computer program, and that genetic programming provides a way to successfully conduct the search for a computer program in the space of computer programs. Similar to GAs, GP works by imitating aspects of natural evolution: A population of solution candidates evolves through many generations towards a solution using evolutionary operators (crossover and mutation) and a "survival-of-the-fittest" selection scheme. Whereas GAs are intended to find an array of characters or integers representing the solution of a given problem, the goal of a GP process is to produce a computer program solving the optimization problem at hand. As in every evolutionary process, new individuals (in GP's case, new programs) are created. They are tested, and the fitter ones in the population succeed in creating children of their own. Unfit ones die and are removed from the population (Langdon & Poli, 2002). This procedure is graphically illustrated in Fig. 2.

3.2 Considerations about selected theoretical aspects of evolutionary computation techniques

Fig. 1 indicates that this classification - especially of the bionic methods - is mainly inspired by the natural role-model. For a more directed consideration of algorithmic concepts of the different methods, it is reasonable to differentiate these methods by their basic idea. One possible (and especially in the context of further considerations drawn in this paper) well-suited classification is the distinction between neighbourhood-based and non-neighbourhood-based search techniques as illustrated in Fig. 3.

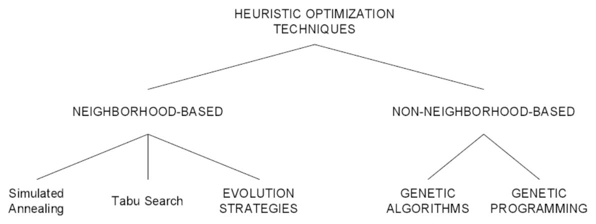


Fig. 3. Classification of heuristic optimization techniques due to their mode of operation

As some kind of approximation for the gradient information which is not available for problems of combinatorial optimization, a conventional neighbourhood search aims to obtain information about the descent/increase of the objective function in the local neighbourhood at a certain point. Conventional neighbourhood searches start from an arbitrary point in the solution space and iteratively move to more and more promising points along a given neighbourhood structure (with respect to the objective function) as long as no better solution can be detected in the local neighbourhood. The self-evident drawback of this method is that for more complex functions the algorithm converges and gets stuck in the next attracting local optimum which is often far away of a global optimum. It is a common feature of all methods based upon neighbourhood searches to counteract this essential handicap. Simulated Annealing, on the one hand, also allows moves to worse neighbourhood solutions with a certain probability which decreases as the search process progresses in order to scan the solution space broader at the beginning, and to become more and more goal-oriented as the search process goes on. A Tabu Search on the other hand introduces some kind of memory in terms of a so-called tabu list which stores moves that are considered to lead to already visited areas of the search space. However, Evolution Strategies (ES), a well-known representative of Evolutionary Computation, also have to be considered as some kind of parallel neighbourhood search, as asexual mutation (a local operator) is the only way to create new individuals (solution candidates) in the standard ES-versions. Therefore, in the case of multimodal test functions, global optima can be detected by Evolution Strategies only if one of the starting values is located in the absorbing region of a global optimum.

Genetic Algorithms (and certainly also GP), the non-neighbourhood-based search techniques in our classification of heuristic methods, take a fundamentally different approach to optimization, by considering recombination (crossover) as their main operator, whereas the essential difference to neighbourhood-based techniques is given by the fact that recombination is a sexual operator, i.e. properties of individuals from different regions of the search space are combined in new individuals. Therefore, provided that the used problem representation and the operators are adequate, the advantage of applying GAs to hard optimization problems lies in their ability to search broader regions of the solution space than heuristic methods based upon neighbourhood search do. Nevertheless, GAs are also frequently faced with a problem which, at least in its impact, is quite similar to the problem of stagnating in a local but not global optimum. This drawback, called premature convergence in the terminology of GAs, occurs when the population of a GA reaches such suboptimal state that the genetic operators can no longer produce offspring which outperform their parents (Fogel, 1994).

A very essential question about the general performance of a GA is, whether or not good parents are able to produce children of comparable or even better fitness (the building block hypothesis implicitly relies on this). In natural evolution, this is almost always true. For Genetic Algorithms this property is not so easy to guarantee. The disillusioning fact is that the user has to take care of an appropriate coding in order to make this fundamental property hold. In order to overcome this strong requirement we have developed an advanced selection mechanism (Affenzeller & Wagner 2004) which is based on the idea to consider not only the fitness of the parents, in order to produce a child for the ongoing evolutionary process. Additionally, the fitness value of the evenly produced offspring is compared with the fitness values of its own parents. The offspring is accepted as a candidate for the further evolutionary process if and only if the reproduction operator was able to produce an offspring that could outperform the fitness of its own parents. This strategy guarantees that evolution is presumed mainly with crossover results that were able to mix the properties of their parents in an advantageous way. Via these means we are already in a position to attack one of the reasons for premature convergence. Furthermore, this strategy has proven to act as a precise mechanism for self-adaptive selection pressure steering, which is of major importance in the migration phases of parallel Evolutionary Algorithms. All these new generic concepts are very promisingly combined in the SASEGASA-algorithm (Affenzeller & Wagner, 2004). Even if the aspect of parallelization is mainly used to improve global convergence in our research so far, the next obvious step is to transform these massively parallel concepts to parallel computing environments. Furthermore, already established parallel GAs should benefit from the recently developed new theoretical concepts as the essential genetic information can be assembled much more precisely in the migration phases.

4. Advanced algorithmic concepts for genetic algorithms

4.1 General remarks on variable selection pressure within genetic algorithms

Our first attempt for adjustable selection pressure handling was the so-called Segregative Genetic Algorithm (SEGA) (Affenzeller, 2001) which introduces birth surplus in the sense of a (μ, λ)-Evolution Strategy (Beyer, 1998) into the general concept of a GA and uses this enhanced flexibility primary for adaptive selection pressure steering in the migration phases of the parallel GA in order to improve achievable global solution quality. The SASEGASA, which stands for Self Adaptive Segregative Genetic Algorithm with aspects of Simulated Annealing, is a further development of SEGA and distinguishes itself mainly in its ability to self-adaptively adjust selection pressure in order to achieve progress in solution quality without loosing essential genetic information which would lead to unwanted premature convergence. The SASEGASA is generic in that sense that all algorithmic extensions are problem-independent so that they do not depend on a certain problem representation and the corresponding operators.

Therefore we have decided to combine the further deloped algorithmic concepts of SASEGASA with Genetic Programming (GP). However, we have observed two major differences when combining SASEGASA and Genetic Programming compared to the experience in the application of SASEGASA in other domains like combinatorial optimization or real-valued optimization (Affenzeller, 2005):

- The potential in terms of achievable solution quality in comparison with the standard algorithms seems to be considerably higher in the field of GP than in standard applications of GAs.
- By far not all algorithmic extensions of SASEGASA are relevant in GP. Only some algorithmic aspects of the rather complex SASEGASA concept are really relevant in the GP domain which makes the handling and especially parameter adjustment easier and more robust.

Therefore, the discussion in this article will focus on the algorithmic parts of SASEGASA which are really relevant for GP. In doing so, this section is structured as follows: The first subsection describes the general idea of SASEGASA in a quite compact way, whereas the second subsection focusses on that parts of SASEGASA in further detail which are really relevant for the present contribution and discusses the reasons for that.

For a more detailed description of all involved algorithmic aspects the interested reader is referred to the book (Affenzeller, 2005).

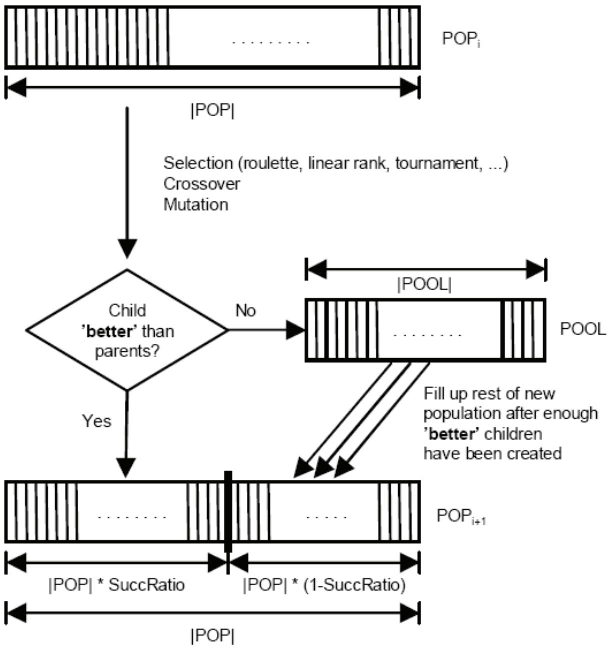
In principle, the SASEGASA introduces two enhancements to the basic concept of Genetic Algorithms. Firstly, it brings in a variable and self-adaptive selection pressure in order to control the diversity of the evolving population in a goal-oriented way w.r.t. the objective function. The second concept introduces a separation of the population to increase the broadness of the search process and joins the subpopulation after their evolution in order to end up with a population including all genetic information sufficient for locating a global optimum.

At the beginning of the evolutionary process the whole population is divided into a certain number of subpopulations. These subpopulations evolve independently from each other until the fitness increase stagnates in all subpopulations because of too similar individuals within the subpopulations, i.e. local premature convergence. Thanks to offspring selection

this can be triggered exactly when an upper limit of selection pressure is exceeded (cf. Subsection 4.2). Then a reunification from *n* to *(n-1)* subpopulations is performed by joining an appropriate number of adjacent subpopulation members.
Metaphorically speaking this means, that the villages (subpopulations) at the beginning of the evolutionary process are slowly growing together to bigger towns, ending up with one big city containing the whole population at the end of evolution. By this approach of width-search essential building blocks can evolve independently in different regions of the search space at the beginning and during the evolutionary process.

4.2 Offspring selection in SASEGASA

In (Affenzeller & Wagner, 2004) it has been shown that the aspect of segregation and reunification is highly relevant in order to systematically improve the achievable global solution quality of combinatorial optimization problems as for example the travelling salesman problem (TSP). Still, we have not used this parallel approach for our GP-based modelling studies. On the one hand, this would lead to a high increase of runtime consumption; on the other hand, anyway, we do not expect any significant increase of solution quality using this concept for GP-based modelling as results summarized in (Affenzeller, 2005) indicate that this parallel approach does not remarkably effect the solution quality of optimization problems others than combinatorial problems.
A very essential question about the general performance of GAs or GP is, whether or not good parents are able to produce children of comparable or even better fitness (the building block hypothesis implicitly relies on this). In natural evolution, this is almost always true. For artificial evolution and exceptionally for Genetic Programming this property is not so easy to guarantee. Offspring selection assures exactly that property.
Offspring selection considers not only the fitness of the parents, in order to produce a child for the ongoing evolutionary process. Additionally, the fitness value of the evenly produced offspring is compared with the fitness values of its own parents. The offspring is accepted as a candidate for the further evolutionary process if and only if the reproduction operator was able to produce an offspring that could outperform the fitness of its own parents. This strategy guarantees that evolution is presumed mainly with crossover results that were able to mix the properties of their parents in an advantageous way.
As in the case of conventional GAs, or GP, offspring are generated by parent selection, crossover, and mutation. In a second (offspring) selection step, the number of offspring to be generated is defined to depend on a predefined ratio-parameter giving the quotient of next generation members that have to outperform their own(!) parents (success ratio, *SuccRatio*). As long as this ratio is not fulfilled, further children are created and only the successful offspring will definitely become members of the next generation; this procedure is illustrated in Fig. 4. When the postulated ratio is reached, the rest of the next generation members are randomly chosen from the children that did not reach the success criterion. Within our new selection model, selection pressure is defined as the ratio of generated candidates to the population size. An upper limit for selection pressure gives a quite intuitive termination heuristics: If it is no more possible to find a sufficient number of offspring that outperform their parents, the algorithm terminates in the simple version as being used here or new genetic information is brought in by reunification in the more general formulation of the parallel SASEGASA.



POP$_i$
|POP|
Selection (roulette, linear rank, tournament, ...)
Crossover
Mutation
|POOL|
Child 'better' than parents?
No
POOL
Yes
Fill up rest of new population after enough 'better' children have been created
POP$_{i+1}$
|POP| * SuccRatio
|POP| * (1-SuccRatio)
|POP|

Fig. 4. Flowchart for embedding offspring selection into a Genetic Algorithm.

5. Data based systems identification

Data mining is understood as the practice of automatically searching large stores of data for patterns. Incredibly large (and quickly growing) amounts of data are collected not only in commercial, administrative, and scientific, but also in medical databases; this is the reason why intelligent computer systems that can extract useful information (such as general rules or interesting patterns) from large amounts of observations are needed. In short, "data mining is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" (Fayyad et al. 1996). This is why data based machine learning algorithms have to be applied in order to retrieve additional insights into human biological processes, how environment factors influence human health or how certain human parameters are related. The following three classes of data analysis problems are relevant within medical data analysis: Regression, classification and time series analysis. In any of these cases, statistical algorithms are supposed to "learn" functions by analyzing a set of input-output examples ("training samples").

In statistics, **regression analysis** is understood as the act of modelling the relationship between variables, namely between one or more target ("dependent") variables and other variables (also called input or explanatory variables). I.e., the goal is to find a mathematical function f which can be used for calculating the target variable Y using the input variables $X_{1..p}$:

$$Y = f(X_1, ..., X_p)$$

Classification is understood as the act of placing an object into a set of categories, based on the object's properties. Objects are classified according to an (in most cases hierarchical) classification scheme also called taxonomy. A statistical classification algorithm is supposed to take feature representations of objects and map them to a special, predefined classification label. Such a classification algorithm is designed to learn a function f which maps a vector of object features $X_1,\dots,X_p$ into one of several classes. A given sample x_i can so be classified using f and $X_1,\dots,X_p$:

$$\text{Class}(x_i) = f(X_{1(i)}, \dots, X_{p(i)})$$

There are several approaches which are nowadays used for solving classification problems; the most common ones are (as described in (Mitchell, 2000), e.g.) decision tree learning, instance-based learning, inductive logic programming (such as in Prolog, e.g.) and reinforcement learning.

Finally, there are two main goals of **time series analysis**: On the one hand one tries to identify the cause of a phenomenon represented by a sequence of observations and its relationships with other sequences of observations, and on the other hand the goal is to predicting future values of time series variables. Both of these goals require that the pattern of observed time series data is identified and more or less formally described. I.e., for the target variable Y one wants to identify a function f so that Y at time t can be calculated using values of other variables and (if available) also information about the history of Y:

$$Y(t) = f(X_{1(t-\{0..z\})}, \dots\dots, X_{p(t-\{0..z\})}, Y_{(t-\{0..z\})})$$

where z is the maximum time offset for variables used in f. Detailed discussions of time series and methods applicable can for example be found in (Box & Jenkins, 1976) or Kendall & Ord, 1990).

6. GP-Based structure identification

6.1 Introduction, general remarks

The concept of structure identification is not very common in the literature. Indeed, it is well known that every model consists of an equation set (the structure) and of values (parameters). System identification actually implies both, but usually the definition of the structure is considered either obvious or as the less critical issue, while the consistent estimation of the parameters especially in presence of noise receives the largest part of the attention. By its very general problem statement, GP allows to approach the problem of structure identification and the problem of parameter identification simultaneously. As a consequence, GP techniques are used for identifying various kinds of technical systems; some approaches use genetic programming to identify the structure in addition to standard parameter estimation techniques, many other ones use GP for determining both the structure and the parameters of the model of a nonlinear system as for example described in (Rodriguez et al., 2000) and (Beligiannis et al., 2005).

GP-based, data driven systems identification works on a set of training examples with known properties ($X_1 \dots X_n$). One of these properties (X_t) has to represent the system's target values. On the basis of the training examples, the algorithm tries to evolve (or, as one could also say, to "learn") a solution, i.e. a formula, that represents the function that maps a vector of input values to the respective target values. In other words, each presented instance of the structure identification problem is interpreted as an instance of an optimization problem; a solution is found by a heuristic optimization algorithm. Further details about the operators used are given for example in (Winkler et al., 2006a). The goal of the implemented GP identification process is to produce an algebraic expression from a database containing the measured results of the experiments to be analyzed. Thus, the GP algorithm works with solution candidates that are tree structure representations of symbolic expressions. These tree representations consist of nodes and are of variable length; the nodes can either be nonterminal or terminal ones:

- *Nonterminal* nodes represent functions performing some actions on one or more property values within the structure to produce the values of the target property (which should be the property which indicates which class the objects belong to);
- A *terminal* node represents an input variable (i.e., a pointer to one of the objects' properties) or a constant.

The nonterminal nodes have to be selected from a library of possible functions, a pool of potential nonlinear model structural components; as with every GP modeling process, the selection of the library functions is an important part since this library should be able to represent a wide range of systems. When the evolutionary algorithm is executed, each individual of the population represents one structure tree.

Since the tree structures have to be usable by the evolutionary algorithm, mutation and crossover operators for the tree structures have to be designed. Both crossover and mutation processes are applied to randomly chosen branches (in this context a branch is the part of a structure lying below a given point in the tree). Crossing two trees means randomly choosing a branch in each parent tree and replacing the branch of the tree, that will serve as the root of the new child (randomly chosen, too), by the branch of the other tree.

Mutation in the context of genetic algorithms means modifying a solution candidate randomly and so creating a new individual. In the case of identifying structures, mutation works by choosing a node and changing it: A function symbol could become another function symbol or be deleted, the value of a constant node or the index of a variable could be modified. This procedure is less likely to improve a specific structure but it can help the optimization algorithm to reintroduce genetic diversity in order to re-stimulate genetic search.

Examples of genetic operations on tree structures are shown in Fig. 5: The crossover of parent1 (representing the expression “$5/x_1(t-5)+\ln(x_2(t-2))$” and parent2 (“$x_3(t) * x_2(t-1)-1.5$”) yields child1 (“$5/x_1(t-5)+x_3(t)*x_2(t-1)$”), child2 and child3 are possible mutants of child1 representing “$5/x_1(t-5)+x_3(t)$” and “$5-x_1(t-5)+x_3(t-1)*x_2(t)$”.

Since the GP algorithm tries to maximize or minimize some objectiv fitness function (better model structures evolve as the GP algorithm minimizes the fitness function), every solution candidate has to be evaluated. In the context of data based modeling, this function should be an appropriate measure of the level of agreement between the original target variable's values and those calculated using the model to be evaluated. Calculating the sum of squared errors *J* between original values o_i and calculated values c_i is a simple as well as robust measurement of the quality of the formula at hand:

$$\mathrm{J} = \sum_{i=1}^{n} (o_i - e_i)^2$$

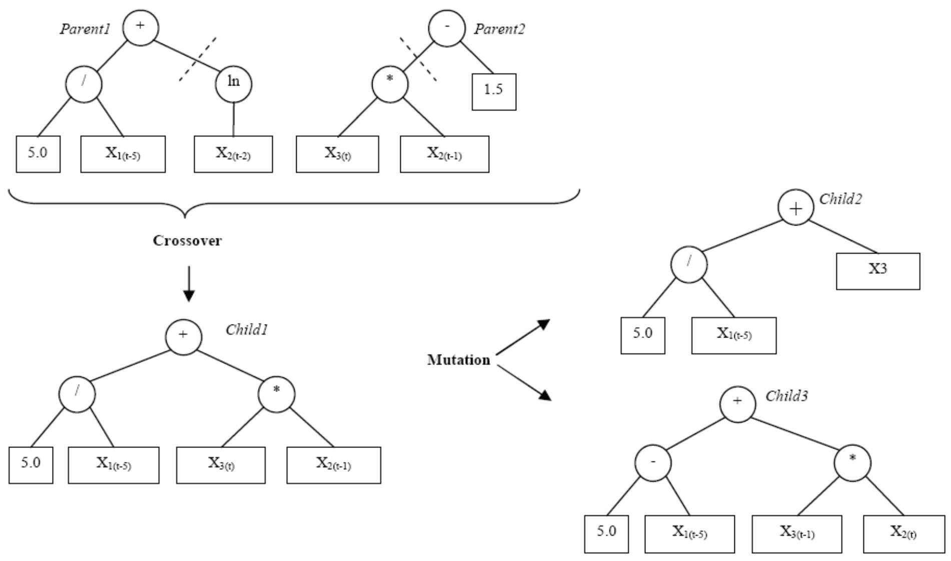


Fig. 5. Genetic Operations on Tree Structures.

6.2 GP Based structure identification: the HeuristicModeler

On the basis of preliminary work on the identification of nonlinear structures in dynamic technical systems (Winkler et al, 2005a), (Winkler et al, 2005b), (Del Re et al, 2005) as well as several other enhanced algorithmic and problem specific mechanisms we have implemented the HeuristicModeler (Winkler et al, 2006c), a multi-purpose machine learning algorithm that is able to evolve models for various different machine learning problem classes. The framework used for the implementation of the HeuristicModeler is the HeuristicLab (Wagner & Affenzeller, 2005), a framework for prototyping and analyzing optimization techniques for which both generic concepts of evolutionary algorithms and many functions for analyzing them are available.

The algorithmic basis for the HeuristicModeler is the SASEGASA (for an explanation see Section 4. There are several new hybrid evolutionary concepts combined in this algorithmic basis, the most important ones being on the one hand the self-adaptive selection pressure steering and on the other hand the so-called Offspring Selection concept.

The selection pressure measures how hard it is to produce individuals out of the current population that improve the overall fitness. As soon as this internal selection pressure reaches a pre-defined maximum value, the algorithm is terminated and presents the best actual model as the result of the training process. Details can be found in (Affenzeller & Wagner, 2004) and (Affenzeller, 2005).

As already explained in further detail in Section 4, the basic idea of Offspring Selection is that individuals are first compared to their own parent solution candidates and accepted as members of the new generation's population if they meet certain criteria. In the context of structure identification and machine learning we have realized that the use of very rigid settings yields best results (Winkler et al., 2006b).

Research results obtained within the last two years have lead to the conclusion that a simplified version of offspring selection together with a slightly modified parent selection shows the best and most robust results in the context of GP-applications. Thus, *SuccRatio* should be set to *1.0*, i.e. every offspring for the next generation is forced to pass the success criterion. Furthermore, it is beneficial in GP applications to state that a child is better than its parents if and only if it is better than the better of the two parents. In the context of combinatorial optimization problems where some intermediate value of the parents fitness values is used as a threshold value for the success criterion, such settings would massively tend to support premature convergence. But in the field of Genetic Programming applications these parameter settings lead to high-quality results quite robustly.

However, there is one aspect concerning parent selection that is to considered in this application domain. It is - applying the parameter settings of offspring selection mentioned above – most effective to use different selection methods for the selection of the two parents which are chosen for crossover. In the present context this gender specific selection aspect (Wagner & Affenzeller, 2005) is implemented most effectively by selecting one parent conventionally by roulette-wheel selection and the other parent randomly.

All together, this especial variant of adapted sexual selection combined with a simplified version of offspring selection aims to cross one above-average parent with a randomly selected parent (which brings in diversity) as long as a whole new population could be filled up with children that were better than their better parent. An upper limit for selection pressure acts as termination criterion in that sense that the algorithm stops, if too many trials (|*POP*| * *maxSelPress*) were already taken and still no new population consisting of successful offspring could be generated. In other words, this indicates that it is not possible to generate a sufficient amount of children that outperform their parents out of the current gene pool; obviously, this seems to be a reasonable termination criterion for an Evolutionary Algorithm. This special version of SASEGASA or offspring selection respectively is schematically shown in Fig. 6.

The GP-based structure identification methods described in the previous section have been implemented as plug-ins for the HeuristicLab forming the problem specific basis of the HeuristicModeler. The following modeling specific extensions have been integrated into the general GP workflow:

- During the execution of a structure identification algorithm it can easily happen that a model showing a very suitable structure is assigned a very bad fitness value only due to inadequate parameter settings. Therefore we have implemented an additional local parameter optimization stage based on real-values encoded Evolution Strategies and integrated it into the execution of the Genetic Programming algorithm.
- As the GP-based model training algorithm tries to evolve better models, it can easily happen that models become more and more complex; the more complex models are, the better they can fit given training data, but they are also negative effects, namely increasing runtime consumption as well as the danger of overfitting. Therefore a heuristic tree pruning algorithm has also been integrated into the HeuristicModeler; in certain intervals, selected models included in the actual models pool are selected and pruned systematically, i.e. formula parts that do not seem to have a measurable influence on the model's evaluation are deleted in order to retrieve simpler models without significantly losing quality.

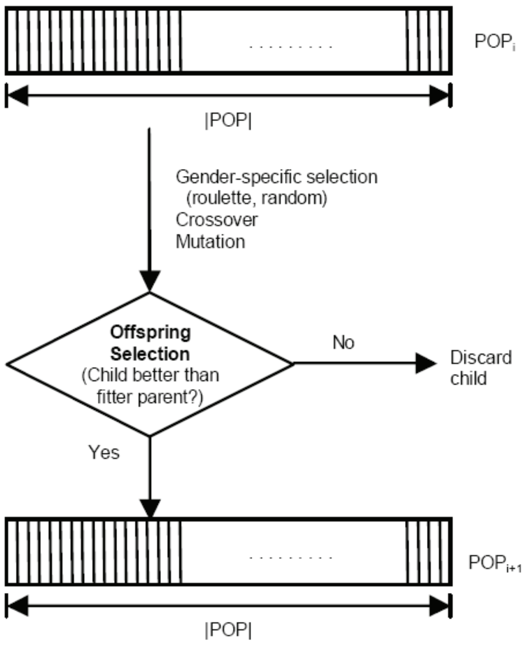

POP$_i$
|POP|
Gender-specific selection (roulette, random)
Crossover
Mutation
Offspring Selection (Child better than fitter parent?)
No
Discard child
Yes
POP$_{i+1}$
|POP|

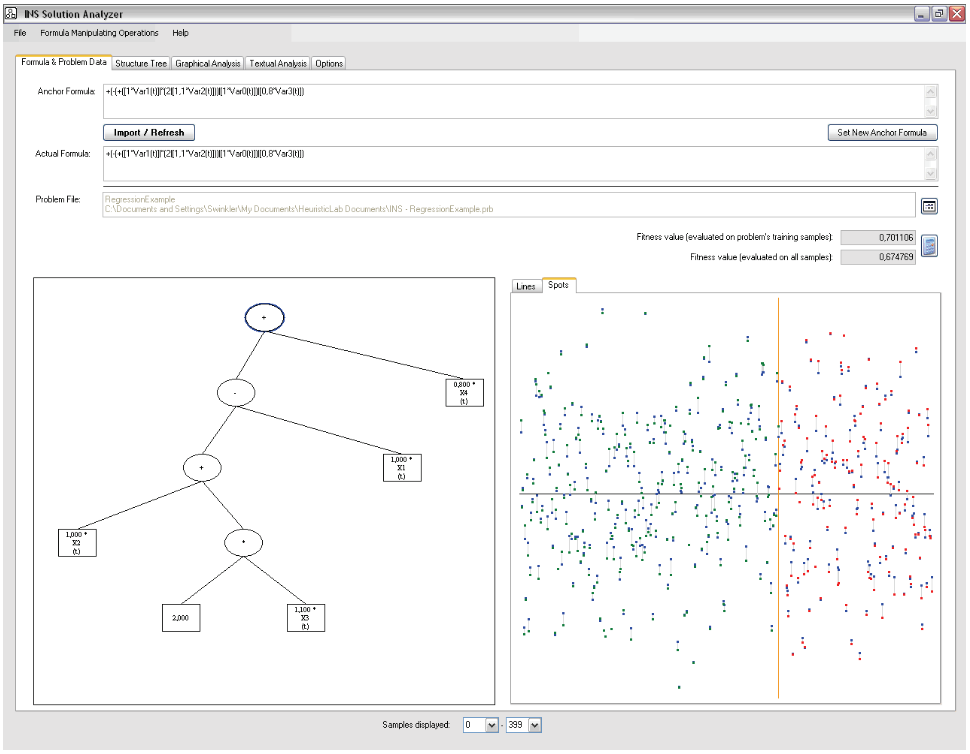
Fig. 6. Flowchart for embedding a simplified version of offspring selection into the GP process.

Due to its flexible and wide functional basis and the extended concepts described above, the GP-based modelling concept implemented in the HeuristicModeler is less exposed to the danger of overfitting than other machine learning algorithms; recent results and comparisons to other data-based modelling techniques are for example summarized in (DelRe et al., 2005), (Winkler et al, 2006f) and (Winkler et al., 2006a). Furthermore, as we will show in the following section, the results generated using the HeuristicModeler can easily be analyzed and interpreted using the HeuristicModelAnalyzer, a tool for analyzing solutions for data analysis problems that includes several enhanced evolutionary modelling aspects.

7. Examples and applications of GP in data based structure identification

7.1 Regression

For demonstrating the use of our evolutionary machine learning approach for attacking regression problems we have generated a synthetic data set including 5 variables and 400 samples. This data was analyzed using the HeuristicModeler and a model was trained; this model is graphically shown in Fig. 7. There are several possibilities how to evaluate a regression model using the HeuristicModelAnalyzer: Apart from drawing the (original and estimated) values and a graphical representation of the formula as a structure tree, the average squared error can be calculated as well as an overview of the errors distribution (as exemplarily shown later in Fig. 11.



Anchor Formula: +(-(+([1*Var1(t)]|*(2|[1,1*Var2(t)]))|[1*Var0(t)])|[0,8*Var3(t)])

Actual Formula: +(-(+([1*Var1(t)]|*(2|[1,1*Var2(t)]))|[1*Var0(t)])|[0,8*Var3(t)])

Problem File: RegressionExample
C:\Documents and Settings\Swinkler\My Documents\HeuristicLab Documents\INS - RegressionExample.prb

Fitness value (evaluated on problem's training samples): 0,701106

Fitness value (evaluated on all samples): 0,674769

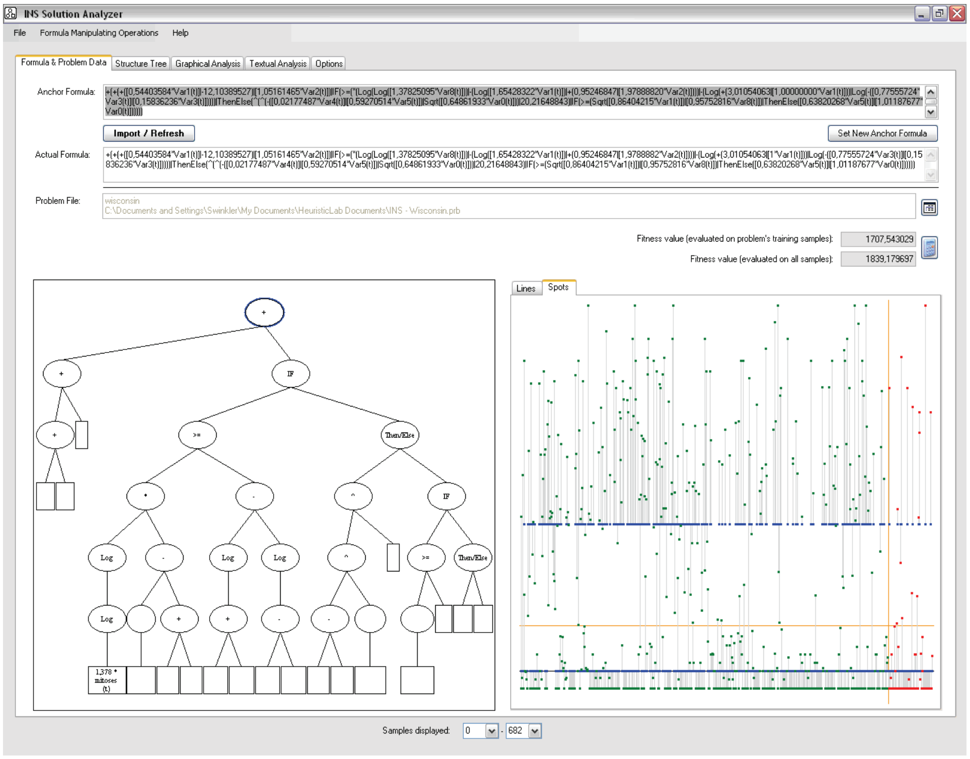
Samples displayed: 0 - 399

Fig. 7. A solution to a regression problem, analyzed using the HeuristicModelAnalyzer.

7.2 Classification

Several widely used benchmark classification datasets storing medical data (mainly survey records and diagnosis information) have already been analyzed using HeuristicModeler and HeuristicModelAnalyzer. In (Winkler et al., 2006b), (Winkler et al., 2006a) and (Winkler et al., 2006e) we have documented the results achieved for several medical classification benchmark problems, for example for the Wisconsin and the Thyroid datasets, which are parts of the UCI Machine Learning Repository (http://www.ics.uci.edu/~mlearn/). Summarizing the results documented in the publications mentioned above, GP-based training of classifiers is able to outperform other training methods (kNN classification, linear modeling and ANNs) especially on test data. There are several possibilities how to evaluate a classification model using the HeuristicModelAnalyzer:

Apart from drawing the (original and estimated) values and a graphical representation of the formula as a structure tree and calculating the average squared error, confusion matrices and (enhanced) receiver operating characteristics (ROC) curves can be generated. Furthermore, optimal thresholds are also identified automatically on the basis of a misclassification matrix storing information about how to weight misclassification dependent on the respective classes involved. This matrix is initially set so that all misclassifications are weighted equally; in various different applications it can be necessary to manipulate this weighting as it is, for example in the context of medical data analysis, more critical misclassifying a diseased patient as not diseased than vice versa. In Fig. 8 we show a graphical representation of a solution for the Wisconsin classification problem that was generated using the HeuristicModeler and analyzed using the HeuristicModelAnalyzer. As confusion matrices are also frequently used for evaluating classifiers, these are also automatically displayed when analyzing a model using the HeuristicModelAnalyzer.



Anchor Formula: +(+(+([0,54403584*Var1(t)]-12,10389527)|[1,05161465*Var2(t)]|IF(>=(*(Log(Log([1,37825095*Var8(t)]))-(Log([1,65428322*Var1(t)])+(0,95246847|[1,97888820*Var2(t)])))-(Log(+(3,01054063|[1,00000000*Var1(t)]))|Log(-([0,77555724*Var3(t)]|[0,15836236*Var3(t)]))))|ThenElse(^(^(-([0,02177487*Var4(t)]|[0,59270514*Var5(t)])|Sqrt([0,64861933*Var0(t)]))|20,21648843)|IF(>=(Sqrt([0,86404215*Var1(t)])|[0,95752816*Var8(t)])|ThenElse([0,63820268*Var5(t)]|[1,01187677*Var0(t)])))))

Actual Formula: +(+(+([0,54403584*Var1(t)]-12,10389527)|[1,05161465*Var2(t)]|IF(>=(*(Log(Log([1,37825095*Var8(t)]))-(Log([1,65428322*Var1(t)])+(0,95246847|[1,9788882*Var2(t)])))-(Log(+(3,01054063|[1*Var1(t)]))|Log(-([0,77555724*Var3(t)]|[0,15836236*Var3(t)]))))|ThenElse(^(^(-([0,02177487*Var4(t)]|[0,59270514*Var5(t)])|Sqrt([0,64861933*Var0(t)]))|20,21648843)|IF(>=(Sqrt([0,86404215*Var1(t)])|[0,95752816*Var8(t)])|ThenElse([0,63820268*Var5(t)]|[1,01187677*Var0(t)])))))

Problem File: wisconsin
C:\Documents and Settings\Swinkler\My Documents\HeuristicLab Documents\INS - Wisconsin.prb

Fitness value (evaluated on problem's training samples): 1707,543029

Fitness value (evaluated on all samples): 1839,179697

Samples displayed: 0 - 682

Fig. 8. A solution for the Wisconsin classification problem, generated by the HeuristicModeler and analyzed using the HeuristicModelAnalyzer.

Of course, classification problems occur not only in medical data analysis, but for example also in the context of data based quality pre-assessment in steel production. In (Winkler et al., 2006f) we report on an analysis done within an enhanced data processing process in cooperation with a large-scale industrial partner in steel industry. It was shown successfully that GP based structure identification is able to identify relationships between process parameters and the quality of steel products; on the basis of these results, high quality classification pre-estimators for the quality of the final results were formed.

Last, but not least the HeuristicModelAnalyzer enables the evaluation of classifiers for multi-class classification problems on the basis of a multi-class extension of ROC curves. Basic ROC analysis provides a convenient graphical display of the trade-off between true and false positive classification rates for two class problems (Zweig & Vampell, 1993). In the context of two class classification, ROC curves are calculated as follows: For each possible threshold value discriminating two given classes, the numbers of true and false classifications for one of the classes are calculated. For example, if the two classes "true" and "false" are to be discriminated using a given classifier, a fixed set of equidistant thresholds is tested and the true positives (TP) and the false positives (FP) are counted for each of them. Each pair of TP and FP values produces a point of the ROC curve. The main idea of Multi-ROC charts as presented in (Winkler et al., 2006d) is that for each given class c_i the numbers of true and false classifications are calculated for each possible pair of thresholds between the classes c_{i-1} and c_i as well as between c_i and c_{i+1} (assuming that the n classes can be represented as real numbers and that $c_i < c_{i+1}$ holds for every $i \in [1,(n-1)]$). The resulting tuples of (FP,TP) values are stored in a matrix which can be plotted easily. This obviously yields a set of points which can be interpreted analog to the interpretation of "normal" ROC curves: the closer the point are located to the left upper corner, the higher is the quality of the classifier at hand. For getting sets of ROC curves instead of ROC points, an arbitrary threshold ta between the classes c_{i-1} and c_i is fixed and the FP and TP values for all possible thresholds tb between c_i and c_{i+1} are calculated. This produces one single ROC curve; it is executed for all possible values of ta. An example showing 10 ROC curves is given in Fig. 9; this MROC chart was generated for a classifier learned for a synthetical data set storing 2000 samples divided into 6 classes and is taken from (Winkler et al., 2006d).

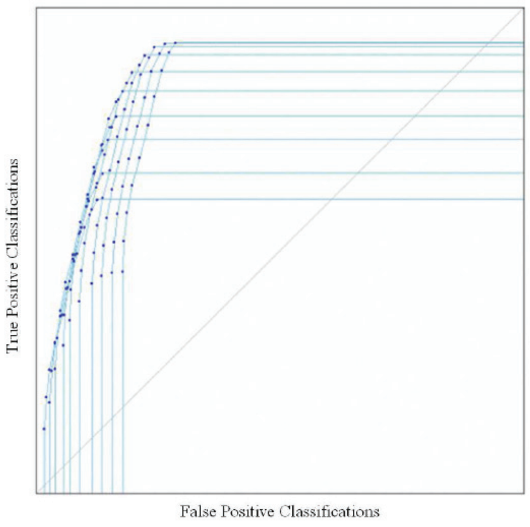
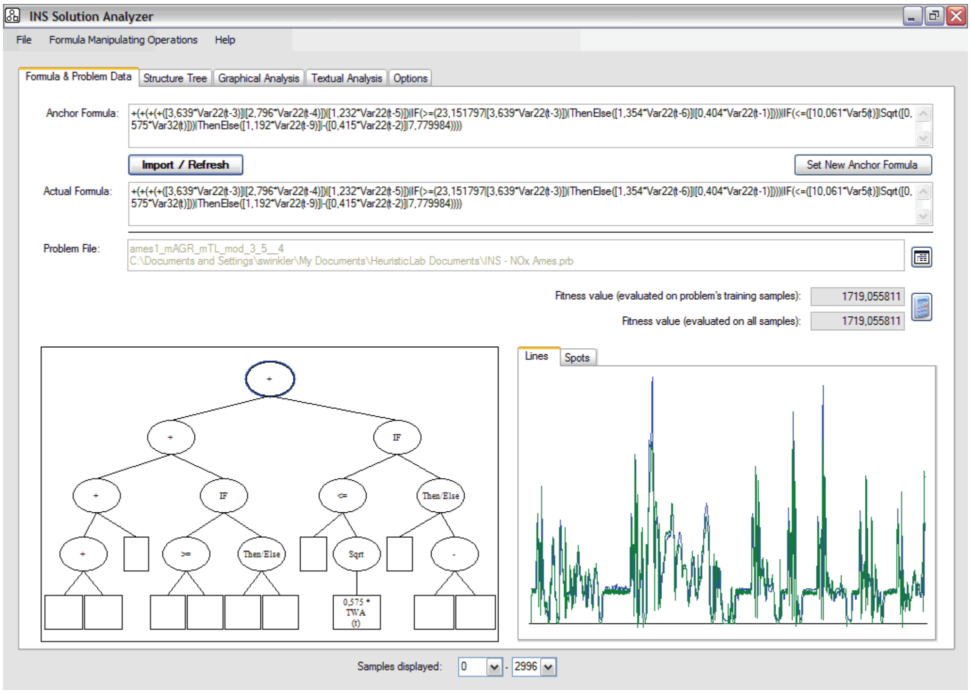


Fig. 9. An exemplary Multi-ROC chart.

7.3 Timeseries analysis

There is a lot of experience using the HeuristicModeler for solving time series problems on data recorded in the context of mechatronical systems. For example, in (Del Re et al., 2005) and (Winkler et al., 2005b) we report on models trained for the NO_x emissions of Diesel engines using the GP-based identification method incorporated in the HeuristicModeler. Fig. 10 and 11 show the evaluation of one of these models using the HeuristicModelAnalyzer: Apart from drawing the (original and estimated) values and a graphical representation of the formula as a structure tree, an overview of the errors distribution is given.



Anchor Formula:

```
+(+(+(+([3,639*Var22(t-3)]|[2,796*Var22(t-4)])|[1,232*Var22(t-5)])|IF(>=(23,151797|[3,639*Var22(t-3)])|ThenElse([1,354*Var22(t-6)]|[0,404*Var22(t-1)])))|IF(<=([10,061*Var5(t)]|Sqrt([0,
575*Var32(t)]))|ThenElse([1,192*Var22(t-9)]|-([0,415*Var22(t-2)]|7,779984))))
```

Actual Formula:

```
+(+(+(+([3,639*Var22(t-3)]|[2,796*Var22(t-4)])|[1,232*Var22(t-5)])|IF(>=(23,151797|[3,639*Var22(t-3)])|ThenElse([1,354*Var22(t-6)]|[0,404*Var22(t-1)])))|IF(<=([10,061*Var5(t)]|Sqrt([0,
575*Var32(t)]))|ThenElse([1,192*Var22(t-9)]|-([0,415*Var22(t-2)]|7,779984))))
```

Problem File: ames1_mAGR_mTL_mod_3_5__4
C:\Documents and Settings\swinkler\My Documents\HeuristicLab Documents\INS - NOx Ames.prb

Fitness value (evaluated on problem's training samples): 1719,055811

Fitness value (evaluated on all samples): 1719,055811

Samples displayed: 0 - 2996

Fig. 10. A model for the NO$_x$ emissions of a BMW Diesel engine, generated using the HeuristicModeler.

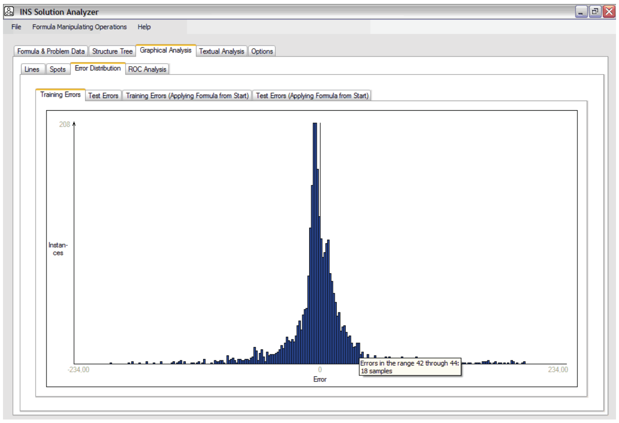


Fig. 11. Evaluation of the model shown in Figure 10.

8. Conclusion

In this paper we have described a multi-purpose machine learning approach based on various evolutionary computation concepts that is applicable for several data mining aspects in data driven systems identification. We have exemplarily shown how regression, classification and time series problems can be attacked using this algorithm. Especially in the context of analyzing time series problems of mechatronical systems as well as medical data sets we have already achieved very good results. Furthermore, we have also demonstrated how to analyze the results for data mining problems as well as selected aspects of the underlying enhanced evolutionary algorithm.

9. Acknowledgements

The work described in this paper was partially done within the Translational Research Program project L284-N04 "GP-Based Techniques for the Design of Virtual Sensors" sponsored by the Austrian Science Fund (FWF). This project is executed as a joint venture by the Upper Austrian University of Applied Sciences Hagenberg, Austria and the Johannes Kepler University Linz, Austria.

10. References

Affenzeller, M. (2001). Segregative Genetic Algorithms (SEGA): A Hybrid Superstructure Upwards Compatible to Genetic Algorithms for Retarding Premature Convergence. *International Journal of Computers, Systems and Signals (IJCSS),* Vol. 2, No. 1, (Feb. 2002) 18 -- 32, ISSN 1608-5655

Affenzeller, M. (2005). *Population Genetics and Evolutionary Computation: Theoretical and Practical Aspects,* Trauner Verlag, ISBN 3-85487-823-0, Linz, Austria

Affenzeller M & Wagner S. (2004). SASEGASA: A New Generic Parallel Evolutionary Algorithm for Achieving Highest Quality Results. *Journal of Heuristics - Special Issue on New Advances on Parallel Meta-Heuristics for Complex Problems,* Vol. 10, 239--263, ISSN

Affenzeller M. & Wagner S. (2003). A Self-Adaptive Model for Selective Pressure Handling within the Theory of Genetic Algorithms, *Proceedings of EUROCAST 2003*, pp. 384-393, LNCS 2809, Las Palmas, Feb. 2003, Springer, Heidelberg

Beligiannis, G.N.; Skarlas, L.V.; Likothanassis, S.D.. & Perdikouri, K. (2003). Nonlinear Model Structure Identication of Complex Biomedical Data Using a Genetic Programming Based Technique, *Proceedings of IEEE International Symposium on Intelligent Signal Processing*.

Beyer, H. G. (1998) The Theory of Evolution Strategies. Springer-Verlag Berlin Heidelberg New York, 1998.

Box; G.E. & Jenkins, G.M. (1976). Time Series Analysis: Forecasting and Control. Holden-Day Inc., San Francisco, 1976.

Del Re, L.; Langthaler, P.; Furtmller, C.; Affenzeller, M. & Winkler, S. (2005). NO_x Virtual Sensor Based on Structure Identification and Global Optimization. Proceedings of the SAE World Congress 2005.

Dumitrescu, D.; Lazzerini, B.; Jain, L.C. & Dumitrescu A. (2000). *Evolutionary Computation*. CRC Press,2000.

Fogel, D.B. (1994). An introduction to simulated evolutionary optimization. *IEEE Trans. on Neural Network*, Vol.5, No. 1. 3--14, 1994.

Goldberg, D.E. (1989) *Genetic Alogorithms in Search, Optimization and Machine Learning*. Addison Wesley Longman, 1989.

Holland, J.H. (1975). *Adaption in Natural and Artificial Systems.* University of *Michigan* Press, 1975.

Kendall, S.M. & Ord, J.K. (1990). *Time Series*. Edward Arnold, London, 1990.

Koza, J.R. (1992). *Genetic Prohramming: On the Programming of Computers by means of Natural Selection*. The MIT Press, 1992.

Langdon, B. & Poli, R. (2002). *Foundations of Genetic Programming*. Springer-Verlag Berlin Heidelberg New York, 2002.

Michaliwicz, Z. (1996). *Genetic Algorithms + Data Structurs = Evolution Programs.* Springer-Verlag Berlin Heidelberg New York, 3. edition, 1996.

Mitchell, T.M. (2000). *Machine Learning*. McGraw-Hill, New York, 2000.

Rechenberg, I. (1973). *Evolutionsstrategie*. Friedrich Frommann Verlag, 1973.

Rodrguez-Vzquez, K. & Fleming, P.J. (2000). Use of Genetic Programming in the Identification of Rational Model Structures. *Third European Conference on Genetic Programming (EuroGP'2000)*, pp 181--192, 2000.

Schwefel, H.P. (1994). *Numerische Optimierung von Computer-Modellen mittels Evolutionsstrategie*. Birkhaeuser Verlag. Basel, 1994.

Tomassini M. (1995). A survey of genetic algorithms. Annual Reviews of Computational Physics, Vol.3: 87--118, 1995.

Wagner, S. & Affenzeller, M. (2005a). Heuristiclab: A generic and extensible optimization environment. Adaptiveand Natural Computing Algorithms – Proceedings of ICANNGA 2005, pp. 538 -- 541, 2005.

Wagner, S. & Affenzeller, M (2005b). SexualGA: Gender-specific selection for genetic algorithms. Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI), pages 76--81, 2005.

Winkler, S.; Affenzeller, M. & Wagner, S. (2005a). New methods for the identification of nonlinear model structures based upon genetic programming techniques. *Journal of Systems Science*, Vol.31, 5--13, 2005.

Winkler, S.; Affenzeller, M. & Wagner, S. (2006a). Advances in applying genetic programming to machine learning focussing on classi¯cation problems. Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2006), 2006.

Winkler, S.; Affenzeller, M. & Wagner, S. (2006b). Automatic data based patient classification using genetic programming. Cybernetics and Systems 2006, 1:251--256, 2006.

Winkler, S.; Affenzeller, M. & Wagner, S. (2006c). Heuristic Modeler: A Multi-Purpose Evolutionary Machine Learning Algorithm and its Applications in Medical Data Analysis. Proceedings of the International Mediterranean Modelling Multiconference I3M 2006, pp. 629--634, 2006.

Winkler, S.; Affenzeller, M. & Wagner, S. (2006d). Sets of Receiver Operating Characteristic Curves and their use in the Evaluation of Multi-class Classification. Proceedings of the Genetic and Evolutionary Computation Conference 2006 (GECCO2006), 2: pp. 1601--1602, 2006.

Winkler, S.; Affenzeller, M. & Wagner, S. (2006e). Using Enhanced Genetic Programming Techniques for Evolving Classiffers in the Context of Medical Diagnosis - an Empirical Study. Proceedings of the GECCO 2006 Workshop on Medical Applications of Genetic and Evolutionary Computation (MedGEC 2006), 2006.

Winkler, S.; Efendic,H.; Affenzeller, M.; Del Re, L. & Wagner, S. (2005b) On-Line Modeling Based on Genetic Programming. Proceedings of the 1st International Workshop on Automatic Learning and Real-Time (ALaRT'05), pp. 119--130, 2005.

Winkler, S.; Efendic,H.& Del Re, L. (2006f). Quality Pre-assesment in Steel Industry using Data Based Estimators. Proceedings of the IFAC Workshop MMM2006 on Automation in Mining, Mineral and Metal Industry, pp. 185--190, 2006.

Zweig, M. H. & Campbell, G. (1993). Receiver-Operating Characteristics (ROC) Plots: A Fundamental Evaluation. *Clinical Chemistry*, Vol. 39: 561--577, 1993.

FPBIL: A Parameter-free Evolutionary Algorithm

Gustavo Caldas and Roberto Schirru
CNEN and COPPE/UFRJ
Brazil

1. Introduction

The purpose of this chapter is to describe a new algorithm named FPBIL (parameter-Free PBIL), an evolution of PBIL (Population-Based Incremental Learning). FPBIL, as well as PBIL (Baluja, 1994), Genetic Algorithms (GAs) (Holland, 1992) and others are general purpose population-based evolutionary algorithms. The success of GAs is unquestionable (Goldberg, 1989). Despite that, PBIL has shown to be superior in many aspects.
PBIL is a evolutionary algorithm developed as an attempt to mimic the behavior of the Genetic Algorithms in an advanced stage of its execution, "in equilibrium". The result shows unexpectedly that the PBIL surpasses (Baluja, 1995) the genetic algorithms in almost all aspects. The PBIL is faster and finds better results (Machado, 1999). However, PBIL depends on five parameters which need to be adjusted before each application. For example, variations in the learning rate produce completely different behaviors (Baluja, 1994).
Up to today, every evolutionary algorithm, like PBIL, just mentioned, depends on at least one parameter which, if not adjusted properly, can cause the algorithm to be very inefficient. Consequently, the less parameters an algorithm has, the minor the risk of it not reaching all its potential in some particular application; and the less the time spent in finding the appropriate parameter's values.
One of the benefits of FPBIL—perhaps the most important—is that it is a parameter free algorithm (the origin of the F in FPBIL), which means that a parameter optimization, an application-dependent procedure required by other algorithms in order to achieve better results, is not necessary in FPBIL. Parameter optimization demands intense computational effort, a precious time often not taken into account when somebody claims that an algorithm finds a better result in a shorter amount of time.
Based on PBIL, FPBIL is built with the guarantee of a better performance than that of PBIL, which also means (whenever the PBIL has a good outcome) a better performance in comparison to other algorithms, besides the advantage of none additional computational cost in adjusting parameters.
We begin this chapter by describing the PBIL algorithm and, then, we present the main steps to the FPBIL algorithm it self. Afterwards, we compare the performance of FPBIL against other algorithms in typical benchmark problems and finally we propose some concluding remarks.

2. PBIL algorithm

The PBILwas created in 1994, by Shumeet Baluja. It was inspired in its previous work with Ari Juels (Juels et al., 1993) in an attempt to simulate the behavior of the genetic algorithms

(Holland, 1992; Goldberg, 1989) in "equilibrium state", after repeated applications of the crossover operator. The algorithm referenced here by "PBIL" had its publication later, in 1995 (Baluja, 1995), in which 27 problems, commonly explored in the literature of genetic algorithms, were examined by seven different optimization techniques, PBIL having achieved optimum performance in more than 80% of the cases. PBIL algorithm is shown in figure 1.

```
**  *** Initialize Probability Vector
01  for ($j = 1 \dots n$)   $\mathscr{P}[j] = 0.5$
**  ***
02  while (not Termination_Condition))
03      for ($i = 1 \dots \mathcal{P}$)
04          for ($j = 1 \dots n$)
**              *** Criation of $\mathscr{I}_i$ from $\mathscr{P}$
05              if (random $(0,1) < \mathscr{P}[j]$)    $\mathscr{I}_i[j] = 1$
06              else                                  $\mathscr{I}_i[j] = 0$
**              ***
07          Calculate $\mathcal{F}(\mathscr{I}_i)$
08      Assign $\mathscr{I}^+$, the best individual
09      Assign $\mathscr{I}^-$, the worst individual
10      for ($j = 1 \dots n$)
**          *** Update $\mathscr{P}$ towards $\mathscr{I}^+$
11          $\mathscr{P}[j] = (1-\alpha) \cdot \mathscr{P}[j] + \alpha \cdot \mathscr{I}^+[j]$
**          ***
12          if ($\mathscr{I}^-[j] \neq \mathscr{I}^+[j]$)
**              *** Update $\mathscr{P}$ away from $\mathscr{I}^-$
13              $\mathscr{P}[j] = (1-\beta) \cdot \mathscr{P}[j] + \beta \cdot \mathscr{I}^+[j]$
**              ***
**      *** Mutation on $\mathscr{P}$
14      for ($j = 1 \dots n$)
15          if (random $(0,1) < P_{\mathscr{M}}$)
16              if (random $(0,1) < 0.5$)    $D_{\mathscr{M}} = 1$
17              else                         $D_{\mathscr{M}} = 0$
18              $\mathscr{P}[j] = (1-\gamma) \cdot \mathscr{P}[j] + \gamma \cdot D_{\mathscr{M}}$
**      ***
```

$\mathcal{P}$:	Population Size (100).
α:	Learning Rate (0.1).
β:	Negative Learning Rate (0.075).
$P_{\mathscr{M}}$:	Mutation Probability (0.02).
γ:	Mutation Rate. (0.05).

Fig. 1. PBIL Algorithm.

In PBIL, a subset $\mathcal{S}_{\mathcal{B}}$ of the search space $\mathcal{B}$ of some optimization problem is explored from one hypercube $\mathcal{H}_n \equiv [0, 1]^n$, in such a way that each vertex of $\mathcal{H}_n$, that is, each point of $\check{\mathcal{H}}_n \equiv \{0, 1\}^n$ corresponds to a point of $\mathcal{S}_{\mathcal{B}}$. This correspondence is made by the mapping

$$\begin{aligned} \mathcal{M}^n_{\mathcal{S}_\mathcal{B}} : \check{\mathcal{H}}_n &\longrightarrow \mathcal{S}_\mathcal{B} \\ \mathscr{I} \equiv (I_1, I_2, \ldots, I_n) &\longmapsto \mathcal{M}^n_{\mathcal{S}_\mathcal{B}}(\mathscr{I}), \end{aligned} \tag{1}$$

with $I_k \equiv \mathscr{I}[k] \in \{0, 1\}$, meaning that $\mathcal{M}^n_{\mathcal{S}_\mathcal{B}}$ maps a bit vector with n bits into a point of $\mathcal{S}_\mathcal{B}$ — a candidate solution to the problem.
After the vertices of $\mathcal{H}_n$ are duly mapped into $\mathcal{S}_\mathcal{B}$, the PBIL works exactly in the same way, independently of the current application and this is what makes the PBIL algorithm[1] versatile, meaning that the necessary and sufficient condition in order that an optimization problemcan be boarded by PBIL is the existence of $\mathcal{M}^n_{\mathcal{S}_\mathcal{B}}$.

A point $\mathscr{P} \in \mathcal{H}_n$ is called probability vector and it plays a central role in PBIL-like algorithms. Its n components $p_k \equiv \mathscr{P}[k] \in [0, 1]$ are suitable for representing the probability of choosing by chance the number 1 in a set $\Omega = \{0, 1\}$. From $\mathscr{P}$ is possible to construct an army of $\mathscr{I}$ objects. All we have to do is to pick I_k to be 1 or 0, probabilistically, according to p_k—the more p_k is close to 1, the more is I_k likely to be 1.
At the beginning of PBIL each point of $\mathcal{S}_\mathcal{B}$ must be treated as potential best solution and $\mathcal{P}$ vertices of $\mathcal{H}_n$ are, therefore, chosen randomly from a uniform probability distribution. This uniform probability distribution is nothing but $\mathscr{P}_0 = (0.5, 0.5, \ldots, 0.5)$, the center of $\mathcal{H}_n$.
In PBIL's terminology, the $\mathcal{P}$ vertices $\mathscr{I}_k$ of $\mathcal{H}_n$ selected from $\mathscr{P}_\mathcal{G}$ forma "population"—the "generation" $\mathcal{G}$—and each $\mathscr{I}_k$ is called an "individual". The PBIL algorithm consists in, once established the individuals of generation 0, constructing $\mathscr{P}_1$, which will generate the next population—generation 1. The process is repeated until an individual of some generation is considered to be good enough. In this sense, the PBIL algorithm may be viewed as the motion of $\mathscr{P}$ inside $\mathcal{H}_n$ until $\mathscr{P}$ gets close enough to some point of $\check{\mathcal{H}}_n$ corresponding to a satisfactory solution; the laws of motion being the PBIL rules by which $\mathscr{P}$ is updated from generation to generation.
The measure of how good an individual is, is given by the fitness function

$$\begin{aligned} \mathcal{F} \colon \check{\mathcal{H}}_n &\longrightarrow \mathbb{R}^+ \\ \mathscr{I} &\longmapsto \mathcal{F} = \mathcal{F}(\mathscr{I}), \end{aligned} \tag{2}$$

whose form depends explicitly on the application.
The construction of $\mathscr{P}_{\mathcal{G}+1}$ from the individuals of the generation $\mathcal{G}$ is the main process in a PBIL-like algorithm. Any point $\mathscr{P}_{\mathcal{G}+1}$ of $\mathcal{H}_n$ different from $\mathscr{P}_0$ generates a non-uniform probability distribution on $\check{\mathcal{H}}_n$. The strategy is to modify, generation after generation, this probability distribution trying to turn ever more likely the sprouting of $\mathscr{I}^\dagger$, the optimum solution. In PBIL, $\mathscr{P}_{\mathcal{G}+1}$ is constructed in two steps.
In the first step, the following operations are carried through:

$$\mathscr{P}_{\mathcal{G}+1/2} = \mathscr{P}_\mathcal{G} + \alpha \cdot (\mathscr{I}^+ - \mathscr{P}_\mathcal{G}) \tag{3}$$

[1] This is also true for other algorithms working with bitstrings, such as Genetic Algorithms.

$$\mathscr{P}'_{\mathcal{G}+1}[j] = \begin{cases} \mathscr{P}_{\mathcal{G}+1/2}[j] + \beta \cdot (\mathscr{I}^+[j] - \mathscr{P}_{\mathcal{G}+1/2}[j]) & \text{if } \mathscr{I}^+[j] \neq \mathscr{I}^-[j] \\ \mathscr{P}_{\mathcal{G}+1/2}[j] & \text{if } \mathscr{I}^+[j] = \mathscr{I}^-[j] \end{cases} \tag{4}$$

where $\mathscr{I}^+$ and $\mathscr{I}^-$ are respectively the best and worst individuals of generation $\mathcal{G}$. That is, $\mathscr{P}_{\mathcal{G}}$ initially is dislocated towards $\mathscr{I}^+$ and then, away from $\mathscr{I}^-$, with the intention to favor the occurrence of better individuals in the following generation.
In the second step $\mathscr{P}'_{\mathcal{G}+1}$ suffers mutation, whose objective is to allow that some component of $\mathscr{P}$ reaching the value 1 (or 0) has the possibility to evolve again—since, once $p_k = 1$ (or $p_k = 0$), it can not change by means of equation (4). Such mutation consists of moving the components of $\mathscr{P}'_{\mathcal{G}+1}$ in the direction of $D_{\mathcal{M}}$ (randomly 0 or 1). This means that each component $\mathscr{P}'_{\mathcal{G}+1}[j]$ will suffer, or not, a displacement (according to the "mutation probability") in the form of:

$$\mathscr{P}_{\mathcal{G}+1}[j] = \mathscr{P}'_{\mathcal{G}+1}[j] + \gamma \cdot (D_{\mathcal{M}} - \mathscr{P}'_{\mathcal{G}+1}[j]). \tag{5}$$

As can be verified in figure 1, the PBIL algorithm needs five parameters to work, whose values were determined experimentally in order to maximize the average performance of the algorithm in a set of different applications. In the next section, we will show how to extend PBIL to be parameter-free.

3. FPBIL: parameter-Free PBIL

FPBIL is a variation of PBIL which basically tries to eliminate the necessity of the PBIL's parameter by modifying some of its fundamental principles. The result is a more efficient algorithm, with a superior search power and without parameters.
As in PBIL, the FPBIL algorithm presents a probability vector $\mathscr{P}$, with n components $p_k \in [0, 1]$, from which $\mathcal{P}$ individuals $\mathscr{I}_k$ of some generation are created. The characteristic that differentiates them is that FPBIL uses generic mechanisms to become free of parameters, especially in the way $\mathscr{P}$ is updated and the mutation is implemented. The FPBIL Algorithm is presented in figure 2.

3.1 $\mathscr{P}$ update: eliminating the parameters α and β

In the algorithm PBIL, the probability vector is updated by suffering a small displacement approaching to the best individual and another displacement moving away from the worst individual. In some variants of the PBIL (Baluja & Caruana, 1995; Baluja & Davies, 1998; Machado, 2005), only the best individual is used, or only the worst individual, or also, the average of the first best individuals. The fact is that, in order to evaluate who are the best and worst individuals, all the individuals must be evaluated, which means that all PBIL algorithms waste almost all the information available about the search space.

```
*** Definition of Functions $C_x$ and $D_x$
$D_x \equiv 1/(1+x)$
$C_x \equiv D_x^{-1} \equiv 1/x - 1$
***
*** Initialization of the Probability Vector, $\mathcal{P}_0$ and $d$
for $(j = 1 \dots n)$   $\mathscr{P}[j] = 0.5$
$\mathcal{P}_0 = 7(1 + 1/n)^n$
$d = 1/3$
***
while (not Termination_Condition))
    *** Determination of $\mathcal{P}$
    if (Fluctuation in $c$)
        $\mathcal{P}_0 = \mathcal{P}_0 + 1$
        if $(\Delta\langle c\rangle_{\mathcal{G}} < 1\%)$
            $d = 1/3$
            $\mathscr{P} = \mathscr{P}_0$
    $c = C_d$
    $\mathcal{P} = \texttt{(int)}\mathcal{P}_0(1 + 1/c)^c(\mathcal{P}_0/7)^{-c/n}$
    ***
    for $(i = 1 \dots \mathcal{P})$
        Creation of $\mathscr{I}_i$ from $\mathscr{P}$, as in PBIL
        Calculate $\mathcal{F}(\mathscr{I}_i)$
    for $(j = 1 \dots n)$
        *** Update $\mathscr{P}$
        $\mathscr{P}[j] = \left(\sum_{i=1}^{\mathcal{P}} \mathcal{F}(\mathscr{I}_i) \cdot \mathscr{I}_i[j]\right) / \left(\sum_{i=1}^{\mathcal{P}} \mathcal{F}(\mathscr{I}_i)\right)$
        ***
    *** Mutation in $\mathscr{P}$
    $c \ = $ Count of Cases $(\mathscr{P}[j] \le d$          or $\mathscr{P}[j] \ge 1 - d)$
    $c' = $ Count of Cases $(\mathscr{P}[j] \le D_{C_d - 1}$ or $\mathscr{P}[j] \ge 1 - D_{C_d - 1})$
    if         $(c > C_d)$          $d = D_{c+1}$
    else if $(c' < C_d - 1)$   $d = D_{c-1}$
    if $(d > 1/3)$   $d = 1/3$
    for $(j = 1 \dots n)$
        if $(\mathscr{P}[j] < d)$         $\mathscr{P}[j] = d$
        if $(\mathscr{P}[j] > 1 - d)$   $\mathscr{P}[j] = 1 - d$
    ***
```

Fig. 2. FPBIL Algorithm.

The rule according to which the FPBIL updates its probability vector is

$$\mathscr{P}'_{\mathcal{G}+1} = \frac{\sum_{i=1}^{\mathcal{P}} \mathcal{F}_i \cdot \mathscr{I}_i}{\sum_{i=1}^{\mathcal{P}} \mathcal{F}_i}, \tag{6}$$

which reflects exactly an average in which all $\mathcal{P}$ individuals are used. The difference is that this average is weighed by the fitness $\mathcal{F}_i \equiv \mathcal{F}(\mathscr{I}_i)$ of each individual. In order to appreciate better the change caused by this detail, it can be deduced that

$$\mathscr{P}'_{\mathcal{G}+1} = \frac{\sum_{i=1}^{\mathcal{P}} \mathscr{I}_i}{\mathcal{P}} + \xi \cdot \mathscr{D}, \tag{7}$$

with

$$\xi \equiv \frac{1}{\mathcal{P}} \sum_{\mathcal{F}_i > \langle \mathcal{F} \rangle} \frac{\mathcal{F}_i - \langle \mathcal{F} \rangle}{\langle \mathcal{F} \rangle} \tag{8}$$

and

$$\mathscr{D} \equiv \frac{\sum_{\mathcal{F}_i > \langle \mathcal{F} \rangle} (\mathcal{F}_i - \langle \mathcal{F} \rangle) \cdot \mathscr{I}_i}{\sum_{\mathcal{F}_i > \langle \mathcal{F} \rangle} (\mathcal{F}_i - \langle \mathcal{F} \rangle)} - \frac{\sum_{\mathcal{F}_i < \langle \mathcal{F} \rangle} (\langle \mathcal{F} \rangle - \mathcal{F}_i) \cdot \mathscr{I}_i}{\sum_{\mathcal{F}_i < \langle \mathcal{F} \rangle} (\langle \mathcal{F} \rangle - \mathcal{F}_i)}. \tag{9}$$

Note that $\sum_{i=1}^{\mathcal{P}} \mathscr{I}_i / \mathcal{P}$ is approximately $\mathscr{P}_{\mathcal{G}}$ so that equation (7) resembles the structure of equations (3) and (4) corresponding to PBIL.
The advantage in using $\mathscr{D}$ is that the direction of the displacement is not based only on the best and worst individuals, but in all the available information about the search space at some generation ($\mathcal{P}$ evaluated individuals). Another detail about $\mathscr{D}$ is that the averages are not simple, but weighed by the differences between the fitness of each individual and the average fitness, so that very bad or very good individuals exert more influence than others with fitness next to the average.
It is worth noting that each point $\mathscr{P}$ of $\mathcal{H}_n$ can be associated to an average fitness ${}_{\mathscr{P}}\langle \mathcal{F} \rangle$ through

$$\begin{aligned} {}_{\mathscr{P}}\langle \mathcal{F} \rangle : \mathcal{H}_n &\longrightarrow \mathbb{R} \\ \mathscr{P} &\longmapsto {}_{\mathscr{P}}\langle \mathcal{F} \rangle = \sum_{i=1}^{2^n} \mathcal{F}_i \cdot \mathscr{P}(\mathscr{I}_i). \end{aligned} \tag{10}$$

The reason is that from $\mathscr{P}$, each individual $\mathscr{I}_i$ has a probability $\mathscr{P}$ ($\mathscr{I}_i$) of being picked. After $\mathcal{P}$ tries, the individual $\mathscr{I}_i$ is picked $\mathcal{P}_i$ times. In the limit when $\mathcal{P}$ becomes sufficiently big, we have

$$\lim_{\mathcal{P}\to\infty} \langle \mathcal{F} \rangle = \lim_{\mathcal{P}\to\infty} \frac{\sum_{j=1}^{\mathcal{P}} \mathcal{F}_j}{\mathcal{P}} = \lim_{\mathcal{P}\to\infty} \sum_{i=1}^{2^n} \mathcal{F}_i \cdot \frac{\mathcal{P}_i}{\mathcal{P}} \tag{11}$$

$$= \sum_{i=1}^{2^n} \mathcal{F}_i \cdot \mathscr{P}(\mathscr{I}_i). \tag{12}$$

Since ${}_{\mathscr{P}}\langle \mathcal{F} \rangle$ is such an average, it is continuous, differentiable and it doesn't have any local maximum or minimum in $\mathcal{H}_n$ - $\check{\mathcal{H}}_n$, which means that the extreme points of ${}_{\mathscr{P}}\langle \mathcal{F} \rangle$ in $\mathcal{H}_n$ occurs for $\mathscr{I}^{\dagger}$ and $\mathscr{I}^{\perp}$ in $\check{\mathcal{H}}_n$, with ${}_{\mathscr{I}^{\dagger}}\langle \mathcal{F} \rangle = \mathcal{F}(\mathscr{I}^{\dagger})$ and ${}_{\mathscr{I}^{\perp}}\langle \mathcal{F} \rangle = \mathcal{F}(\mathscr{I}^{\perp})$ — where $\mathscr{I}^{\perp}$ represents the worst individual in $\check{\mathcal{H}}_n$. And that is just interesting.

In each generation of FPBIL, we have $\langle \mathcal{F} \rangle \approx {}_{\mathscr{P}}\langle \mathcal{F} \rangle$ and the $\mathcal{P}$ individuals $\mathscr{I}_k$ are divided into two groups: those with $\mathcal{F}(\mathscr{I}_k) > \langle \mathcal{F} \rangle$ and those with $\mathcal{F}(\mathscr{I}_k) < \langle \mathcal{F} \rangle$. If we represent each of these groups respectively by the points

$$\langle \mathscr{P} \rangle_{>} \equiv \frac{\sum_{\mathcal{F}_i > \langle \mathcal{F} \rangle} (\mathcal{F}_i - \langle \mathcal{F} \rangle) \cdot \mathscr{I}_i}{\sum_{\mathcal{F}_i > \langle \mathcal{F} \rangle} (\mathcal{F}_i - \langle \mathcal{F} \rangle)} \tag{13}$$

and

$$\langle \mathscr{P} \rangle_{<} \equiv \frac{\sum_{\mathcal{F}_i < \langle \mathcal{F} \rangle} (\langle \mathcal{F} \rangle - \mathcal{F}_i) \cdot \mathscr{I}_i}{\sum_{\mathcal{F}_i < \langle \mathcal{F} \rangle} (\langle \mathcal{F} \rangle - \mathcal{F}_i)} \tag{14}$$

we see from equation (9) that FPBIL works in such a way that $\mathscr{P}$ moves in the direction that ${}_{\mathscr{P}}\langle \mathcal{F} \rangle$ grows, leading, theorically at least, to $\mathscr{I}^{+}$. Just to compare, in PBIL, $\mathscr{I}_{+}$ and $\mathscr{I}_{-}$ are used instead of $\langle \mathscr{P} \rangle_{>}$ and $\langle \mathscr{P} \rangle_{<}$, which means that PBIL is much easier to get caught by local optimums.

Obviously we can only bet that the approximation $\langle \mathcal{F} \rangle \approx {}_{\mathscr{P}}\langle \mathcal{F} \rangle$ is good enough. Only in the limit $\mathcal{P} \to \infty$ can we be sure. The same limit when we would have already evaluated every element of $\check{\mathcal{H}}_n$, so that we would no longer need a search algorithm. Fortunately, the FPBIL algorithm also have proper mechanisms that compensate for the finiteness of $\mathcal{P}$. ξ can be considered to be one of those.

It can be verified that ξ plays a similar role just like α or β, related to the intensity of the displacement suffered by $\mathscr{P}$. While α and β are constants, ξ varies in accordance to the fitness distribution of each generation. More precisely, ξ is the half of the mean absolute deviation, relative to the average, of the fitness:

$$\xi = \frac{1}{\langle \mathcal{F} \rangle} \frac{1}{\mathcal{P}} \sum_{\mathcal{F}_i > \langle \mathcal{F} \rangle} \mathcal{F}_i - \langle \mathcal{F} \rangle \tag{15}$$

$$= \frac{1}{2} \frac{1}{\langle \mathcal{F} \rangle} \Big(\frac{1}{\mathcal{P}} \sum_{i=1}^{\mathcal{P}} |\mathcal{F}_i - \langle \mathcal{F} \rangle| \Big) \tag{16}$$

$$= \frac{1}{2} \frac{\delta}{\langle \mathcal{F} \rangle} = \frac{1}{2} \delta_r . \tag{17}$$

The mean absolute deviation (δ) is a measure of dispersion of a distribution, just like the standard deviation. δ_r is only another way to express the same dispersion relative to the average.

At the beginning of an execution of the FPBIL, the individuals generally possess a very bad fitness. While no individual detaches, ξ is small—the algorithm does not take risks by making a decision on which direction to follow. When the first good individuals appear, ξ increases considerably. As the average fitness goes up, ξ diminishes gradualy—preventing itself from premature convergence. Finally, when the optimum solution is near, ξ becomes very small, making sure that $\mathscr{P}$ will not have great oscillations around it but, instead, it might be reached.

3.2 Mutation: eliminating the parameters $P_{\mathcal{M}}$ and γ

The role of mutation is to give "second chances" to the components of $\mathscr{P}$ that reach the values 0 or 1 when they were not supposed to do so. In the limit $\mathcal{P} \to \infty$, FPBIL would not need mutation at all, as we have already discussed. But in a real situation, mutation is another mechanism that compensates for finite $\mathcal{P}$, and it is essential to FPBIL.

The PBIL carries mutation probabilistically (in accordance to $P_{\mathcal{M}}$) through random displacements (proportional to γ) in the components of $\mathscr{P}$. The FPBIL algorithm follows a more direct strategy, exploring the meaning of the probability vector. First, the algorithm hinders any component of $\mathscr{P}$ from reaching the values 0 or 1. This way the emergence of any individual in $\check{\mathcal{H}}_n$ is always possible. That is accomplished by restricting every component p_k of $\mathscr{P}$ to the interval $[d, 1 - d]$. As a consequence, the probability of choosing by chance any individual from $\mathscr{P}$ will always be between d^n and $(1 - d)^n$.

Given any value d, the number c of components of $\mathscr{P}$ with $p_k \leq d$ or $p_k \geq 1 - d$ is considere to be the number of components which are in the correct position. Then it is possible to find the optimum value of d, so that it maximizes the probability of choosing from $\mathscr{P}$ an individual with the corresponding c correct components and so that is also capable of inverting the trend of some component going toward the wrong direction. The probability which must be maximized is, therefore,

$$p(d) = d(1 - d)^c, \tag{18}$$

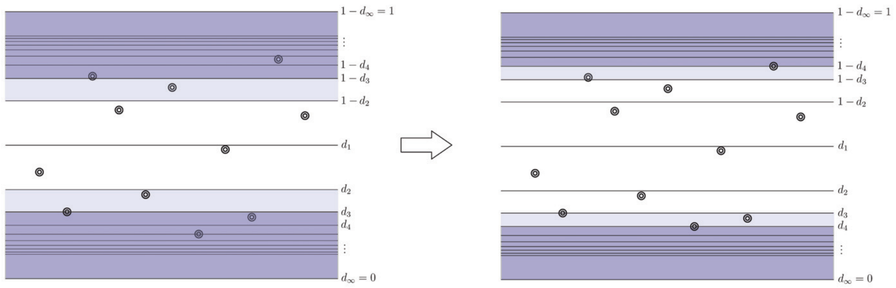
giving

$$d_c = \frac{1}{1 + c}. \tag{19}$$

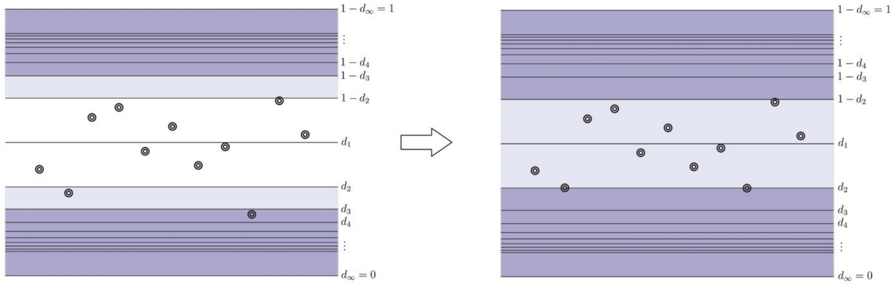
The FPBIL algorithm takes d to be initially (in generation 0) $d_2 = 1/3$—the biggest value of d_c different from 0.5. After $\mathscr{P}_{\mathcal{G}}$ is updated to $\mathscr{P}'_{\mathcal{G}+1}$, we count how many ($c$) components of $\mathscr{P}'_{\mathcal{G}+1}$ satisfy $p_k \leq d_2$ (or $p_k \geq 1 - d_2$). If $c \geq 3$, d becomes $d_3 = 1/4$. If $d = d_3$ and $c \geq 4$ (the number of components of $\mathscr{P}'_{\mathcal{G}+1}$ that satisfy $p_k \leq d_3$ (or $p_k \geq 1 - d_3$)), d becomes $d_4 = 1/5$, and so on. Thus, it is possible to diminish d gradually as P gets close to some point in $\check{\mathcal{H}}_n - \mathscr{I}^+$, expectedly.

But there is also a mechanism that allows d to grow. If, for example, $d = d_5 = 1/6$ but $c \ngeq 6$, we count how many (c') components of $\mathscr{P}'_{\mathcal{G}+1}$ satisfy $p_k \leq d_4$ (or $p_k \geq 1 - d_4$). If $c' < 5$, d becomes $d_4 = 1/5$. If $d = d_4$, $c \ngeq 5$ and $c' < 4$ (the number of $\mathscr{P}'_{\mathcal{G}+1}$ components that satisfy $p_k \leq d_3$ (or $p_k \geq 1 - d_3$)), d becomes $d_3 = 1/4$, and so on, until d hits the value $d_2 = 1/3$, the biggest allowed.

After we count c and $c'$and update d, mutation do its real job: it brings back to d (or to $1-d$) any $\mathscr{P}'_{\mathcal{G}+1}$ component smaller than d (or bigger than $1-d$), transforming $\mathscr{P}'_{\mathcal{G}+1}$ into $\mathscr{P}_{\mathcal{G}+1}$. FPBIL's mutation is illustrated in figure 3, where each point ⊚ represents a component of $\mathscr{P}$. As we can see, d values work as "gates" that open or close depending on the values of c and c'.



$\mathscr{P}'_{\mathcal{G}+1}; d = d_3.$ $\mathscr{P}_{\mathcal{G}+1}; d = d_4.$



$\mathscr{P}'_{\mathcal{G}+1}; d = d_3.$ $\mathscr{P}_{\mathcal{G}+1}; d = d_2.$

Fig. 3. Two examples of mutation: in the first, d diminishes; in the second, it grows.

3.3 Variable population size and reinitializations: eliminating the parameter $\mathcal{P}$

The size of $\check{\mathcal{H}}_n$ is 2^n, which is usually very large. The population sizes commonly used in PBIL are very small fractions of this value. Therefore, it is reasonable to use the relation

$$\mathcal{P} = 2^{\frac{n}{w}} \tag{20}$$

for some w.

Perhaps the most remarkable aspect of FPBIL (and PBIL) is that the population size does not have to be a constant—sheer nonsense for GA users. Since every population is generated from $\mathscr{P}_\mathcal{G}$ instantly after $\mathscr{P}_\mathcal{G}$ is created, it does not matter whether we generate only one or a thousand individuals. There is no higher complexity involved than choosing how many individuals we want.

As the number c of correct components of $\mathscr{P}$ increases, we must, therefore, need only

$$\mathcal{P}_c = \frac{k}{(1-d)^c} \cdot 2^{\frac{n-c}{w}} \tag{21}$$

individuals, where the factor $k/(1-d)^c$ only appears to assure that the correct component are reproduced with 99.9% of probability (for $k = 7$) (Caldas, 2006). Using equation (19), it can be written as

$$\mathcal{P}_c = \left(1 + \frac{1}{c}\right)^c \cdot \mathcal{P}_0 \left(\frac{\mathcal{P}_0}{k}\right)^{-\frac{c}{n}} \tag{22}$$

where $\mathcal{P}_0$ is the initial population, corresponding $c = 0$. FPBIL is initiated with $\mathcal{P}_0 = \mathcal{P}_n$ and every time c suffers a fluctuation, $\mathcal{P}_0$ is increased by 1. That occurs because, when the time average $\langle c \rangle_{\mathcal{G}}$ of c stops varying, the algorithm must be imprisoned in a local optimum, so it must be reinitiated. The difference is that in each reinitialization $\mathcal{P}_0$ will be each time bigger (due to the fluctuations of c), increasing gradually the power of search of the FPBIL. A fluctuation in c will be computed whenever c does not grow or decrease directly, that is, whenever c, as a function of $\mathcal{G}$, reaches a minimum, a maximum or simply remains constant; and $\langle c \rangle_{\mathcal{G}}$ stands for the time average of c between reinitializations.

3.4 About the fitness function

Although FPBIL is parameters-free, it still depends on the form of the fitness function. There are several functional forms for $\mathcal{F}$ capable of determining the same order $\mathcal{F}(\mathscr{I}^-) \leq \mathcal{F}(\mathscr{I}_i) \leq \mathcal{F}(\mathscr{I}_j) \leq \cdots \leq \mathcal{F}(\mathscr{I}^+)$ and each one of them can generate different $\mathscr{D}$ and ξ values, which would result in equally different behaviors. Consider the analysis of equations (8) and (9) in two simple examples:

1. With the transformation $\mathcal{F}'_i = f \cdot \mathcal{F}_i$ ($f \in \mathbb{R}$), one has $\mathscr{D}' = \mathscr{D}$ e $\xi' = \xi$, that is, the multiplication of the fitness by a constant factor, does not modify anything in the behavior of FPBIL.
2. With the transformation $\mathcal{F}'_i = t + \mathcal{F}_i$ ($t \in \mathbb{R}$), however, $\mathscr{D}' = \mathscr{D}$, but

$$\xi' = \frac{\langle \mathcal{F} \rangle}{\langle \mathcal{F} \rangle + t}\, \xi, \tag{23}$$

meaning that if $t >> \langle \mathcal{F} \rangle$ then $\xi' \approx 0$, that is, the addition of the fitness to a constant term modifies the intensity of the steps of the FPBIL, making the FPBIL impracticable for big values of t.

Item 2 suggests that one good practice may be the use of the fitness $\mathcal{F}'_i = \mathcal{F}_i - \mathcal{F}(\mathscr{I}^-)$, guaranteeing that ξ will never be smaller than necessary. Following such recommendation, a generic procedure was adopted to construct the fitness—based on the procedure used by Koza in the genetic programming algorithm (Koza, 1992)—described as follows.

The raw fitness $\mathcal{F}_r$ is the natural amount of the problem that one desires to maximize or minimize, also called objective function. From the raw fitness, the standard fitness $\mathcal{F}_s$ is constructed, which possesses the characteristic of having $0 \leq \mathcal{F}_s(\mathscr{I}_a) < \mathcal{F}_s(\mathscr{I}_b)$ whenever $\mathscr{I}_a$ is better than $\mathscr{I}_b$, and, preferentially, with $\mathcal{F}_s(\mathscr{I}^+) = 0$. From the standard fitness, the adjusted fitness $\mathcal{F}_a$ is calculated from

$$\mathcal{F}_a(\mathscr{I}_i) = \frac{1}{1 + \mathcal{F}_s(\mathscr{I}_i)}. \tag{24}$$

Finally, following the recommendation of having $\mathcal{F}'_i = \mathcal{F}_i - \mathcal{F}(\mathscr{I}^-)$, the fitness function used everywhere in this work (excep when expressly told) will be

$$\mathcal{F}(\mathscr{I}_i) = \begin{cases} \mathcal{F}_a(\mathscr{I}_i) - \mathcal{F}_a(\mathscr{I}^-_{\mathcal{G}-1}), & \text{if } \mathcal{F}_a(\mathscr{I}_i) - \mathcal{F}_a(\mathscr{I}^-_{\mathcal{G}-1}) \geq 0; \\ 0, & \text{if } \mathcal{F}_a(\mathscr{I}_i) - \mathcal{F}_a(\mathscr{I}^-_{\mathcal{G}-1}) < 0, \end{cases} \tag{25}$$

where $\mathcal{F}_a(\mathscr{I}^-_{\mathcal{G}-1})$ it is the adjusted fitness of the worse individual of the previous generation. The excuse for using $\mathcal{F}_a(\mathscr{I}^-_{\mathcal{G}-1})$ is that, to find $\mathcal{F}(\mathscr{I}^-)$, it is necessary to evaluate all the individuals of a generation, which implies that, in order to calculate $\mathcal{F}'_i = \mathcal{F}_i - \mathcal{F}(\mathscr{I}^-)$, all the individualsmust be evaluated twice every generation or all the individuals of a generation must be stored in some data structure. The adopted solution, besides economical, does not harm too much the original recommendation since generally $\mathcal{F}_a(\mathscr{I}^-_{\mathcal{G}-1}) \approx \mathcal{F}_a(\mathscr{I}^-)$.
Next, we will see how to put all this into practice.

4. Problems

This section is intended to show how PBIL and FPBIL behave in different problems of growing complexity. These problems belong to specific classes, which are, ultimately, numerical or combinatorial, so we can learn how to proceed in both cases. Besides the opportunity to see how these two algorithms works in practice, we will use the results then achieved to quantitatively compare them and, whenever interesting, compare their results to those of other techniques. Let us begin with the simplest.

4.1 A simple problem in $\mathcal{H}_2$

In order to visualize better the differences between FPBIL and PBIL, we will use them in a very simple problem: to find the greatest number in $\mathcal{B} = \mathbb{N}_4 \equiv \{1, 2, 3, 4\}$. We can chose $\mathcal{S}_{\mathcal{B}} = \mathcal{B}$, so that we need only $n = 2$ bits to cover all $\mathcal{S}_{\mathcal{B}}$ (because $2^2 = 4$ = number of elements in $\mathcal{S}_{\mathcal{B}}$)—FPBIL and PBIL will work in $\mathcal{H}^2$, which is nothing but a simple (easy to visualize) square. That means that we can correspond each point of $\check{\mathcal{H}}_2$ to a member of $\mathcal{S}_{\mathcal{B}}$. We may choose, for example, $\mathcal{M}^n_{\mathcal{S}_{\mathcal{B}}}$ to be the following map:

$$\mathcal{M}^n_{\mathcal{S}_{\mathcal{B}}}(0,0) = 1; \mathcal{M}^n_{\mathcal{S}_{\mathcal{B}}}(0,1) = 3; \mathcal{M}^n_{\mathcal{S}_{\mathcal{B}}}(1,0) = 4; \mathcal{M}^n_{\mathcal{S}_{\mathcal{B}}}(1,1) = 1. \tag{26}$$

Given the simplicity of this problem and the fact that we are, in this first moment, more interested in seeing what happens inside the hypercube, a few simplifications will be done: we will fix the population size; there will be no reinitializations; and the fitness will be the raw fitness, which we choose to be:

$$\mathcal{F}(\mathscr{I}) = \mathcal{F}_r(\mathscr{I}) = \mathcal{M}^n_{\mathcal{S}_{\mathcal{B}}}(\mathscr{I}). \tag{27}$$

We make two experiments. In the first, we fix the population size to be $\mathcal{P}$= 1, 000, 000, which compared to the size of $\mathcal{S}_{\mathcal{B}}$ can be considered to be infinite. The result is shown in figure 4. The contour lines represent constant values of $\mathscr{P}\langle\mathcal{F}\rangle$, according to equation (10) for the fitness defined in equation (27). The lines describe the movement of FPBIL's and PBIL's probability vectors.

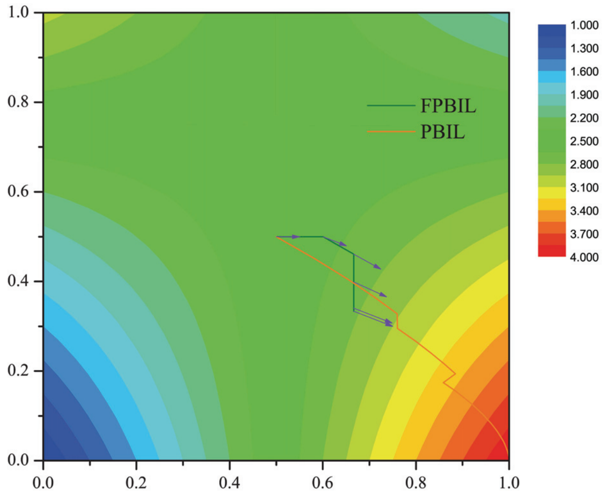


Fig. 4. Comparison between FPBIL and PBIL in $\mathcal{H}_2$; $\mathcal{P}$=1,000,000.

We see clearly that PBIL certainly finds the result to be "4", but FPBIL's line ends mysteriously. This is FPBIL's mutation in action. Since n = 2, the minimum value of d allowed is d_n = d_2 = 1/3—FPBIL's $\mathscr{P}$ can move only inside $[1/3, 2/3]^2$. This doesn't mean FPBIL can't find the result "4". In fact, from point (2/3, 1/3), the probability of getting the result "4" is 4/9, 2 times higher than the probability of getting "1" or "2" and 4 times higher than that of getting the result "3".The mutation in PBIL is more subtle and can be observed in the two sudden breaks suffered by PBIL's line.

We also highlight, in the same figure, the blue arrows which represent the gradient of $\mathscr{P}\langle\mathcal{F}\rangle$. Note that before the FPBIL's line reach the limits of $[1/3, 2/3]^2$, it (differently from PBIL's line) follows a direction very near from that of the gradient, which is just excellent, considering the discussion in section 3.1, meaning that in the limit of big values of $\mathcal{P}$, FPBIL's line can follow the gradient of $\mathscr{P}\left\langle \mathcal{F} \right\rangle$ to the optimum solution.

From this first experiment we are tempted to think PBIL is much better. But let us not forget this was an "almost infinite" population size experiment. In real applications we generally cannot span $\check{\mathcal{H}}_n$completely (whenever we can, we surely will not need FPBIL). Hence, in the second experiment, we fix $\mathcal{P}$ = 2 (at maximum, half the elements of $\mathcal{S}_{\mathcal{B}}$). The results are in figure 5. This time we see what generally happens in a real world problem. Both PBIL and FPBIL get more confused, but while FPBIL's mechanisms keep it doing its search inside $[1/3, 2/3]^2$, PBIL converges prematurely to a local optimum.

The next problem is, in a sense, a tougher version of this first.

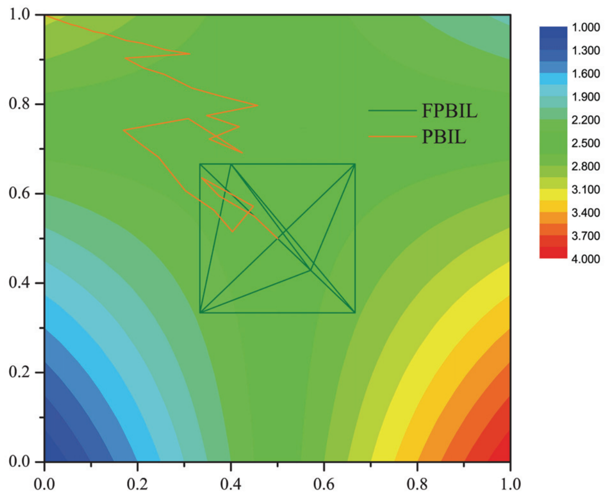


Fig. 5. Comparison between FPBIL and PBIL in $\mathcal{H}_2$; $\mathcal{P}$=2.

4.2 Banana

The banana problem consists in minimizing the Rosenbrocks function (Gill et al., 1981):

$$B(x,y) = 100\left(y - x^2\right)^2 + (1-x)^2. \tag{28}$$

From a simple observation of the expression of this equation, we may conclude, without trouble, that a minimum of $B(x, y)$ occurs for $(x, y) = (1, 1)$. Also it is not difficult to show analytically that this is the only point where $B(x, y)$ becomes stationary. However, looking at the graph of $B(x, y)$ it is impossible to come to the same conclusion.

It is quite obvious the existence of a valley located at $y = x^2$, but finding the exact point of the valley where $B(x, y)$ is minimal is not simple at all. The difficulty in having such a view is due to the factor 100 that multiplies only $(y-x^2)^2$, leaving out the term $(1-x)^2$. Only when observed in a logarithmic scale, such as in figure 6, does the region where the minimum is located become apparent. The white line is a contour line that shows the banana shape, which names the problem.

The Rosenbrocks function have been classically used to test optimization algorithms, exactly because of the difficulty that this function imposes, especially for gradient-based search algorithms.

The set $\mathcal{S}_{\mathcal{B}} \subset \mathbb{R}$ to be codified into binary vectors, for the use of PBIL and FPBIL algorithms will be [- 4.194304; 4.194304)^2, with a granularity of 0.000001 in both variables x and y. This means that each variable needs 23 bits to represent $\mathcal{S}_{\mathcal{B}}$, resulting in a total of 2^{46} = 70, 368, 744, 177, 664 possibilities.

More formally we have, with n = 2 ┌23 = 46,

$$\mathcal{M}^{n}_{\mathcal{S}_{\mathcal{B}}}(\mathscr{I}) = (x, y), \tag{29}$$

With

$$x = -4.194304 + \frac{G_1(\mathscr{I})}{1000000}; \tag{30}$$

$$y = -4.194304 + \frac{G_2(\mathscr{I})}{1000000}, \tag{31}$$

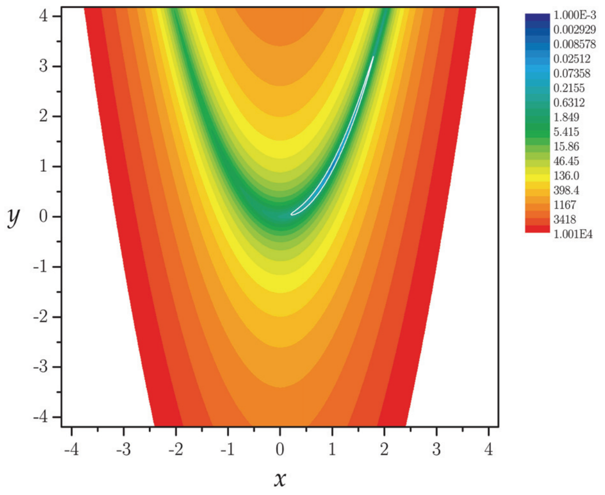


Fig. 6. Contour lines of $\log_{10} B(x, y)$. The white curve, in a banana shape, highlights the blue area where the minimum occurs.

Where $G_1(\mathscr{I})$ is the decoding of the first half of $\mathscr{I}$ and $G_2(\mathscr{I})$, of the second, both using Gra code[2] (Knuth, 2002). The fitness function of the banana problem used in this work is simply $B(x, y)$:

$$\mathcal{F}_b(\mathscr{I}_i) = B(x, y) = B \circ \mathcal{M}^{n}_{\mathcal{S}_{\mathcal{B}}}(\mathscr{I}_i). \tag{32}$$

[2] The use of Gray code may improve results considerably (Baluja, 1995).

Since we are dealing with a minimization problem and $\mathcal{F}_p(\mathscr{I}^+) = 0$, the standard fitness we use will be the raw fitness itself:

$$\mathcal{F}_p(\mathscr{I}_i) = B \circ \mathcal{M}^n_{\mathcal{S}_B}(\mathscr{I}_i). \tag{33}$$

In order to compare FPBIL to PBIL, we executed each algorithm 100 times and computed the average of the corresponding best individuals after a number of fitness evaluations. The result is shown in figure 7. We can see that the initial advantage of PBIL is amply overcome in the last fitness evaluations (approximately by a factor of 10^6). PBIL stagnates after 2, 000 fitness evaluations while FPBIL keeps finding better results in a constant rate until the end.
The next problem is a classical one concerning evolutionary search algorithms based on bit vectors.

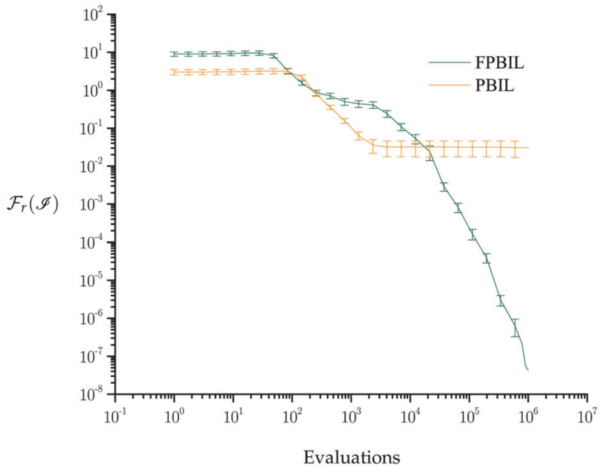


Fig. 7. Comparison between FPBIL and PBIL.

4.3 The four peaks problem

Consider the two functions defined on $\check{\mathcal{H}}_{100}$:

$$\mathrm{O}(\mathscr{I}) = \text{number of contiguous 1's of } \mathscr{I} \text{ starting in position 1;} \tag{34}$$

$$\mathrm{Z}(\mathscr{I}) = \text{number of contiguous 0's of } \mathscr{I} \text{ ending in position 100;} \tag{35}$$

where, for example, $O(011 \cdots 111) = 0$, $O(111 \cdots 111) = 100$, $Z(111 \cdots 110) = 1$ and $Z(000 \cdots 010) = 1$. Consider also the reward function

$$R(Z(\mathscr{I}), O(\mathscr{I}); T) = \begin{cases} 100 + T, & \text{if} \quad Z(\mathscr{I}) \geq T \quad \text{and} \quad O(\mathscr{I}) \geq T; \\ 0, & \text{otherwise.} \end{cases} \tag{36}$$

defined on $\{0, 1, 2, \ldots, 100\}^2 \times \{0, 1, 2, \ldots, 50\}$. In the four peaks problem, the objective is to maximize the function

$$F_T(\mathscr{I}) = \max\{Z(\mathscr{I}), O(\mathscr{I})\} + R(Z(\mathscr{I}), O(\mathscr{I}); T). \tag{37}$$

Observing F_T's plot in figure 8, one perceives that the four peaks problem is highly deceptive. There are two regions. One rewarded, corresponding to the upper surface, and another one, not rewarded, corresponding to the lower one. No point of the not-rewarded region (which increases with T) supplies us with any indication of the existence of the reward, giving the wrong impression of the existence of just peaks P_1 and P_2—corresponding to $F_T(\mathscr{I}) = 100$—while there still are the peaks P_3 and P_4—corresponding to $F_T(\mathscr{I}) = 200$, the global optimums.

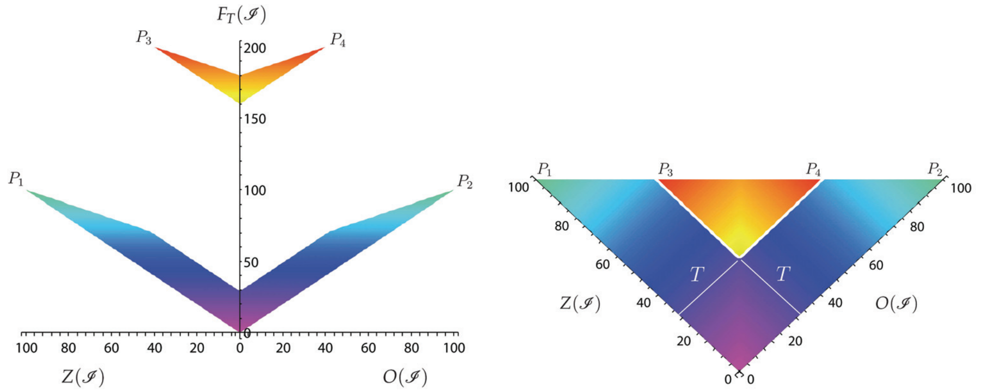


Fig. 8. Plot of $F_T(\mathscr{I})$, the objective function of the four peaks problem.

All the tests of the four peaks problem , carried through in this work have had $T = 30$ corresponding to a great bigger difficulty than the maximum difficulty used in (Baluja & Caruana, 1995), when, amongst a 25 total executions, the PBIL prematurely converged 20 times (the best result) and the genetic algorithms, between 22 and 25 times.

The raw fitness used in the four peaks problem was simply the value of $F_T(\mathscr{I})$: $\mathcal{F}_r(\mathscr{I}_i) = F_T(\mathscr{I}_i)$. Since one is dealing with a maximization problem and $\mathcal{F}_s(\mathscr{I}^+) = 200$, the standard fitness was $\mathcal{F}_s(\mathscr{I}_i) = 200 - F_T(\mathscr{I}_i)$. Figure 9 shows the comparison between FPBIL and PBIL, where the averages of the best fitness, after a number of fitness evaluations, are plotted for each algorithm. In 100 runs, PBIL was not able to reach the rewarded region, while the FPBIL did it every time, having as worst result $\mathcal{F}_r(\mathscr{I}_i) = 178$.

In the four peaks problem, the observation of the probability vector's evolution gives avery interesting insight into the algorithms. Figure 10, for example, illustrates a typical FPBIL run. It can be very clearly seen that during the first 1, 000, 000 fitness evaluations there were 4 reinitializations. After the second reinitialization, around the 2000th generation, FPBIL clearly reaches the global optimum. The PBIL, on the other hand, as shown in figure 11, converges, by the 2000th generation, to P_2. It is also worth noting the occurrence of mutation in PBIL. The white region corresponds to the probability vector's component equal to 1. The many red spots are the effects of mutation on the several components, making them change toward the value 0.5.

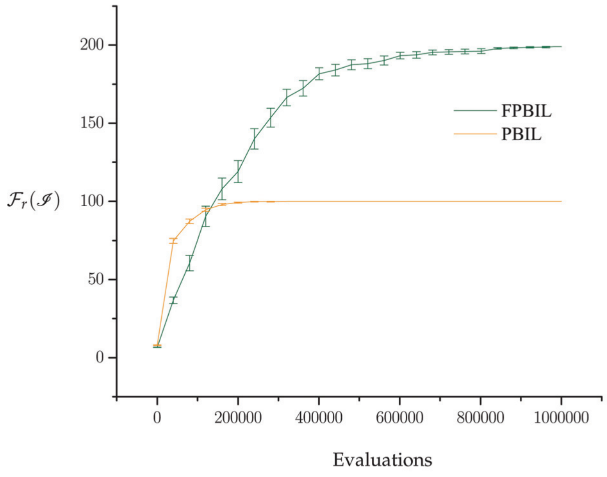


Fig. 9. Comparison between FPBIL and PBIL.

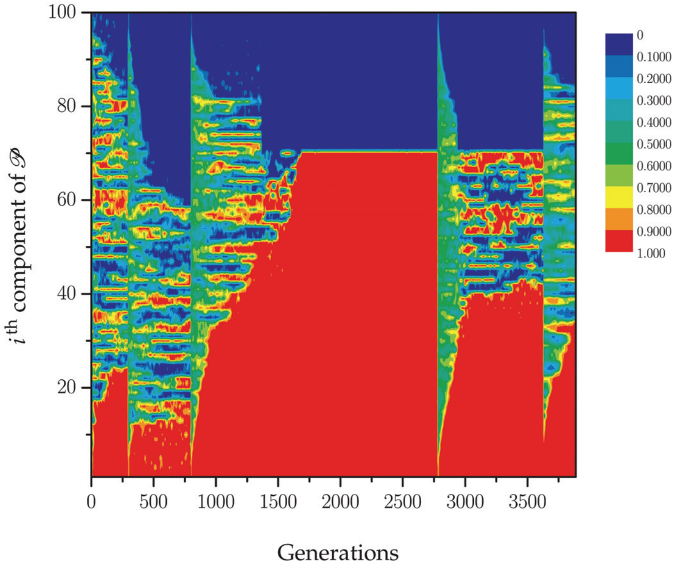


Fig. 10. Typical evolution of the probability vector in FPBIL.

4.4 TSP Rykel48

A traveling salesman must visit N cities, returning, in the end, to the city of origin, so that no city is visited twice. There are several possible routes (for $N > 2$). In fact, the number of routes is $(N-1)!$. The traveling salesman problem (TSP) consists in finding the shortest route.

The TSP is a NP problem, meaning that there is not yet an algorithm of polynomial order that can solve it. TheNP class can be considered as an intermediary computational complexity class, between classes P and EXP; as only a great amount of combinations is responsible for the demand of time (Lewis & Papadimitriou, 2000), the evaluation of each combination is usually the easy part.

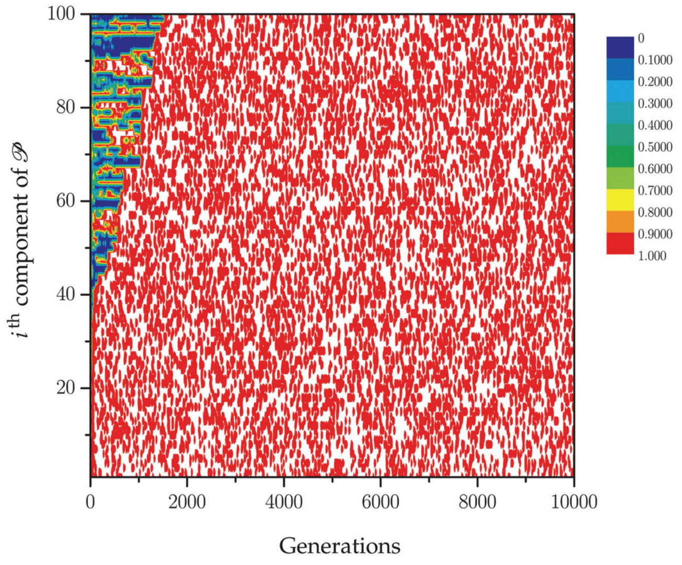


Fig. 11. Typical evolution of the probability vector in PBIL.

Rykel48 (TSPLIB, 2006) is a asymmetrical TSP with 48 cities resulting in a total of 258,623, 241,511,168,180,642,964,355,153,611,979,969,197,632,389,120,000,000,000 possible routes. In an asymmetric TSP the distance from one city A to another city B may be different from the distance from B to A, modeling, perhaps, single handed roads. Although the symmetric and asymmetric TSPs share the same number of routes (for the same amount of N), the asymmetry mixes up the search space topology, resulting in more complexes TSPs.

An important difference between Rykel48 TSP and the former problems is that the restriction that no city can be visited more than once prevents the direct codification of routes into bit vectors. The routes must be represented in an indirect way. In this work, we used the random keys representation (Bean, 1994; Caldas, 2006).

The Rykel48 TSP's raw fitness used in this work was simply the length of each route C_i corresponding to individual $\mathscr{I}_i$:

$$\mathcal{F}_b(\mathscr{I}_i) = C_i. \tag{38}$$

Since it is a minimization problem, we could have $\mathcal{F}_p(\mathscr{I}_i) = \mathcal{F}_b(\mathscr{I}_i)$. But since $\mathcal{F}_p(\mathscr{I}^+) = 14,422 \neq 0$, the standard fitness used will be

$$\mathcal{F}_p(\mathscr{I}_i) = C_i - 14,422. \tag{39}$$

Figure 12 shows the result. As it can be seen, FPBIL keeps the lead formost of its execution, especially in the latest 500,000 fitness evaluations.

Figure 13 shows the minimum and maximum values found after a number of the algorithms execution. The shortest route found by FPBIL was 14, 674, only 1.75% higher than the global optimum. Note that the PBIL presented a greater dispersion around the average.
At this point, it must be emphasized that the route length 14, 422 is not easily reached by any general purpose search algorithm. For example, the genetic algorithms only reach values close to 16, 500 (Machado, 1999) and the algorithms based on ant colonies—designed specifically to find smaller routes—achieve the optimum value only when processed in parallel, even so, only when assisted with heuristics (de Lima, 2005). Fig. 13 shows that PBIL is capable of reaching values just below 15, 000. The fact that FPBIL finds routes with the length of 14, 674 is a remarkable achievement.

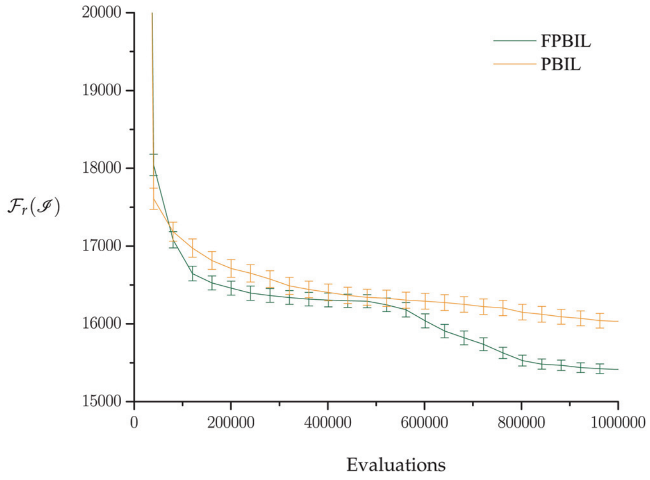


Fig. 12. Comparison between FPBIL and PBIL.

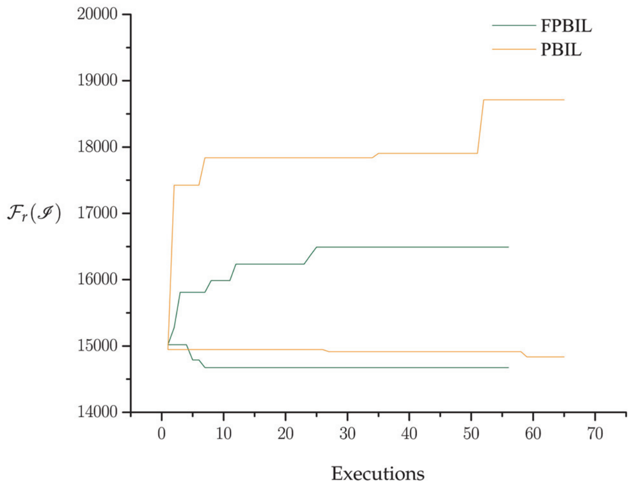


Fig. 13. Maximum and minimum values after a number of executions.

5. Conclusion

It can be affirmed, in conclusion to this chapter, that the FPBIL is an evolutionary algorithm, competitive with the best current optimization techniques, compact, relatively modest in the use of computational resources—like PBIL—, well founded, efficient, robust, self-adaptable, simple and parameterless.

Furthermore, the examples show that the FPBIL is efficient at both numerical and combinatorial problems. Here we should highlight the Four Peaks Problem, a highly deceptive problem handled very well by FPBIL.

FPBIL is conceptually simple and intuitive, since it does not require much sophisticated knowledge; it is compact, in the sense that it can be programmed with a few lines of code; and uses little amount of memory, since there is no need to store individuals of a population in some data structure.

The radically different way the mutation is handled in FPBIL is based on the probabilitie distribution inherent of the probability vector itself. This is updated using all the available information in each generation. These modifications enable the FPBIL to acquire self-adjustable features—such as the mechanism of variable population size—making the algorithm more efficient and more robust. Efficient in the sense that it finds solutions in less time; robust, meaning it has more resources to escape from local optimums.

With the proposition of FPBIL, we expect to have added relevant theoretical and practical tools, presenting feasible improvements with a considerable economic return, in both cost and benefit.

There still are, however, improvements which might be incorporated into FPBIL. After escaping from a local optimum, the FPBIL tends to approach the global optimum more slowly than other algorithms—PBIL, for example. Considering the process as a whole, the FPBIL takes advantage (since PBIL get caught more easily), but maybe it is possible to combine FPBIL with some other fast search algorithm, resulting in an even more efficient algorithm.

Other improvements can appear by constructing a multi-objective FPBIL— adapting the techniques from (Machado, 2005)—or even a parallel FPBIL—based on the techniques of (de Lima, 2005). One can still try to incorporate some kind of heuristic to the FPBIL perhaps some described in (de Lima, 2005). Works in these directions prove that these complementary techniques tends to produce better solutions.

6. Acknowledgments

In first place, I must thank God for all the great things he has done for me; for giving me strength and wisdom to conclude this work.

I also thank the invitation done by Dr. Vedran Kordic who started all of this; who has shown always very interested and helpful in his cordial emails.

Rubem and Lucia Caldas contributed with a fat slice of the costs for the publication of this material. I Thank them very much for this and also for the affection always demonstrated.

I would like to make a special thanks to Rubem Natan, Gisele Flores and Walter Aiken III, for the time they have spent trying to eliminate the maximum number of errors that I hid inside this chapter while I was preparing it. You guys are great!

I thank also all my colleagues from CNEN and COPPE/UFRJ for the support, specially those guys from room 429. You are insane!
I thank a lot my family, which have always encouraged and loved me. Rubem, Lúcia, Mrs. Rose, Mr. Antônio, Gisele, Elso, Ricardo, Anna, Luciana, Rosimar, Alexandre, Antônio Jr. and Tarsila. In particular, I would like to thank my wife, Renata, who promptly changed our vacation for this project. Love you! (not because of that!!!).
I also wanna thank my newest group of friends: Elthom, Susanne, Gustavo, Graciana, Victor and Leilane. You guys have been very important to me lately. You rock!

7. References

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. CMU-CS-94-163, Pittsburgh, PA. URL http://citeseer.ist.psu.edu/baluja94population.html

Baluja, S. (1995). An empirical comparison of seven iterative and evolutionary function optimization heuristics. Tech. Rep. CMU-CS-95-193. URL http://citeseer.ist.psu.edu/baluja95empirical.html

Baluja, S., & Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. In A. Prieditis, & S. Russel (Eds.) *The Int. Conf. on Machine Learning 1995*, (pp. 38–46). San Mateo, CA: Morgan Kaufmann Publishers. URL http://citeseer.ist.psu.edu/baluja95removing.html

Baluja, S., & Davies, S. (1998). Fast probabilistic modeling for combinatorial optimization. In *AAAI/IAAI*, (pp. 469–476). URL http://citeseer.ist.psu.edu/baluja98fast.html

Bean, J. C. (1994). "genetic algorithms and random keys for sequencing and optimization". *ORSA Journal on Computing*, *6*(2).

Caldas, G. H. F. (2006). *Algoritmo Evolucionário não Parametrizado Aplicado ao Problema da Otimiza¸cão de Recargas de Reatores Nucleares*. Ph.D. thesis, COPPE/UFRJ, Brazil.

de Lima, A. M. M. (2005). *Recarga de Reatores Nucleares Utilizando Redes Conectivas de Colônias Artificiais*. Ph.D. thesis, COPPE/UFRJ, Rio de Janeiro.

Gill, P. E., Murray, W., & Wright, M. H. (1981). *Practical Optimization*. San Diego: Academic Press.

Goldberg, D. E. (1989). *GENETIC ALGORITHMS in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems*. Massachusetts: MIT Press, second ed.

Juels, A., Baluja, S., & Sinclair, A. (1993). The equilibrium genetic algorithm and the role of crossover. URL http://citeseer.ist.psu.edu/juels93equilibrium.html

Knuth, D. E. (2002). *The Art of Computer Programming*. Stanford: Addison-Wesley, pre fascicle 2A ed.

Koza, J. R. (1992). Genetic Programming - On the Programming of Computers by Means of Natural Selection. Cambridge: MIT Press.

Lewis, H. R., & Papadimitriou, C. H. (2000). *Elementos de teoria da Computação*. Porto Alegre: Bookman, second ed.

Machado, M. D. (1999). *Um Novo Algoritmo Evolucionário com Aprendizado LVQ para a Otimização de Problemas Combinatórios como a Recarga de Reatores Nucleares*. Master's thesis, COPPE/UFRJ, Rio de Janeiro.

Machado, M. D. (2005). *Algoritmo Evolucionário PBIL Multi Objetivo Aplicado ao Problema da Recarga de Reatores Nucleares*. Ph.D. thesis, COPPE/UFRJ, Rio de Janeiro.

TSPLIB (2006). TSPLIB - a library of traveling salesman problem and related problem instances. URL http://nhse.cs.rice.edu/softlib/catalog/tsplib/tsp/

Part II:

Hybrid and Harmony Search

A Memetic Algorithm Assisted by an Adaptive Topology RBF Network and Variable Local Models for Expensive Optimization Problems

Yoel Tenne and S.W. Armfield
School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, Australia

1. Introduction

A common practice in modern engineering is that of **simulation-driven optimization**. This implies replacing costly and lengthy laboratory experiments with **computer experiments**, i.e. computationally-intensive simulations which model real world physics with high fidelity. Due to the complexity of such simulations a single simulation run can require up to several hours of CPU time of a high-performance computer [45, 56, 61].

With computer experiments the simulation-driven optimization process is cast as a nonlinear optimization problem having three distinct features:

- There is typically no analytic expression for the relation between inputs (candidate designs) and outputs, i.e. it is a **black-box function**.
- Each simulation run is expensive so only a small number ($\sim$ 200) of runs can be made.
- The underlying real-world physics and/or numerical solution often yield an inputs–output landscape which is multimodal and nonsmooth.

A promising approach to tackle such problems is the **surrogate-assisted memetic optimization**. A memetic algorithm combines an evolutionary algorithm (EA) with an efficient local search so as to obtain both efficient exploration and exploitation during the optimization search [21, 65]. A surrogate-model is a computationally cheaper mathematical approximation of the expensive objective function and is used during the optimization search in lieu of the expensive function [2, 45] (in some references the term **metamodel** is used synonymously while 'surrogate-model' is reserved for a lower-fidelity simulation [42, 87]). Thus, using surrogate-models circumvents the problem of simulation cost and allows evaluation of many candidate designs.

In this study we propose a surrogate-assisted memetic algorithm which builds upon recent advances in computational intelligence and optimization [9, 53, 60, 83–85, 94]. The proposed algorithm aims to address four open issues:

- Obtaining a global model with a small generalization error is too expensive: analysis has shown the number of sites required to achieve a fixed generalization error grows exponentially with the problem dimension [79]. To avoid allocating all function evaluations to the global model we employ **a combination of global and local surrogate-models** to achieve an efficient optimization search.

- The accuracy of a global Lagrangian model can degrade due to over-fitting: a Lagrangian model learns the exact features of the data which can lead to over-fitting and degrades its generalization ability. To address this we use as a global surrogate-model an **artificial neural network based on a RBF network (RBFN) with an adaptive network topology**. We describe an efficient method for adapting and training the network.
- Convergence to a false optimum: the local search relies on local models, hence if these are badly inaccurate the local search may converge to a false optimum. To address this we employ a **trust-region framework applied to general nonlinear local models**. Such models can describe a complicated landscape better than the quadratic models of the classical trust-region approach. We propose a framework for safeguarding and improving the models' accuracy.
- Difficulty in selecting an optimal model: different models can be used during the local search, e.g. RBF and Kriging. Due to lack of information the user typically chooses an inoptimal model which degrades the local search performance. To address this we describe **a method for model selection based on an approximate generalization error**. The method results in local models which vary during the local search.

Accordingly, in this chapter we propose a framework of memetic optimization using variable global and local surrogate-models for expensive optimization problems. To obtain a global model with good generalization ability it uses an RBFN artificial neural network. During the local search it makes an extensive use of accuracy assessment to select the local models and to improve them if necessary. It also employs the trust-region approach but replaces the quadratic models with the more general RBF and Kriging models. Rigorous performance analysis shows the proposed algorithm outperforms several variants of a reference surrogate-assisted EA.

This chapter is organized as follows: Sect. 2 reviews related work and Sect. 3 describes in detail the proposed algorithm. This is followed by Sect. 4 which provides the performance analysis and lastly Sect. 5 summarizes this chapter.

2. Related work

2.1 Expensive optimization problems

Since EAs require many function evaluations to converge several approaches have been studied so as to make them applicable to expensive optimization problems.

One such approach is **fitness inheritance**, where only a fraction of the offspring are evaluated with the computationally expensive objective function and the rest inherit their fitness from their parents [32, 75].

A second approach is that of **hierarchical** or **variable-fidelity** optimization which uses several computer simulations of varying computational cost (fidelity); promising candidate solutions migrate from low- to high-fidelity simulations and vice versa [15, 68, 71].

A third approach, which we adapt in this study, is that of **surrogate-assisted** optimization [2, 20, 26, 30, 53, 63, 77, 83, 85, 94]. As mentioned, a surrogate-model is a mathematically-cheaper approximation of the expensive function (typically an interpolant). A least-squares quadratic model (originally designed for real-world experiments which are noisy) are used in the Response Surface Methodology [5, 48]. Recent studies have used neural-networks [29, 61], Kriging [63, 72] and radial basis functions [85, 94]. The framework of surrogate-assisted optimization also involves the design of computer experiments [25, 73] and accuracy assessment of surrogate-models [42, 74].

2.2 Memetic optimization

Heuristics using random processes, such as EAs, are efficient in exploring the objective function landscape and can escape non-global optima. However, in late stages the optimization search focuses on a small subset of the search space so exploiting the local function behavior is preferred. This motivates the hybridization of random-based heuristics with efficient local search algorithms to balance **exploration–exploitation**, i.e. an efficient global and local search [88]. Within the framework of evolutionary optimization such algorithms are termed **hybrid algorithms** or **memetic algorithms**.
Examples include hybridization of an EA with a quasi-Newton and conjugate directions algorithms [21, 62, 66] and various direct search methods [33, 65, 91, 92]. Multiobjective memetic algorithms were studied in [19, 61] and a parallel algorithm was studied in [10]. An algorithm for selection among candidate local searchs was studied in [52]. Memetic algorithms aimed for expensive optimization problems were studied in [53, 54, 83, 84, 93, 94].

3. The proposed algorithm

3.1 Initialization and main loop

Analysis shows the number of sites required to achieve a fixed interpolation error grows exponentially with problem dimension [79]. This implies it is inefficient to allocate most or even all function evaluations to a single model as this may still result in an inaccurate model. Accordingly, we use a sequential approach where we only aim for a coarse global model and then use the remaining function evaluations to converge to an optimum [87]. As such, the algorithm begins by generating a Latin Hypercube sample (LHS) of $N_0 = 0.2fe_{\max}$ where $fe_{\max}$ is the prescribed limit on evaluations of the expensive function. This provides a space-filling sample which improves the model accuracy [41, 73]. The sites are evaluated with the true objective function to obtain their corresponding responses and both are copied into a cache which is initially empty. Next, a global model is generated based on all cached sites using the procedure described in Sect. 3.2. We then search for an optimum of this model using a memetic algorithm. Lastly in the optimization iteration, a local search is initiated from the predicted optimum so as to converge to an optimum of the expensive function, as described in Sect. 3.4. The main loop terminates when the number of function evaluations reaches the prescribed limit $fe_{\max}$ ($fe_{\max}$ = 100, 150 and 200 were used for performance analysis). A pseudocode of the main algorithm is given in Algorithm 1.

Algorithm 1: A pseudocode of the main loop.

```
generate initial sites with LHS and evaluate them;
copy sites into the cache;
while $fe \leqslant fe_{\max}$ do
    generate a global model based on the cache;
    perform a local search;
    update cache;
```

3.2 A variable-topology RBFN global model

A global model which is a Lagrangian interpolant, i.e. satisfying the conditions of exact interpolation

$$\mathcal{S}(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)\,,\; i = 1 \ldots n\ (=\text{sample size})\,, \tag{1}$$

can suffer from two demerits: a) it can generalize poorly due to over-fitting to the given data [4, 7, 34] and b) it can become computationally-expensive (since it accounts for all sites) and numerically unstable (due to ill-conditioning) [6, 11, 28].

To circumvent these issues we use for the global model an artificial neural network with radial basis functions neurons (processing units), a design termed an **RBF network** (RBFN). Such networks have two merits: a) both theoretical analysis and real-world experience have shown they generalize well [22, 43, 59, 81] and b) they have a simpler topology compared to other networks and hence are more easily implemented and trained [46, 57, 58].

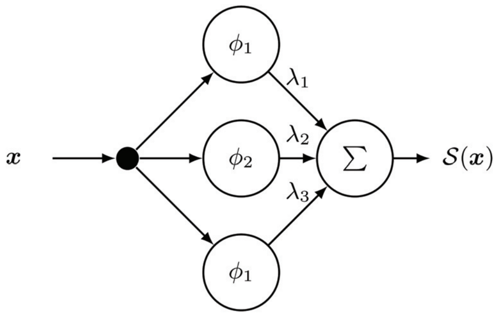


Fig. 1. An RBFN with three neurons (processing units).

Figure 1 shows a diagram of a typical RBFN. It comprises of three layers: the input layer, the processing layer comprised of neurons and the output layer which is a weighted sum of the neuron responses. An RBFN generalizes well and avoids over-fitting since it generates an abstraction of the data set. This is achieved by using fewer neurons than sample sites (so the centres of the neuron RBFs typically do not coincide with any of the data sites) and careful training of the network parameters. The response of an RBFN is given as

$$\mathcal{S}(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{t}_j\|_2}{c_j}\right)\,, \tag{2}$$

where N is the number of neurons, λ_j is a coefficient, $\boldsymbol{t}_j$ is a basis-function (or kernel) centre and c_j is a shape parameter (or hyper-parameter). The neurons are RBF Gaussian functions which assist in modelling nonlinear functions [22, 43, 49, 57].

To avoid ill-conditioning and expensive calculation the network needs to be compact (minimizing the number of neurons N) while still be capable of generalizing well. Also, it is difficult to prescribe an optimal topology so the network should be self-adaptive [18, 30, 31, 39, 58]. Accordingly, we implement such a self-adaptive network which operates as follows.

Initially, the data set is split into a training set ($\mathcal{X}_{\text{tra}}$) and a testing set ($\mathcal{X}_{\text{tra}}$) which are disjoint (we use a 80–20 training–testing ratio). Starting from a single neuron, the network is trained with $\mathcal{X}_{\text{tra}}$ and is tested with $\mathcal{X}_{\text{tes}}$, an approach termed **holdout** [23, 82]. The generalization error is measured by the normalized root mean square error (NRMSE) over $\mathcal{X}_{\text{tes}}$, i.e.

$$\text{NRMSE} = \sqrt{\frac{\sum_{i=1}^{|\mathcal{X}_{\text{tes}}|} (\mathcal{S}(\boldsymbol{x}_i) - f(\boldsymbol{x}_i))^2}{\text{Var}(f(\boldsymbol{x}_i))}} \,, \tag{3}$$

where $\boldsymbol{x}_i$ is the ith site in the testing set $\mathcal{X}_{\text{tes}}$ and the numerator is the sum of the Gaussian loss-function (or discrepancy)

$$L(\boldsymbol{x}) = (\mathcal{S}(\boldsymbol{x}) - f(\boldsymbol{x}))^2 \tag{4}$$

over then training set [34]. The denominator is the variance of the responses in the testing set. Besides the NRMSE the loss-function values over the training and testing set are also calculated, i.e.

$$L_{\text{tes}} = \sum_{i=1}^{|\mathcal{X}_{\text{tes}}|} (\mathcal{S}(\boldsymbol{x}_i) - f(\boldsymbol{x}_i))^2 \tag{5}$$

and similarly for the training set yielding L_{tra} . If NRMSE > NRMSE$^{\ulcorner}$ where NRMSE$^{\ulcorner}$ is prescribed than $0.1\,|\,\mathcal{X}_{\text{tra}}\,|$ neurons are added to the network and the new network is trained as explained below. The network stops growing if NRMSE ≤ NRMSE$^{\ulcorner}$ or if the number of neurons equals the number of training sites $(N = |\,\mathcal{X}_{\text{tra}}\,|)$. After the network stopped growing the chosen topology is that which had the lowest weighted error

$$e_{\text{w}} = 0.8L_{\text{tes}} + 0.2L_{\text{tra}} \,, \tag{6}$$

where a larger weight is given to the testing error over the training error.
For each number of neurons the network parameters (RBF centres, coefficients, shape parameters) need to be trained to achieve good generalization. While it is possible to train the network in a fully supervised manner by minimization of the generalization error convergence is slow [46]. Accordingly, we implement a fully unsupervised learning where the RBF centres are obtained by a k-means clustering algorithm [31, 46], the shape parameters are obtained from

$$c_j = \max\{0.1\bar{d}\,,\, 1\}\,, \quad j = 1 \ldots N \,, \tag{7}$$

where $\bar{d}$ is the mean l_2 distance between all sites in the data set $\mathcal{X}$ (related to the Gaussian rate of decay) [57, 58]), and the coefficients λ are obtained from the normal least-squares equations

$$\boldsymbol{\Phi}^{\text{T}}\boldsymbol{\Phi}\boldsymbol{\lambda} = \boldsymbol{\Phi}^{\text{T}}\boldsymbol{f}, \tag{8}$$

where $\boldsymbol{f}$ is the vector of responses and $\boldsymbol{\Phi}$ is the interpolation matrix

$$\boldsymbol{\Phi} : \Phi_{i\,,\,j} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{t}_j\|_2}{c_j}\right) . \tag{9}$$

Figure 2 shows an example of a model training with the variations in L_{tra} and L_{tes} . When the network is over-trained the testing error begins to grow. The parameters are taken from the cycle which minimized e_{w} before over-training. Figure 3 shows an example of the adaptation of the proposed RBFN. Algorithm 2 gives a pseudocode of the proposed algorithm for the adaptive RBFN.

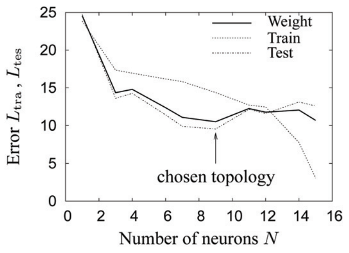
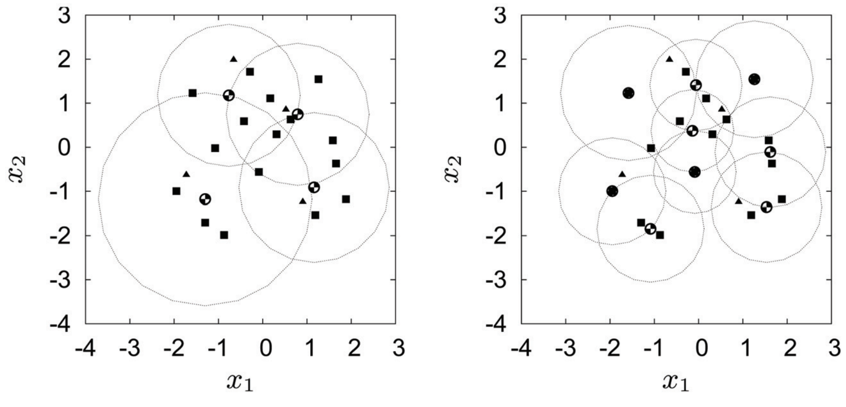


Fig. 2. An example of the RBFN training with the Rastrigin-5D objective function. As the number of neurons increases both training error and testing error decrease until overtraining commences at 9 neurons (indicated by an increase in the testing error). The chosen topology has the minimal weighted error.



(a) 4 neurons (b) 9 neurons

Fig. 3. An example of the RBFN topology adaptation with the Rastrigin-2D function. A sample of 20 sites was split into training (■) and testing (▲) sites. We show each topology by its RBFN centres (◕) and the corresponding shape parameters (the radius of the circles).

Algorithm 2: The proposed algorithm for the adaptive topology RBFN

```
inputs
    $\mathcal{X}$;                    /* set of sites and responses */

split $\mathcal{X}$ into a training set and a testing set which are disjoint;
set number of neurons $N = 1$;
repeat
    train network;
    calculate NRMSE and weighted error;
    if $NRMSE > NRMSE^\star$ then
        increase number of neurons;
until $NRMSE \leqslant NRMSE^\star$ or ( number of neurons is $|\mathcal{X}_{\text{tra}}|$ ) ;
select network topology yielding the smallest weighted error;
```

3.3 Memetic search for an optimum of the global model

After generating the global model $\mathcal{S}(\ \boldsymbol{x}\ )$ we use a memetic algorithm to search for an optimum of it. The memetic algorithm first employs a real-coded EA [8] for efficient exploration. The EA uses a population size $s_{\text{pop}} = 50$, linear ranking, stochastic universal sampling (SUS), intermediate recombination, elitism with a generation gap $g_{\text{gap}} = 0.9$ and the breeder-genetic-algorithm mutation operator with probability $p_{\text{m}} = 0.05$ [47]. The evolutionary search is stopped when no improvement is observed after $g_{\text{n.i.}} = 10$ generations; the small setting for $g_{\text{n.i.}}$ is since we do not require the EA to converge to a very accurate solution, as this is accomplished by the following step. The optimum found by the EA is then used as the initial solution for an SQP solver which uses the finite-differences quasi-Newton BFGS algorithm. This yields $\boldsymbol{x}_{\text{c}}$ an improved predicted optimum of the global model. During the memetic optimization stage approximate function values are obtained from the surrogate-model (the objective function is not used).

3.4 The local search

Since the global model is coarse $\boldsymbol{x}_{\text{c}}$ may be a bad approximation to a true optimum of the expensive function. Accordingly, we use $\boldsymbol{x}_{\text{c}}$ as an initial guess for a local search to search for a true optimum. Two considerations with the local search are efficiency (which suggests using local models requiring fewer sites than the global model) and accuracy (which suggests using a procedure to safeguard against convergence to a false optimum). Both of these goals are accomplished by using a trust-region approach, as described below. To further improve the local search we propose a method for selecting the model type (as either RBF or Kriging) and to improve the models, if necessary; this results in local models which vary during the local search.

3.4.1 A trust-region approach

The classical trust-region approach generates at each iteration a quadratic model and obtains its constrained optimum (a truncated Newton step) as a quadratic programming problem. However, such models cannot adequately describe a complicated or multimodal landscape so instead we generate more flexible local models (either RBF or Kriging) and obtain their constrained optimum (in the trust-region) using a memetic search. The trust-region framework safeguards the model accuracy and ensures convergence to an optimum of the expensive objective function, i.e. it is a **framework for managing models** [1, 12, 68].

The initial trust-region is taken as a cuboid centred at $\boldsymbol{x}_c$ the predicted optimum of the global model and is of size Δ (with an initial size $\Delta_0 = 0.1$), i.e.

$$\mathcal{T} = \{\boldsymbol{x} : \|\boldsymbol{x}_\mathrm{c} - \boldsymbol{x}\|_\infty \leqslant \Delta\} . \tag{10}$$

All cached sites which are in the trust-region are used to generate the local surrogate-model. We exclude remote sites to emphasize only the local function behaviour.

The model type is selected using the algorithm described in Sect. 3.2 and the constrained optimum of the local model in $\mathcal{T}$, $\boldsymbol{x}_\mathrm{m}$, is obtained by the memetic search described in Sect. 3.3.

Following the classical trust-region approach the predicted optimum is evaluated with the true objective function and a merit value is calculated

$$\rho = \frac{f(\boldsymbol{x}_\mathrm{m}) - f(\boldsymbol{x}_\mathrm{c})}{\mathcal{S}(\boldsymbol{x}_\mathrm{m}) - \mathcal{S}(\boldsymbol{x}_\mathrm{c})} , \tag{11}$$

where $\mathcal{S}(\boldsymbol{x}_\mathrm{m})$ now denotes the current local surrogate-model.

A main difference to the classical trust-region framework is that the latter assumes the quadratic model is accurate (i.e. based on an exact gradient and Hessian) while here we also need to account for model inaccuracy due to the interpolation on a finite set. As such, the model may be inaccurate due to an insufficient number of sites in the trust-region. Reducing the trust-region size too quickly due to model inaccuracy can lead to premature termination of the local search [9]. To avoid this we relate the model accuracy to the number of sites in the trust-region, denoted as s. A reasonable criterion to consider the model accurate is when $s \geq d + 1$ (d being the problem dimension). This threshold is based on the number of sites required to model the gradient of the objective function (and hence to identify a descent direction) by well-established methods like quasi-Newton finite-differences or polynomial interpolation [9]. However, if the allowed number of function evaluations fe_max is small and the problem dimension is high too many sites are needed to consider the model accurate. Accordingly, we use the threshold value $s^\ulcorner = \min\{d + 1 , 0.1fe_\mathrm{max}\}$.

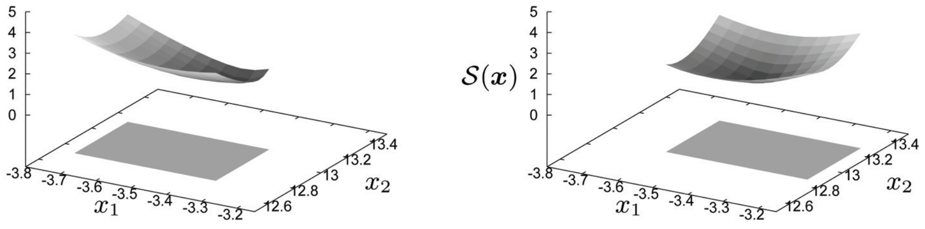
Based on ρ, s and $s^\ulcorner$ the proposed algorithm performs one of the following updates:

- if $\rho > 0$: then the surrogate-model is accurate since a better solution has been found. Following the classical trust-region framework we centre the trust-region at the new optimum $\boldsymbol{x}_\mathrm{m}$ and increase the trust-region size by a factor δ_+.
- if $\rho \leq 0$ and $s < s^\ulcorner$: the local model is inaccurate but this is attributed to an insufficient number of sites in the trust-region. Thus we improve the accuracy of the local model in the trust-region by adding a site using the model improvement algorithm (Sect. 3.4.3).
- if $\rho \leq 0$ and $s \geq s^\ulcorner$: the local model is based on a sufficient number of sites but fails to predict an improvement due to the trust-region size. Following the classical trust-region framework we decrease the trust-region size by a factor δ_-.

After the model and trust-region have been updated the current local search iteration is finished. The local search is stopped if the trust-region is small enough $\Delta < \Delta_{\min}$ (we use $\Delta_{\min} = \Delta_0 \cdot \delta_-^2$) or if the number of evaluations of the true objective function exceeds $fe_{\max}$.
Some additional comments on the local search:

- At most only two evaluations of the true function are performed at each local search iteration.
- All sites evaluated during the local search are added to the cache for later use.

Figure 4 shows an example of a local search with the proposed trust-region approach used with the Branin function. A pseudocode of the proposed trust-region local search is given in Algorithm 3.



(a) Iteration 2 (b) Iteration 5

Fig. 4. An example of the trust-region local search using local models (RBF or Kriging). The objective function is Branin. For iterations 2 and 5 the chosen model (Kriging) and the corresponding trust-region are shown.

Algorithm 3: A pseudocode of the local search.

```
inputs
    $\mathcal{X}$;                              /* set of sites */
    $\mathcal{Y}$;                          /* set of responses */
    $\boldsymbol{x}_c$;           /* predicted optimum of global model */

$s^\star = \min\{d+1\,,\,0.1fe_{\max}\}$ ;        /* model accuracy threshold */
repeat
    define a cuboid trust-region with a face size $\Delta$ and centred at $\boldsymbol{x}_c$;
    find the cached sites inside the trust-region;
    generate a local surrogate-model using these sites (use model selection);
    use a memetic search to find $\boldsymbol{x}_m$ the optimum of the local model in the
    trust-region;
    calculate a trust-region merit value $\rho$;
    perform a trust-region update:
    $\begin{cases} \rho \geqslant 0 & \boldsymbol{x}_c \leftarrow \boldsymbol{x}_m \text{, increase } \Delta \\ \rho < 0 \bigcap s < s^\star & \text{improve local model (add a site)} \\ \rho < 0 \bigcap s \geqslant s^\star & \text{decrease } \Delta \end{cases}$
until $\Delta < \Delta_{\min}$ or $fe \geqslant fe_{\max}$ ;
```

3.4.2 Model selection

To assist the optimization search we wish to generate a surrogate-model which is optimal, i.e. as accurate as possible. We select among two candidate models, namely radial basis functions (RBFs) or Kriging, as these have performed well in benchmark tests against other models [17, 25, 73, 74].

The RBF surrogate-model is a Lagrangian interpolant which is a linear combination of basis functions. To ensure the non-singularity of the interpolation matrix we consider an RBF model which uses linear basis functions [44] such that

$$\mathcal{S}(\boldsymbol{x}) = \sum_{i=1}^{n} \lambda_i \phi_i(\boldsymbol{x}) \,, \quad \phi_i(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_i\|_2 \,, \tag{12}$$

where $\boldsymbol{n}$ is the number of sites, $\phi_i(\boldsymbol{x})$ are the linear radial basis functions and the coefficients λ_i are obtained from the linear system

$$\boldsymbol{\Phi}\boldsymbol{\lambda} = \boldsymbol{f} \,. \tag{13}$$

A Kriging (or a spatial-correlation) model uses a global 'drift' function o which a stationary Gaussian process is overlaid; the former captures the global trend while the latter provides local adjustments [40, 45, 69]. We adapt the common approach where the drift function is taken as constant (e.g. set to 1) so the model is given by

$$\mathcal{S}(\boldsymbol{x}) = \beta + Z(\boldsymbol{x}) \,, \tag{14}$$

where β is the drift function coefficient and $Z(\boldsymbol{x})$ is the Gaussian process function [45, 69]. The Gaussian process is assumed to have a mean zero and variance σ. Deviating from the random error approach of the Response Surface Methodology, the response at any site is considered correlated with other sites. The correlation between two sites $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is defined by a covariance function

$$C(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma^2 R(\boldsymbol{x}_1, \boldsymbol{x}_2) \,, \tag{15}$$

where $R(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is a prescribed spatial correlation function (SCF). Following [45] we consider the exponential SCF

$$R(\boldsymbol{x}_1 \,, \boldsymbol{x}_2) = \prod_{j=1}^{d} \exp\left(-\theta |\boldsymbol{x}_{1\,,j} - \boldsymbol{x}_{2\,,j}|\right) \,. \tag{16}$$

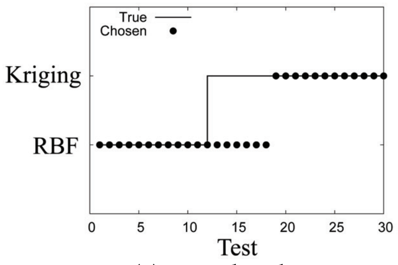
The Kriging model is defined once β and θ have been fixed. For a given data set the value of θ is obtained by maximum likelihood estimation [37]. Having found the optimal θ and assuming a constant drift function then the Kriging model is

$$\mathcal{S}(\boldsymbol{x}) = \hat{\beta} + \boldsymbol{r}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{R}^{-1}(\boldsymbol{f} - \boldsymbol{1}\hat{\beta}) \,, \tag{17}$$

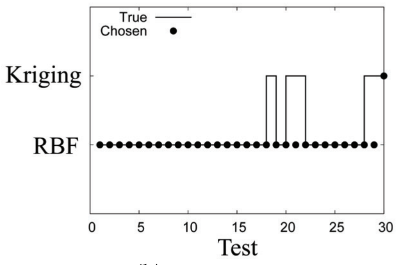
where R is the correlation matrix for data set $\mathcal{X}$, $\boldsymbol{r}$ is the correlation vector between $\boldsymbol{x}$ and $\mathcal{X}$ and $\hat{\beta}$ is the least-squares estimate of β

$$\hat{\beta} = (\mathbf{1}^{\mathrm{T}}\boldsymbol{R}^{-1}\mathbf{1})^{-1}\mathbf{1}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{f}\,. \tag{18}$$

Details of the Kriging code implementation are given in [78].
The two different possible models, namely linear RBF and Kriging, introduces the issue of **model selection**. To assist the local search we wish to select the most accurate model, i.e. having the least generalization error. Similarly to Section 3.2 we approximate the generalization error based on the available data set. While it is possible to use the holdout method for approximating the generalization error a better estimate is obtained if repeated models are generated and all sites are used both for training and for testing, an approach known as the **leave-one-out cross-validation** (LOOCV) [42, 80]. The estimate is obtained as follows: given a candidate model (in our case a linear RBF or Kriging) then for each site $\boldsymbol{x}_i$, $i = 1 \ldots n$ a surrogate-model is generated using all sites except $\boldsymbol{x}_i$ and the Gaussian loss-function of this surrogate-model is calculated at $\boldsymbol{x}_i$. The estimated generalization error is then the mean of all observed errors. The model corresponding to the smallest LOOCV error is assumed to be the most accurate. In this basic form the LOOCV procedure requires generating n surrogate-models, which is expensive. To circumvent this, for the RBF we use an efficient procedure proposed in [67] while for the Kriging we use a procedure proposed in [45].



(a) Rosenbrock-10D



(b) Rastrigin-20D

Fig. 5. Examples of the model selection algorithms. The solid line (—) indicates which model was more accurate based on a large sample of 250 sites while the dot (•) indicates which model was selected by the proposed method based on a small sample.

Figure 5 shows two examples of the proposed model selection algorithm. The following procedure was repeated 30 times to obtain statistically significant results. We used the Rosenbrock-10D and Rastrigin-20D test functions and 50 sites generated by LHS. The proposed method was used to select between an RBF model and a Kriging model. A separate testing sample of 250 LHS sites was used to obtain a more accurate estimate of the true generalization error of the models. It follows the proposed method selects (in the large majority of cases) the model whose true generalization error is indeed smaller.
The outcome of the model selection is that the proposed memetic algorithm uses **variable surrogate-models** (either linear RBF or Kriging) during the local search. A pseudocode of the model selection algorithm is given in Algorithm 4.

Algorithm 4: A pseudocode for model selection.

```
inputs
    $\mathcal{X}$;                                        /* set of sites */
    $\mathcal{Y}$;                                    /* set of responses */

for model type = linear RBF, Kriging do
    calculate LOOCV error for the model;
select the model having the smaller LOOCV error.
```

3.4.3 Model improvement

If the local model is deemed inaccurate, i.e. there is an insufficient number of sites in the trust-region then a new site is generated to improve the model accuracy (reduce its generalization error). Analysis of various surrogate-models (polynomial, RBF and Kriging) relates their generalization error to the distribution of sites [9, 28, 70]. Clustered sites do not provide sufficient information on the objective function and lead to an ill-conditioned interpolation matrix which further degrades the model accuracy. The distribution of sites is measured by the **fill distance**

$$h = \sup \min \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \,, \tag{19}$$

i.e. the radius of the largest ball in the feasible domain $\mathcal{F}$ which does not contain any sites in its interior [36, 70]. To improve the model accuracy (increase h) new sites should be added such that they are remote from existing sites. Thus, to improve the model in the trust-region we seek a site which maximizes the fill distance for the augmented set $\{\mathcal{X}_{\mathcal{T}} \cup \boldsymbol{x}_i\}$ where $\mathcal{X}_{\mathcal{T}}$ is the set of sites in the trust-region. To obtain the model-improving site $\boldsymbol{x}_i$ we formulate the nonlinear optimization problem

$$\begin{aligned} &\max \min\{\|\boldsymbol{x} - \boldsymbol{x}_j\|_2\} \,, \quad j = 1 \ldots |\mathcal{X}_{\mathcal{T}}| \\ &\text{s.t. } \boldsymbol{x} \in \mathcal{T} \end{aligned} \tag{20}$$

We solve (20) by generating an initial sample of candidate sites and starting an SQP solver from the best one (having the maximum separation distance). This results in sites distributed similarly to the **maximin design** [27]. After $\boldsymbol{x}_i$ has been found it is evaluated with the true objective function and is added to the cache. A pseudocode of the model improvement iteration is given in Algorithm 5.

Algorithm 5: A pseudocode for model improvement

```
inputs
    $\mathcal{X}$;                                        /* set of sites */

search for a site $\boldsymbol{x}_{\mathrm{i}} \in \mathcal{T}$ which maximizes the fill distance of $\{\mathcal{X}_{\mathcal{T}} \bigcup \boldsymbol{x}\}$;
evaluate $\boldsymbol{x}_{\mathrm{i}}$ and add $\boldsymbol{x}_{\mathrm{i}}$, $f(\boldsymbol{x}_{\mathrm{i}})$ to the cache;
```

3.5 Additional remarks

In this section we provide additional remarks on the complete algorithm.

- The local search is initiated only if the distance of the predicted optimum $\boldsymbol{x}_c$ from all cached sites is larger than $\Delta_{\min}$
- As the cache grows the interpolation matrix $\boldsymbol{\Phi}$ becomes ill-conditioned and this degrades the solution accuracy of (8) [28]. To circumvent this we solve (13) by the numerically stable **truncated singular value decomposition (TSVD)** such that

$$\boldsymbol{U}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{V} = \boldsymbol{\Phi}\,, \tag{21}$$

where $\boldsymbol{\Sigma}$ is the diagonal matrix of singular values σ_i of $\boldsymbol{\Phi}$. Given the responses vector $\boldsymbol{f}$ and defining

$$\boldsymbol{C} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{f} \tag{22}$$

and a vector $\boldsymbol{y}$ such that

$$\boldsymbol{y} : y_i = \begin{cases} c_i/\sigma_i & \sigma_i \geqslant \epsilon_{\mathrm{SVD}} \\ 0 & \sigma_i < \epsilon_{\mathrm{SVD}} \end{cases} \tag{23}$$

the solution vector is

$$\boldsymbol{\lambda} = \boldsymbol{V}\boldsymbol{y}\,. \tag{24}$$

Thus the solution vector is generated by the span of the vectors corresponding to a sufficiently large singular value. We use $\epsilon_{\mathrm{SVD}} = 10\epsilon$, where ϵ is the machine precision.

4. Performance analysis

We assessed the performance of the proposed algorithm using both mathematical test functions and a real-world problem of airfoil shape optimization. In these tests the proposed algorithm was also benchmarked against two variants of a reference surrogate-assisted EA which is representative of many others [64]; Algorithm 6 gives its pseudocode.

Algorithm 6: A pseudocode of the reference surrogate-assisted EA.

```
generate initial sites with LHS and evaluate them;
copy sites into the cache;
generate a global model based on the cache;
while $fe \leqslant fe_{\max}$ do
    search for model optimum using an EA for 10 generations;
    evaluate top 10% of elites with true objective function;
    add elites to cache;
    update global model and population fitness;
```

The two variants differ by the surrogate-model they use, namely either a linear RBF model or a Kriging model with an exponential spatial correlation function.

	General parameters	
$fe_{\max}$	max. (true) objective function evaluations	200
N_0	sample size for initial global model	$0.2 \cdot fe_{\max}$
	RBFN global model	
δ_{h}	holdout ratio	0.2
NRMSE*	threshold NRMSE	0.1
	Memetic algorithm	
s_{pop}	population size	50
g_{gap}	generation gap	0.9
$g_{\max}$	maximum generations	20
p_{m}	mutation probability	0.05
$g_{\mathrm{n.i.}}$	no-improvement generations to stop	10
	Trust-region algorithm	
$\varDelta_0$	initial trust-region radius	0.1
δ_+	trust-region size increase factor	2
δ_-	trust-region size decrease factor	0.5
$\varDelta_{\min}$	minimum trust-region radius	$\varDelta_0 \cdot \delta_-^2$
$\varDelta_{\max}$	maximum trust-region radius	$\varDelta_0 \cdot \delta_+^2$

Table 1. Parameter Settings for the Proposed Memetic Algorithm

All relevant parameters (e.g. initial surrogate-model sample and evolutionary parameters) were the same as in the proposed algorithm. Parameter settings are given in Table 1. To obtain statistically-significant results 30 runs were repeated for each test with the proposed algorithm and the two variants.

4.1 Mathematical test functions

For the mathematical tests functions we used the well-known Branin, Hartman 3 and Hartman 6 functions with a maximum evaluations limit of $fe_{\max} = 100$ [13]. To asses the impact of the 'curse of dimensionality' [3] we also used the well-known chained Rosenbrock (high epistasis) and Rastrigin function (high multimodality) functions with $fe_{\max} = 200$ [86, 90]. We set these small values for $fe_{\max}$ to measure performance under a constraint of resources (as function evaluations are considered expensive) [89]. Details of the test functions are given in Table 2. Test statistics are given in Table 3 which indicate the proposed algorithm outperformed the two surrogate-assisted EAs.

To determine in a rigorous manner if there is a statistically-significant difference between the results of the proposed algorithm and the two variants we applied the nonparametric one-tailed Mann–Whitney (or Wilcoxon) test which provides a test statistic U [35]. The null and alternative hypothesis are:

$$H_0 : P(x_i < x_p) \geqslant 0.5 \tag{25a}$$

$$H_1 : P(x_i < x_p) < 0.5\,, \tag{25b}$$

where $P(\boldsymbol{x}_i < \boldsymbol{x}_p)$ is the probability that a score of the proposed algorithm is larger (worse) than a score of one of the variants (i = 1, 2). Table 4 provides the test statistics for comparisons with the two variants over the five test functions and the decision rules. For the

Function	d	Definition	Feasible Domain	$f(\boldsymbol{x}_{\mathbf{g}})$[1]
Branin	2	$(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$	$[-5, 10] \times [0, 15]$	0
Hartman 3	3	$\sum_{i=1}^{4} c_i \exp\left[\sum_{j=1}^{4} a_{i,j}(x_i - p_{i,j})^2\right]$	$[0, 1]^d$	−3.86
Hartman 6	6	$\sum_{i=1}^{4} c_i \exp\left[\sum_{j=1}^{6} a_{i,j}(x_i - p_{i,j})^2\right]$	$[0, 1]^d$	−3.32
Rastrigin	20	$\sum_{i=1}^{n} \left\{x_i^2 - 10 \cdot \cos(2\pi x_i) + 10\right\}$	$[-5, 5]^d$	0
Rosenbrock	30	$\sum_{i=1}^{d}(2x_{i-1} - x_i^2)^2 + (1 - x_i)^2$	$[-2, 2]^d$	0

Table 2. Mathematical Test Functions

Function		Proposed	Ref.-RBF	Ref.-Kriging
	Mean	$\mathbf{9.542e-01}$	$1.324e+00$	$2.599e+00$
	S.D.	$\mathbf{6.706e-01}$	$8.142e-01$	$2.230e+00$
Branin	Median	$\mathbf{6.518e-01}$	$1.107e+00$	$1.719e+00$
	Best	$\mathbf{3.988e-01}$	$4.209e-01$	$4.268e-01$
	Worst	$\mathbf{2.593e+00}$	$3.287e+00$	$8.875e+00$
	Mean	$\mathbf{-3.764e+00}$	$-3.544e+00$	$-3.346e+00$
	S.D.	$\mathbf{1.045e-01}$	$2.157e-01$	$3.478e-01$
Hartman 3	Median	$\mathbf{-3.779e+00}$	$-3.533e+00$	$-3.426e+00$
	Best	$\mathbf{-3.862e+00}$	$-3.846e+00$	$-3.829e+00$
	Worst	$\mathbf{-3.325e+00}$	$-2.768e+00$	$-2.354e+00$
	Mean	$\mathbf{-3.014e+00}$	$-1.955e+00$	$-1.935e+00$
	S.D.	$\mathbf{2.611e-01}$	$6.159e-01$	$5.732e-01$
Hartman 6	Median	$\mathbf{-3.114e+00}$	$-1.769e+00$	$-2.020e+00$
	Best	$\mathbf{-3.259e+00}$	$-3.079e+00$	$-2.958e+00$
	Worst	$\mathbf{-2.200e+00}$	$-8.777e-01$	$-7.943e-01$
	Mean	$\mathbf{1.213e+02}$	$1.654e+02$	$1.719e+02$
	S.D.	$\mathbf{2.949e+01}$	$1.999e+01$	$2.686e+01$
Rastrigin 20	Median	$\mathbf{1.224e+02}$	$1.662e+02$	$1.693e+02$
	Best	$\mathbf{5.797e+01}$	$1.310e+02$	$1.063e+02$
	Worst	$\mathbf{1.789e+02}$	$2.173e+02$	$2.242e+02$
	Mean	$\mathbf{2.039e+01}$	$5.525e+01$	$7.333e+01$
	S.D.	$\mathbf{3.970e+00}$	$1.486e+01$	$2.508e+01$
Rosenbrock 30	Median	$\mathbf{1.988e+01}$	$5.145e+01$	$6.850e+01$
	Best	$\mathbf{1.637e+01}$	$3.324e+01$	$3.576e+01$
	Worst	$\mathbf{3.656e+01}$	$9.150e+01$	$1.418e+02$

S.D.: standard deviation

Table 3. Results for Mathematical Tests Functions

Function	Proposed-RBF	Proposed-Kriging
Branin	$2.040e{+}00$	$3.814e{+}00$
Hartman 3	$4.568e{+}00$	$5.707e{+}00$
Hartman 6	$5.914e{+}00$	$6.239e{+}00$
Rastrigin-20D	$5.219e{+}00$	$5.470e{+}00$
Rosenbrock-30D	$6.638e{+}00$	$6.623e{+}00$

Reject H_0 at $\alpha = 0.05$ if $U \geqslant 1.644$.
Reject H_0 at $\alpha = 0.01$ if $U \geqslant 2.326$.

Table 4. Mann–Whitney Test Statistics

Branin function and the RBF variant we cannot reject the null hypothesis at the 0.01 significance level, which is attributed to the relative low difficulty of the problem (d = 2) so the difference between the proposed algorithm and the variant is not statistically-significant. For all other tests we reject H_0 at both significance levels α = 0.05 and 0.01 and accept there is a statistically significant difference between the results obtained by the proposed algorithm and by each of the variants for both test functions, i.e. the proposed algorithm outperforms the two variants of the reference algorithm.

4.2 A real-world application

We have also applied the proposed algorithm to a real-world application of airfoil shape optimization. The goal is to find an airfoil shape which maximizes the lift-to-drag ratio (equivalently minimizes the drag-to-lift ratio) at the prescribed cruise conditions [51, 56], i.e.

$$\begin{aligned} \min\ & c_D/c_L \text{ (ratio of drag and lift coefficients)} \\ \text{s.t. } & t^\star = 0.12 \text{ (min. allowed thickness at 0.2–0.8 of chord)} \\ & \alpha = 2^\circ \text{ (cruise angle of attack)} \\ & \mathrm{M} = 0.7 \text{ (cruise Mach number)} \\ & \mathrm{h} = 30,000[\mathrm{ft}] \text{ (cruise altitude)} \end{aligned} \tag{26}$$

where the thickness constraint is based on [55] and the cruise conditions are based on [16, p.484–487] (modified from M = 0.57 , h = 25, 000[ft]) .

Accordingly, to normalize the objectives c_D / c_L and the thickness to the interval [0,1] we defined the objective function

$$f = \frac{c_D}{c_{D,\max}} \cdot \frac{c_{L,\min} + \min(0.1\,,\ -1.1c_{L,\min})}{c_L + \min(0.1\,,\ -1.1c_{L,\min})} + \frac{\max\{t^\star - t\,,\ 0\}}{t^\star}\,, \tag{27}$$

where $c_{L,\min} = -0.5$, $c_{D,\max} = 0.2$ are the assumed minimum c_L and maximal c_D, respectively. For the latter two only reasonable estimates are needed as they are only used to normalize the objectives.

Candidate airfoils were generated using the PARSEC parameterization [50, 76] which involves 11 design variables as shown in Figure 6. Bounds for these design variables were set according to previous studies [24, 56] and are given in Table 5. To ensure a closed airfoil shape the leading edge gap was set as Δz_{TE} = 0. Candidate airfoils were evaluated with XFoil, an analysis code for subsonic isolated airfoils based on the panel method [14]. Each airfoil evaluation required approximately 30 seconds on a desktop computer. We set the limit of function evaluations to $fe_{\max} = 150$.

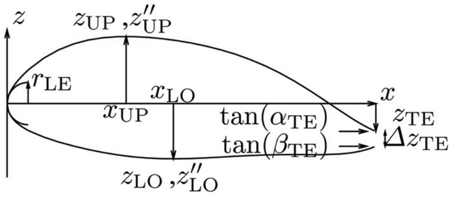

z
$z_{\mathrm{UP}}, z''_{\mathrm{UP}}$
r_{LE}
x_{LO}
x
x_{UP}
$\tan(\alpha_{\mathrm{TE}})$
$\tan(\beta_{\mathrm{TE}})$
z_{TE}
Δz_{TE}
$z_{\mathrm{LO}}, z''_{\mathrm{LO}}$

Fig. 6. PARSEC design variables.

Variable	Meaning	min.	max.
r_{LE}	leading-edge radius	0.002	0.030
x_{up}	max. upper thickness location	0.2	0.7
z_{up}	max. upper thickness	0.08	0.18
z''_{up}	max. upper curvature	-0.6	0.0
x_{lo}	max. lower thickness location	0.2	0.6
z_{lo}	max. upper thickness	-0.09	0.02
z''_{lo}	max. lower curvature	0.2	0.9
z_{TE}	trailing edge height	-0.01	0.01
Δz_{TE}	trailing edge thickness	0	0
α_{TE}[1]	upper trailing edge angle°	165	180
β_{TE}[1,2]	lower trailing edge angle°	165	190

[1] measured anti-clockwise from the x-axis.
[2] $\beta_{\mathrm{TE}} \geqslant \alpha_{\mathrm{TE}}$ to avoid intersecting curves.

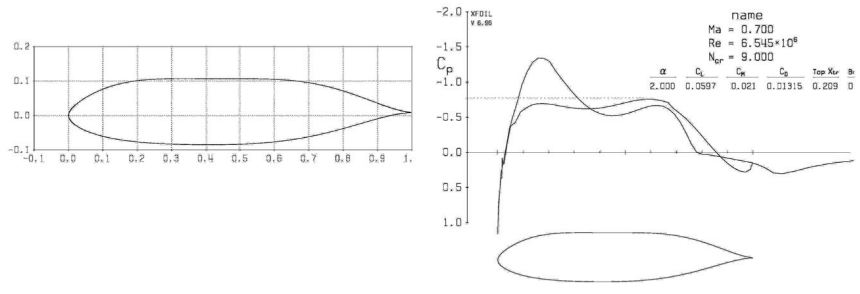
Table 5. PARSEC design variables bounds.

Figure 7 shows an airfoil found by the proposed algorithm and the distribution of the pressure coefficient along its upper and lower surfaces. The airfoil yields a lift to drag ratio of $c_L/c_D = 4.557$ and satisfies the minimum thickness requirement (minimum thickness at 0.2–0.8 of chord is $t = 0.120$).
We benchmarked the proposed algorithm against the two reference algorithms from the previous subsection and test statistics are given in Table 6. A nonparametric analysis similar to that of the previous section gives a Mann–Whitney test statistic of $U = 3.918$ and 4.110 for the RBF and Kriging variants respectively.

Statistic	Proposed	RBF	Kriging
Mean	**$2.982e{-}01$**	$3.145e{-}01$	$3.186e{-}01$
S.D.[1]	**$1.720e{-}02$**	$2.425e{-}02$	$4.068e{-}02$
Median	**$2.940e{-}01$**	$3.068e{-}01$	$3.068e{-}01$
Best	**$2.810e{-}01$**	$2.864e{-}01$	$2.885e{-}01$
Worst	**$3.799e{-}01$**	$3.864e{-}01$	$4.699e{-}01$

[1] standard deviation

Table 6. Benchmarks for the airfoil shape optimization



(c) Airfoil geometry (d) Pressure distribution

Fig. 7. Obtained airfoil.

We thus reject the null hypothesis at both $\alpha = 0.05$ and 0.01 and accept there are statistically-significant differences between the results. This shows that also in this real-world application the proposed algorithm outperformed the surrogate-assisted variants.

5. Summary

We have proposed a surrogate-assisted memetic algorithm for expensive optimization problems. The algorithm combines global and local models and makes extensive use of model selection to assist the optimization search. The global model is an RBF artificial neural network (RBFN) whose topology is adapted incrementally to achieve both a compact network and good generalization. For the local models the proposed algorithm selects between an RBF and a Kriging model based on an accuracy assessment of the models. To ensure convergence to a true optimum of the expensive function these models are used in a trust-region framework, i.e. they replace the quadratic models; the proposed trust-region framework safeguards the accuracy of the local models and improves them, if necessary. Extensive performance analysis shows the proposed algorithm outperforms variants of a reference surrogate-assisted EA.

6. References

N. M. Alexandrov, J. E. Dennis, R. M. Lewis, and V. Torczon. A trust region framework for managing use of approximation models in optimization. *Structural Optimization*, 15(1):16–23, 1998.

J. F. M. Barthelemy and R. T. Haftka. Approximation concepts for optimum structural design—a review. *Structural optimization*, 5:129–144, 1993.

R. E. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton, N.J., 1961.

A. G. Bors and M. Gabbouj. Minimal topology for a radial basis functions neural network for pattern classification. *Digital Signal Processing*, 4:173–188, 1994.

G. E. P. Box and N. R. Draper. *Empirical Model Building and Response Surface*. John Wiley and Sons, New York, NY, 1987.

R. R. Burton. Metamodeling: A state of the art review. In D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, editors, *Proceedings of the 30th Conference on Winter Simulation–WSC 1998*, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

S. Chen, C. Cowan, and P. M. Grant Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309, 1991.

A. Chipperfield, P. Fleming, H. Pohlheim, and C. Fonseca. *Genetic Algorithm TOOLBOX For Use with MATLAB, Version 1.2*. Department of Automatic Control and Systems Engineering, University of Sheffield.

A. R. Conn, K. Scheinberg, and P. L. Toint. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming*, 79:397–414, 1997.

P. S. de Souza and S. N. Talukdar. Genetic algorithms in asynchronous teams. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 392–397, San Mateo, Calif, 1991. Morgan Kaufmann.

J. W. Demmel. The geometry of ill-conditioning. *Computer Journal*, 3(2):201–229, 1987.

J. E. Dennis, Jr and V. Torczon. Managing approximation models in optimization. In N. M. Alexandrov and M. Y. Hussaini, editors, *Multidisciplinary Design Optimization: State of the Art*, pages 330–347. SIAM, Philadelphia, 1997.

L. C. W. Dixon and G. P. Szegö. The global optimization problem: An introduction. In L. C. W. Dixon and G. P. Szegö, editors, *Towards Global Optimisation 2*, pages 1–15. North-Holland Publishing Company, Amsterdam; New York; Oxford, 1978.

M. Drela and H. Youngren. *XFOIL 6.9 User Primer*. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, 2001.

D. Eby, R. C. Averill, W. F. I. Punch, and E. D. Goodman. Evaluation of injection island GA performance on flywheel design optimization. In *Proceedings of the Third Conference on Adaptive Computing in Design and Manufacturing*, pages 121–136, Plymouth, England, 1998. Springer Verlag.

A. Filippone. *Flight Performance of Fixed and Rotary Wing Aircraft*. Elsevier, first edition, 2006.

R. Franke. Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, 38(157):181–200, 1982.

B. Fritzke. Fast learning with incremental RBF networks. *Neural Processing Letters*, 1(1):2–5, 1994.

A. Gaspar-Cunha and A. Vieira. A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. *International Journal of Computers, Systems and Signals*, 6(1):18–36, 2005.

K. C. Giannakoglou. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *International Review Journal Progress in Aerospace Sciences*, 38(1):43–76, 2002.

W. E. Hart and R. K. Belew. Optimization with genetic algorithm hybrids that use local search. In R. K. Belew and M. Mitchell, editors, *Adaptive Individuals in Evolving Populations: Models and Algorithms*, Santa Fe Institute Studies in the Sciences of Complexity, chapter 27, pages 483–496. Addison-Wesley, Reading, MA, 1995.

E. J. Hartman, J. D. Keeler, and J. M. Kowalski. Layered neural networks with Gaussian hidden units as universal approximations. *Neural Computation*, 2:210–215, 1990.

W. H. Highleyman. The design and analysis of pattern recognition experiments. *Bell Systems Technical Journal*, 41:723–744, 1962.

T. L. Holst and T. H. Pulliam. Aerodynamic shape optimization using a real-numberencoded genetic algorithm. Technical Report 2001-2473, AIAA, 2001.

R. Jin, W. Chen, and T. W. Simpson. Comparative studies of metamodeling techniques under multiple modeling criteria. *Structural Optimization*, 23(1):1–13, 2001.

Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Journal of Soft Computing*, 9(1):3–12, 2005.

M. E. Johnson, L. M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26(2):131–148, 1990.

E. J. Kansa and Y.-C. Hon. Circumventing the ill-conditioning problem with multiquadric radial basis functions: Applications to elliptic partial differential equations. *Computers and Mathematics with Applications*, 39(7):123–137, 2000.

M. K. Karakasis and K. C. Giannakoglou. On the use of surrogate evaluation models in multi-objective evolutionary algorithms. In *Proceedings of the European Conference on Computational Methods in Applied Sciences and Engineering–ECCOMAS 2004*, 2004.

M. K. Karakasis and K. C. Giannakoglou. Metamodel-assisted multi-objective evolutionary optimization. In R. Schilling, editor, *Proceedings of the 6th on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems–Eurogen 2005*, 2005.

N. B. Karayiannis and G. W. Mi. Growing radial basis neural networks:Merging supervised and unsupervised learning with network growth techniques. *IEEE Transactions on Neural Networks*, 8(6):1492–1506, 1997.

H.-S. Kim and S.-B. Cho. An efficient genetic algorithm with less fitness evaluation by clustering. In *Proceedings of 2001 IEEE Conference on Evolutionary Computation*, pages 887–894. IEEE, 2001.

K.-H. Liang, X. Yao, and C. Newton. Evolutionary search of approximated N dimensional landscapes. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 4(3):172–183, 2000.

H. Linhart and W. Zucchini. *Model Selection*. Wiley Series in Probability and Mathematical Statistics. Wiley-Interscience Publication, New York; Chichester, 1986.

H. B. Mann and D. R. Whitney. On a test whether one of two variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18:50–60, 1947.

S. D. Marchi. On optimal center locations for radial basis function interpolation: Computational aspects. *Rendiconti del Seminario Matematico*, 61(3):343–358, 2003.

K. Marida and R. Marshall. Maximum likelihood estimation of models for residual covariance in spatial regression. *Biometrika*, 71(1):135–146, 1984.

J. D. Martin and T. W. Simpson. Use of kriging models to approximate deterministic computer models. *AIAA Journal*, 43(4):853–863, 2005.

S. Matej and R. M. Lewitt. Practical considerations for 3-D image reconstruction using spherically symmetric volume elements. *IEEE Transactions on Medical Imaging*, 15(1):68–78, 1996.

C. Matheron. Principles of geostatistics. *Economic Geology*, 58:1246–1266, 1963.

M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

M. Meckesheimer, A. J. Booker, R. R. Burton, and T. W. Simpson. Computationally inexpensive metamodel assessment strategies. *AIAA Journal*, 40(10):2053–2060, 2002.

H. N. Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation*, 8:164–177, 1995.

C. A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.

T. J. Mitchell and M. D. Morris. Bayesian design and analysis of computer experiments: Two examples. *Statistica Sinica*, 2, 359–379 1992.

J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

H. Mühlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm I: Continuous parameter optimization. *Evolutionary Computations*, 1(1):25– 49, 1993.

R. H. Myers and D. C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. John Wiley and Sons, New York, 1995.

M. Niranjan and F. Fallside. Neural networks and radial basis functions in classifying static speech patterns. *Computer Speech and Language*, 4(3):275–289, 1990.

S. Obayashi. Airfoil shape optimization for evolutionary computation. In *Genetic Algorithms for Optimization in Aeronautics and Turbomachinery*, VKI Lecture Series 2000-07. Rhode Saint Genese, Belgium and Von Karman Institute for Fluid Dynamics, 2000.

S. Obayashi, K. Nakahashi, A. Oyama, and N. Yoshino. Design optimization of supersonic wings using evolutionary algorithms. In *Proceedings of the European Conference on Computational Methods in Applied Sciences and Engineering–ECCOMAS 1998*, pages 575–579, 1998.

Y.-S. Ong and A. J. Keane. Meta-Lamarckian learning in memetic algorithm. *IEEE Transactions On Evolutionary Computation*, 8(2):99–110, 2004.

Y.-S. Ong, P. B. Nair, and A. J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *American Institute of Aeronautics and Astronautics Journal*, 41(4):687–696, 2003.

Y.-S. Ong, P. B. Nair, and K. Y. Lum. Max-min surrogate-assisted evolutionary algorithm for robust aerodynamic design. *IEEE Transactions on Evolutionary Computation*, 10(4):392–404, 2006.

A. Oyama, S. Obayashi, and K. Nakahashi. Real-coded adaptive range genetic algorithm and its application to aerodynamic design. *International Journal of the Japan Society of Mechanical Engineering*, 43(2):124–129, 2000.

A. Oyama, S. Obayashi, and T. Nakahashi. Real-coded adaptive range genetic algorithm applied to transonic wing optimization. In M. Schoenauer, editor, *The 6th International Conference on Parallel Problem Solving from Nature–PPSN VI*, pages 712–721, Berlin ; New York, 2002. Springer.

J. Park and I. W. Sandberg. Universal approximation using radial basis function networks. *Neural Computation*, 3:247–257, 1991.

J. Platt. A resource–allocating network for function interpolation. *Neural Computation*, 3(2):213–225, 1991.

T. Poggio and F. Girosi. A theory of networks for approximation and learning. A. I. Memo 1140, Massachusetts Institute of Technology, Artificial Intelligence Laboratory and Center for Biological Information Processing, 1989.

T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

C. Poloni, A. Giurgevich, L. Onseti, and V. Pediroda. Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 186:403–420, 2000.

D. Quagliarella and A. Vicini. Coupling genetic algorithms and gradient based optimization techniques. In D. Quagliarella, J. Périaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms in Engineering and Computer Science*, chapter 14, pages 288–309. John Wiley and Sons, 1997.

A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximations. In A. E. Eiben, Bäck, Thomas, M. Schoenauer, and H. -P. Schwefel, editors, *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature–PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 87– 96, Berlin Heidelberg, 1998. Springer-Verlag.

A. Ratle. Optimal sampling strategies for learning a fitness model. In *The 1999 IEEE Congress on Evolutionary Computation–CEC 1999*, pages 2078–2085, New York, 1999. IEEE.

J.-M. Renderes and H. Bersini. Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways. In A. Sebald and L. J. Fogel, editors, *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 312–317. World Scientific, 1994.

J.-M. Renderes and S. P. Flasse. Hybrid methods using genetic algorithms for global optimization. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 26(2):243–258, 1996.

S. Rippa. An algorithm for selecting a good value for the parameter *c* in radial basis function interpolation. *Advances in Computational Mathematics*, 11(2–3):193–210, 1999.

J. F. Rodríguez, J. E. Renaud, and L. T. Watson. Convergence of trust region augmented Lagrangian methods using variable fidelity approximation data. *Structural Optimization*, 15(3/4):141–156, 1998.

J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.

R. Schaback. Multivariate interpolation and approximation by translates of a basis function. In C. Chui and L. Schumaker, editors, *Approximation Theory VIII*, pages 491–514. Vanderbilt University Press, 1996.

M. Sefrioui and J. Périaux. A hierarchical genetic algorithm using multiple models for optimization. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. M. Guervós, and H.-P. Schwefel, editors, *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature–PPSN VI*, number 1917 in Lecture Notes in Computer Science, pages 879–888. Springer-Verlag, 2000.

T. W. Simpson, A. J. Booker, D. Ghosh, A. Giunta, P. Koch, and R.-J. Yang. Approximation methods in multidisciplinary analysis and optimization: a panel discussion. *Structural and Multidisciplinary Optimization*, 27:302–313, 2004.

T. W. Simpson, D. K. J. Lin, and W. Chen. Sampling strategies for computer experiments: Design and analysis. *International Journal of Reliability and Applications*, 2(3):209–240, 2001.

T.W. Simpson, J. D. Poplinski, P. N. Koch, and J. K. Allen. Metamodels for computer based engineering design: Survey and recommendations. *Engineering with Computers*, 17:129–150, 2001.

R. E. Smith, B. Dike, and S. Stegmann. Fitness inheritance in genetic algorithms. In K. M. George, editor, *Proceedings of the 1995 ACM Symposium on Applied Computing–SAC 95*, pages 345–350. ACM Press, 1995.

H. Sobieckzy. Parametric airfoils and wings. In K. Fujii and G. S. Dulikravich, editors, *Recent Development of Aerodynamic Design Methodologies-Inverse Design and Optimization*, volume 68 of *Notes on Numerical Fluid Mechanics*, pages 71–88. Vieweg Verlag, Germany, 1999.

J. Sobieszczansk-Sobieski and R. Haftka. Multidisciplinary aerospace design optimization: Survey of recent developments. *Structural Optimization*, 14(1):1–23, 1997.

L. N. Søren, H. B. Nielsen, and J. Søndergaard. DACE: A MATLAB Kriging toolbox. Technical Report IMM-TR-2002-12, Informatik and Mathematical Modelling, Technical University of Denmark, 2002.

C. J. Stone. Optimal global rates of convergence for nonparametric regression. *Annals of Statistics*, 10(4):1040–1053, 1982.

M. Stone. Cross-validatory choice and assessment of statistical predictors. *Journal of the Royal Statistical Society Series B (Methodological)*, 36(2):111–147, 1974.

L. Sukhan and R. M. Kil. A Gaussian potential function network with hierarchically self-organizing learning. *Neural Networks*, 4:207–224, 1991.

Y. Tenne. Metamodel accuracy assessment in evolutionary optimization. In *Proceedings of the IEEE World Congress on Computational Intelligence–WCCI 2008*. IEEE, 2008. 1505-1512, 2008. IEEE.

Y. Tenne and S. W. Armfield. A memetic algorithm using a trust-region derivative-free optimization with quadratic modelling for optimization of expensive and noisy blackbox functions. In S. Yang, Y.-S. Ong, and Y. Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence*, pages 389–415. Springer-Verlag, 2007.

Y. Tenne and S. W. Armfield. A versatile surrogate-assisted memetic algorithm for optimization of computationally expensive functions and its engineering applications. In A. Yang, Y. Shan, and L. Thu Bui, editors, *Success in Evolutionary Computation*, volume 92/2008, pages 43–72. Springer, Berlin; Heidelberg, 2008.

Y. Tenne, S. Obayashi, and S. W. Armfield. Airfoil shape optimization using an algorithm for minimization of expensive and discontinuous black-box functions. In *Proceedings of the AIAA InfoTec 2007*, number AIAA-2007-2874. American Institute for Aeronautics and Astronautics (AIAA), 2007.

P. L. Toint. Some numerical results using a sparse matrix updating formula in unconstrained optimization. *Mathematics of Computation*, 32(143):839–851, 1978.

V. Torczon and M. W. Trosset. Using approximations to accelerate engineering design optimization. In *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pages 738–748, Reston, Va, 1998. American Institute for Aeronautics and Astronautics (AIAA).

A. Törn. Global optimization as a combination of global and local search. In *Proceedings of Computer Simulation Versus Analytical Solutions for Business and Economic Models*, number 17 in BAS Business Administration Studies, pages 191–206. School of Business Administration, Gothenburg, 1973.

A. Törn, M. M. Ali, and S. Viitanen. Stochastic global optimization: Problems classes and solution techniques. *Journal of Global Optimization*, 14:437–447, 1999.

A. Törn and A. Žilinskas. *Global Optimization*. Number 350 in Lecture Notes In Computer Science. Springer-Verlag, Berlin; Heidelberg; New York; London, 1989.

J. Yen, J. C. Liao, B. Lee, and D. Randolph. A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 28(2):173–191, 1998.

Y. Yun, M. Gen, and S. Seo. Various hybrid methods based on genetic algorithm with fuzzy logic controller. *Journal of Intelligent Manufacturing*, 14:401–419, 2003.

Z. Zhou, Y.-S. Ong, M. Lim, and B. Lee. Memetic algorithms using multi-surrogates for computationally expensive optimization problems. *Journal of Soft Computing*, 11(10):957–971, 2007.

Z. Zhou, Y.-S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum. Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Transactions On Systems, Man and Cybernetics-Part C*, 37(1):66–76, 2007.

An Adaptive Evolutionary Algorithm Combining Evolution Strategy and Genetic Algorithm (Application of Fuzzy Power System Stabilizer)

Gi-Hyun Hwang and Won-Tae Jang
Dept. of Computer Information Engineering, Dongseo University, Pusan 609-735, South Korea

1. Introduction

The research of power system stabilizer (PSS) for improving the stability of power system has been conducted from the late 1960's. Conventionally lead-lag controller has been widely used as PSS. Root locus and Bode plot to determine the coefficient of lead-lag controller (Yu, 1983; Larsen and Swann, 1981; Kanniah et al., 1984), pole-placement and eigenvalue control (Chow & Sanchez-Gasca, 1989; Ostojic & Kovacevic, 1990) and a linear optimal controller theory (Fleming & Jun Sun, 1990; Mao et al., 1990) have been used. These methods, using a model linearlized in the specific operating point, show a good control performance in the specific operating point. But these approaches are difficult to obtain a good control performance in case of operating conditions such as change of load or three phase fault, etc. Therefore, several methods based on adaptive control theory (Chen et al., 1993; Park & Kim, 1996) have been proposed to give an adaptive capability to PSS for nonlinear characteristic of power system. These methods can improve the dynamic characteristic of power system, but these approaches cannot be applied for the real time control because of long execution time.
Recently the research for intelligence control method such as fuzzy logic controller (FLC) and neural network for PSS has greatly improved the dynamic characteristic of power system (Hassan et al., 1991; Hassan & Malik, 1993). Fuzzy rules and membership functions shape should be adjusted to obtain the best control performance in FLC. Conventionally the adjustment is done by the experience of experts or trial and error methods. Therefore it is difficult to determine the suitable membership functions without the knowledge of the system. Recently, evolutionary computations (EC) that is a kind of a probabilistic optimal algorithm is employed to adjust the membership functions and fuzzy rules of FLC.
The EC is based on the natural genetics and evolutionary theory. The results of this approach show a good performance (Abido and Abdel-Magid, 1998, 1999).
EC is based on the principles of genetics and natural selection. There are three broadly similar avenues of investigation in EC: genetic algorithm (GA), evolution strategy (ES), and evolutionary programming (EP) (] Fogel, 1995). GA simulates the crossover and mutation of natural systems, having a global search capability (Goldberg, 1989), whereas ES simulates the evolution of an asexually reproducing organism. ES can find a global minimum, and by combining another EC it also could be efficient local search technique (Gong et al., 1996).

The performance of EC is influenced by parameters such as size of population, fitness, probability of crossover, and mutation, etc. If these parameters are not adequately selected, execution time will be longer and premature convergence to local minimum can occur. To solve problems above, several approaches have been proposed. To enhance the performance of GA, the population size, the probability of crossover, mutation and operation method should be adaptively modified in each generation (Arabas et al., 1994; Schlierkamp-Voosen & Muhlenbein, 1996). To enhance the performances of ES and EP, the mutation parameters should be adapted while running ES and EP (Goldberg, 1989; Fogel et al., 1991).
In conventional ES, parameter values and operator probabilities for the GA and ES are adapted to find a solution efficiently. In this paper, however, we propose adaptive evolutionary algorithm (AEA). The ratio of population to which GA and ES will apply is adaptively modified in reproducing according to the fitness. We use ES to optimize locally, while the GA optimizes globally. The resulting hybrid scheme produces improved and reliable results by using the "global" nature of the GA as well as the "local" improvement capability of the ES.
AEA was applied to search the optimal parameters of the membership functions and the suitable gains of the inputs and outputs for fuzzy power system stabilizer (FPSS). The effectiveness of FPSS is demonstrated by computer simulation for single-machine infinite bus system (SIBS) and multi-machine power system (MPS). To show the superiority of FPSS, its performances are compared with those of conventional power system stabilizer (CPSS). The proposed FPSS shows the better control performances than the CPSS in three-phase fault under a heavy load, which is system condition in tuning FPSS. To show the robustness of the proposed FPSS, it is applied to the system with disturbances such as change of the mechanical torque and three-phase fault under nominal and heavy load conditions.

2. Adaptive evolutionary algorithm

2.1 Motivation

GA, one of the probabilistic optimization methods, is robust and is able to solve complex and global optimization problem. But GA can suffer from the long computation time before providing an accurate solution because it uses prior knowledge minimally and does not exploit local information (Renders & Flasse, 1996). ES, which simulates the evolution of asexually reproducing organisms, has efficient local search capability. To solve complex problem, however, it better to a hybrid EC (Gong et al., 1996).
In this paper, to reach the global optimum accurately and reliably in a short execution time, we designed an AEA by using GA and ES together. In AEA, GA operators and ES operators are applied simultaneously to the individuals of the present generation to create the next generation. Individual with higher fitness value has the higher probability of contributing one or more chromosomes to the next generation. This mechanism gives greater rewards to either GA or ES operation depending on what produces superior offspring.

2.2 Adaptive evolutionary algorithm

In AEA, the number of individuals created by GA and ES operations is changed adaptively. An individual is represented as a real-valued chromosome that makes it possible to hybridize GA and ES operations.
ES forms a class of optimization technique motivated by the reproduction of biological system and the population of individuals evolves toward the better solutions by means of

the mutation and selection operation. In this paper, we adopted a (μ, λ)-ES. That is, only the λ offspring generated by mutation competes for survival and the μ parents are completely replaced in each generation. Also, self-adaptive mutation step sizes are used in ES.

For AEA to self-adapt its use of GA and ES operators, each individual has an operator code for determining which operator to use. Suppose a '0' refers to GA, and a '1' to ES. At each generation, if it is more beneficial to use the GA, '0's should appear at the end of individuals. If it is more beneficial to use the ES, '1's should appear. After reproduction by roulette wheel selection according to the fitness, GA operations (crossover and mutation) are performed on the individuals that have the operator code of '0' and the ES operation (mutation) is performed on the individuals that have an operator code of '1'. Elitism is also used. The best individual in the population reproduces both the GA population and ES population in the next generation. The major procedures of AEA are as follows:

1) Initialization: The initial population is randomly generated. Operator code is randomly initialized for each individual. According to the operator code, GA operations are performed on the individuals with operator code '0', while ES operations are applied where the operator code is '1'.
2) Evaluation and Reproduction: Using the selection operator, individual chromosomes are selected in proportional to their fitness, which is evaluated by the defined objective function. After reproduction, GA operations are performed on the individuals having an operator code of '0' and the ES operations are performed on the individuals having an operator code '1'. At each generation, the percentages of '1's and '0's in the operator code indicate the performance of GA and ES operators.
3) Preservation of Minimum Number of Individuals: At each generation, AEA may fall into a situation where the percentage of the offspring by one operation is nearly 100% and the offspring by other operation dies off. Therefore, it is necessary for AEA to preserve certain amount of individuals for each EC operation. In this paper, we randomly changed the operator code of the individuals with a higher percentage until the numbers of individuals for each EC operation become higher than a certain amount of individuals to be preserved. The predetermined minimum number of individuals to be preserved is set to 20% of the population size.
4) Genetic Algorithm and Evolution Strategy: The real-valued coding is used to represent a solution (Michalewicz, 1992; Mitsuo Gen and Cheng, 1997). Modified simple crossover and uniform mutation are used as genetic operators. The modified simple crossover operator is a way to generate offstrings population, selecting two strings randomly in parent population, as shown in Fig. 1. If crossover occurs in *k*-th variable, selecting randomly two strings in *t*-th generation, offstrings of *t+1*-th generation are shown in Fig. 1.

In uniform mutation, we selected a random *k*-th gene in an individual. If an individual and the *k*-th component of the individual is the selected gene, the resulting individual is as shown in Fig. 2.

Only the λ offspring generated by mutation operation competes for survival and the μ parents are completely replaced in each generation. Mutation is then performed independently on each vector element by adding a normally distributed Gaussian random variable with mean zero and standard deviation (σ), as shown in Eq. (1). After adapting the mutation operator for ES population, if the improved ratio of individual number is lesser than δ, standard deviation for the next generation is decreased in proportion to decreased rates of standard deviation (c_d), Otherwise, standard deviation of the next generation is increased in proportion to increased rates of standard deviation (c_i,), as shown in Eq. (2) (Fogel, 1995).

< Before Crossover > < After Crossover >

$$\begin{array}{lcl} S_v^t = [V_1, \ \dots, \ V_k, \ \dots, \ V_n] & & S_v^{t+1} = [V_1, \ \dots, \ V_k', \ V_{k+1}' \dots, \ V_n'] \\ & \longrightarrow & \\ S_w^t = [W_1, \dots, \underset{\uparrow}{W_k}, \dots, W_n] & & S_w^{t+1} = [W_1, \ \dots, \ W_k', \ W_{k+1}' \dots, \ W_n'] \end{array}$$

Crossover point

where, $V_j' = a_1 \ V_j \ + \ a_2 \ W_j$

$W_j' = a_1 \ W_j \ + \ a_2 \ V_j$

a_1, a_2 : Random numbers from [0, 1]

V_j : j-th gene of the vector S_v

W_j : j-th gene of the vector S_w

n : Number of parameters

Fig. 1. Modified simple crossover method

< Before Mutation > < After Mutation >

$$S_v^t = [V_1, \ \dots, \ \underset{\uparrow}{V_k}, \ \dots, \ V_n] \longrightarrow S_v^{t+1} = [V_1, \ \dots, \ V_k', \ V_{k+1} \dots, \ V_n]$$

Mutation point

where, V_k' : Random value between upper bound and lower bound

Fig. 2. Uniform mutation method

$$v_k^{t+1} = v_k^t + N(0, \sigma^t) \tag{1}$$

$$\sigma^{t+1} = \begin{cases} c_d \times \sigma^t, & if \quad \varphi(t) < \delta \\ c_i \times \sigma^t, & if \quad \varphi(t) > \delta \\ \sigma^t, & if \quad \varphi(t) = \delta \end{cases} \tag{2}$$

where, $N(0,\sigma^t)$: Vector of independent Gaussian random variable with mean of zero and standard deviations σ

V_k^t: k-th variable at t-th generation

$\phi(t)$: Improved ratio of individual number after adapting mutation operator for population of ES in t-th generation

δ: Constants

5) Elitism: The best individual in a population is preserved to perform GA and ES operation in the next generation. This mechanism not only forces GA not to deteriorate temporarily, but also forces ES to exploit information to guide subsequent local search in the most promising subspace.

3. Design of fuzzy power system stabilizer using AEA

Conventionally, we have used the knowledge of experts and trial and error methods to tune FLC's for a good control performance, but recently many other ways using EC are proposed to modify fuzzy rule and shape of fuzzy membership function (Abido and Abdel-Magid, 1998, 1999). Scaling factors of input/output and parameters of membership function of FPSS are optimized by means of AEA using GA and ES adaptively, as described in chapter 2.
Fig. 3 shows the architecture for tuning scaling factors of input/output and membership function shape of FPSS using AEA. As shown in Fig. 3, the rotor speed deviation of generator and the change rate for rotor speed deviation are used as inputs of FPSS. The control signals of the FPSS are used for enhancing power system damping by supplementary control signals of generators.

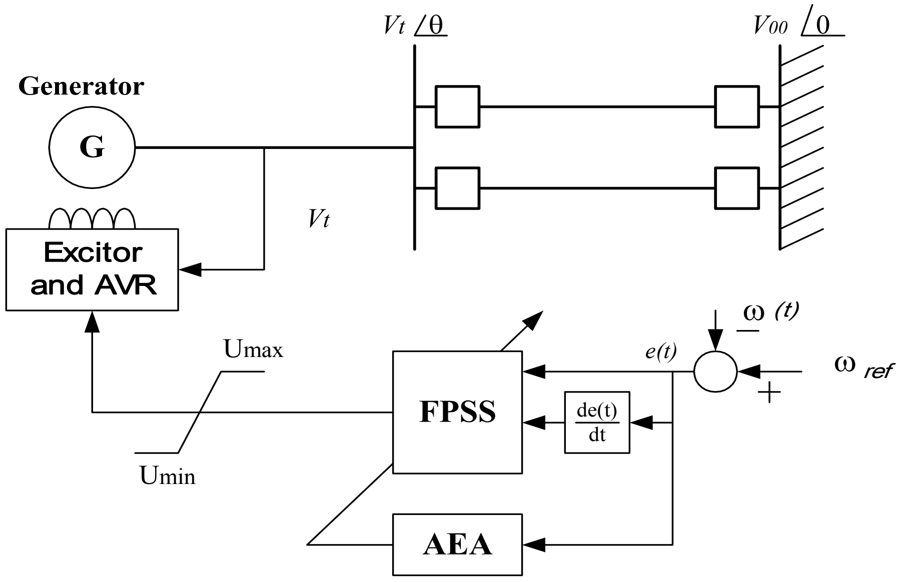


Fig. 3. Configuration for tuning of FPSS using AEA.

The FPSS parameters used in this paper are given below.
- Number of input/output variables : 2/1
- Number of input/output membership functions : 7/7
- Fuzzy inference method : max-min method
- Defuzzification method : center of gravity

Because deviation and change-of-deviation are used as input variables of the FPSS, proportional-derivative (PD)-like FPSS is used. Rule base for the PD-like FPSS from the two-dimensional phase plane of the system in terms of deviation (e) and change-of-deviation (de) is shown in Table 1. As shown in Table 1, the phase plane is divided into two semi-planes by means of switching-line. Within the semi-planes, positive and negative control signals are produced, respectively. The magnitude of the control signals depends on the distance of the state vector from the switching line.

When AEA is tuning the membership functions, fuzzy rules are used for PD-type, as shown in Table 1, where, linguistic variable NB means "Negative Big", NM means "Negative Medium", NS means "Negative Small", etc. Fig. 4 shows triangular membership function used in this paper. Because we use 7 fuzzy variables (PB, PM, ··· ,NM, NB) respectively, for input/output of FPSS, the total membership functions will be 21, so 63 variables that include the center and width of all the membership function will be adjusted, but it takes a long calculation time to tune 63 variables using AEA, and suffers from undesirable converging characteristic. In this paper, we fixed center of ZE to 0 and positive and negative membership functions are constructed symmetrical for the 0. So the number of parameters of FPSS will be reduced to 21, which means 3 centers and 4 widths for each variable as shown in Fig. 4.

e \ de	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NM	NM	NS	ZE
NM	NB	NB	NM	NM	NS	ZE	PS
NS	NB	NM	NM	NS	ZE	PS	PM
ZE	NM	NM	NS	ZE	PS	PM	PM
PS	NM	NS	ZE	PS	PM	PM	PB
PM	NS	ZE	PS	PM	PM	PB	PB
PB	ZE	PS	PM	PM	PB	PB	PB

Table 1. Fuzzy rules of proportional-differential type

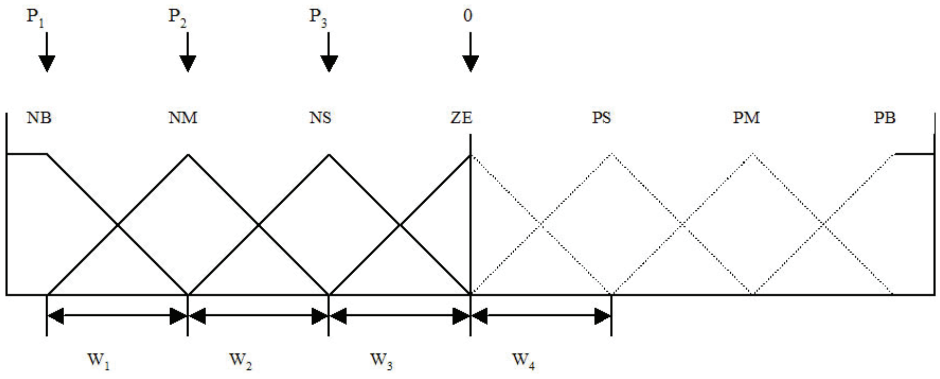
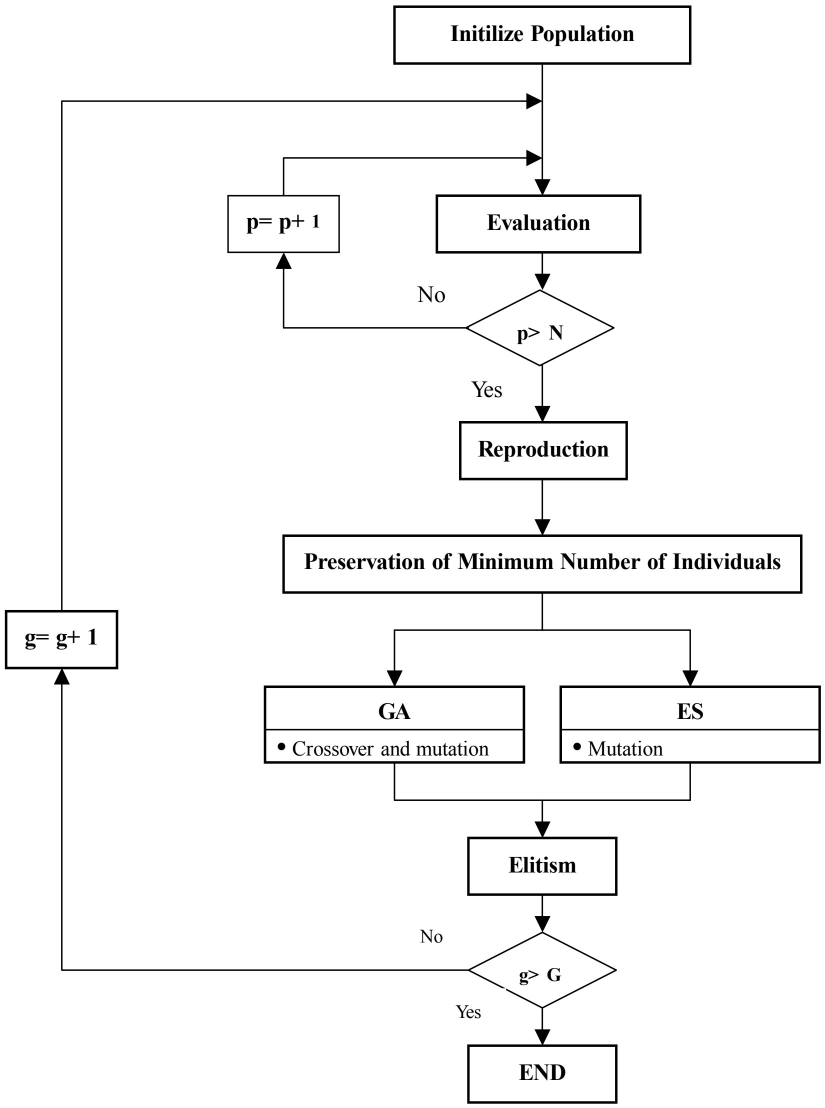


Fig. 4. Symmetrical membership functions

The flowchart for the design of FPSS using the proposed AEA is shown in Fig. 5. The procedure for the design of FPSS using AEA is as follows:



where, P : Number of population
G : Specified generation

Fig. 5. Flowchart for the design of FPSS using AEA

Step1) Initialize population

Strings are randomly generated between upper bounds and lower bounds of the membership function parameters and scaling factors of FPSS. The operator code is randomly set to decide if each string is individual of GA or ES. The configuration of population is described in Fig. 6. Also scaling factors of the FPSS are tuned by the AEA.

S_1	P_{11}	• • •	P_{19}	W_{11}	• • •	W_{112}	SF_{11}	SF_{12}	SF_{13}	*
S_2	P_{21}	• • •	P_{29}	W_{21}	• • •	W_{212}	SF_{21}	SF_{22}	SF_{23}	*
						•				
						•				
S_n	P_{n1}	• • •	P_{n9}	W_{n1}	• • •	W_{n12}	SF_{n1}	SF_{n2}	SF_{n3}	*

where, n : population size
P_{ij} : Center of the membership functions
W_{ij} : Width of the membership functions
SF_{ij} : Scaling factors
* : Operator code

Fig. 6. String architecture for tuning membership functions and scaling factors.

Step 2) Evaluation

Each string generated in Step 1 is evaluated using the fitness function in Eq. (3). As shown in Eq. (3), the absolute deviation between the rotor speed and the reference rotor speed of generator is used. The flowchart for evaluation part is shown in Fig. 7.

$$Fitness = \frac{1}{1 + \int_{t=0}^{T} | \omega_{ref} - \omega_{(t)} |} \tag{3}$$

where, ω_{ref} : Reference rotor speed of generator
$\omega(t)$: Rotor speed of generator
T : No. of data acquired during specified time

Step 3) Reproduction

We used roulette wheel to reproduce in proportion to fitness. After reproduction, the individual operator code of '0' is inserted in the population of GA, the individual operator code of '1' is inserted in the population of ES.

Step 4) Preservation of Minimum Number of Individuals

Among GA and ES, depending on which is stronger, we guarantee minimum number of individuals to offsprings being disappearing by the remaining iterations.

Step 5) GA and ES operation

The individual with operator code of '0' applied crossover and mutation in GA operators and generates offsprings. The individual with operator code of '1' apply mutation in ES operator and generates offsprings.

Step 6) Elitism

We use elitism reproducing the best individual of fitness to GA and ES population by each one.

Step 7) Convergence criterion

We iterate Step 2 – Step 6 until being satisfied of the specified generation.

4. Simulation studies

4.1 Simulation cases of single-machine infinite bus system

We performed nonlinear simulation for SIBS in Fig. 8 to demonstrate the performance of the proposed FPSS. A machine has been represented by third order one-axis nonlinear model, as

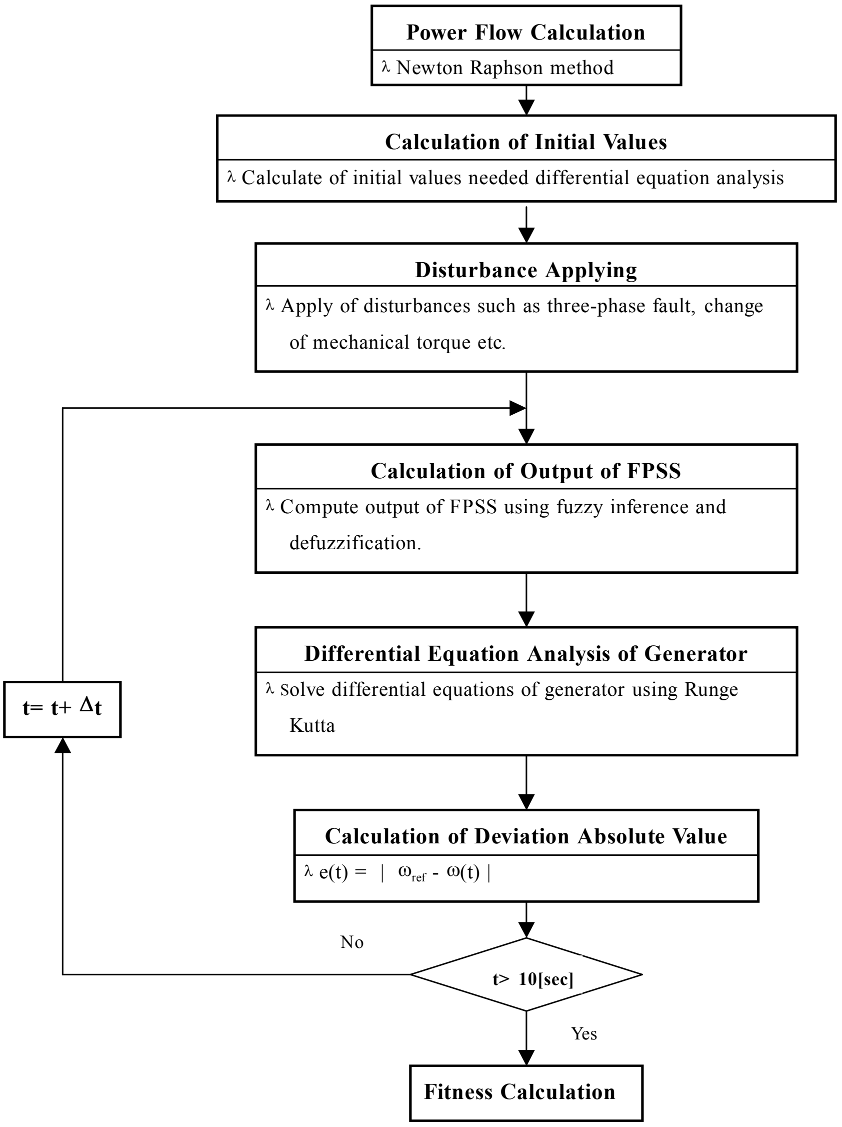

Power Flow Calculation
λ Newton Raphson method
Calculation of Initial Values
λ Calculate of initial values needed differential equation analysis
Disturbance Applying
λ Apply of disturbances such as three-phase fault, change of mechanical torque etc.
Calculation of Output of FPSS
λ Compute output of FPSS using fuzzy inference and defuzzification.
Differential Equation Analysis of Generator
λ Solve differential equations of generator using Runge Kutta
Calculation of Deviation Absolute Value
λ e(t) = | ω_ref - ω(t) |
t= t+ Δt
No
t> 10[sec]
Yes
Fitness Calculation

Fig. 7 Flowchart for evaluation part

shown in appendix. Details of the system data are given in Yu, 1983. Table 2 shows the simulation coefficients of AEA used in nonlinear simulation. The execution time in PC 586 (300 MHz) takes about 30 minutes to tune the parameters of FPSS under the condition in Table 2. Fig. 9 shows membership functions shape of FPSS tuned by AEA, where scaling constant of deviation is 0.24, scaling constant of deviation rate is 3.50 and scaling constant of output part is 2.75. We reviewed the performance of FPSS proposed in this paper and compared it with CPSS (Yu, 1983). In CPSS, time constants (T_1, T_2) were designed based on phase compensation as in Eq. (4), where washout filter (T_w) is 3 sec, stabilization gain (K_{pss}) is 7.09, and T_1, T_2 are 0.1 sec, 0.065 sec respectively.

$$V_S = \frac{sT_w}{1+sT_w} K_{pss} \left(\frac{1+sT_1}{1+sT_2} \right) \quad (4)$$

where, Vs : Output of PSS

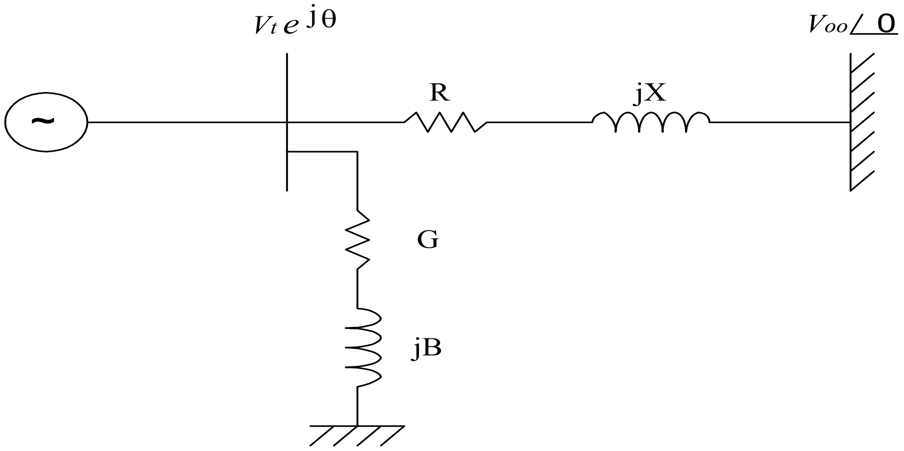
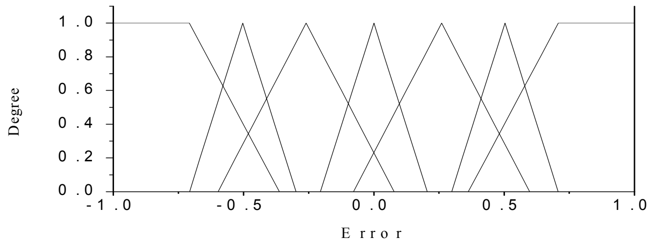


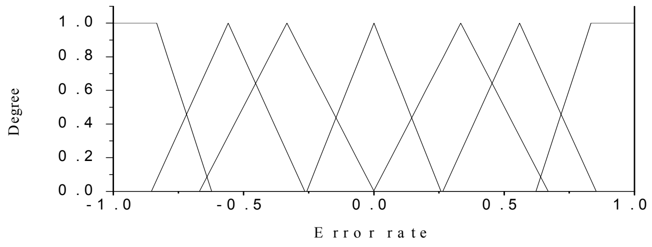
Fig. 8. Single-machine infinite system used in performance evaluation

Methods	AEA	
	SIBS	MPS
Size of population	50	100
Crossover probability	0.95	0.95
Mutation probability	0.005	0.005
δ	0.5	0.5
C_d	0.95	0.95
C_I	1.05	1.05
Number of Generation	100	200

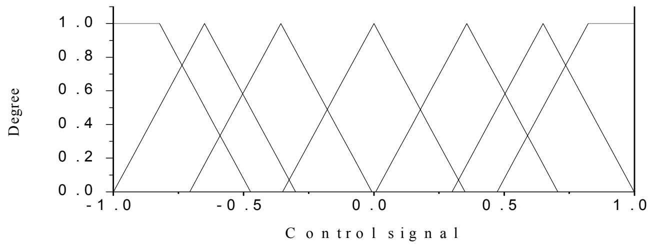
Table 2. Coefficients for simulation using AEA



(a) Membership function of deviation



(b) Membership function of change-of-deviation



(c) Membership function of output part

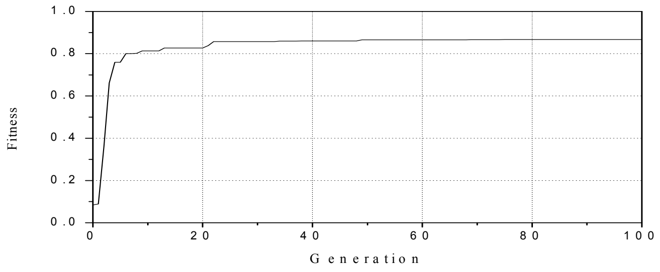
Fig. 9. Tuned membership function of FPSS

Fig. 10 (a) shows the fitness values by AEA in each generation. Fig. 10 (b) shows the number of individuals for GA and ES in the AEA. As shown in Fig. 10, the number of individuals of GA is higher than that of individuals of ES in early generation. But, from generation to generation, the number of individuals of ES goes higher than that of individuals of GA. The AEA produces the improved reliability by exploiting the "global" nature of the GA initially as well as the "local" improvement capabilities of the ES from generation to generation.

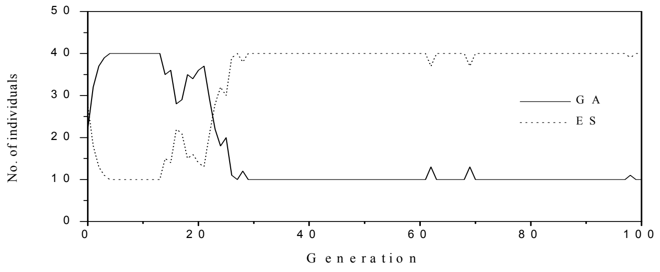
Analysis conditions used for comparing control performance of CPSS with FPSS optimized by AEA are summarized in Table 3. Table 3 is classified into four cases according to the power system simulation cases used in designing FPSS and in evaluating the robustness of FPSS. As shown in Table 3, Case-1 is used to design FPSS and tune scaling constant of input/output variable and membership functions of FPSS by AEA. We used Case-2 and Case-4 in evaluating the robustness of FPSS.

1) Heavy load condition

Fig. 11 shows generator angular velocity and the phase angle both without PSS and with CPSS and FPSS under Case-1 in Table 3. As shown Fig. 11, the FPSS shows the better control performance than CPSS in terms of settling time and damping effect. To evaluate the robustness of FPSS, Fig. 12 shows generator response characteristic in case that PSS is not applied. In this case, CPSS and proposed FPSS are applied under Case-2 of Table 3. As shown in Fig. 12, FPSS shows the better control performance than CPSS in terms of settling time and damping effect.



(a) fitness



(b) Number of individuals of GA and ES in AEA

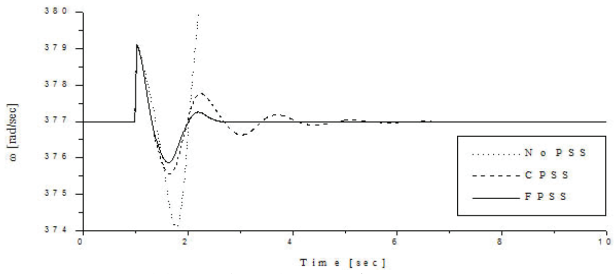
Fig. 10. Fitness and number of individuals of GA and ES in each generation

Simulation cases	Operating conditions	Disturbance	Fault time [msec]
Case-1	Heavy load $P_e = 1.3$ [pu] $Q_e = 0.015$ [pu]	A	40
Case-2		B	-
Case-3	Nominal load $P_e = 1.0$ [pu] $Q_e = 0.015$ [pu]	A	40
Case-4		B	-

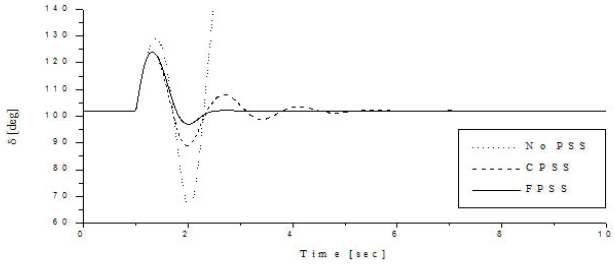
A: Three phase fault

B: Mechanical torque was changed as 0.1 [pu]

Table 3. Simulation cases used in evaluation of controller performance

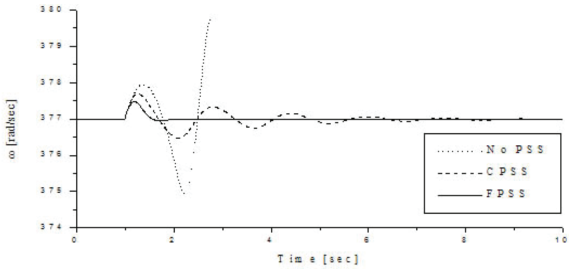


(a) Angle velocity of generator

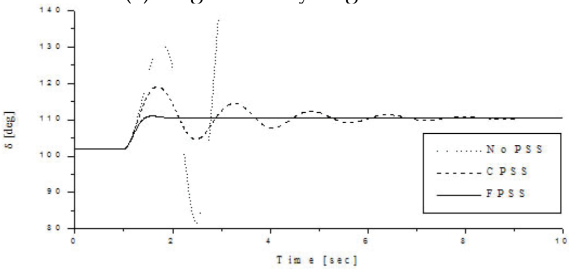


(b) Angle of generator

Fig. 11. Responses of generator when three-phase fault was occurred in heavy load



(a) Angle velocity of generator



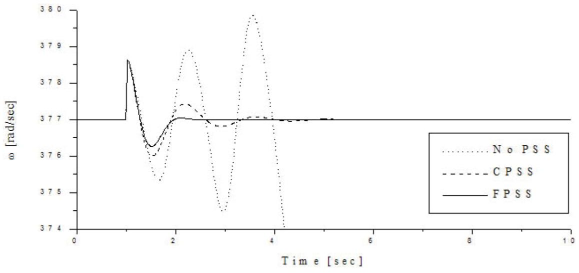
(b) Angle of generator

Fig. 12. Responses of generator when mechanical torque was changed into 0.1[pu] in heavy load

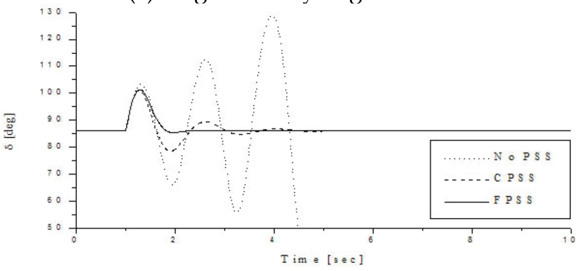
2) Nominal load condition

To evaluate the robustness of FPSS, Fig. 13-14 show generator response characteristic in case that PSS is not applied, and CPSS and proposed FPSS are applied under Case-3 and 4 of

Table 3. As shown in Fig. 13-14, the FPSS shows the better control performance than CPSS in terms of settling time and damping effect.

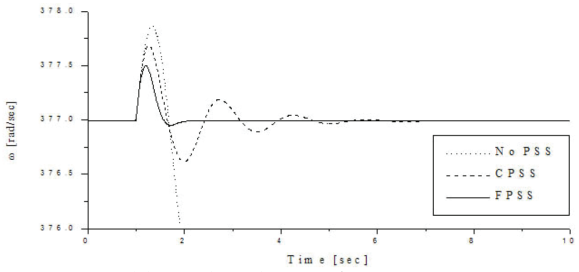


(a) Angle velocity of generator

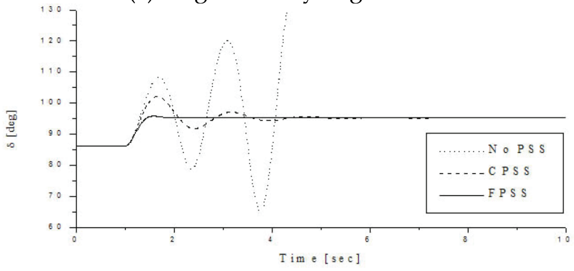


(b) Angle of generator

Fig. 13. Responses of generator when three-phase fault was occurred in nominal load



(a) Angle velocity of generator



(b) Angle of generator

Fig. 14. Responses of generator when mechanical torque was changed into 0.1[pu] in nominal load

3) Dynamic stability margin

To evaluate the dynamic stability margin (He & Malik, 1997) of CPSS and FPSS, a simulation study is conducted with the initial operating condition of light, nominal and heavy load as given in Table 3. The mechanical torque is increased gradually. The dynamic stability margin is described by the maximum active power in which the system losses synchronism. Table 4 shows the dynamic stability margin. In Table 4, we can find FPSS increases the dynamic stability of generator.

Conditions \ Methods		CPSS	FPSS
Light load	Maximum active power [pu]	1.02	1.06
	Maximum generator phase angle [rad]	2.44	2.46
Nominal load	Maximum active power [pu]	1.22	1.27
	Maximum generator phase angle [rad]	2.35	2.45

Table 4. Dynamic stability margin (SIBS)

4.2 Simulation cases of multi-machine power system

To demonstrate the performance of the proposed FPSS, we performed nonlinear simulation for WSCC 3-machine, 9-bus system (Anderson & Found, 1977) as in Fig. 15. Constants of generator and exciter, load admittance, and load condition used in generator dynamic characteristic analysis are shown in Appendix (Abido & Abdel-Magid, 1999). Coefficients for simulation of AEA are shown in Table 2. We compared the proposed FPSS with the conventional power system stabilizer, CPSS, for multi-machine power system. In CPSS, time constants (T_1, T_2) were designed based on phase compensation as in Eq. (5), where washout filter (T_w) is 1.5 sec, stabilization gain (K_{pss}) is 15, and T_1, T_2 are 0.29 sec, 0.029 sec respectively.

$$V_S = \frac{sT_w}{1+sT_w} K_{pss} \left(\frac{1+sT_1}{1+sT_2} \right)^2 \tag{5}$$

As shown in Table 5, simulation cases used in comparing control performance of FPSS with CPSS are classified into Case-1 to Case-4. Case-1 was for the power operating condition used in designing FPSS. Case-2 and Case-4 were for evaluating the robustness of FPSS

Simulation cases	Operating conditions	Disturbanc e	Fault time [msec]
Case-1	Heavy load	A	70
Case-2		B	70
Case-3	Nominal load	A	70
Case-4		B	70

A: Three phase fault in bus-7
B: Three phase fault between bus-5 and bus-7
Table 5. Simulation cases used in evaluation of controller performance

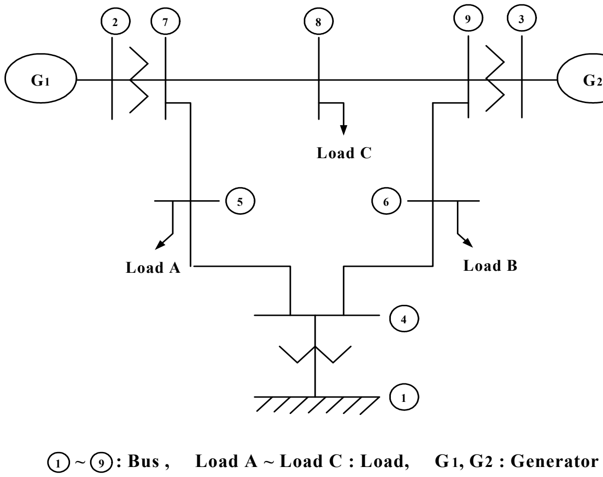
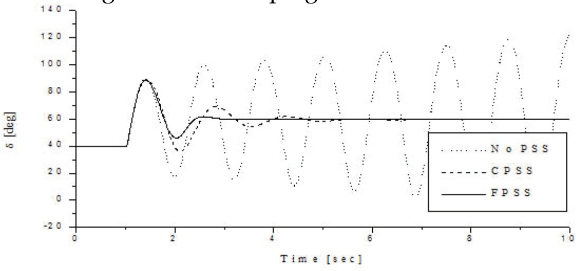


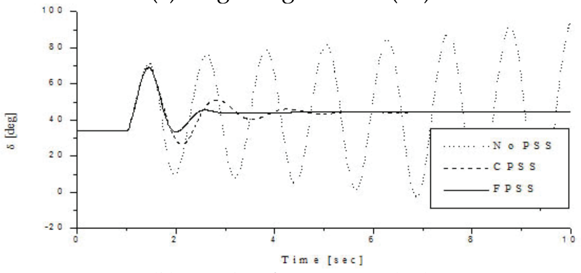
Fig. 15. WSCC 3-machine, 9-bus system

1) Heavy load condition

Fig. 16 shows generator phase angles (G_1, G_2) both without PSS and with CPSS and FPSS under Case-1 in Table 5. As shown Fig. 16, the FPSS shows the better control performance than CPSS in terms of settling time and damping effect. To evaluate the robustness of FPSS, Fig. 17 shows generator phase angles (G_1, G_2) both without PSS and with CPSS and FPSS under Case-2 in Table 5. As shown in Fig. 17, FPSS shows the better control performance than CPSS in terms of settling time and damping effect.

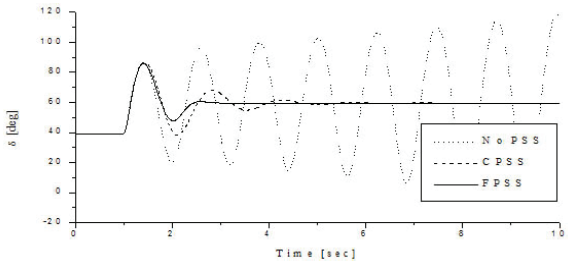


(a) Angle of generator (G_1)

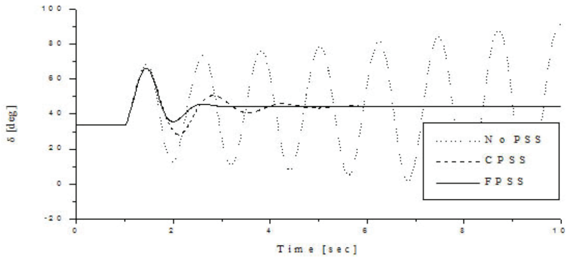


(b) Angle of generator (G_2)

Fig. 16. Responses of generator when three-phase ground fault was occurred at bus-7 under heavy load condition



(a) Angle of generator (G_1)

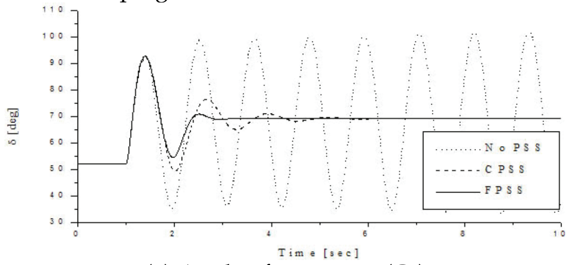


(b) Angle of generator (G_2)

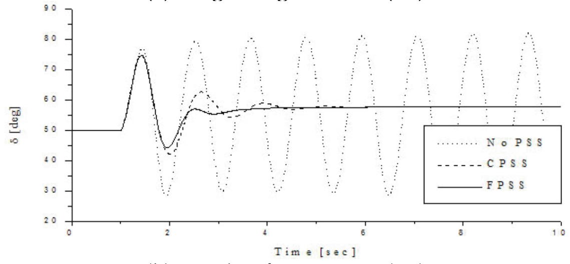
Fig. 17. Responses of generator when three-phase ground fault was occurred at bus-5 and bus-7 under heavy load condition

2) Nominal load condition

To evaluate the robustness of FPSS, Fig. 18-19 shows generator response characteristic in case that PSS is not applied, and CPSS and proposed FPSS are applied under Case-3 and 4 in Table 3. As shown in Fig. 18-19, the FPSS shows the better control performance than CPSS in terms of settling time and damping effect.

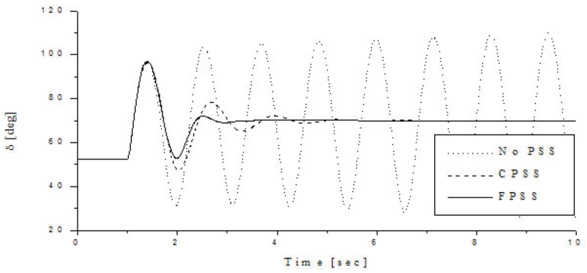


(a) Angle of generator (G_1)

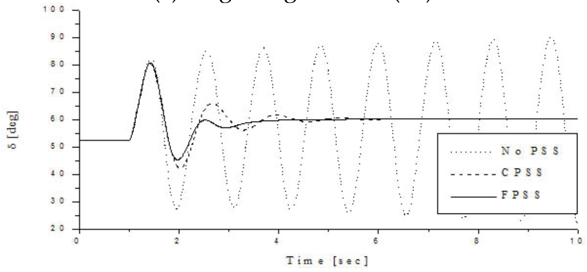


(b) Angle of generator (G_2)

Fig. 18. Responses of generator when three-phase ground fault was occurred at bus-7 under nominal load condition



(a) Angle of generator (G_1)



(b) Angle of generator (G_2)

Fig. 19. Responses of generator when three-phase ground fault was occurred at bus-5 and bus-7 under nominal load condition

3) Dynamic stability margin

Table 6 shows the dynamic stability margin (He and Malik, 1997) of CPSS and FPSS when the mechanical torque was increased gradually. In Table 6, we can find FPSS increases the dynamic stability of generator.

Methods / Conditions		CPSS		FPSS	
		G_1	G_2	G_1	G_2
Heavy load	A	3.04	2.44	3.11	2.51
	B	2.25	1.39	2.25	1.46
Nominal load	A	2.84	2.29	2.91	2.36
	B	2.52	1.58	2.52	1.63

A : Maximum active power [pu]
B : Maximum generator phase angle [rad]

Table 6. Dynamic stability margin (MPS)

5. Conclusions

In this paper, we tuned membership functions shape and input/output gain of FPSS using AEA that is algorithm that ratio of population to which GA and ES will adapt is adaptively modified in reproduction according to the fitness. In the SIBS and MPS, we analyzed simulation results of FPSS and CPSS. The results are as following:

① As a result of applying AEA to the design of FPSS, in the early generation, it is shown the number of population of GA is higher than that of population of ES, also the number of population of ES grows as the number of generation increases. This shows that the global search is executed through GA in the early generation and the local search is executed adaptively by means of ES as the number of generation increases.

② FPSS showed the better control performance than CPSS in terms of settling time and damping effect when three- phase fault under heavy load that is used in tuning FPSS occurs. To evaluate the robustness of FPSS, we analyzed dynamic characteristic of generator for changeable mechanical torque in heavy load, and change of mechanical torque and three-phase fault in nominal. FPSS showed the better damping effect than CPSS.

③ As result of finding dynamic stability margin and successive peak damping ratio, FPSS more increased dynamic stability margin and showed the better result than CPSS in terms of successive peak damping ratio.

6. Acknowledgments

This research was supported by the Program for the Training of Graduate Students in Regional Innovation which was conducted by the Ministry of Commerce Industry and Energy of the Korean Government.

7.Appendix

A. System Model

$$\frac{dE_q'}{dt} = -\frac{1}{T_{do}'}[E_q' + (X_d - X_d')I_d - E_{fd}]$$

$$\frac{d\delta}{dt} = \omega - \omega_{ref}$$

$$\frac{d\omega}{dt} = \frac{\omega_{ref}}{2H}[T_m - E_q' I_q - (X_q - X_d')I_d I_q]$$

$$\frac{dE_{ef}}{dt} = \frac{K_a}{T_a}(V_{ref} - V_t + V_s) - \frac{1}{Ta}E_{fd}$$

where, $V_t = \sqrt{V_d^2 + V_q^2}$

$$I_d = \frac{1}{\Delta}[\mathrm{R_e}(E_d' - V_\infty \sin\delta) + (X_e + X_q')(E_q' - V_\infty \cos\delta)]$$

$$I_q = \frac{1}{\Delta}[\mathrm{R_e}(E_q' - V_\infty \cos\delta) - (X_e + X_d')(E_d' - V_\infty \sin\delta)]$$

$$V_d = E_d' + \frac{X_q'}{\Delta}[R_e(E_q' - V_\infty \cos\delta) - (X_e + X_d')(E_d' - V_\infty \sin\delta)]$$

$$V_q = E_q' - \frac{X_d'}{\Delta}[R_e(E_d' - V_\infty \sin\delta) + (X_e + X_q')(E_q' - V_\infty \cos\delta)]$$

$$\Delta = R_e^2 + (X_e + X_d')(X_e + X_q')$$

B. Nomenclature

δ	: Rotor angle of generator
ω	: Rotor speed of generator
ω_{ref}	: Reference rotor speed of generator
H	: Inertia constant of generator
T_m	: Mechanical input of generator
X_d	: d-axis synchronous reactance of generator
X_d'	: d-axis transient reactance of generator
X_q	: q-axis synchronous reactance of generator
E_q'	: q-axis voltage of generator
E_{fd}	: Generator field voltage
T_{do}'	: d-axis transient time constant of generator
I_d	: d-axis current of generator
I_q	: q-axis current of generator
V_t	: Terminal voltage
V_{ref}	: Reference voltage
V_s	: PSS signal
V_{oo}	: Voltage of infinite bus
K_a	: AVR gain
T_a	: Exciter time constant
R_e	: Equivalent resistance of transmission line
X_e	: Equivalent reactance of transmission line

C. Multi-machine Power System

1. Constants of generator and exciter

Parameters / Generators	H [sec]	X_d [pu]	X'_d [pu]	X_q [pu]	T'_{do} [pu]	T'_{qo} [pu]
G1	6.4	0.8958	0.1198	0.8645	6.0	0.535
G2	5.4	1.3125	0.1813	1.2578	5.89	0.6

2. Load admittance

Load	Nominal load	Heavy load
Load A	1.261 - j0.504	2.314 – j0.925
Load B	0.878 – j0.293	2.032 – j0.677
Load C	0.969 – j0.339	1.584 – j0.634

3. Loading conditions

Loading condition \ Generators		G_1	G_2
Nominal load	P [pu]	1.35	0.80
	Q [pu]	0.02	- 0.12
Heavy load	P [pu]	1.65	1.05
	Q [pu]	0.53	0.35

8. References

Yu ,Y. N., 1983. Electric Power System Dynamics, Academic Press.

Larsen, E. V., Swann, D. A., 19981. Applying Power System Stabilizers Part Ⅰ: General Concepts. IEEE Transactions on Power Apparatus and System, PAS-100 (6), 3017-3024.

Kanniah, J., Malik, O. P., Hope, G. S., 1984. Excitation Control of Synchronous Generators Using Adaptive Regulators Part Ⅰ- Theory and Simulation Result. IEEE Transactions on Power Apparatus and Systems, PAS-103 (5), 897-904.

Chow, J. H., Sanchez-Gasca, J. J., 1989. Pole-Placement Design of Power System Stabilizers. IEEE Transactions on Power Systems, 4 (1), 271-277.

Ostojic, D., Kovacevic, B., 1990. On the Eigenvalue Control of Electromechanical Oscillations by Adaptive Power System Stabilizer. IEEE Transactions on Power Systems, 5 (4), 1118-1126.

Fleming, R. J., Jun Sun, 1990. An Optimal Multivariable Stabilizer for a Multimachine Plant. IEEE Transactions on Energy Conversion, 5 (1), 15-22.

Mao, C., Malik, O. P., Hope, G. S., Fun, J., 1990. An Adaptive Generator Excitation Controller Based on Linear Optimal Control. IEEE Transactions on Energy Conversion, 5 (4), 673-678.

Chen, G. P., Malik, Hope, O. P., G. S., Qin, Y. H., Xu, G. Y., 1993. An Adaptive Power System Stabilizer Based On The Self-Optimization Pole Shifting Control Strategy. IEEE Transactions on Energy Conversion, 8 (4), 639-644.

Park, T. M., Kim, W., 1996. Discrete-Time Adaptive Sliding Mode Power System Stabilizer with Only Input/Output Measurement. Electrical Power Energy Systems, 18 (8), 509-517.

Hassan, M. A. M., Malik, O. P., Hope, G. S., 1991. A Fuzzy Logic Based Stabilizer for a Synchronous Machine. IEEE Transactions on Energy Conversion, 6 (3), 407-413.

Hassan, M. A. M., Malik, O. P., 1993. Implementation and Laboratory Test Results for a Fuzzy Logic Based Self-tuned Power System Stabilizer. IEEE Transactions on Energy Conversion, 8 (2), 221-227.

Abido, M. A., Abdel-Magid, Y. L., 1998. A Genetic-Based Power System Stabilizer. Electric Machines and Power Systems, 26, 559-571.

Abido, M. A., Abdel-Magid, Y. L., 1999. Hybridizing Rule-Based Power System Stabilizers with Genetic Algorithms. IEEE Transactions on Power Systems, 14 (2), 600-607.

Fogel, D. B., 1995. Evolutionary Computation: Towards New Philiosophy of Machine Intelligence. IEEE Press, Piscataway, NJ.

Goldberg, D. E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley publishing Company, INC.

Gong, D., Yamazaki, G., Gen, M., 1996. Evolutionary program for Optimal Design of Material Distribution System. IEEE International Conference on Evolutionary Computation, 139-143.

Arabas, J., Michalewicz, Z., Mulawka, J., 1994. GAVaPS-a Genetic Algorithm with Varying Population Size. IEEE International Conference on Evolutionary Computation, 73-78.

Schlierkamp-Voosen, D., Muhlenbein, H., 1996. Adaptation of Population Sizes by Competing Subpopulations. IEEE International Conference on Evolutionary Computation, 330-335.

Fogel, D. B., Fogel , L. J., Atmas, J. W., 1991. Meta-Evolutionary Programming. Proceedings 2sth Asilomar Conference on Systems, Signals and Computers, 540-545.

Michalewicz, Z., 1992. Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag.

Renders, J. M., Flasse, S. P., 1996. Hybrid Methods Using Genetic Algorithms for Global Optimization. IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, 26 (2).

Mitsuo Gen, Cheng, R., 1997. Genetic Algorithms & Engineering Design. A Wiley-Interscience Publication.

He, J., Malik, O. P., 1997. An Adaptive Power System Stabilizer Based on Recurrent Neural Networks. IEEE Transactions on Energy Conversion, 12 (4), 413-418.

Anderson, P. M., Found, A. A., 1977. Power System Control and Stability. The IOWA State Unversity Press.

A Simple Hybrid Particle Swarm Optimization

Wei-Chang Yeh
Department of Industrial Engineering and Engineering Management
National Tsing Hua University
Taiwan, R.O.C.

1. Introduction

As a novel stochastic optimization technique, the Particle Swarm Optimization technique (PSO) has gained much attention towards several applications during the past decade for solving the global optimization problem or to set up a good approximate solution to the given problem with a high probability. PSO was first introduced by Eberhart and Kennedy [Kennedy and Eberhart, 1997]. It belongs to the category of Swarm Intelligence methods inspired by the metaphor of social interaction and communication such as bird flocking and fish schooling. It is also associated with wide categories of evolutionary algorithms through individual improvement along with population cooperation and competition. Since PSO was first introduced to optimize various continuous nonlinear functions, it has been successfully applied to a wide range of applications owing to the inherent simplicity of the concept, easy implementation and quick convergence [Trelea 2003].

PSO is initialized with a population of random solutions. Each individual is assigned with a randomized velocity based to its own and the companions flying experiences, and the individuals, called particles, are then flown through hyperspace. PSO leads to an effective combination of partial solutions in other particles and speedens the search procedure at an early stage in the generation. To apply PSO, several parameters including the population (N), cognition learning factor (c_p), social learning factor (c_g), inertia weight (w), and the number of iterations (T) or CPU time should be properly determined. Updating the velocity and positions are the most important parts of PSO as they play a vital role in exchanging information among particles. The details will be given in the following sections.

The simple PSO often suffers from the problem of being trapped in local optima. So, in this this paper, the PSO is revised with a simple adaptive inertia weight factor, proposed to efficiently control the global search and convergence to the global best solution. Moreover, a local search method is incorporated into PSO to construct a hybrid PSO (HPSO), where the parallel population-based evolutionary searching ability of PSO and local searching behavior are reasonably combined. Simulation results and comparisons demonstrate the effectiveness and efficiency of the proposed HPSO.

The paper is organized as follows. Section 2 describes the acronyms and notations. Section 3 outlines the proposed method in detail. In Section 4, the methodology of the proposed HPSO is discussed. Numerical simulations and comparisons are provided in Section 5. Finally, Concluding remarks and directions for future work are given in in Section 6.

2. Acronym and notations

Acronym:

PSO :	Particle Swarm Optimization Algorithm
SPSO :	Traditional PSO
IPSO :	An improved PSO proposed in [Jiang *et. al.* 2007]
HPSO :	The proposed Hybrid PSO

Notations:

D :	The number of dimensions.
N :	The number of particles in each replication.
T :	The number of generations in each replication.
R :	The total number of independent replications.
$r_\bullet$:	The random number uniformly distributed in [0, 1].
c_p, c_g :	The cognition learning factor and the social learning factor, respectively.
w :	The inertia weight.
$x_{t,i,j}$:	The dimension of the position of particle i at iteration t, where t=1,2,…,T, i=1,2,…,N, and j=1,2,…,D.
$X_{t,i}$:	$X_{t,i}=(x_{t,i,1},\ldots,x_{t,i,D})$ is the position of particle i at iteration t, where t=1,2,…,T, i=1,2,…,N, and j=1,2,…,D.
$v_{t,i,j}$:	the dimension of the velocity of particle i at iteration t, where t=1,2,…,T, i=1,2,…,N, and j=1,2,…,D.
$V_{t,i}$:	$V_{t,i}=(v_{t,i,1},\ldots,v_{t,i,D})$ is the velocity of particle i at iteration t, where t=1,2,…,T, i=1,2,…,N, and j=1,2,…,D.
$P_{t,i}$:	$P_{t,i}=(p_{t,i,1},\ldots,p_{t,i,D})$ is the best solution of particle i so far until iteration t, i.e., the pBest, where t=1,2,…,T, i=1,2,…,N, and j=1,2,…,D.
G_t :	$G_t=(g_{t,1},\ldots,g_{t,D})$ the best solution among $P_{t,1},P_{t,2},\ldots,P_{t,N}$ at iteration t, i.e., the gBest, where t=1,2,…,T.
F(•) :	The fitness function value of •.
U(•),L(•) :	The upper and lower bounds for •, respectively.

3. The PSO

In PSO, a solution is encoded as a finite-length string called a particle. All of the particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles [Parsopoulos *et. al.* 2001]. PSO is initialized with a population of random particles with random positions and velocities inside the problem space, and then searches for optima by updating generations. It combines the local and global search resulting in high search efficiency. Each particle moves towards its best previous position and towards the best particle in the whole swarm in every iteration. The former is a local best and its value is called *pBest*, and the latter is a global best and its value is called *gBest* in the literature. After finding the two best values, the particle updates its velocity and position with the following equation in continuous PSO:

$$v_{t,i,j} = wv_{t-1,i,j}+c_p r_{p,i,j}(p_{t-1,i,j}-x_{t-1,i,j})+c_p r_{p,i,j}(g_{t-1,j}-x_{t-1,i,j}), \quad (1)$$

$$x_{t,i,j}=x_{t-1,i,j}+v_{t,i,j}. \quad (2)$$

The values $c_p r_{p,i,j}$ and $c_g r_{g,i,j}$ determine the weights of the two parts, and c_p+c_g is usually limited to 4 [Trelea 2003]. Generally, the value of each component in $X_{t,i}$ and $V_{t,i}$ can be clamped to the range $[L(X), U(X)]$ and $[L(V), U(V)]$, respectively, to limit each particle to ensure its feasibility.
For example, let

$$X_{3,4}=(1.5, 3.6, 3.7, -3.4, -1.9), \tag{3}$$

$$V_{2,4}=(0.4, 0.1, 0.7, -2.7, -3.5), \tag{4}$$

$$P_{3,4}=(1.6, 3.7, 3.5, -2.1, -1.9), \tag{5}$$

$$G_3=(1.7, 3.7, 2.2, -3.5, -2.5), \tag{6}$$

$$R_p=(0.21, 0.58, 0.73, 0.9, 0.16), \tag{7}$$

$$R_g=(0.47, 0.45, 0.28, 0.05, 0.77), \tag{8}$$

$$L(X)= (0, 0, 0, -3.6, -3), \tag{9}$$

$$U(X)=(2, 4, 4, 0, 0), \tag{10}$$

$$L(V)=(-4, -4, -4, -4, -4), \tag{11}$$

$$U(V)=(4, 4, 4, 4, 4), \tag{12}$$

$$w=.9, \tag{13}$$

$$c_p=c_g=2. \tag{14}$$

Then, from Eq.(1), we have

$$V_{3,4}=(0.59, 0.296, -0.502, -0.1, -4.074). \tag{15}$$

Since -4.074<-4, $V_{3,4}$ needs to be adjustmented in the following:

$$V_{3,4}=(0.59, 0.296, -0.502, -0.1, -4). \tag{16}$$

Under the guidance of Eq.(2),

$$X_{4,4}=(2.09, 3.896, 3.198, -3.5, -5.974), \tag{17}$$

and

$$X_{4,4}=(2.0, 3.896, 3.198, -3.5, -3.0) \tag{18}$$

after the adjustment according to the upper/lower-bounds of X.
We conducted the preliminary experiments, and the complete computational procedure of the PSO algorithm can be summarized as follows.
STEP 1: Initialize: Initialize parameters and population with random positions and velocities.

STEP 2: Evaluation: Evaluate the fitness value (the desired objective function) for each particle.
STEP 3: Find the *pBest*: If the fitness value of particle *i* is better than its best fitness value (*pBest*) in history, then set current fitness value as the new *pBest* to particle *i*.
STEP 4: Find the *gBest*: If any *pBest* is updated and is better than the current *gBest*, then set *gBest* to the current value.
STEP 5: Update and adjustment velocity: Update velocity according to Eq.(1). Adjust the velocity to meet its range if necessary.
STEP 6: Update and adjustment position: Update velocity and move to the next position according to Eq.(2). Adjust the position to meet their range if necessary.
STEP 7: Stopping criterion: If the number of iterations or CPU time are met, then stop; otherwise go back to STEP 2.

4. The proposed HPSO

To overcome the weakness of PSO for local searches, this paper aims at creating HPSO by combining PSO, local search (LS), and vector based (VB) with a linearly varying inertia weight. The PSO part in the proposed HPSO is similar to the SPSO proposed in section 3. Hence, only the differences, i.e. the linearly varying inertia weight, LS and VB, are elaborated in this section.

4.1 Initial population

The initial population is generated randomly in the feasible space such that its lower-/upper-bounds are satisfied. To construct a direct relationship between the problem domain and the PSO particles in this study, the *i*th dimension in the pariticle stands for the value of the *i*th variable in the solution.

4.2 The linearly varying inertia weight

One of the most important issues to find the optimum solution effectively and efficiently while designing the PSO algorithm is its parameters. The inertia weight represents the influence of previous velocity which provides the necessary momentum for particles to move across the search space. Hence, the inertia weight dictates the balance between exploration and exploitation in PSO [Jiang *et. al.* 2007]. Shi and Eberhart (2001) made a significant improvement in the performance of the PSO with a linearly varying inertia weight over the generations, which linearly varies from 0.9 at the beginning of the search to 0.4 at the end. Thus the linearly varying inertia weight is adapted in the proposed HPSO to achieve trade-off between exploration and exploitation, i.e. the inertia weight of the *i*th generation is

$$w_i=U(w)-(i-1)[U(w)-L(w)]/N. \tag{19}$$

4.3 Vector based PSO

The underlying principle of the traditional PSO is that the next position of each particle is a compromise of its current position, the best position in its history so far, and the best position among all existing particles. The vector synthesis is the original mathematical foundation of PSO, as shown in the following figure.

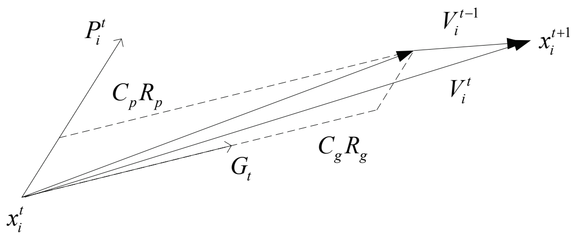


Fig. 1. The vector synthesis of PSO.

Eqs.(1) and (2) are more complicated than the concept of the vector synthesis in increasing the diversity of the dimensions of each particle. Hence, the following equations are implemented in the proposed HPSO instead of Eqs.(1) and (2):

$$V_{t,i}=w_i V_{t-1,i}+c_p r_p(P_{t-1,i}-X_{t-1,i})+c_g r_g(G_{t-1}-X_{t-1,i}), \tag{20}$$

$$X_{t,i}=X_{t-1,i}+V_{t,i}. \tag{21}$$

In the traditional PSO, each particle needs to use two random number vectors (e.g., R_p and R_q) to move to its next position. However, only two random number (e.g., r_p and r_q) are needed in the proposed HPSO. Besides, Eqs(20) and (21) are very easy and efficient in deciding the next positions for the problems with continuous variables. For example, let $P_{3,4}$, $X_{3,4}$, $P_{3,4}$, G_3, $L(X)$, $U(X)$ are the same as defined in the example in Section 1. Assume r_p=0.34 and r_g=0.79. From Eq.(19),

$$w_i=0.9-(4-1)[0.9-0.4]/1000=0.8985. \tag{22}$$

Plug w_i, r_p, r_g and the other required value into Eq.(20), we have

$$V_{3,4}=(0.6974, 0.28985, -1.56505, -1.75795, -3.97275), \tag{23}$$

$$V_{3,4}=(0.6974, 0.28985, -1.56505, -1.75795, -3.97275), \tag{24}$$

where $X_{4,4}$ is adjustmented from

$$(2.1974, 3.88985, 2.13495, -5.15795, -5.87275). \tag{25}$$

4.4 Local search method

One of the major drawbacks of PSO is is its very slow convergence. To surmount this drawback, to guide the search towards unexplored regions in the solution space and to avoid being trapped into local optimum, LS is implemented for constructing the proposed HPSO.

In PSO, proper control of global exploration and local exploitation is crucial in finding the optimum solution efficiently [Liu 2005]. A local optimizer is applied to the best particle (i.e. *gBest*) for each run in order to push it to climb the local optimum [Liu 2005]. With the hybrid method, PSOs are used to perform global exploration around particles except the *gBest* to maintain population diversity, while the local optimizer is used to perform local exploitation to the best particle. Since the properties of PSOs and conventional local optimizers are complementary, HPSOs are often better than either method operating alone from the computation exprements shown in Section 5.

The proposed LS is very simple and similar to the famous local improvement method the pairwise exchange procedure. In LS, the *i*th dimension of both the current best particle of all population (i.e., *gBest*) are replaced by the current best particle of the *j*th particle (i.e., *pBest*). If the fitness function value is improved, the the current *gBest* is updated accordingly. Otherwise, there is no need to change the current *gBest*. The above procedure in the proposed HPSO is repeated until all dimensions in the *gBest* are performed.

To minimize the number of duplicated computations of the same fitness function in LS, only one non-*gBest* is randomly selected to each dimension of *gBest* in the local search. The complete procedure of the local search part of the proposed HPSO can be summarized as in the following:

STEP 0. Let d=1.

STEP 1. Let n=1.

STEP 2. If $G_d=P_{t,n}$ or $g_{t,d}=p_{t,n,d}$, go to STEP 4. Otherwise, let $F^*=F(G_d)$, $G_d=P_{t,n}$, and $g_{t,d}=p_{t,n,d}$.

STEP 3. If $F(G_d)$ is better than F^*, then let $F^*=F(G_d)$. Otherwise, let $g_{t,d}=g$.

STEP 4. If $n<N$, let $n=n+1$ and go to STEP 2.

STEP 5. If $d<D$, let $d=d+1$ and go to STEP 1.

5. Numerical examples

To evaluate the performance of the proposed algorithms, four famous benchmark optimization problems [Jiang *et. al.* 2007] are used, which are described as follows.

Function	Formula	Range	Optima	Solution
Rosenbrock	$f_1(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	$[-30,30]^n$	0	(1,…,1)
Rastrigrin	$f_2(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	$[-5.12,5.12]^n$	0	(0,…,0)
Griewark	$f_3(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]^n$	0	(0,…,0)

Table 1. Benchmark functions.

Features of the above three functions are the following: Rosenbrock is an unimodal function and its variables are strongly dependent and gradient information often misleads algorithms; Rastrigin is ultimodal function with many local optima; Griewank is strongly multi-modal with significant interactions between its variables (caused by the product term) and the number of local minima increases with dimensionality [Jiang *et. al.* 2007].

These problems are implemented using the proposed HPSO, SPSO, and the best-known PSO (IPSO) proposed in by Jiang *et. al.* (2007) with regard to these three benchmark problems and the results of the experiments were compared. The proposed HPSO, SPSO and IPSO were implemented in C programming language on an Intel Pentium 2.6 GHz PC with 2G memory. To facilitate fair comparison, the number of iterations, the number of runs, and the populations for the proposed HPSO, SPSO and IPSO are taken directly from Jiang *et. al.* (2007). All these methods use a linearly varying inertia weight over the generations, varying from 0.9 at the beginning of the search to 0.4 at the end. In addition, $c_g=c_p=2$, $X_{\max}=V_{\max}=UB$ and $X_{\min}=V_{\min}=LB$ are used.

As in Jiang *et. al.* (2007), the proposed HPSO, SPSO and IPSO were tested on four group problems (population sizes of 20, 40, 80, and 160). The population sizes of each group are equal and each group problem contains 3 data sets. Each data set was first run with 3 sets of dimensions: 10, 20, and 30 and the corresponding maximum number of generations are set as 1000, 1500 and 2000, respectively. Hence, there are 12 different test sets to each benchmark problem as follows:

Set A1: N=20, D=10, T=1000;
Set A2: N=20, D=20, T=1500;
Set A3: N=20, D=30, T=2000;
Set B1: N=40, D=10, T=1000;
Set B2: N=40, D=20, T=1500;
Set B3: N=40, D=30, T=2000;
Set C1: N=80, D=10, T=1000;
Set C2: N=80, D=20, T=1500;
Set C3: N=80, D=30, T=2000;
Set $D$1: N=160, D=10, T=1000;
Set $D$2: N=160, D=20, T=1500;
Set $D$3: N=160, D=30, T=2000;

Each algorithm with each set of parameter is executed in 50 independent runs. The average fitness values of the best particle found for the 50 runs for the three functions are listed in Table 2. The shaded number shows the best result with respect to the corresponding function and the set. From the Table 2, it can be seen that IPSO outperforms the HPSO in Rosenbrock functions for POP=20 and 80. However, the remaining cases of Rosenbrock functions and for the Griewark function, the proposed HPSO has almost achieved better results than SPSO and IPSO. Furthermore HPSO is superior to SPSO and IPSO at all instances of the Rastrigrin function.

SET	Rosenbrock			Rastrigrin			Griewark		
	PSO	IPSO	HPSO	PSO	IPSO	HPSO	PSO	IPSO	HPSO
A1	42.6162	10.5172	3.3025	5.2062	3.2928	0	0.0920	0.0784	0.0071
A2	87.2870	75.7246	124.3305	22.7724	16.4137	0.4975	0.0317	0.0236	0.0168
A3	132.5973	99.8039	122.7829	49.2942	35.0189	1.0760	0.0482	0.0165	0.0190
B1	24.3512	1.2446	0	3.5697	2.6162	0	0.0762	0.0648	0.0002
B2	47.7243	8.7328	0.0797	17.2975	14.8894	0	0.0227	0.0182	0.0026
B3	66.6341	14.7301	120.7434	38.9142	27.7637	0	0.0153	0.0151	0.0012
C1	15.3883	0.1922	0.0797	2.3835	1.7054	0	0.0658	0.0594	0
C2	40.6403	1.5824	60.3717	12.9020	7.6689	0	0.0222	0.0091	0
C3	63.4453	1.5364	4.7461	30.0375	13.8827	0	0.0121	0.0004	0
D1	11.6283	0.0598	0	1.4418	0.8001	0	0.0577	0.0507	0
D2	28.9142	0.4771	0	10.0438	4.2799	0	0.0215	0.0048	0
D3	56.6689	0.4491	0	24.5105	11.9521	0	0.0121	0.0010	0
Average	39.48832	3.22272	20.66896	15.67783	9.50649	0	0.03396	0.02483	0.00044

Table 2. Mean Fitness function values 50 independent runs.

The final statistical result including the Success Rate, the fitness function values, CPU times and convergence iterations of all 50 runs related to Rosenbrock function, Rastrigrin function, and Griewark function are listed in Tables 3-5. The Success Rate is defined to be the percentage of the number of final searching solution that is equal to the global optimal value in 50 independent runs. Convergence iterations denote the number of iterations required for convergence. These data are divided into three categories: maximum, minimum, average, and standard deviations (denoted by max, min, mean, and std., respectively).

	Success	Fitness Function Value				Running Time (sec.)				Convergence Iterations			
SET	Rate	max	min	mean	std	max	min	Mean	std	max	min	mean	std
A1	92%	153.17	0	3.3025	21.65	0.03	0.02	0.02	0.00	962	520	720.3	73.77
A2	90%	3018.59	0	124.3305	597.18	0.12	0.05	0.06	0.01	1264	520	1130.3	141.50
A3	82%	3018.59	0	122.7829	597.14	0.29	0.09	0.11	0.04	1987	914	1603.5	268.45
B1	100%	0	0	0	0	0.05	0.04	0.05	0.00	787	607	660.1	39.34
B2	98%	3.99	0	0.0797	0.56	0.25	0.09	0.14	0.03	1129	813	1024.7	64.66
B3	96%	3018.59	0	120.7434	597.52	0.47	0.19	0.23	0.07	1635	1176	1477.9	114.45
C1	98%	3.99	0	0.0797	0.56	0.13	0.07	0.11	0.01	782	556	602.7	37.37
C2	98%	3018.59	0	60.3717	426.89	0.46	0.24	0.37	0.04	1089	649	946.9	60.43
C3	98%	237.31	0	4.7461	33.56	0.91	0.44	0.66	0.10	1457	1217	1346.7	56.36
D1	100%	0	0	0	0	0.27	0.21	0.25	0.02	681	533	567.4	28.35
D2	100%	0	0	0	0	1.00	0.78	0.89	0.05	1048	818	887.3	47.31
D3	100%	0	0	0	0	2.41	1.52	1.78	0.17	1452	1073	1232.0	73.77

Table 3. Experimental results on Rosenbrock function of 50 independent runs.

	Success	Fitness Function Value				Running Time (sec.)				Convergence Iterations			
SET	Rate	max	min	mean	std	max	min	mean	std	max	min	mean	std
A1	100%	0	0	0	0	0.05	0.04	0.05	0.00	112	5	30.3	20.52
A2	98%	24.87	0	0.4975	3.52	0.16	0.15	0.16	0.00	329	11	56.1	49.95
A3	96%	28.92	0	1.0760	5.34	0.37	0.33	0.34	0.01	357	19	86.4	74.91
B1	100%	0	0	0	0	0.09	0.08	0.08	0.00	58	7	25.4	12.34
B2	100%	0	0	0	0	0.27	0.25	0.25	0.00	104	11	36.4	20.96
B3	100%	0	0	0	0	0.58	0.53	0.55	0.01	196	20	47.7	28.55
C1	100%	0	0	0	0	0.17	0.15	0.16	0.00	56	5	19.3	11.02
C2	100%	0	0	0	0	0.49	0.45	0.47	0.01	106	9	32.9	17.43
C3	100%	0	0	0	0	1.01	0.92	0.96	0.02	127	18	45.2	25.74
D1	100%	0	0	0	0	0.32	0.30	0.31	0.00	38	5	15.6	7.43
D2	100%	0	0	0	0	0.92	0.86	0.89	0.01	63	9	26.4	10.87
D3	100%	0	0	0	0	1.91	1.74	1.79	0.03	97	9	34.7	15.96

Table 4. Experimental results on Rastrigrin function of 50 independent runs.

	Success	Fitness Function Value				Running Time (sec.)				Convergence Iterations			
SET	Rate	max	min	mean	std	max	min	mean	std	max	min	mean	std
A1	86%	0.10	0	0.0071	0.02	0.08	0.08	0.08	0.00	711	6	110.7	169.22
A2	74%	0.13	0	0.0186	0.04	0.30	0.27	0.28	0.01	1401	18	262.1	303.67
A3	76%	0.21	0	0.0190	0.04	0.72	0.64	0.67	0.03	812	15	275.0	272.40
B1	98%	0.01	0	0.0002	0.00	0.14	0.13	0.14	0.00	518	4	53.1	74.79
B2	96%	0.07	0	0.0026	0.01	0.49	0.44	0.45	0.01	823	10	90.7	151.76
B3	96%	0.03	0	0.0012	0.01	1.06	0.99	1.00	0.01	738	10	95.7	140.38
C1	100%	0	0	0	0	0.26	0.25	0.26	0.00	124	2	20.5	23.07
C2	100%	0	0	0	0	0.81	0.78	0.80	0.01	120	10	40.8	26.87
C3	100%	0	0	0	0	1.74	1.67	1.70	0.02	131	8	48.1	28.07
D1	100%	0	0	0	0	0.51	0.49	0.50	0.00	30	6	14.8	6.09
D2	100%	0	0	0	0	1.53	1.46	1.49	0.01	104	9	29.0	16.99
D3	100%	0	0	0	0	3.18	3.04	3.10	0.03	77	9	32.9	14.44

Table 5. Experimental results on Griewark function of 50 independent runs.

As the dimension increases, the solution space get more complex, and PSO algorithm is more likely to be trapped into local optima. Experimental data shown in Table 2 does not clearly indicate that the HPSO outperforms the other PSOs in the measures of average fitness function values. However, the Success Rates are all over 74%. Therefore, the proposed HPSO can find global optima with very high probability, and it is concluded that HPSO has the strongest exploration ability and it is not easy to be trapped into local optima. Table 3 shows that the proposed HPSO uses only 3.18 seconds in worst case and 0.6 seconds in average. Thus, HPSO is very effective, efficient, robust, and reliable for complex numerical optimization.

6. Conclusions

A successful evolutionary algorithm is one with a proper balance between exploration (searching for good solutions), and exploitation (refining the solutions by combining information gathered during the exploration phase). In this study, a new hybrid version of PSO called HPSO is proposed. The HPSO constitutes a vector based PSO method with the linearly varying inertia weight, along with a local search.

A novel, simpler, and efficient mechanism is employed to move the *gBest* to its next position in the proposed HPSO. The HPSO combines the population-based evolutionary searching ability of PSO and local searching behavior to effciently balance the exploration and exploitation abilities. The result obtained by HPSO has been compared with those obtained from traditional simple PSO (SPSO) and improved PSO (IPSO) proposed recently. Computational results show that the proposed HPSO shows an enhancement in searching efficiency and improve the searching quality. In summary, the results presented in this work are encouraging and promising for the application of the proposed HPSO to other complex problems.

Further analysis is necessary to see how other soft computing method (e.g., the genetic algorithm, the taboo search, etc.) react to local searches for future researchers who may want to develop their own heuristics and to make further improvements. Our research is still very active and under progress, and it opens the avenues for future efforts in this directions such as: how to adjust parameters, increase success rates, reduce running times, using other local search, and the aggregation of different and new concepts to PSO.

7. References

B. Liu, L. Wang, Y.-H. Jin, F. Tang, D.-X. Huang (2005), “Improved particle swarm optimization combined with chaos”, Chaos Solitons & Fractals, Vol. 25, 2005, pp. 1261–1271.

I.C. Trelea (2003), The particle swarm optimization algorithm: convergence analysis and parameter selection, *Information Processing Letters*, Vol. 85, 2003, pp. 317–325.

J. Kennedy and R.C. Eberhard (1995), “Particle swarm optimization”, *Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, USA,* 1995, pp. 1942-1948.

J. Kennedy and R.C. Eberhard and Y. Shi, “*Swarm intelligence*”, San Francisco, CA: Morgan Kaufmann; 2001.

J. Kennedy and R.C. Eberhart (1997), “A discrete binary version of the particle swarm algorithm”, Systems, Man, and Cybernetics, *Computational Cybernetics and Simulation, IEEE International Conference*, Vol. 5, No. 12-15, 1997/10, pp. 4104-4108.

J. Moore and R. Chapman (1999), "Application of particle swarm to multiobjective optimization", *Department of Computer Science and Software Engineering, Auburn University*.

K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis (2001), Improving particle swarm optimizer by function stretching, Advances in Convex Analysis and Global Optimization, 2001, 445–457.

R.C. Eberhart and Y. Shi (2001), "Particle Swarm Optimization: Developments, Application and Resources", Proceedings of the 2001 Congress on Evolutionary Computation, Seoul, South Korea, Vol. 1, pp. 81-86.

Y. Jiang, T. Hu, C. Huang, and X. Wu (2007), "An improved particle swarm optimization algorithm", *Applied Mathematics and Computation*, Vol. 193, pp. 231–239.

Recent Advances in Harmony Search

Zong Woo Geem[1], M. Fesanghary[2], Jeong-Yoon Choi[3], M. P. Saka[4], Justin C. Williams[1], M. Tamer Ayvaz[5], Liang Li[6], Sam Ryu[7] and A. Vasebi[8]
[1]Johns Hopkins University, [2]Amirkabir University of Technology, [3]Montgomery College, [4]Middle East Technical University, [5]Pamukkale University, [6]Da Lian University of Technology, [7]SOFEC, [8]K.N.Toosi University of Technology
[1,3,7]USA, [2,8]Iran, [4,5]Turkey, [6]China

1. Introduction

The harmony search (HS) is a music-inspired evolutionary algorithm, mimicking the improvisation process of music players (Geem et al., 2001). The HS is simple in concept, few in parameters, and easy in implementation, with theoretical background of stochastic derivative (Geem, 2007a). The algorithm was originally developed for discrete optimization and later expanded for continuous optimization (Lee & Geem, 2005).
The following pseudo code describes how the HS algorithm works:

```
procedure HS
  // initialize
  initiate parameters
  initialize the harmony memory
  //main loop
  while (not_termination)
    for I = 1 to number of decision variables (N) do
      R1 = uniform random number between 0 and 1
      if (R1 < $P_{HMCR}$)  (memory consideration)
        X[I] will be randomly chosen from harmony memory
        R2 = uniform random number
        if (R2 < $P_{PAR}$)  (pitch adjustment)
          X[I] = X[I] $\pm \Delta$
        end if
      else  (random selection)
        X[I] = X $\in \Phi$  ($\Phi$ = Value Set)
      end if
    end do
  // evaluate the fitness of each vector
  fitness_X = evaluate_fitness(X)
  // update harmony memory
  update_memory(X, fitness_X) % if applicable

  end while
end procedure
```

Ensemble harmony search (EHS) is another variant of the HS where ensemble consideration is added to the original algorithm structure (Geem, 2006a). The new operation considers the relationship among decision variables. The EHS could overcome the drawback of genetic algorithm's building block theory which does not work well if less-correlated variables locate closely in a chromosome.

Mahdavi et al. (2007) proposed an improved harmony search (IHS), in which dynamic parameter adjusting is used in improvisation step. As the search progresses, P_{PAR} is increased linearly while adjusting amount is decreased exponentially. This modification improves the local exploitation capability of the HS algorithm.

Recently, Omran & Mahdavi (2007) proposed a new variant of harmony search, called the global-best harmony search (GHS), in which the concepts from swarm intelligence are borrowed to enhance the performance of HS such that the new harmony can mimic the best harmony in the harmony memory (HM).

The HS algorithm has been successfully applied to various artificial intelligence and engineering problems including music composition (Geem & Choi, 2007), Sudoku puzzle solving (Geem, 2007b), structural design (Lee & Geem, 2004; Saka, 2007), ecological conservation (Geem & Williams, 2008), aquifer parameter identification (Ayvaz, 2007), soil slip determination (Cheng et al., 2008), offshore structure mooring (Ryu et al., 2007), power economic dispatch (Vasebi et al., 2007), pipeline network design (Geem, 2006b), and dam operation (Geem, 2007c).

The goal of this chapter is to review various recent applications of the HS algorithm, helping other researchers to draw a big picture of the HS ability and to apply it to their own problems.

2. Recent applications

2.1 Music composition

The HS algorithm composed music pieces (Geem & Choi, 2007). When HS was applied to the organum (an early form of polyphonic music) composition, it was able to successfully compose harmony lines based on original Gregorian chant lines.

Gregorian chant is a monophonic religious song in the middle ages, and organum is an early form of harmonized music which accompanies the Gregorian chant melody. HS generates the harmony line (vox organalis) to accompany the original Gregorian chant (vox principalis).

The organum has the following composing rules: the harmony line progresses in parallel; for the parallel motion, the interval of perfect fourth is preferred; and, in order to distinguish the vox principalis from vox organalis, the former should always be located above the latter.

The above-mentioned rules were formulated as a optimization problem. Then, HS solved the problem, obtaining aesthetically pleasing organum as shown in Figure 1.

Figure 1 shows a Gregorian chant “Rex caeli Domine” and its organum composed by HS. The upper line in the figure is the Gregorian chant melody and the lower line is the organum line.

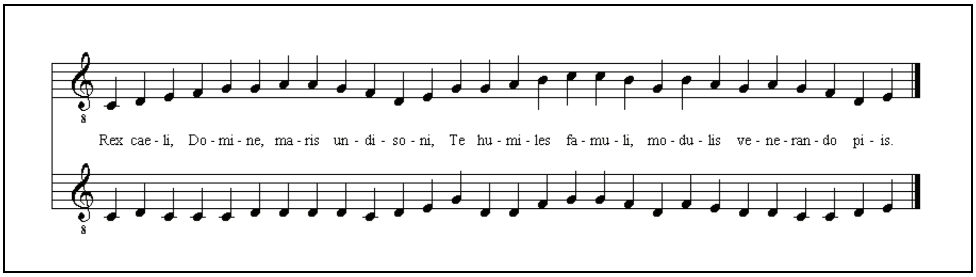
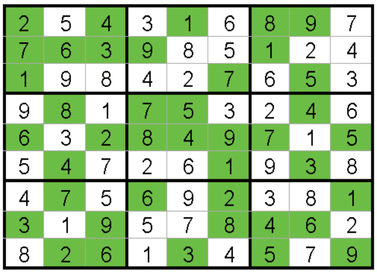

Rex cae - li, Do - mi - ne, ma - ris un - di - so - ni, Te hu - mi - les fa - mu - li, mo - du - lis ve - ne - ran - do pi - is.

Fig. 1. Organum Composed by HS algorithm

2.2 Sudoku puzzle solving

HS was applied to a Sudoku puzzle (Geem, 2007b), which is formulated as an optimization problem with number-uniqueness penalties.

Sudoku means "singular number" in Japanese, and consists of 9 × 9 grid and 3 × 3 blocks for all the 81 cells. Each puzzle starts with some cells that already have numbers as shown in Figure 2 (the numbers in white cells are originally given). The goal of the puzzle is to find numbers for the remaining cells with three rules: (1) Each horizontal row should contain the numbers 1 - 9, without repeating any; (2) Each vertical column should contain the numbers 1 - 9, without repeating any; and (3) Each 3 × 3 block should contain the numbers 1 - 9, without repeating any.



2	5	4	3	1	6	8	9	7
7	6	3	9	8	5	1	2	4
1	9	8	4	2	7	6	5	3
9	8	1	7	5	3	2	4	6
6	3	2	8	4	9	7	1	5
5	4	7	2	6	1	9	3	8
4	7	5	6	9	2	3	8	1
3	1	9	5	7	8	4	6	2
8	2	6	1	3	4	5	7	9

Fig. 2. Sudoku Puzzle Solved by HS algorithm

The HS model found the optimal solution without any violation of three rules after 285 function evaluations as shown in Figure 2.

2.3 Structural design

Structural design involves in decision making about cross sectional dimensions of the members that constitute the structure and sometimes the geometry and topology of the structure itself. In the design of a steel frame, the decision making process necessitates selecting W or any other type of steel sections from practically available set of steel section tables for the members of the frame such that the response of the frame to external loads is within the limitations described in the steel design codes. It is not very difficult to imagine that one can come up with large number of different combinations selected from the available steel section set which may satisfy these requirements. However, the designer is interested in finding the combination which not only satisfies design code limitations but also minimizes the material weight or the overall cost. This is the optimal design. HS method is quite effective in finding the optimum solution of such combinatorial optimization problems. In this section the HS algorithm is applied to determine the solution of optimum design of grillage system, optimum geometry design of a steel dome and the optimum design of reinforced concrete continuous beam.

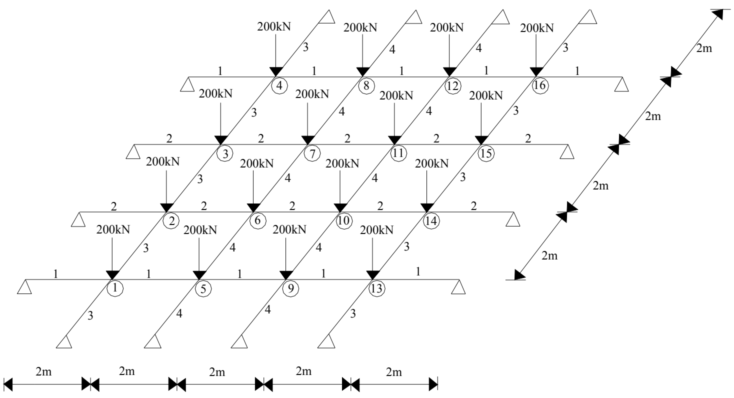


Fig. 3. 40-Member Grillage System

Optimum Design of 40-Member Grillage System: The grillage system shown in Figure 3 has 40 members which are collected in four groups such that the outer and inner longitudinal beams are considered to belong to groups 1 and 2 while the outer and inner transverse beams are taken as groups 3 and 4 respectively. This system is originally designed using HS (Erdal, 2007). The displacement and stress constraints are considered in the formulation of this design problem. The external loading that the grillage system is subjected to also shown in the figure. Under this loading it is required that the vertical displacements of joints 6, 7, 10 and 11 should not exceed 25mm. Furthermore it is the condition of the design criteria that nowhere in the longitudinal and transverse beams the bending stress should exceed the allowable bending stress of 250MPa. The 272 W-sections starting from W100X19.3 to W1100X499 are selected from LRFD-AISC (Manual of Steel Construction) as an available discrete design set for the optimum design procedure to select from. The task of the optimum design algorithm is to decide the appropriate W sections from this list for longitudinal and transverse beams of the grillage system such that the displacement and stress constraints described above are satisfied while the weight of the grillage system is the minimum. The solution of this problem is obtained by using HS as well as genetic algorithm (GA). The GA algorithm utilized in the solution of this design problem is a simple genetic algorithm where the initial population size is taken as 50 and two-point crossover is used to swap the genetic information between mating parents. While GA obtained the optimum solution after 40,000 structural analyses (function evaluations), HS required only 10,000 structural analyses to reach the optimum result. The optimum design (minimum weight = 7,075.84 kg) obtained by the HS method is 14% lighter than the one (8,087.91kg) determined by the GA in this particular design problem.

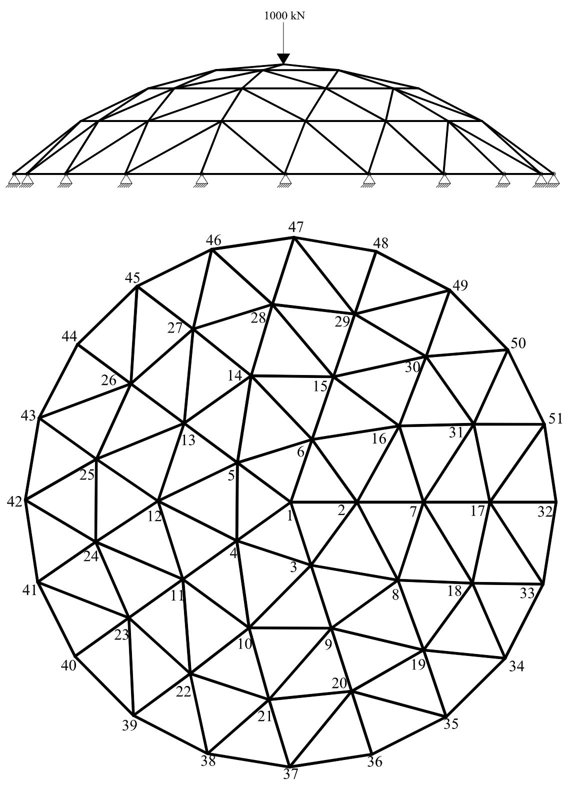


Fig. 4. Geodesic dome

Optimum Geometry Design of Geodesic Domes: Domes are economical structures in terms of materials that are used to cover large areas such as exhibition halls and stadiums where they provide a completely unobstructed inner space. Domes are given different names depending upon the way their surface is formed. Geodesic dome shown in Figure 4 is a typical example of a braced dome which is widely used in the construction of exhibition halls all over the world. A geodesic dome is comprised of a complex network of triangles that form a roughly spherical surface. Generally the area that is to be covered by the dome is provided by a client and the structural designer is required to come up with dimensions of

pipe sections that are usually adopted for the dome members and also specify the height of the crown.
The design problem considered here is to determine the optimum height and circular steel hollow section designations for the geodesic dome that is suppose to cover the circular area of 20m as shown in Figure 4. The modulus of elasticity of the material is taken as 205kN/mm^2. The grade of steel adopted is grade 43. The dome is considered to be subjected to equipment loading of 1000kN at its crown. The formulation of the design problem and the construction of these constraints are explained in detail by Saka (2007). The solution of the design problem is obtained by HS. There are altogether 32 values for the HS algorithm to choose from.
It is apparent from Figure 4 that there are 3 rings in the dome. This number can also be treated as design variable. However for the simplicity here it is not taken as design variable. Two design problems are considered. In the first one all the members are decided to be made out of the same pipe section which means all the members are belong to the same group. In this case HS obtains the optimum height of the dome as 1.75m and PIP886 is adopted for the dome members. The minimum weight for this dome is 3750.6kg. It is noticed that while the displacements of the restricted joints are much smaller than their upper limits the strength ratios of some members are at their upper bound. This indicates that in the optimum design problem the strength constraints were dominant. Later, it is decided that those members between each ring are to be made one group and the members on each ring are another group. For example, if grouping is carried out such a way that the diagonal members between the crown and the first ring are group 1, the members on the first ring are group 2, the members between ring 1 and 2 are group3 and the group number of members on the ring 2 is 4 and so forth, then the total number of groups in the dome becomes twice the number of rings in the dome. In this case HS method determines the optimum height of the crown as 2m while the sectional designations for six groups of the dome members were PIP1143, PIP603.6, PIP483.2, PIP423.2 and PIP213.2. The minimum weight of the dome was 1244.42kg. Once more it is noticed that the strength constraints were dominant in the design problem
Optimum Design of Reinforced Concrete Continuous Beams: In the formulation of the optimum design problem of reinforced concrete continuous beams, design variables are selected as the width and height of beams and the reinforcement areas of longitudinal bars. These longitudinal bars are tensile reinforcements at each mid-span and supports and the shear reinforcement bar diameters for each beam. The general description of the design variables for four span continuous beams is given in Figure 5. The objective function is the total cost of the continuous beams which consists of cost of concrete, formwork and reinforcement steel. The design constraints consist of the ultimate strength requirements in bending and shear and minimum and maximum percentage of tensile and shear reinforcements. The details of these constraints are given by Akin (2007). The optimum design determined by the HS algorithm has the minimum cost of \$11,406 while GA obtained \$11,836.
Three different structural design problems are considered to demonstrate the robustness and effectiveness of the HS algorithm. The first problem is a size optimization problem where the HS method has selected optimum W sectional designations for longitudinal and transverse beams of grillage systems out of 272 discrete set of W steel sections. The solution obtained by HS is better than the one determined by simple genetic algorithm. The second

design example is optimum geometry design of a geodesic dome where the HS algorithm has also effectively determined the optimum height of the crown as well as the optimum pipe designations for the dome members. Finally in the third design example, it is shown that HS can be successfully employed to determine the optimum cross sectional dimensions for beams as well as required reinforcement diameters and their total number in the design of reinforced concrete continuous beams.

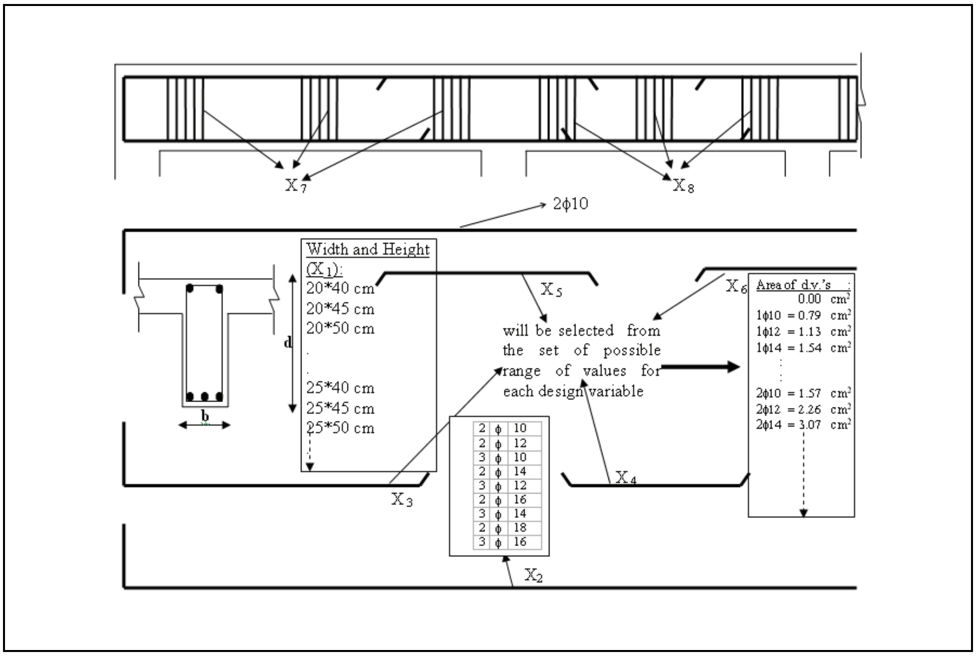


Fig. 5. Design Variables for Four Span Symmetrical Reinforced Concrete Continuous Beam

2.4 Ecological conservation

In today's industrialized life, to conserve ecosystem and its species becomes very important. In order to achieve the goal, quantitative techniques have been so far developed and utilized for the problem. HS was also applied to a natural reserve selection problem for preserving species and their habitats (Geem & Williams, 2008). The problem was formulated as an optimization problem (maximal covering species problem) to maximize the number of species protected within the reserve system given a specified number of sites that can be selected (ReVelle et al., 2002). The HS model developed for this problem was tested with real-world problem in the state of Oregon, USA, which consists of 426 species and 441 candidate sites as shown in Figure 6.

Harmony Search was applied to 24 cases, each involving a different limit on number of parcels that could be selected. HS found 15 global optimum solutions and 9 near-optimal solutions. When compared with simulated annealing (SA), the HS algorithm found better solutions than those of SA in 14 cases while the former found worse solution only once (Csuti et al., 1997).

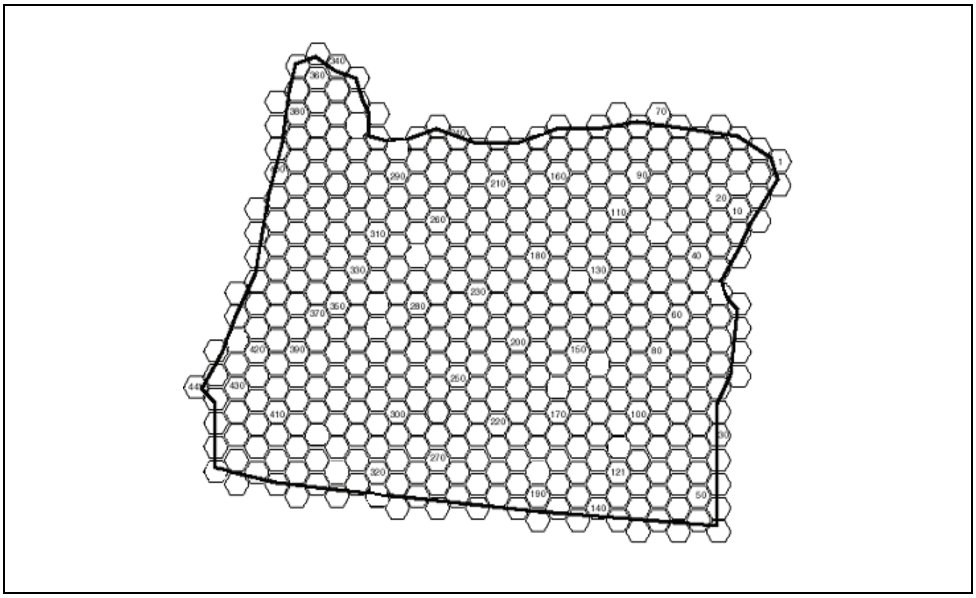


Fig. 6. Hex Map of Oregon

Another advantage of the HS algorithm is that it gives many alternative solutions because it handles multiple solutions as a time. For example, the HS found 25 alternative solutions for the case of 24 selected sites.

2.5 Aquifer parameter identification

Mathematical simulation models are widely used in the management of aquifer systems. These models require the spatial distributions of some hydrologic and hydro-geologic parameters for the solution process. However, aquifers are heterogeneous geological structures and usually distribution of their parameters is unknown. Thus, the determination of both aquifer parameters and their corresponding parameter structures based on field observations becomes an important step. The main goal of this study is to propose an S/O approach for simultaneously identification of transmissivity values and associated zone structures of a heterogeneous aquifer system. In the simulation model, the governing equation of groundwater flow is numerically solved using a block-centered finite difference solution scheme. The zone structure identification problem is solved through fuzzy c-means clustering (FCM) algorithm, and the HS algorithm is used as an optimization model to determine the optimum locations of cluster centroids and the associated transmissivity values within each zone (Ayvaz , 2007).

The main reason for applying FCM and HS to the groundwater inverse problem is to determine the zone structure and associated transmissivity values within each zone. The parameter zone structure of the aquifer is initiated using random cluster centroids and random transmissivity values are assigned to each cluster. Cluster centroids and transmissivity values are then optimized using HS by minimizing the residual error (RE) between the simulated and observed hydraulic heads at several observation wells.

The performance of the proposed S/O approach is tested on a hypothetical example. Figure 7 (Left) shows the plain view of two-dimensional confined aquifer.

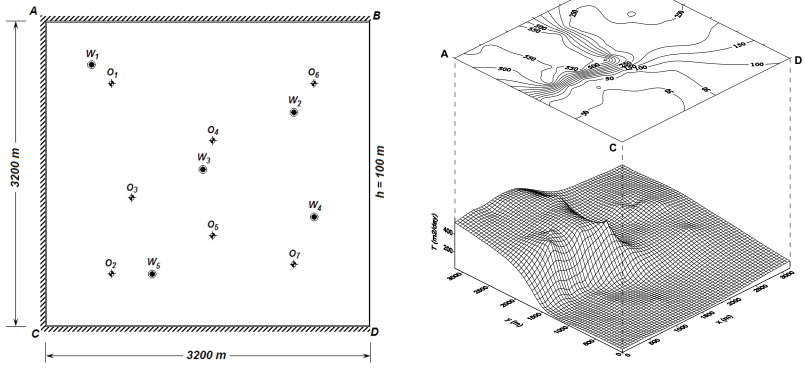


Fig. 7. (Left) Plain View of Confined Aquifer and (Right) True Transmissivity Field

As can be seen in Figure 7 (Left), the boundary conditions of the aquifer are 100 m constant head in the BD side and the no-flow in the other sides. The storage coefficient of the aquifer is the 0.0002. There are five pumping wells having the pumping rates of 4,000 cmd for Wells 1 to 4 and 2,000 cmd for Well 5. All the pumping wells are continuously operated for 10 days. There are seven observation wells and head observations are collected at the end of each day. The Gaussian noise of zero mean and 0.1 m standard deviation is added to the head observations. The true transmissivity field of the aquifer is shown in Figure 7 (Right). The main goal is to determine the best zonation pattern to satisfy the true transmissivity field. For the optimization process, five cases with different algorithm parameters are taken into account. Maximum number of improvisations (iteration) is set as 50,000 and the search process ends when the RE value remains unchanged through 1,000 improvisations. Note that, for comparison, the number of zones is fixed as 4 and the bounds of transmissivity values are set as 20 ~ 600 smd.

HS obtained the minimum RE (2.33) after 29,370 of function evaluations. Note that, GA (Tsai et al., 2003) solved the same problem, obtaining RE of 2.62 after 40,000 function evaluations. Although there are some differences, the identified transmissivity structures well capture the true transmissivity field.

2.6 Soil slip determination

Soil slopes are general in civil engineering and their stability assessment is of great importance to engineers. Up to now, limit equilibrium method is widely used by engineers and researchers for slope stability analysis. By using limit equilibrium method, a value F_s, also named the factor of safety can be estimated without the knowledge of the initial stress conditions and a problem can be defined and solved within a relatively short time. Limit equilibrium method is a statically indeterminate problem and different assumptions on the

internal forces distributions are adopted for different methods of analyses. At present, the famous method proposed by Morgenstern and Price (1965) is used to give the factor of safety for specified slip surface.
The minimum factor of safety of a slope and the corresponding critical failure surface are critical for the proper design of slope stabilization measures. The HS algorithm is employed to locate the critical failure surface in slope stability analysis. The generation of slip surfaces is as follows.
Consider the Cartesian system of reference Oxy as shown in Figure 8.

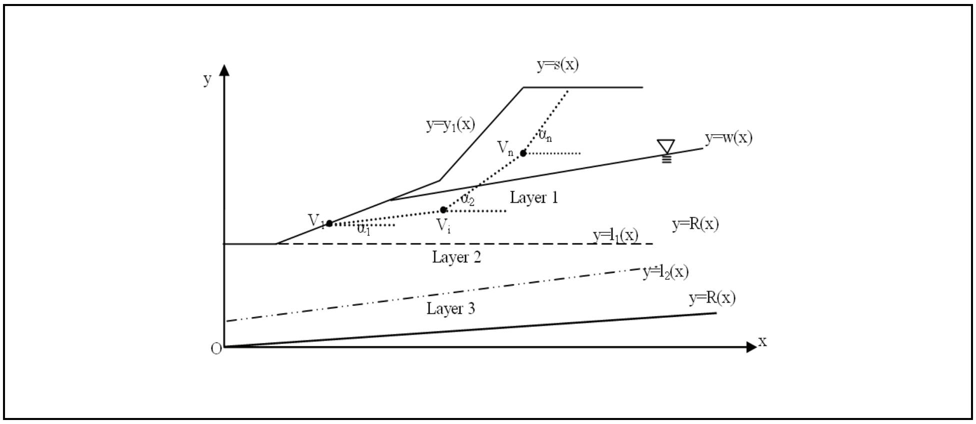


Fig. 8. Slip Surface and the Cross Section of a Slope

Function $y = y_1(x)$ describes the ground profile while the water table is represented by $y = w(x)$. The bed rock surface is represented by the function $y = R(x)$ and function $y = l_i(x)$ can be introduced to represent boundary between different soils. The trial failure surface is described by using the function $y = s(x)$.

To obtain the values of F_s requires the failure soil mass to be divided into n vertical slices and the slip surface is represented by n+1 vertices. Each slice can be identified by two adjacent vertices. Generally speaking, the potential slip surfaces are concave upward (kinematically acceptable requirement) with only few exceptions. The concave upward requirement can be formulated as follows:

$$\alpha_1 \le \alpha_2 \le \ldots \le \alpha_n \tag{1}$$

where α_i is the base inclination of slice i as shown in Figure 8. Every slip surface can be mathematically identified by the control variable vector $\mathbf{X}$ as follows:

$$\mathbf{X} = [x_1, y_1, x_2, y_2, \ldots, x_n, y_n, x_{n+1}, y_{n+1}]^T \tag{2}$$

The vector $\mathbf{X}$ is analogous to the harmony in music, and the HS algorithm can be performed to determine the critical slip surface with the minimum factor of safety.

The example is the one proposed by Zolfaghari (2005), where a slope in layered soil is analyzed using the GA and the Morgenstern and Price method. The number of slices n used in this study is assumed to be 20, 25, and 30. While GA (Zolfaghari, 2005) found minimum safety factor of 1.24, HS found 1.20 with 30 slices.

2.7 Mooring design of offshore floating structures

The mooring design of offshore platforms requires relatively significant amount of design cycles since a desired solution must satisfy the complex design constraints and be economically competitive. The complexity of these mooring design constraints may result from coupling between platform motion and mooring/riser system, maximum offset constraint of the riser system, multiple numbers of design parameters defining anchor leg system components, and uniqueness of site-dependent environmental conditions including water depth, wave/current/wind condition, seabed condition, etc. When the optimal cost is sought for this complex mooring design, the design process becomes even more complex.

Mooring design is to find an appropriate stiffness which is stiff enough and soft enough at the same time since the mooring system needs to satisfy mainly two design constraints: (1) required maximum horizontal offset and (2) reduction of extreme forces acting on the platform caused by interactions between environmental forces and platform responses. To reduce the trial and error effort in mooring design, Fylling (1997) addresses an application of mooring optimization of deepwater mooring systems. A nonlinear optimization program with frequency-domain analysis of mooring systems was presented, and the results showed that the suggested optimization could be a powerful tool for concept development and finding a feasible solution (Fylling, 1997). Fylling and Kleiven (2000) presented the simultaneous optimization of mooring lines and risers.

A single point mooring of a Floating, Production, Storage, and Offloading (FPSO) system was adopted for a case study. Deepwater and ultra-deepwater application of FPSOs becomes more attractive since they have advantages in early production and relatively big storage capacity compared to other types of offshore platforms. As we target for deeper water oil/gas fields, more technical challenges are confronted. For instance, prediction of deepwater oil offloading buoy motion becomes more difficult (Duggal and Ryu, 2005; Ryu, et al., 2006). Technical challenges due to deepwater and ultra-deepwater oil fields and project execution challenges due to the fast track schedule become a trend in FPSO projects. This deeper water and fast track trend naturally suggests a way of fast finding of a site and requirement specific feasible mooring design.

This section addresses a HS-based mooring optimization determining the length and diameter of each mooring component. In this design, only three design constraints were applied: (1) maximum platform offset, (2) factor of safety (FS) for intact case top tension, and (3) no uplift of the bottom chain. The objective function is the total cost of mooring system.

A total of 2,000 iterations were performed to find optimal mooring designs. Figure 9 presents the search history of optimal mooring cost as a function of iteration, and Figure 10 shows one final solution the HS algorithm found.

A mooring optimization design tool using the HS algorithm and a frequency domain global analysis tool was proposed to minimize the cost of the mooring system. This proposed cost-optimal mooring design tool successfully finds feasible mooring systems. A case study on a permanent turret mooring system for an FPSO in deepwater was conducted. The results show that the objective function (i.e. mooring system cost) converges well and HS provides several feasible mooring systems. In conclusion, a new HS-based mooring optimization tool, has a potential for fast finding the cost-optimal mooring system.

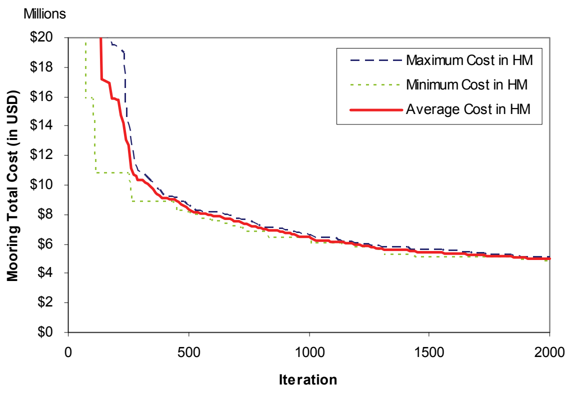


Fig. 9. Max, Min, and Mean Costs in Harmony Memory

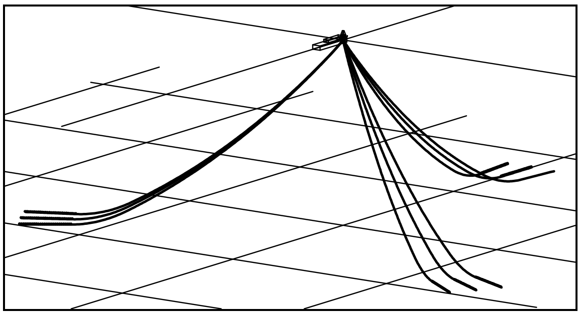


Fig. 10. Mooring Configurations

2.8 Heat & power generation

The conversion of primary fossil fuels, such as coal and gas, to electricity is a relatively inefficient process. Even the most modern combined cycle plants can only achieve efficiencies of between 50–60%. Most of the energy that is wasted in this conversion process is released to the environment as waste heat. The principle of combined heat and power (CHP), also known as cogeneration, is to recover and make beneficial use of this heat, significantly raising the overall efficiency of the conversion process. The best CHP schemes can achieve fuel conversion efficiencies of the order of 90%. In order to obtain the optimal utilization of CHP units, economic dispatch must be applied. The primary objective of

economic dispatch is to minimize the total cost of generation while honoring the operational constraints of the available generation resources. Complication arises if one or more units produce both electricity and heat. In this case, both of heat and power demands must be met concurrently. This section will show the application of the HS algorithm to solve the CHPED problem.
Figure 11 shows the heat-power Feasible Operation Region (*FOR*) of a combined cycle cogeneration unit. The feasible operation region is enclosed by the boundary curve ABCDEF.

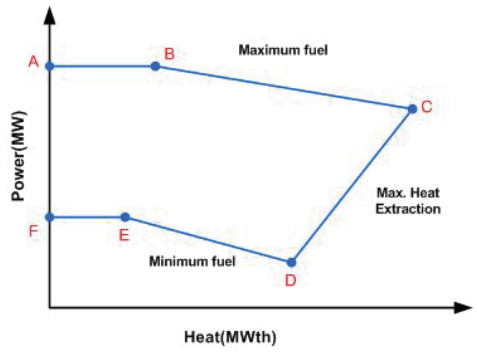


Fig. 11. Feasible Operation Region for a Cogeneration Unit

An example which is taken from the literature is used to show the validity and effectiveness of the HS algorithm. This example has been previously solved using a variety of other techniques (both evolutionary and traditional mathematical programming methods) after originally proposed by Guo et al. (1996). The problem consists of a conventional power unit, two cogeneration units and a heat-only unit. The objective is to find the minimum overall cost of units subject to constraints on heat and power production and demands.
After 25,000 function evaluations, the best solution is obtained with corresponding function value equal to $9257.07 (Vasebi et al., 2007). No constraints are active for this solution. The best solution of this problem obtained using the HS algorithm is compared with solutions reported by other researchers, showing that the result of HS is the same as the best known solution in the literature: $9257.07 by Lagrangian Relaxation (Guo et al., 1996); $9267.20 by GA (Song & Xuan, 1998); $9452.20 by ant colony search algorithm (Song et al., 1999); $9257.07 by improved GA (Su & Chiang, 2004).
Comparison between the results obtained by the HS method and those generated with other (evolutionary and mathematical programming) techniques reported in the literature clearly demonstrate that the HS method is practical and valid for CHPED applications.

3. Conclusions

This study reviews recent applications of the music-inspired HS algorithm, such as music composition, Sudoku puzzle solving, structural design, ecological conservation, aquifer

parameter identification, soil slip determination, offshore structure mooring, and power economic dispatch.
As observed in most applications, the HS algorithm possesses a potential for obtaining good solutions in various optimization problems. Thus, the authors expect to see more successful applications in other scientific and engineering fields in near future. Also, theoretical progress in finding better solutions is expected.

4. Acknowledgements

The first author would like to thank many people who have helped in development of the HS algorithm, including Joel Donahue, Jack Galuardi, Teresa Giral, Kang Seok Lee, Audrey Leslie, Chung-Li Tseng, and Ronald Wiles.

5. References

Akin, A. (2007). *Optimum Design of Reinforced Concrete Continuous Beams using Harmony Search Algorithm*, PhD Progress Report, Engineering Sciences Department, Middle East Technical University, Ankara, Turkey

Ayvaz, M. T. (2007). Simultaneous Determination of Aquifer Parameters and Zone Structures with Fuzzy C-Means Clustering and Meta-Heuristic Harmony Search Algorithm. *Advances in Water Resources*, Vol. 30, No. 11, 2326-2338

Cheng, Y. M.; Li, L.; Lansivaara, T.; Chi, S. C. & Sun, Y. J. (2008). An Improved Harmony Search Minimization Algorithm Using Different Slip Surface Generation Methods for Slope Stability Analysis. *Engineering Optimization*, Vol. 40, No. 2, 95-115

Csuti, B.; Polasky, S.; Williams, P. H.; Pressey, R. L.; Camm, J. D.; Kershaw, M.; Kiester, A. R.; Downs, B.; Hamilton, R.; Huso, M. & Sahr, K. (1997). A Comparison of Reserve Selection Algorithms Using Data on Terrestrial Vertebrates in Oregon," *Biological Conservation*, Vol. 80, 83-97

Duggal, A. & Ryu, S. (2005). The Dynamics of Deepwater Offloading Buoys, In: *Fluid Structure Interactions and Moving Boundary Problems*, WIT Press, pp. 269-278

Erdal, F. (2007). *Optimum Design of Grillage Systems using Harmony Search Algorithm*, MSc Thesis, Engineering Sciences Department, Middle East Technical University, Ankara, Turkey

Fylling, I. (1997). Optimization of Deepwater Mooring Systems, *Proceedings of the Offshore Mediterranean Conference*, March 19-21, Ravenna, Italy

Fylling, I. & Kleiven, G. (2000). Integrated Optimized Design of Riser and Mooring System for Floating Production Vessels, *Proceedings of OMAE 2000 Conference*, February 14-16, New Orleans, USA.

Geem, Z. W. (2006a). Improved Harmony Search from Ensemble of Music Players. *Lecture Notes in Artificial Intelligence*, Vol. 4251, 86-93

Geem, Z. W. (2006b). Optimal Cost Design of Water Distribution Networks using Harmony Search. *Engineering Optimization*, Vol. 38, No. 3, 259-280

Geem, Z. W. (2007a). Novel Derivative of Harmony Search Algorithm for Discrete Design Variables. *Applied Mathematics and Computation*, doi:10.1016/j.amc.2007.09.049

Geem, Z. W. (2007b). Harmony Search Algorithm for Solving Sudoku. *Lecture Notes in Artificial Intelligence*, Vol. 4692, 371-378

Geem, Z. W. (2007c). Optimal Scheduling of Multiple Dam System Using Harmony Search Algorithm. *Lecture Notes in Computer Science*, Vol. 4507, 316-323

Geem, Z. W. & Choi, J. Y. (2007). Music Composition Using Harmony Search Algorithm. *Lecture Notes in Computer Science*, Vol. 4448, 593-600

Geem, Z. W.; Kim, J. H. & Loganathan, G. V. (2001). A New Heuristic Optimization Algorithm: Harmony Search. *Simulation*, Vol. 76, No. 2, 60-68

Geem, Z. W. & Williams, J. C. (2008). Ecological Optimization Using Harmony Search, *Proceedings of American Conference on Applied Mathematics* (MATH '08), Harvard University, March 24-26, Cambridge, MA, USA

Guo, T.; Henwood, M.I. & Ooijen, M. van (1996). An Algorithm for Combined Heat and Power Economic Dispatch, *IEEE Transactions on Power System*, Vol. 11, No. 4, 1778–1784

Lee, K. S. & Geem, Z. W. (2004). A New Structural Optimization Method Based on the Harmony Search Algorithm. *Computers & Structures*, Vol. 82, No. 9-10, 781-798

Lee, K. S. & Geem, Z. W. (2005). A New Meta-Heuristic Algorithm for Continuous Engineering Optimization: Harmony Search Theory and Practice. *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, No. 36-38, 3902-3933

Mahdavi, M. ; Fesanghary, M. & Damangir, E. (2007). An Improved Harmony Search Algorithm for Solving Optimization Problems. *Applied Mathematics and Computation*, Vol. 188, No. 2, 1567-1579

Morgenstern, N. R. & Price, V. E. (1965). The Analysis of the Stability of General Slip Surfaces, *Geotechnique*, Vol. 15, 79-93

Omran, M. G. H. & Mahdavi, M. (2007). Global-Best Harmony Search. *Applied Mathematics and Computation*, doi:10.1016/j.amc.2007.09.004

ReVelle, C. S.; Williams, J. C. & Boland, J. J. (2002), Counterpart Models in Facility Location Science and Reserve Selection Science," *Environmental Modelling and Assessment*, Vol. 7, 71-80

Ryu, S. ; Duggal, A. S. ; Heyl, C. N. & Geem, Z. W. (2007). Mooring Cost Optimization via Harmony Search, *Proceedings of the 26th ASME International Conference on Offshore Mechanics and Arctic Engineering* (OMAE 2007), June 2007, San Diego, CA, USA

Ryu, S.; Duggal, A. S.; Heyl, C. N. & Liu, Y. (2006). Prediction of Deepwater Oil Offloading Buoy Response and Experimental Validation, *International Journal of Offshore and Polar Engineering*, Vol. 16, No. 4, 290-296

Saka, M. P. (2007). Optimum Geometry Design of Geodesic Domes Using Harmony Search Algorithm. *Advances in Structural Engineering*, Vol. 10, No. 6, 595-606

Song, Y. H.; Chou, C. S. & Stonham, T. J. (1999). Combined Heat and Power Economic Dispatch by Improved Ant Colony Search Algorithm, *Electric Power Systems Research*, Vol. 52, 115–121

Song, Y. H. & Xuan, Q. Y. (1998). Combined Heat and Power Economic Dispatch using Genetic Algorithm Based Penalty Function Method, *Electric Machines and Power Systems*, Vol. 26, 363–372

Su, C. T. & Chiang, C. L. (2004). An Incorporated Algorithm for Combined Heat and Power Economic Dispatch, *Electric Power Systems Research*, Vol. 69, 187–195

Tsai, F. T. –C. ; Sun, N. Z. & Yeh, W. W. G. (2003). A Combinatorial Optimization Scheme for Parameter Structure Identification in Ground-Water Modeling. *Groundwater*, Vol. 41, No. 2, 156-169

Vasebi, A. ; Fesanghary, M. & Bathaeea, S. M. T. (2007). Combined Heat and Power Economic Dispatch by Harmony Search Algorithm. *International Journal of Electrical Power & Energy Systems*, Vol. 29, No. 10, 713-719

Zolfaghari, A. R.; Heath, A. C. & McCombie, P. F. (2005). Simple Genetic Algorithm Search for Critical Non-Circular Failure Surface in Slope Stability Analysis. *Computers and Geotechnics*, Vol. 32, 139-152

A Hybrid Evolutionary Algorithm and its Application to Parameter Identification of Rolling Elements Bearings

Eric Yonghan Kim[1], Bo-Suk Yang[2] and Andy Chit Chow Tan[1]
[1] Queensland University of Technology, [2] Pukyong National University
[1] Australia, [2] South Korea

1. Introduction

Genetic algorithms (GAs) are powerful stochastic search techniques and are the most widely known types of evolutionary algorithms (EAs). This method performs a search by evolving a population of candidate solutions through the use of non-deterministic operators and by improving incrementally the individuals forming the population by mechanisms inspired from those of genetics (e.g. crossover and mutation). They are known to offer significant advantages over traditional methods by using simultaneously several search principles and heuristics, of which the most important ones are: population-wide search, continuous balance between exploitation (convergence) and exploration (maintained diversity) and the principle of building-block combination. However, GA can suffer from excessively slow convergence before providing an accurate solution. This is because of its fundamental requirement of using minimal prior knowledge without exploiting local information. Since the introduction of global search algorithms in engineering applications, many modified versions of GA have been reported to reduce the searching time and to raise the global search capability. Many researchers have proposed improved versions of GA which GA operator works adaptively (Wu et al., 1999; He et al., 2001; Fung et al., 2002). A local search or meta-heuristic algorithm has been incorporated into GA to improve the algorithm (Renders & Flasse, 1996; Berger et al., 1999; Lee et al., 2001; Hsiao et al., 2001; Hagenman et al., 2003; Jiang et al., 2003). The combined GA-SA algorithm has been introduced to improve the efficiency of the global search (Roach & Nagi, 1996; Yu et al., 2000; Ong et al., 2002; Liu et al., 2002; Ponnambalam et al., 2003).
In the first half of this chapter, a new hybrid evolutionary algorithm known as clustering-based hybrid evolutionary algorithm (CHEA) is introduced (Kim et al., 2006). This algorithm utilizes the GA's grouping property which involves gathering a number of individuals around the global candidate according to the generation. Clustering of individuals using artificial neural network (ANN) is incorporated into the GA to evaluate the stage of maturity of genetic evolution and to deal with statistical data of each cluster. After clustering, a local search is carried out for each cluster to accelerate the convergence process and to judge the convexity of each cluster. Finally, an efficient random search is adapted for searching the potential global candidate which may be missed in GA and local search. The efficiency of the proposed algorithm is then verified by applying it to three well-

known benchmark functions namely banana function, multi-modal function and Rastrigin function.

The dynamic behavior of a rotating shaft is significantly influenced by the stiffness and damping characteristics of the bearings. The precise values of stiffness and damping coefficients are difficult to predict. In the past decade, many works have dealt with identification of bearing coefficients using impulse or synchronous/non-synchronous excitation techniques (Burrows & Stanway, 1977; Kraus et al., 1987), and using mathematical formulations using an out-of-unbalance response (Lee & Hong, 1988; Chen & Lee, 1995, 1997). Other researches used the least square method as an optimizer to minimize the error between the measured unbalance response and the estimated one after they have formulated the minimization problem (Edwards et al., 2000; Reddy et al., 2002 and Tiwari et al., 2002). Least square method with sensitivity-based approach is a very effective algorithm that can be used for parameter identification of machinery characteristics. However, the application of least square optimizer cannot guarantee a global minimum, which means the identified parameters may not be the optimum ones for the real rotor-bearing systems which are often influenced by noises or non-linear effects.

Recently, global optimization schemes such as GA and simulated annealing (SA) (Kirpatrick et al., 1983) have been used in the area of parameter identification. These schemes do not involve gradient information and mathematical formulation but require only forward analysis procedure. Unfortunately identification approach based on global optimization algorithms is a highly time consuming task because it is based on the iterative strategy which updates unknown parameters systematically using an analytical output. Therefore, a fast and more efficient search algorithm is required for parameter identification in line with the rapid progress of computer technology.

In the latter half of this chapter, we introduce a method of using a hybrid evolutionary algorithm for parameter identification of ball bearings (Kim et al., 2007). The identification method utilises the hybrid evolutionary algorithm. The capability of the technique is verified using a numerical example and a series of experimentation on a tests. The results reveal that the proposed method can identify not only unknown bearing parameters but also unbalance information of disks. In contrast to other traditional identification techniques, the method can be applied with simple formulation of an optimisation problem using the existing dynamic analysis procedure without any complex mathematical approach.

2. Clustering-based hybrid evolutionary algorithm (CHEA)

The CHEA is a hybrid GA which is combined with neural-network, local search and random search. The flowchart of CHEA process is shown in Fig. 1. The first task is *GA-clustering,* in which GA is combined with the clustering process by using neural network. In this task, all individuals after each generation of GA are classified into several clusters until all individuals are well classified. After *GA-clustering*, the local search (LS) is carried out for each cluster with their best individuals. If all final points of the local search converged close to one point, this point implies a global candidate. This means that, graphically, the objective function is a kind of convex, which has only one global/local minimum in the search space. If all final points do not converged to one point, the objective function is considered to be a multi-modal function, which has many local minima. In this case,

additional local searches are carried out which starts at several random points within each group to determine whether each cluster has only one local minimum or not. Similarly, considering one cluster, if the final points by the local search are nearly the same point, this cluster has one local minimum, which implies the objective function is a convex for the region of this cluster. Otherwise the clusters have many local minima in their regions. In this case, GA is run again with reduced bounds as those of each cluster. The classification and the local search procedure are executed until each cluster has only one local minimum. Finally, an efficient random search is adopted for extra-searching to find the potential global candidate which may be missed in GA and local search.
Adaptive resonance theory-Kohonen neural network (ART-KNN) developed by (Yang et al., 2004) is incorporated for clustering of individuals after each generation in GA. Sequential quadratic programming (SQP) is adopted for the task of local search in this algorithm.

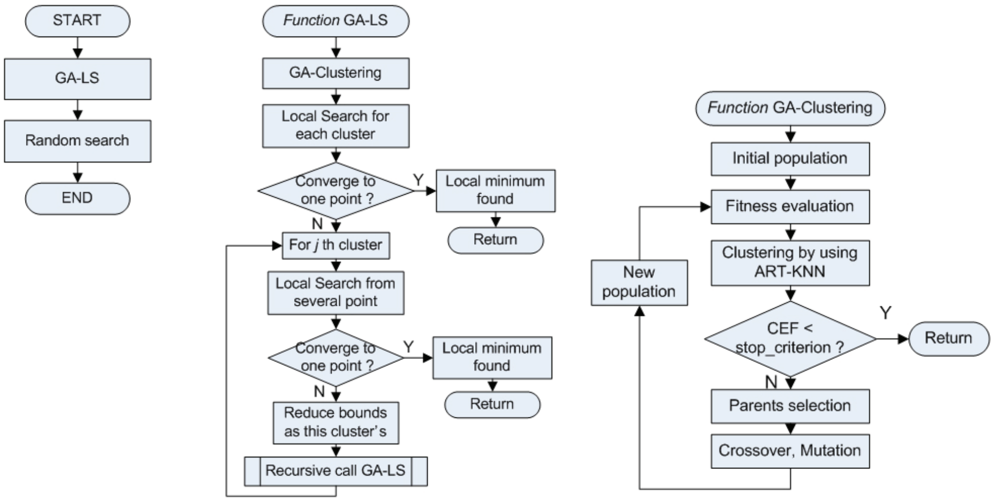


Fig. 1. Flowchart of CHEA

2.1 GA-clustering task

GA improves the genes of individuals based on evolutionary operation. Geometrically, the evolution of GA is that increasing individuals are gathered together around the global or local minimum with respect to the increase of generation as shown in Fig. 2. Generally GA is not efficient for improving the precision of best individuals to global minimum after gathering around the global minimum. However, the ability to gather individuals to a global or local minimum in the first several generations is excellent. Therefore, the proposed hybrid algorithm intends to use the merit of GA and to prevent inefficient calculations after the individuals have gathered around the global or local minimum.
If the objective function is a multi-modal function which has more than two local minimums, clustering or classification of individuals are necessary to divide them into several clusters as shown in Fig. 2(b) and requires a stop criterion for GA.. The clustering evaluation function (*CEF*) is introduced to evaluate the stage of maturity of individuals in each generation. *CEF* is defined by eq (1) using statistical data of each classified cluster:

$$CEF = \sum_{j=1}^{M} \frac{\boldsymbol{\delta}_j \gamma_j}{M \alpha_j \beta_j} \tag{1}$$

$$\boldsymbol{\delta}_j = \sum_{i=1}^{N_j} \left\| \mathbf{v}_{ij} - \mathbf{v}_{0j} \right\|^2, \quad \alpha_j = \left(\frac{N_j}{\sum N_j} \right)^{w_g}, \quad \beta_j = \left(\left\| \mathbf{v}_{0j} - \mathbf{v}_{0all} \right\|_{\min} \right)^{w_m}, \quad \gamma_j = (1-\rho)^{w_r}$$

where, $\mathbf{v}_{ij}$ denotes the ith vector for the jth cluster and $\mathbf{v}_{0j}$ is the center of the jth cluster. w_g, w_m, w_r are weight factors for cluster, distance of each group and similarity by ART-KNN, respectively. $\|\ \|$ denotes Euclidian distance between two vectors. M denotes the number of cluster and N_j is the number of individual for the jth cluster.

In this study, well matured is defined when the average distance from the mid point of each cluster approaches a small value and the average distance among mid points approaches a large value. *CEF* value for *stop criterion* is very important because it is directly related to the efficiency of the search algorithm. GA stopped with a too high *CEF* implies that the GA evolution is not matured yet and individuals may be classified into too many clusters. On the contrast, with a too small *CEF*, most individuals will migrate to only one cluster which contains the best individual. This may lead to lose of useful information about local minimum. Furthermore, if the number of individuals is not sufficient to find all the local minima, most individuals will move to a local minimum. In our study with trying many kind of test functions, the best *stop criterion* is selected as 0.2.

As shown in the flowchart of GA-clustering task in Fig. 1, ART-KNN algorithm was adapted as the traditional GA procedure to classify individuals into several clusters after the evaluation of fitness. After clustering, it is judged whether all individuals are well matured by using the *CEF*. If the *CEF* is smaller than the *stop criterion*, subroutine *GA-clustering* is terminated and returns to the final individuals and provides cluster information to the main program. Otherwise, the general procedure of GA, such as selection, crossover and mutation, is preceded again.

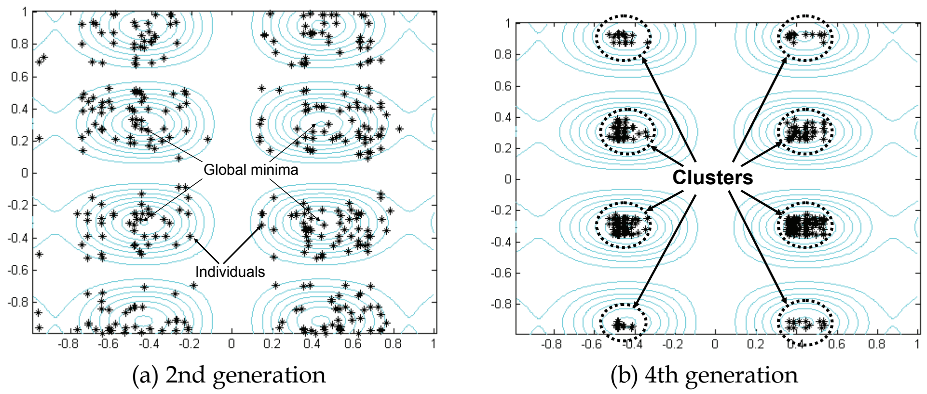


Fig. 2. Distribution of individuals according to generations

2.2 ART-KNN algorithm

The adaptive resonance theory (ART) network (Carpenter & Grossberg, 1988) is a neural network that self-organizes stable recognition codes in real time in response to arbitrary sequences of input patterns. It is also a vector classifier based on mathematical model for the description of fundamental behavioral functions of the biological brain such as the learning, parallel and distributed information storage, short and long-term memory and pattern recognition. The Kohonen neural network (KNN) (Kohonen, 1995) is also called self-organizing feature map network (SOFM). It defines a feed forward two-layer neural network that implements a characteristic non-linear projection from the high dimensional space of sensory or other input signals onto a low-dimensional array of neurons.

Recently, Yang et al. proposed a new algorithm using the adaptive resonance theory-Kohonen neural network (ART-KNN) (Yang et al., 2004), which does not affect the initial training and can adapt with additional training data. The structure of ART-KNN is shown in Fig. 3. It is similar to ART's but excluding the adaptive filter. ART-KNN is formed by two major subsystems: the attentional subsystem and the orienting subsystem. There are two interconnected layers, discernment layer and comparison layer, which are fully connected with both bottom-up and top-down processes and comprise of the attentional subsystem. The application of a single input vector leads to several patterns of neural activity in both layers. The activity in discernment nodes reinforces the activity in comparison nodes due to top-down connections. The interchange of bottom-up and top-down information leads to a resonance in neural activity. As a result, critical features comparison is reinforced with those having the greatest activity. The orienting subsystem is responsible for generating a reset signal to discernment when the bottom-up input pattern and top-down template pattern do not match during comparison process according to a similarity law. In other words, once it has detected that the input pattern is novel, the orienting subsystem must prevent the previously organized category neurons in discernment from learning this pattern (via a reset signal). Otherwise, the category will become increasingly non-specific. When a mismatch is detected, the network adapts its structure by immediately storing the novelty with additional weights. The similarity criterion is set by the value of the similarity parameter. A high value of the similarity parameter means than only a slight mismatch will be tolerated before a reset signal is emitted. On the other hand, a small value means that large mismatches will be tolerated. After the resonance check, if a pattern match is detected according to the similarity parameter, the network changes the weights of the winning node.

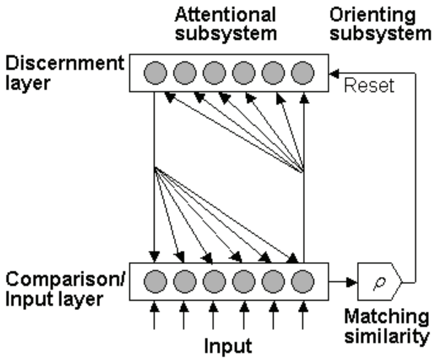


Fig. 3. Structure of ART-KNN

2.3 Clustering by ART-KNN

In the ART-KNN, the determination of a limiting value of similarity (ρ) is important in the optimization problem because the classification result is dependent on ρ. *CEF* detailed in the previous section is used to evaluate the superiority of the classified results based on average distance from mid point of each cluster and the variance of each cluster.

ART-KNN is modified and incorporated into GA procedure for the clustering process according to following sequence:

Step 1: Normalize every individuals of GA from 0 ~ 1.0.

Step 2: Change similarity ρ from 0.4~1.0.

- Classify into clusters using ART-KNN for each ρ.
- Calculate the *CEF* for each ρ.

Step 3: Choose clustering results which correspond to minimum *CEF.*

2.4 Sequential quadratic programming (SQP)

SQP method represents the state of the art in nonlinear programming methods. Schittkowski (Schittkowski, 1985) has implemented and tested a version that outperforms every other tested methods in terms of efficiency, accuracy and percentage of successful solutions over a large number of test problems. Based on the work of Powell (Powell, 1978), the method allows it to closely mimic Newton's method for constrained optimization similar to an unconstrained optimization. At each major iteration, an approximation is made of the Hessian of the Lagrangian function using a quasi-Newton updating method. This is then used to generate a QP sub-problem whose solution is used to form a search direction for a line search procedure. An overview of SQP can be seen in Fletcher (Fletcher, 1980). The general method is not listed here, but MATLAB program provides a full implementation together with the SQP algorithm.

2.5 Efficient random search

The last procedure of CHEA is a complementary random search to find a global minimum candidate, which may be missed in GA and LS procedure. Considering the valley of global minimum is highly narrow and deep as shown in Fig. 4, general stochastic global search algorithms, such as GA and SA, often fail to find the global minimum. This is because not only we use limited number of trials to find the global minimum but heuristics reduce the searching area toward the global candidate which has a relatively wide valley. The mutation operator in GA gives a part of this random search by changing the genes randomly, but it doesn't use previous search history at all. Therefore, this paper proposes an efficient random search method, which uses all previous search points. It works by generating a new search point as far as possible from all previous search points. In the stochastic viewpoint, this random search increases the probability of finding the global minimum.

The steps of the proposed efficient random search are as follows:

Step 1: Generate 5 search points randomly.

Step 2: Calculate Euclidean distance of the nearest point among previous search points.

Step 3: Select one point which has the largest Euclidean distance.

Step 4: Calculate fitness from the objective function.

If the calculated fitness is smaller than the best local minimums from GA-LS,

Step 5: Apply local search using the SQP.

Step 6: Else, go to step 1, repeat the above procedure until the maximum number of iterations is reached.

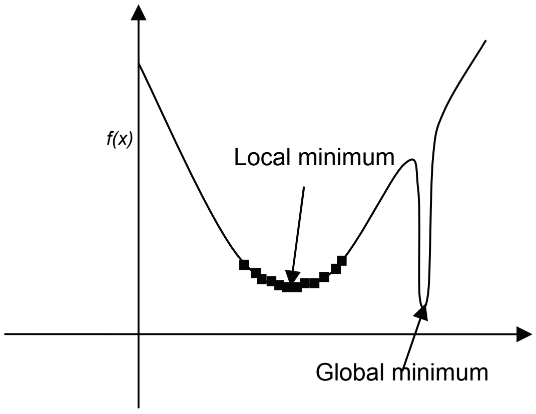


Fig. 4. An objective function which has narrow and deep global minimum

3. Application to test functions

The new optimization algorithm was tested by using several benchmark functions to evaluate its capability and to compare it with other algorithms. Many types of test functions have been used to this subject, however in this study, the three well-known test functions were used to evaluate the algorithm.

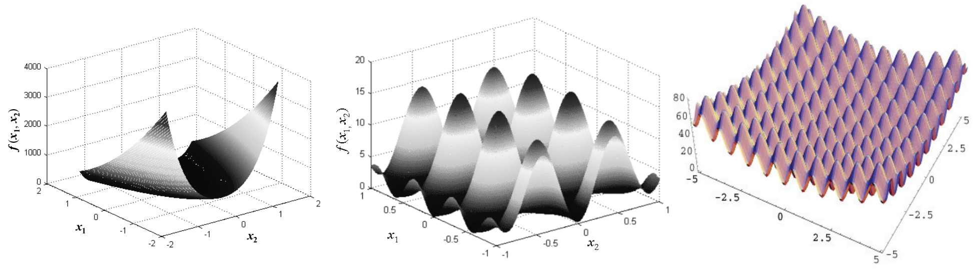
- Test function 1: Banana function which has one global minimum and converges slowly to the global minimum.
- Test function 2: Multi-modal function which has several global minima and several local minima
- Test function 3: Rastrigin function which contains one global minimum and many local minima

$$f_1(x_1,x_2)=100(x_1-x_2)^2+(1-x_1)^2, \quad (-2.0\le x_1,x_2\le 2.0) \tag{2}$$

$$f_2(x_1,x_2) = (\cos 2\pi x_1+\cos 2.5\pi x_1-2.1)\times(2.1-\cos 3\pi x_2-\cos 3.5\pi x_2) \\ (-1.0\le x_1,x_2\le 1.0) \tag{3}$$

$$f_3(x_1,x_2)=2\times 10+\sum_{i=1}^{2}\{x_i^2-10\cos(2\pi i)\} \quad (-5.0\le x_1,x_2\le 5.0) \tag{4}$$

Test function 1, known as a Banana function, has the shape shown in Fig. 5 (a). In general, the convergence speed of an evolution program for this function is very slow and the accuracy of the searched solution is low as well. The objective of this example is to find the variable *x*, which minimizes the objective function. This function has only one optimum solution ($x_1 = 1.0$, $x_2 = 1.0$) at $f(x) = 0$. It is difficult to find the optimum solution because of a valley phenomenon. In general, an objective function which has several global minima and/or local optimum points is called the multi-modal function as shown in Fig. 5 (b). The objective of this test function is to maximize the objective function. This function has four local minima of $f(x)$=14.333087 and four global minima of $f(x)$=16.09172. The Rastrigin function defined in eq. (4) is often used to evaluate the global search capability because there are many local minima around the global minimum as shown in Fig. 5 (c). It is very difficult to find a global minimum within the limited function in this test function. The objective of this test function is to minimize a function. This function has 220 local minima and one global minimum $f(x)$=0 at (0,0).



(a) Banana function (b) Multi-modal function (c) Rastrigin function

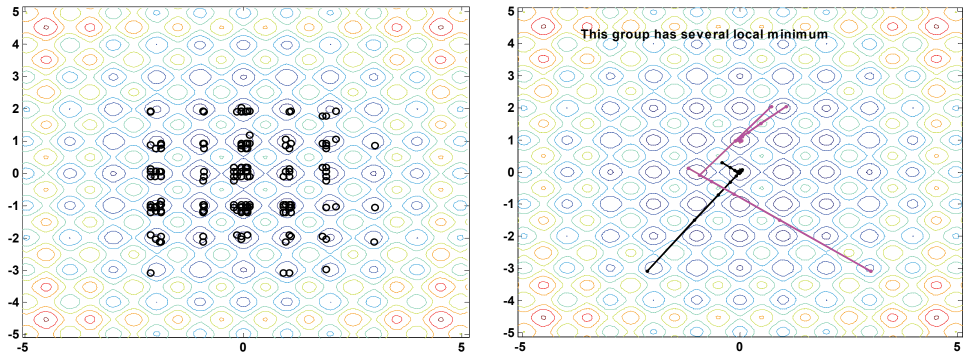
Fig. 5. Benchmark test functions

The convergence speed of the optimization algorithm is evaluated by using test function 1. The ability of searching several global minima simultaneously is evaluated by using test function 2. The global search capability among many local minima is finally evaluated by using test function 3. Table 1 shows the parameters of CHEA used in this paper.

GA	Length of chromosome	12
	Number of population	200
	Crossover probability	40%
	Mutation probability	$0.8\exp(-i_G/2)$, i_G: ith generation
Clustering	CEF	0.2
	w_g	0.9
	w_m	1.5
	w_r	0.9
Random search	Max iteration	400

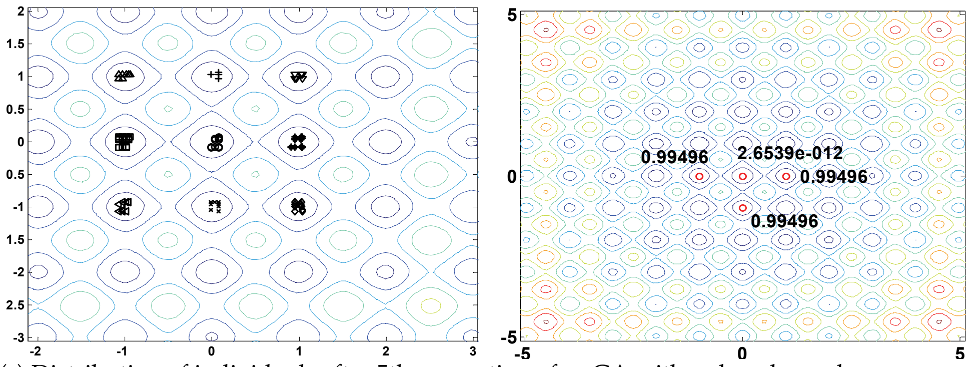
Table 1. Parameters for CHEA

To observe the searching procedure of CHEA, the gradual process of CHEA for Rastrigin function is shown in Fig. 6. GA was terminated in one cluster after six generations as shown in Fig. 6 (a). After *GA-clustering* process, local search was carried out with four randomly selected individuals from each cluster. Since the results of the local search did not converge to a point, CHEA considered this cluster to have many local minima as shown in Fig. 6 (b). Therefore, GA-clustering task was repeated with reduced search bounds. After five generations, all individuals were well clustered as shown in Fig. 6(c) where the GA was terminated. After a local search for each cluster, CHEA produced a global minimum and three local minima as shown in Fig. 6(d). No better global candidate was found during random search.



(a) Distribution of individuals after 4th generation of GA

(b) Local search by SQP with individuals randomly selected



(c) Distribution of individuals after 5th generation of re-GA with reduced search area

(d) Final results by CHEA

Fig. 6. Optimization procedure by CHEA for test function 3

4. Comparison of performance of CHEA

Optimization results by CHEA are compared with EGA (Kim, 2003) and ASA (Ingber & Rosen, 1992) which are known as the advanced version of GA and SA. Table 2 shows the comparison for test function 1. The second column indicates the total number of function call which also represents computation time. Third to fifth columns show the mean optimum values of the design variables and the final value of the objective function, respectively. The result using ASA did not converge well to an optimum value though it spent more computation times than those of CHEA. EGA gave the exact optimum value but took 3183 number of function calls as compared to CHEA which took 1120 functional calls.
The results for test function 2 are shown in Table 3. All algorithms showed the results having similar resolution, but ASA produced only one global minimum as compared with the others which found four global minima. EGA was slower than ASA but found all the global minima. The table shows that CHEA found all global minima and with the smallest number of function call.

Finally, the test results for test function 3 are summarized in Table 4. All algorithms found the local minima, but they often failed to find the global minimum. The last column shows the percentage of success in finding the global minimum. EGA produced the worst results in terms of computation time and success ratio. CHEA although is slower than ASA but the success ratio to global minimum is more superior. Considering the convergence speed, accuracy of results and global search capability, CHEA is found to be the most efficient algorithm among the considered algorithms which are known to be efficient and fast.

	No. of function call	x_1	x_2	f(x)
ASA	1414	0.6995	0.4878	0.0905
EGA	3183	0.9999	1.0000	6.06e-19
CHEA	1120	1.0000	1.0000	9.94e-13

Table 2. Comparison of the results for the test function 1

	No. of function call	x_1	x_2	f(x)
ASA	1391	0.43881	–0.30585	16.09172
EGA	3014	–0.43880	–0.30585	16.09172
		–0.43880	–0.30585	16.09172
		–0.43880	–0.30585	16.09172
		0.43880	0.30585	16.09172
CHEA	1131	–0.43880	–0.30585	16.09172
		–0.43880	–0.30585	16.09172
		–0.43881	–0.30585	16.09172
		0.43881	0.30585	16.09172

Table 3. Comparison of the results for the test function 2

	No. of function call	x_1	x_2	f(x)	Success to global (%)
ASA	1336	1.82e-5	–2.55e-6	–2.55e-7	83
EGA	3131	1.00e-20	1.00e-20	3.16e-13	85
CHEA	2100	–5.72e-9	–2.81e-7	1.57e-11	99
		0.994	4.63e-8	0.994	
		–0.994	9.23e-9	0.994	
		8.69e-7	0.994	0.994	

Table 4. Comparison of the results for the test function 3

5. Unbalance response analysis of rotating shaft

In this study, the vibrations are calculated using general finite element procedures. Since the finite element discretization procedure is well documented in many literatures (Nelson, 1980; Pilkey, 1994; Choi & Yang, 2000), the details are omitted here and only the equations of motions are presented below.

5.1 Disk element

The rigid disk is modeled as a four degrees of freedom rigid body with the generalized coordinates defined as two translations *V, W* of the mass center in the *X* and *Y* directions and two rotations *B* and *Γ* of the plane of the disk about the *X* and *Y* axes. The rigid disk needs to be located at a finite element station. If the spin speed *Ω* is assumed to be constant then the coordinates $\mathbf{q}^d$ are governed by the following equation.

$$\left(\mathbf{M}_T^d + \mathbf{M}_R^d\right) \ddot{\mathbf{q}}^d - \Omega \mathbf{G}^d \dot{\mathbf{q}}^d = \mathbf{F}^d \tag{5}$$

where $\mathbf{M}_T^d$, $\mathbf{M}_R^d$ are the translational and rotational mass matrices respectively, $\mathbf{G}^d$ is the gyroscopic matrix and $\mathbf{F}^d$ is the force vector acting on the disk.

5.2 Shaft element

The shaft element is considered to be initially straight and modeled as an eight degrees of freedom element: two translations and two rotations at each station of the element. The cross-section of the element is taken to be circular and uniform. Continuous shaft mass with a constant density is taken as equivalent lumped mass. The inertia of each element is divided into two parts and applied at both ends of an element.
The equation of motion, in fixed frame and for a shaft element rotating with a constant speed *Ω* are given by,

$$\left(\mathbf{M}_T^e + \mathbf{M}_R^e\right) \ddot{\mathbf{q}}^e - \Omega \mathbf{G}^e \dot{\mathbf{q}}^e + \mathbf{K}^e \mathbf{q}^e = \mathbf{F}^e \tag{6}$$

Here $\mathbf{q}^e$ is a (8×1) displacement vector, corresponding to the translational and rotational displacements (*V, W, B, Γ*) at both ends of the element. $\mathbf{M}_T^e$, $\mathbf{M}_R^e$ are the translational and rotational mass matrices respectively, $\mathbf{G}^e$ is the gyroscopic matrix, $\mathbf{K}^e$ is the stiffness matrix and $\mathbf{F}^e$ is the force vector acting on the shaft element.

5.3 Bearing elements

The nonlinear characteristics of the bearings can be linearized at the static equilibrium position using the assumption of a small vibration. The dynamic characteristics of the bearings are represented by eight stiffness and damping coefficients. The force acting on the shaft can be expressed as

$$\mathbf{C}^b \dot{\mathbf{q}}^b + \mathbf{K}^b \mathbf{q}^b = \mathbf{F}^b \tag{7}$$

where $\mathbf{C}^b$ and $\mathbf{K}^b$ are the damping and stiffness matrices of the bearing elements, respectively.

5.4 Assembly and system equation

Once equations (5) - (7) are established for a typical element, these equations are repeatedly used to generate other equations recursively for other elements. Then they are assembled to find the global equation, which describes the behavior of the entire system. The assembled damped system equation of motion in the fixed frame is of the form

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{F} \tag{8}$$

where, $\mathbf{M} = \mathbf{M}^d + \mathbf{M}^e$, $\mathbf{K} = \mathbf{K}^e + \mathbf{K}^b$, $\mathbf{C} = \mathbf{C}^b - \Omega\mathbf{G}^e - \Omega\mathbf{G}^d$. **M**, **C** and **K** are total mass matrix, damping matrix and stiffness matrix, respectively. **F** is the external force vector acting on the entire system.

5.5 Steady-state unbalance response

In fixed frame coordinates, the unbalance force in eqn. (8) is of the form

$$\mathbf{F} = \mathbf{F}_C \cos \Omega t + \mathbf{F}_S \sin \Omega t \tag{9}$$

The steady state solution is given by,

$$\mathbf{q} = \mathbf{q}_C \cos \Omega t + \mathbf{q}_S \sin \Omega t \tag{10}$$

Substituted eqns. (9) and (10) into (8) yields

$$\begin{bmatrix} \mathbf{q}_C \\ \mathbf{q}_S \end{bmatrix} = \begin{bmatrix} \mathbf{K} - \Omega^2\mathbf{M} & -\Omega\mathbf{C} \\ \Omega\mathbf{C} & \mathbf{K} - \Omega^2\mathbf{M} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{F}_C \\ \mathbf{F}_S \end{bmatrix} \tag{11}$$

The solution of eqn. (11) and substituting back into eqn. (10) provides the system unbalance response.

6. Optimization formulation for identification

6.1 General identification procedure

Fig. 6 shows the general identification procedure for determining the unknown system parameters, such as bearing parameters, position, magnitude and phase of unbalance of rotor-bearing system. It consists of different tasks as shown in Fig. 6. At first, a linear analytical model which is generally described by a differential equation is formulated by including unknown parameters. And then, steady-state unbalance response can be calculated by using the equations described in previous section. Such a response can also be obtained from the measurements of output signals in rotor-bearing system. Finally, in the comparison task, the analytical response is compared with the measured response at the same nodes. If their correlation is poor, the system unknown parameters are renewed and sent to the analytical model. This iterative procedure for improving the system unknown parameters is set if the correlation of model and measurement is good enough. The key issue of this procedure is how much variations of parameters have to be given to the new analytical model. It is very time consuming to do this manually. Thus many optimization techniques have been developed to solve this kind of problem which can be formulated as minimization problem.

6.2 Formulation of optimization problem

The classical nonlinear constrained optimization problem can be written mathematically as:

$$\text{Minimize } f(\mathbf{x}) \tag{12}$$

$$\text{Subject to } g_l(\mathrm{x}) \le 0 \ (l=1, m),\ h_k(\mathrm{x}) = 0 \ (k = 1, n),\ x_i^l \le x_i \le x_i^u \ (i = 1, p) \tag{13}$$

In general, the objective function $f(\mathbf{x})$ as well as the constraint functions $g_l(\mathbf{x})$ and $h_k(\mathbf{x})$ are nonlinear implicit functions with respect to the design variables. Classical optimization algorithms require these functions to be unimodal and continuous, and their first derivatives have to be available. Otherwise, various numerical difficulties and convergence problems may arise. The global optimization algorithms, such as GA and SA, have been developed in order to overcome the above restrictions and difficulties.

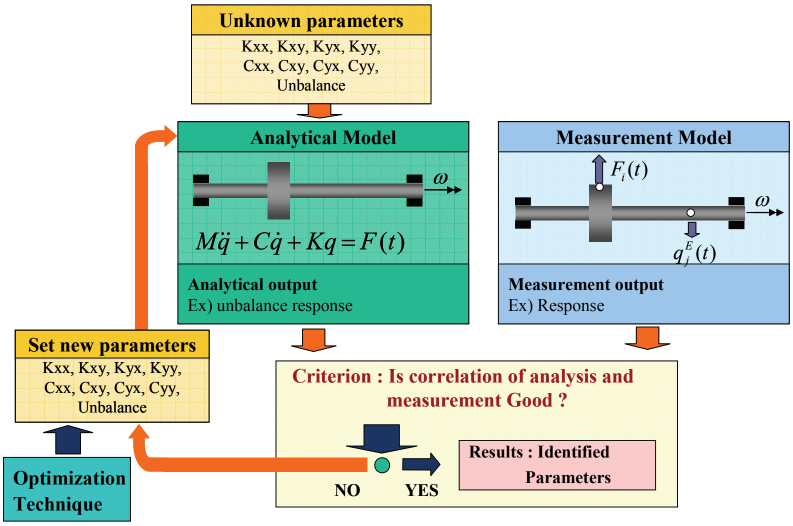


Fig. 6. General identification procedure using optimization technique

It is important to choose the form of the objective function, $f(\mathbf{x})$, in engineering application of optimization algorithms. Three different types of objective functions are considered as shown in equations (14) to (16). The sum-squared difference between the magnitude of the experimental and analytical unbalance responses, as shown in equation (14), is a common choice, but this function performs rather badly in certain practical applications, especially in low damping system. The reasons for this failure are due to the function being dominated by the contributions made at the critical speed and resonant peaks. Another possible approach is to consider the difference of the natural logarithm of the unbalance responses to reduce the weighting of the natural frequencies defined in equation (15). A simple difference function, shown in equation (16), can also be used as an objective function.

$$f_1(\mathbf{x}) = \sum_j \sum_\Omega \left(U_j^X(\Omega,\mathbf{x}) - U_j^A(\Omega,\mathbf{x}) \right)^2 \tag{14}$$

$$f_2(\mathbf{x}) = \sum_j \sum_\Omega \left| \log_{10} U_j^X(\Omega,\mathbf{x}) - \log_{10} U_j^A(\Omega,\mathbf{x}) \right| \tag{15}$$

$$f_3(\mathbf{x}) = \sum_j \sum_\Omega \left| U_j^X(\Omega,\mathbf{x}) - U_j^A(\Omega,\mathbf{x}) \right| \tag{16}$$

where, U denotes the unbalance response and superscripts X and A represent measured and analytical responses, respectively. Ω is the rotating speed of the shaft, j is the measuring node and **x** is the identifying parameter vector.
The optimization problem for parameters identification of rotor-bearing system is formulated as follows:

$$\text{Minimize } f(\mathbf{x})$$

$$\text{subject to: } x_i^l \le x_i \le x_i^u,\ x_i \in \mathbf{x},\ x_i = 1, 2, \ldots, 5 \qquad (17)$$

and the design variables: $\mathbf{x} = (k_{xx}, k_{xy}, k_{yx}, k_{yy}, c_{xx}, c_{xy}, c_{yx}, c_{yy}, u)$

where, x_i is the design variable and superscripts l and u represent the lower and upper bounds of the design variables, respectively. k_{ij}, c_{ij} ($i, j = x, y$) are the stiffness coefficients and damping coefficients of bearing respectively. Subscript x and y denote horizontal and vertical direction, respectively. u denotes the residual unbalance of the disk.
In this study, only the diagonal terms of the stiffness and damping coefficients (k_{xx}, k_{yy}, c_{xx}, c_{yy}) are considered and does not consider inequality or equality constraints. When a journal bearing is used in the rotor-bearing system, cross-coupled terms of stiffness and damping coefficients (k_{xy}, k_{yx}, c_{xy}, c_{yx}) need to be selected as design variables.

7. Numerical application

The proposed methodology is first verified by a simulation study. A simple rotor-bearing model is shown in Fig. 7 and detail specifications of the rotor bearing model are shown in Table 5. The rotor system consists of a shaft of 1.3m in length and 0.1m in diameter, and has three disks. Two bearings support the shaft at the each ends. The dynamic coefficients of the two bearings are of the same values, and hence only the diagonal terms are considered. An unbalance mass was added on disk 2 (6th node) with a magnitude of 200 g·mm and an angle of 0°. The unbalance responses at the 2nd and the 12th nodes were selected as simulated measured responses. To consider the uncertainty of the analytical model and to examine the robustness of identification, 10% of Gaussian noise was applied to the simulated responses.
The stiffness and damping coefficients of the bearing and the magnitude of unbalance mass on disk were chosen as identifying parameters. The formulation of optimization is described in the following section.

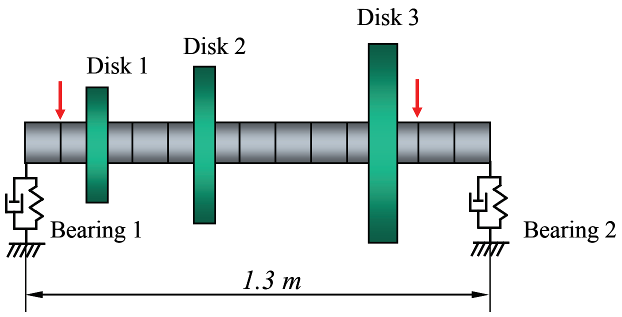


Fig. 7. Rotor bearing model (Lalanne and Ferraris, 1998)

Shaft	Shaft length (m)	1.3
	Shaft diameter (m)	0.1
	Young's modulus (GPa)	200
	Density (kg/m3)	7,800
	Poisson ratio	0.3
Bearing	k_{xx}, k_{yy} (MN/m)	50, 70
	c_{xx}, c_{yy} (kN·s/m)	0.5, 0.7
	k_{xy}, k_{yx}, c_{xy}, c_{yx}	0

Table 5. Model parameters in Lalanne's rotor model

7.1 Formulation of optimization

Objective function

$$f_1(\mathbf{x}) = \sum_{j=v,h} \sum_{\Omega} \left(U_j^X(\Omega, \mathbf{x}) - U_j^A(\Omega, \mathbf{x}) \right)^2$$

$$\text{Minimize} \quad f_2(\mathbf{x}) = \sum_{j=v,h} \sum_{\Omega} \left| \log_{10} U_j^X(\Omega, \mathbf{x}) - \log_{10} U_j^A(\Omega, \mathbf{x}) \right| \tag{18}$$

$$f_3(\mathbf{x}) = \sum_{j=v,h} \sum_{\Omega} \left| U_j^X(\Omega, \mathbf{x}) - U_j^A(\Omega, \mathbf{x}) \right|$$

where, U_j is vertical and horizontal responses at 2nd and 12th nodes, respectively and Ω is rotating speed ranging from 200 to 15000 rpm with a step of 200 rpm.

Design variables (Identifying parameters)

$$\mathbf{x} = (k_{xx}, k_{yy}, c_{xx}, c_{yy}, u) \tag{19}$$

where, k_{xx} and k_{yy} are the horizontal and vertical stiffness coefficients, c_{xx} and c_{yy} are the horizontal and vertical damping coefficients, respectively. u is the magnitude of mass unbalance of disk.

Side constraints

$$10^2 \le k_{xx}, k_{yy} \le 10^9 \text{ (N/m)},\ 10^0 \le c_{xx}, c_{yy} \le 10^7 \text{ (N·s/m)},\ 10^{-7} \le u \le 10^{-2} \text{ (kg·m)} \tag{20}$$

The control parameters for this algorithm are listed in Table 6. These parameters are determined by considering the global search capability and the computation time.

GA	Length of chromosome	12
	Number of population	200
	Crossover probability	40%
	Mutation probability	$0.8\exp(-i_G/2)$, i_G : ith generation
Clustering	CEF	0.2
	w_g	0.9
	w_m	1.5
	w_r	0.9
Random search	Max iteration	500

Table 6. Control parameters for optimization algorithm (CHEA)

7.2 Identification results

Table 7 shows the identification results using the simulated unbalance response without noise. With the objective functions of all cases, all identified parameters have exactly the same reference values and the total call number of the objective function is about 3000. Fig. 8 shows the history of the objective function values. It can be seen that, after 6th generation, GA-clustering task was terminated and yielding the classification to one cluster. In a local search, three points converged to one point and consumed 1300 times of function evaluations. With a total of 500 trials of random searches the algorithms were unable to locate the lower local minimum candidate and the program had to be terminated. The result clearly shows that the shape of objective function needs to be a wide concave type.

Design variables	Reference values	Identified values		
		$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	$f_3(\mathbf{x})$
k_{xx} (MN/m)	50	50	50	50
k_{yy} (MN/m)	70	70	70	70
c_{xx} (kN·s/m)	0.5	0.5	0.5	0.5
c_{yy} (kN·s/m)	0.7	0.7	0.7	0.7
u (g·mm)	200	200	200	200
No. of function call		2,993	2,768	3,150

Table 7. Identification results using the unbalance response without noise

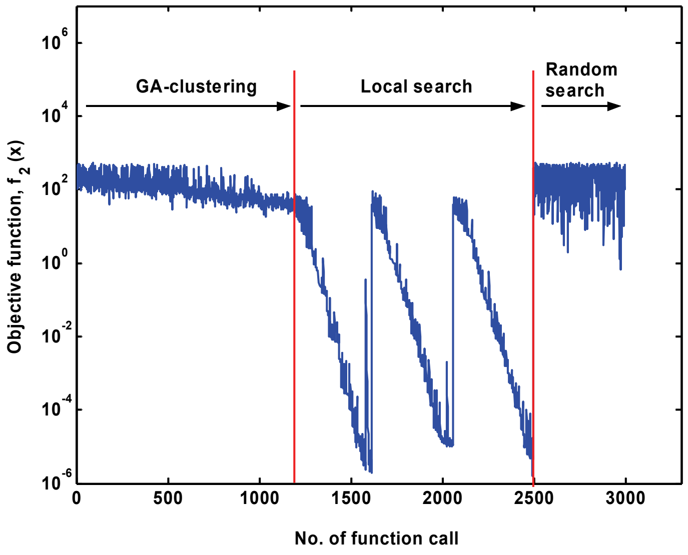


Fig. 8. History of objective function values

The identification results using a simulated response with 10% Gaussian noise added are summarized in Table 8, taking into consideration the three kinds of objective functions. In the case of function $f_1(\mathbf{x})$, the errors of stiffness coefficients varied from 2.4% to 8.1% and are less than 10%. However, the errors due to damping coefficients fluctuate significantly. The results by using function $f_3(\mathbf{x})$ are not acceptable due to the high errors encountered in stiffness coefficients, ranging from 14.8% to 160%. In the case of function $f_2(\mathbf{x})$, which is considered to be the best choice, the stiffness coefficients and magnitude of mass unbalance (u) are well identified with error less than 1% with respect to the reference values. This is obtained by excluding the relative higher errors of damping coefficients. The reasons for the poor results with respect to the damping coefficients are

- The damping coefficients of the bearing strongly affect the magnitude of the unbalance response near the resonant peaks in a low damping system.
- The peak value of the response fluctuates to the higher values than other responses due to the Gaussian noise.

Design variables	Reference value	Objective function (% error)		
		$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	$f_3(\mathbf{x})$
k_{xx} (MN/m)	50	45.94 (8.1)	50.16 (0.3)	112.1 (124)
k_{yy} (MN/m)	70	71.67 (2.4)	69.94 (0.1)	80.38 (14.8)
c_{xx} (kN·s/m)	0.5	2.570 (414)	0.434 (13.8)	0.852 (160)
c_{yy} (kN·s/m)	0.7	0.0015 (99)	0.684 (4.1)	0.834 (19)
u (g·mm)	200	210.4 (5.2)	200.8 (0.3)	115.8 (42)

Table 8. Comparison of identification results for different objective functions in the case 10% Gaussian noise added to unbalance response

From these results, the objective function needs to be selected carefully by considering the shape of the measured response function. Fig. 9 shows the simulated unbalance responses with 10% Gaussian noise added and the calculated unbalance responses using the identified parameters for the case function $f_2(\mathbf{x})$. The identified response is in good agreement with the simulated measured ones.

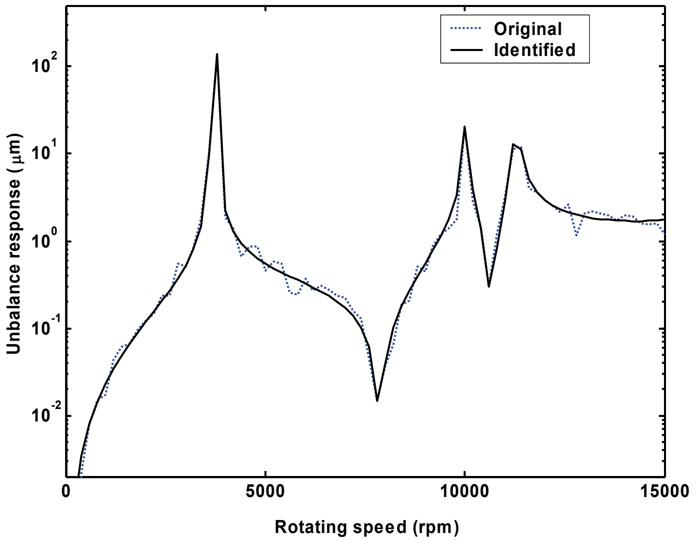


Fig. 9. Original and identified unbalance response

8. Experimental validation

The experimental validation was performed to verify the effectiveness of proposed identification approach. By using a Rotor-Kit system, the stiffness coefficients and unbalance mass of disk are identified simultaneously. The identified results are compared with those obtained by measurement.

8.1 Test rig and measured response

The test rig for experimental validation is shown in Fig. 10. The rotor-system is the RK4 model manufactured by Bently-Nevada. A flexible coupling connects a controllable DC motor to the shaft. Spring-bearing, which has four springs for driving a ball bearing in all directions as shown in Fig. 10, was used to identify the stiffness and damping coefficients. The adjoined two ball bearings in the coupling side are used to prevent slight angular movements which usually occurred in single ball bearing setup. Two proximity probes are incorporated to measure the shaft vibration in the vertical and horizontal directions.

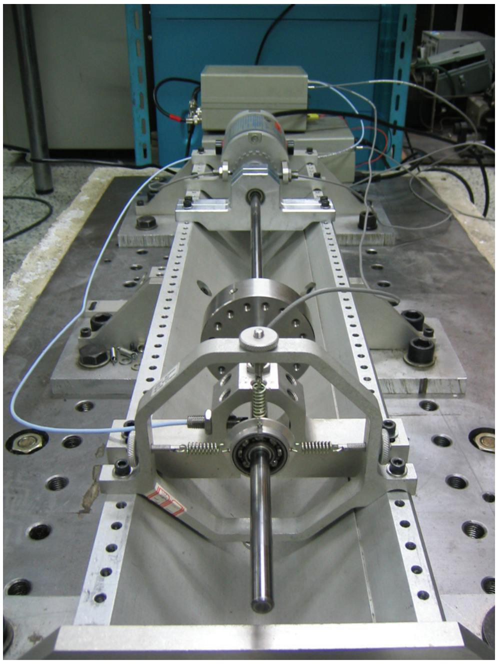


Fig. 10. Experimental test rig

Fig. 11 shows the schematic of the test setup with the spring-bearing. The measured signal was processed by using the DAI-108 and ADRE software. The stiffness of the two adjoined ball bearings in the left side was considered to be rigid because it was significantly greater than the identifying stiffness of the spring-bearing at the right side. The parameters of the shaft, disk and spring-bearing are listed in Table 9. To identify the unknown parameters in a real system, all the other parameters need to be defined. Therefore, Young's modulus and density of shaft listed in Table 5 were updated by using the model updating technique.

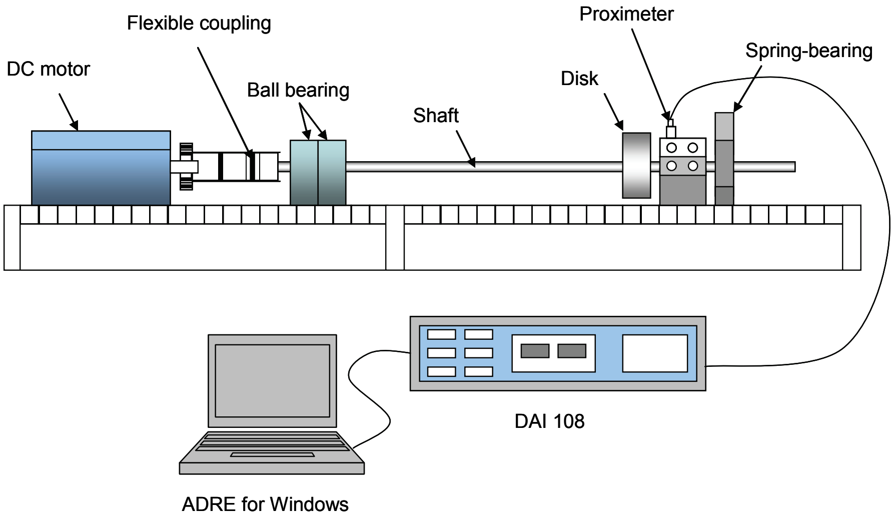


Fig. 11. Schematic of an experimental setup with a spring-bearing

Shaft	Length (mm)	560
	Diameter (mm)	10
	Density (kg/m3)	7,801
	Young's modulus (MPa)	208.11
	Poisson ratio	0.3
Disk	Mass (kg)	0.809
	Polar moment of inertia (kg ·m2)	568.46×10-6
	Trans. moment of inertia (kg ·m2)	327.60×10-6
	Magnitude of unbalance (g ·mm)	15
Bearing	Bearing span (mm)	401
	Horizontal stiffness (kN/m)	33.9
	Vertical stiffness (kN/m)	33.6

Table 9. Parameters of test setup

Fig. 12 shows a 1X filtered measured response of horizontal vibration according to speed-up and speed-down of the motor. Slow roll vector at 500 rpm was used to compensate the original signal. The response below the critical speed was used in the identification process because they increased sharply near the critical speed. In actual fact, many rotating systems operate below the first critical speed. The reason why the measured signal is not smooth enough is because this system has no damping mechanism except internal material damping or friction. Fig. 13 shows, for example, an instantaneous measured signal in the vertical direction at a shaft speed of 1350 rpm. The first peak in the spectrum plot indicates the rotating speed and the second peak is the first natural frequency of the system. This appearance is frequently shown in low damping systems supported by ball bearings. Furthermore, traditional deterministic identification approaches (Lee & Hong, 1988; Chen & Lee, 1995, 1997; Tiwari et al., 2002) often failed to identify the exact parameters.

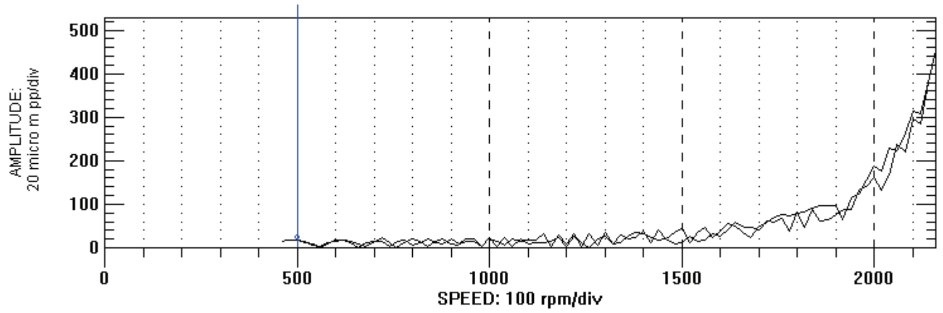
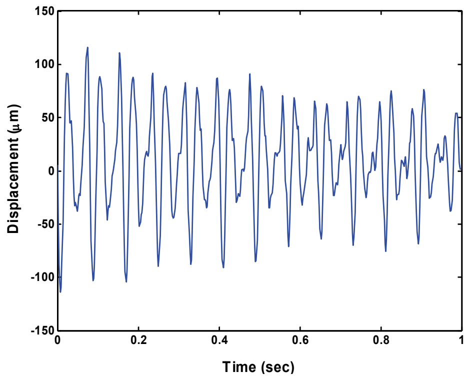
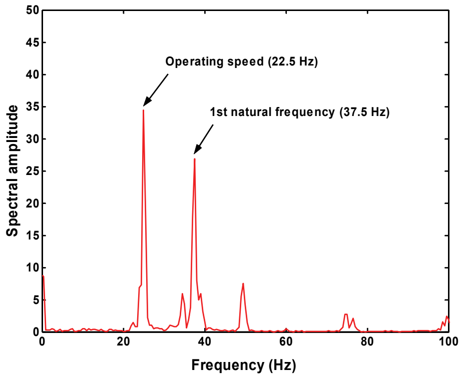


Fig. 12. 1X filtered measured horizontal response



(a) Time base signal



(b) Spectrum

Fig. 13 Instantaneous vibration signal and its spectrum at 1350 rpm

8.2 Optimization formulation and results

The same control parameters for optimization algorithm listed in Table 2 are used this case. By using the above measured responses in the vertical and horizontal directions, optimization for identifying the bearing parameters and unbalance is formulated as follows:

Objective function:

$$f_1(\mathbf{x}) = \sum_j \sum_{\Omega} \left(U_j^X(\Omega, \mathbf{x}) - U_j^A(\Omega, \mathbf{x}) \right)^2$$

$$\text{Minimize } f_2(\mathbf{x}) = \sum_j \sum_{\Omega} |\log_{10} U_j^X(\Omega, \mathbf{x}) - \log_{10} U_j^A(\Omega, \mathbf{x})| \tag{20}$$

$$f_3(\mathbf{x}) = \sum_j \sum_{\Omega} \left| U_j^X(\Omega, \mathbf{x}) - U_j^A(\Omega, \mathbf{x}) \right|$$

where, U_j is response at the position of sensors and Ω is the rotating speed from 480 to 2140 rpm with a step of 20 rpm.

Design variables (Identifying parameters):

$$\mathbf{x} = (k_{xx}, k_{yy}, c_{xx}, c_{yy}, u) \tag{21}$$

where, k_{xx} and k_{yy} are the horizontal and vertical stiffness coefficients, c_{xx} and c_{yy} are the horizontal and vertical damping coefficients, respectively. u is the magnitude of mass unbalance of disk.

Side constraints:

$$10^2 \le k_{xx}, k_{yy} \le 10^6 \text{ (N/m)},\ 10^0 \le c_{xx}, c_{yy} \le 10^3 \text{ (N·s/m)},\ 10^{-7} \le u \le 10^{-3} \text{ (kg·m)}$$

The identification results for the spring-bearing system are summarized in Table 10. The results show an average function call number of 4327 which corresponds to a computation CPU time of 3519 second on the P-IV 3.0 GHz PC. The reference values for the stiffness coefficients were obtained from static deflection tests. The percent error of identified parameters to reference values is given in terms of percentage error (% error).

Design variables	Experimental value	Identified values (% error)		
		$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	$f_3(\mathbf{x})$
k_{xx} (kN/m)	33.900	30.884 (8.9)	30.796 (9.1)	33.491 (1.2)
k_{yy} (kN/m)	34.600	34.203 (1.1)	34.001 (1.7)	36.390 (5.2)
c_{xx} (N·s/m)	–	13.42	11.96	15.44
c_{yy} (N·s/m)	–	16.06	14.11	3.16
u (g·mm)	15	13.82 (7.8)	12.86 (14.3)	16.13 (7.5)
No. of total function call		4,334	4,360	4,288

Table 10. Identification results for the spring-bearing system

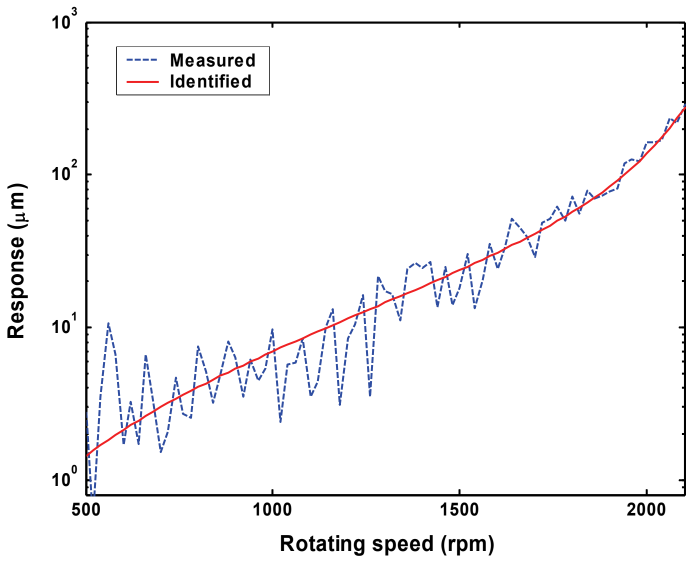


Fig. 14. Measured and Identified horizontal unbalance response for $f_3(\mathbf{x})$

Although the 1X amplitude of the measured signal had significant fluctuation, all the identified parameters fitted well with the reference values. Considering the percentage error to reference values, as shown in the Table 10, the best choice of the objective function is $f_3(\mathbf{x})$, which is the sum of differences between measured and analytical responses. Fig. 14 shows the identified horizontal unbalance response and 1X filtered measured response at the sensor positions. From this result, it is verified that the proposed methodology could be effectively used to identify bearing coefficients with the magnitude of unbalance using the measured unbalance responses.

9. Conclusions

A new hybrid evolutionary algorithm using clustering-based hybrid evolutionary algorithm (CHEA), is proposed in this chapter. The main feature of CHEA is the clustering of individuals introduced for evaluating the degree of maturity of genetic evolution. After the clustering-based genetic algorithm, local search is carried for each cluster in this algorithm. CHEA attempts to find each local minimum from each cluster or continues with GA focusing on the regions of each cluster until all significant local minima are found. Therefore CHEA can lead to local minima as well as global minimum. ART-Kohonen neural network (ART-KNN) is used in the clustering of individuals in GA. Sequential quadratic programming (SQP) is adopted as local search. An efficient random search is introduced for improving the probability of finding the global minimum which may be missed by GA or local search task. The effectiveness of the proposed algorithm was evaluated using three well-known benchmark functions. The results showed that the CHEA reached the global minimum faster than EGA and ASA. It has the ability to find the global minimum as well as the local minima and having higher global search capability than other algorithms.
When using CHEA for parameter identification of bearings, it optimizes the formulation process to achieve an optimum solution. It minimizes the differences between analytical unbalance responses and measured ones by considering the unknown bearing parameters as design variables. Three types of feasible objective functions were applied in evaluation process, namely, sum-squared differences, logarithmic differences and simple differences to find the most competent formulation of the objective function. The magnitude of mass unbalance was also chosen as identifying parameters. Numerical and experimental applications were presented to confirm the effectiveness of this methodology. In the numerical application, 10% of Gaussian noise was added to simulate measured response and to examine the robustness of the methodology. The results showed that the unknown parameters were correctly identified and the logarithmic differences function was concluded as the best objective function in the numerical simulation. When applied to an experimental rotor-bearing system the measured synchronous response fluctuates according to the rotating speeds but the identified parameters fitted well with the reference values. This new algorithm has the potential for use in real life applications. However, further investigations using industrial data are required to test the robustness of the technique before applying the method to industrial rotating machinery.

10. Acknowledgement

The work is partially supported by the CRC for Integrated Engineering Asset Management (CIEAM), Australia.

11. References

Berger, J., Sassi, M. and Salois, M. (1999) A hybrid genetic algorithm for the vehicle routing problem with time windows and itinerary constraints, in Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, USA, pp. 44—51

Burrows, C. R. & Stanway, R. (1977) Identification of journal bearing characteristics. ASME J Dynamic System Measurement and Control, Vol. 99, pp. 167-175

Carpenter GA, Grossberg S (1988) The art of adaptive pattern recognition by a self-organizing neural network. IEEE Trans on Computer, Vol 21, No. 3, pp. 77-88

Chen, J. H. & Lee, A. C. (1995) Estimation of linearized dynamic characteristics of bearings using synchronous response. International Journal of Mechanical Science, Vol. 37, No.2, pp. 197-219

Chen, J. H. & Lee, A. C. (1997) Identification of linearised dynamic coefficients of rolling element bearings. ASME Journal of Vibration and Acoustics, Vol. 119, pp. 60-69

Choi, B. G. & Yang, B. S. (2000) Optimum shape design of shaft using genetic algorithm. J of Vib and Control 6(1): 207- 220

Edwards, S.; Lees, A. W. & Friswell, M. I. (2000) Experimental identification of excitation and support parameters of a flexible rotor- bearings-foundation system from a single run-down. Journal of Sound and Vibration, Vol. 232, No. 5, pp. 963-992

Fletcher, R. (1980) Practical methods of optimization, Constrained Optimization. John Wiley and Sons

Fung, R. Y. K., Tang, J. and Wang, D. (2002) Extension of a hybrid genetic algorithm for nonlinear programming problems with equality and inequality constrains, Computers & Operations Research, Vol. 29, No. 3, pp. 261—274

Hageman, J. A., Wehrens, R., Sprang, H. A. and Buydens, L. M. C. (2003) Hybrid genetic algorithm-tabu search approach for optimizing multilayer optical coatings, Analytica Chimica Acta, Vol. 490, pp. 211—222

He, D., Li, Y. and Wang, F. (2001) Hybrid genetic algorithm based on the operator of pattern search, Information and Control, Vol. 30, No. 3, pp. 276—278

Hsiao, C. T., Chahine, G. and Gumerov, N. (2001) Application of a hybrid genetic algorithm/Powell algorithm and a boundary element method to electrical impedance topography, Journal of Computational Physics, 173, 433-454

Ingber, L. and Rosen, B. (1992) Genetic algorithms and very fast simulated re-annealing: a comparison, Mathematic Computational Modeling, Vol. 16, pp. 87—100

Jiang, Z., Liu, B., Dai, L. and Wu, T. (2003) A hybrid genetic algorithm integrated with sequential linear programming, in Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xian, pp. 1030—1033

Kim, Y. C. (2003) Development of Enhanced Genetic Algorithm and Its Applications to Optimum Design of Rotating Machinery, Ph.D. Dissertation, Pukyong National University, South Korea

Kim, Y. H., Yang, B. S. and Tan, A. C. C. (2006) Clustering-based Hybrid Evolutionary Algorithm for Optimization, Advances in vibration engineering, Vol. 5, No. 2, pp. 163-173

Kim, Y. H., Yang, B. S. and Tan, A. C. C. (2007) Bearing Parameter Identification of Rotor-Bearing System Using Clustering-based Hybrid Evolutionary Algorithm, Structural and Multidisciplinary Optimization, Vol. 33, No. 6, pp. 493-506

Kirpatrick, S. C. D. & Gelatt, M. P. (1983) Optimization by simulated annealing. Science Vol. 220, No. 4598, pp. 671–680

Kohonen, T. (1995) Self-Organizing Maps, New York: Springer-Verlag

Kraus, J.; Blech, J. J. & Braun, S. G. (1987) In situ determination of rolling bearing stiffness and damping by modal analysis. ASME J Vibration, Acoustics, Stress, and Reliability in Design, Vol. 109, pp. 235-240

Lee, C. W. & Hong, S. W. (1988) Identification of bearing dynamic coefficients by unbalance response measurements. Proceedings of Institution of Mechanical Engineers Vol. 203C, pp. 93-101

Lee, C. Y., Gen, M. and Kuo, W. (2001) Reliability optimization design using hybridized genetic algorithm with a neural network technique, IEICE Trans. on Fundamental, Vol. E84-A, No. 2, pp. 627—637

Liu, Y., Ma, L. and Zhang, J. (2002) Reactive power optimization by GA/SA/TS combined algorithms, Electrical Power and Energy Systems, Vol. 24, pp. 765—769, 2002

Nelson, H. D. (1980) A finite rotating shaft element using Timoshenko beam theory. ASME, J Mechanical Design 102: 793-803

Ong, S. K., Ding, J. and Nee, A. Y. C. (2002) Hybrid GA and SA dynamic set-up planning optimization, International Journal of Production Research, Vol. 40, No. 18, pp. 4697—4719

Pilkey, W. D. (1994) Formulus for stress, strain, and structural matrices. John Wiley, New York, USA

Ponnambalam, S. G. and. Reddy, M. M. (2003) A GA-SA multiobjective hybrid search algorithm for integrating lot sizing and sequencing in flow-line scheduling, International Journal of Advanced Manufacturing Technology, Vol. 21, pp. 126—137

Powell, M. J. D. (1978) A fast algorithm for nonlinearly constrained optimization calculations. Numerical Analysis, G.A.Watson ed., Lecture Notes in Mathematics, Springer Verlag, 630

Reddy, V. B.; Tiwari, R. & Kakoty, S. K. (2002) Identification of bearing dynamic parameters from impulse response of rotor bearing systems. Proceedings of VETOMAC-2

Renders, J. M. and Flasse, S. P. (1996) Hybrid methods using genetic algorithms for global optimization, IEEE Transactions on Systems, Man, and Cybernetics Part B, Vol. 26, No. 2, pp. 243—258

Roach, A. and Nagi, R. (1996) A hybrid GA-SA algorithm for just-in-time scheduling of multi-level assemblies, Computers Industrial Engineering, Vol. 30, No. 4, pp. 1047—1060

Schittkowski, K. (1985) NLQPL: A FORTRAN-subroutine solving constrained nonlinear programming problems. Annals of Operations Research 5: 485-500

Tiwari, R.; Lees, A. W. & Friswell, M. I. (2002) Identification of speed-dependent bearing parameters. Journal of Sound and Vibration, Vol. 254, No. 5, pp. 967-986

Wu, Z., Shao, H. and Wu, X. (1999) A new adaptive genetic algorithm & its application in multimodal function optimization, Control Theory and Applications, Vol. 16, No. 1, pp. 127—129

Yang, B. S.; Han, T. & An, J. L. (2004) ART-Kohenen neural network for fault diagnosis of rotating machinery. Mechanical System and Signal Processing, Vol. 18, pp. 645-657

Yu, H., Fang, H., Yao, P. and Yuan, Y. (2000) A combined genetic algorithm/simulated annealing algorithm for large scale system energy integration, Computers and Chemical Engineering, Vol. 24, pp. 2023—2035

Domain Decomposition Evolutionary Algorithm for Multi-Modal Function Optimization

Guangming Lin[1], Lishan Kang[2], Yongsheng Liang[1] and Yuping Chen[2]
[1]*Shenzhen Institute of Information Technology, Shenzhen 518029,*
[2]*School of Computer, China University of Geosciences, Wuhan,*
PRC.

1. Introduction

The Simple Genetic Algorithm (SGA) is applied more and more extensively since it was proposed by J. H. Holland [1] in 1970's. SGA is an optimization method based on population by emulating the evolvement disciplinarian of the nature. It has showed the great advantage of quick search for optimal solutions while applied in the optimization of single-modal functions. But as we all know many problems in reality belong to the optimization of multi-modal function, and if SGA is applied to solve this kind of problems, it has the confliction between the search space and convergence speed: the expansion of search space will slow down the convergence speed and the acceleration of convergence speed will reduce the search space, lead to early convergence and as a result stop research at some local optimal solutions.
Evolutionary algorithms have been used regularly to solve multi-modal function optimization problems, due to their population-based approach and their inherent parallelism, *e.g.* a crowding factor model proposed by De Jong[2], a shared-function model proposed by Goldberg and Richardson[3], an artificial immune system method, a split ring parallel evolutionary algorithm, etc., all of which have attempted to maintain the diversity of the population during the process of evolution. In this chapter, we introduce a new 'Domain Decomposition Evolutionary algorithm (called DDEA) which can solve not only simple nonlinear programming problems effectively and efficiently, but can also find the multiple solutions of multi-modal problems in a single run. The DDEA employs dual strategy approach that searches at two levels of detail (namely global then local). In the first (global) step, a Self-adaptive Mutations with Multi-parent Crossover Evolutionary Algorithm (SMMCEA)[4] is employed to perform a global search to divide the (chromosome) population into several subpopulations or niches in subdomains, which is domain decomposition. In the second (local) step, an evolutionary strategy-like algorithm is employed to perform a local search on each isolated niche independently. Then the best solutions of the multi-modal problem are exploited.
The remainder of the chapter is organized as follows. Section 2 introduces a Self-adaptive Mutations with Multi-parent Crossover Evolutionary Algorithm (SMMCEA); Section 3 introduces Domain Decomposition evolutionary algorithm (DDEA); Section 4 presents the successful results of applying DDEA to several challenging numerical multi-modal optimization problems; Section 5 concludes.

2. Introduction of SMMCEA

2.1 The Problem to Solve

The general non-linear programming (NLP) problem can be expressed in the following form:

$$\text{Minimize } f(X,Y)$$
$$s.t. \quad h_i(X,Y)=0 \quad i=1,2,\ldots,k_1, \quad g_j(X,Y)\le 0 \quad j=k_1+1, k_1+2,\ldots,k \tag{1}$$
$$X^{\text{lower}} \le X \le X^{\text{upper}}, \quad Y^{\text{lower}} \le Y \le Y^{\text{upper}}$$

where $X \in R^p$, $Y \in N^q$, and the objective function $f(X,Y)$, the equality constraints $h_i(X,Y)$ and the inequality constraints $g_j(X,Y)$ are usually nonlinear functions which include both real variable vector X and integer variable vector Y.

Denoting the domain $D = \{(X,Y) \mid X^{lower} \le X \le X^{upper}, \ Y^{lower} \le Y \le Y^{upper}\}$, we introduce the concept of a subspace V of the domain D. m points (X_j,Y_j), $j=1,2,\cdots,m$ in D are used to construct the subspace V, defined as :

$$V = \{(Xv,Yv) \in D \mid (Xv,Yv) = \textstyle\sum_{i=1}^{m} a_i (X_i, Y_i)\}$$

where ai is subject to $\sum_{i=1}^{m} a_i = 1$, $-0.5 \le a_i \le 1.5$.

Because we deal mainly with optimization problems which have real variables and INequality constraints, we assume $k_1 = 0$ and $q = 0$ in the expression (1).

Denoting $w_i(X) = \begin{cases} 0, & g_i(X) \le 0 \\ g_i(X), & \text{otherwise} \end{cases}$ and $W(X) = \sum_{i=1}^{k} W_i(X)$

Then problem (1) can be expressed as follows:

$$\text{Minimize } f(X) \qquad X \in D \tag{2}$$

Subject to

$$W(X)=0 \qquad X \in D$$

We define a Boolean function "*better*" as:

$$better(X_1, X_2) = \begin{cases} W(X_1) < W(X_2) & \text{TRUE} \\ W(X_1) > W(X_2) & \text{FALSE} \\ (W(X_1) = W(X_2)) \wedge (f(X_1) \le f(X_2)) & \text{TRUE} \\ (W(X_1) = W(X_2)) \wedge (f(X_1) > f(X_2)) & \text{FALSE} \end{cases}$$

If *better* (X_1, X_2) is TRUE, this means that the individual X_1 is "better" than the individual X_2.

2.2 Related Work

In 1999, Guo Tao proposed a multi-parent combinatorial search algorithm (GTA) for solving non-linear optimization problems in his PhD thesis [5]. Later it was developed as a kind of subspace stochastic search algorithm [6], that can be described as follows:

Guo Tao's Algorithm (GTA)

```
Begin
    initialize popln $P = \{X_1, X_2, \cdots, X_N\}$;  $X_i \in D$ since ($q = 0$ implies no integer variables)
        generation count $t := 0$;
        $X_{best} = \arg \underset{1 \le i \le N}{Min} f(X_i)$;
        $X_{worst} = \arg \underset{1 \le i \le N}{Max} f(X_i)$;
    while $abs(f(X_{best}) - f(X_{worst})) > \varepsilon$ do
            select randomly $m$ points $X_1', X_2', \cdots, X_m'$ from $P$ to form the subspace $V$;
            select randomly one point $X'$ from $V$;
              If better $(X', X_{worst})$ then $X_{worst} := X'$;
              $t := t + 1$;
              $X_{best} = \arg \underset{1 \le i \le N}{Min} f(X_i)$;
              $X_{worst} = \arg \underset{1 \le i \le N}{Max} f(X_i)$
    end do
        output $t$, $P$;
End
```

where N is the size of population P, $(m - 1)$ is the dimension of the subspace V (if the m points (vectors) that construct the subspace V are linearly independent), t is the number of generations, ε is the accuracy of solution. $X_{best} = \arg \underset{1 \le i \le N}{Min} f(X_i)$ means that X_{best} is the variable (individual) in X_i ($i=1, 2, \cdots, N$) that makes the function $f(X)$ have the smallest value. The sub-population in GTA is families which reproduce sexually through the number of m individuals randomly selected from P. The best individual in the sub-population takes part in competition to replace the worst individual in P, therefore the pressure of elimination through selection is minimum. There is no mutation operator, only using multi-parents crossover in GTA.

2.3 A self-adaptive evolutionary algorithm

Since Guo's algorithm deals mainly with continuous NLP problems with Inequality constraints, to make it a truly universal and robust algorithm for solving general NLP problems, we extend Guo's algorithm by adding to it the following improvements:

(1) Guo selected randomly only one candidate solution from the current subspace V. Although he used the concept of a subspace to describe his algorithm, he did not really use a subspace search, but rather a multi-parent crossover. Because he selected randomly only one individual in the subspace, this action would tend to ignore better solutions in the subspace, and hence influence negatively the quality of the result and the efficiency of the search. If however, we select randomly several individuals from the subspace, and substitute the best one for the worst one in the current population, the search should be better. So we replace the instruction line in Guo's algorithm:

"select randomly one point $X'$from V; "

with the two instruction lines:

" select randomly s points X_1^* , X_2^* , … , X_s^* from V;

$$X' = \arg \underset{1 \le i \le s}{Min} f(X_i^*);"$$

(2)The dimension m of the subspace in Guo's algorithm is fixed (i.e. m parents reproduce). The algorithm always selects a substitute solution in subspaces which have the same dimension, regardless of the characteristics of the solutions in the current population. Thus, when the population is close to the optimal value, the searching range is still large. This would apparently result in unnecessary computation, and affect the efficiency of the search. We can in fact reduce the search range, that is to say, the dimension of the subspaces. We therefore use subspaces with variable dimensions in the new algorithm, by adding the following instruction line to Guo's algorithm:

if $abs\ (f(X_{best}) - f(X_{worst})) \le \eta$ **.and.** $m \ge 3$ **then** $m := m - 1$;

where η depends on the computation accuracy ε, and $\eta > \varepsilon$. For example, if the computation accuracy $\varepsilon = 10^{-14,}$ then we can set $\eta = 10^{-2}$ or $10^{-3.}$

(3) We know in principle that Guo's algorithm can deal with problems containing EQuality constraints. For example, we can use the device of setting two INequality constraints $0 \le h_i(X, Y)$ and $h_i(X, Y) \le 0$ to replace the equality constraint $h_i(X, Y) = 0$, but the experimental results when employing this device are not ideal. However, equality constraints are likely to exist in real-world problems, so we should find methods to deal with them. One such method is to define a new function $W(X, Y)$

Where $W(X, Y) = \sum_{i=1}^{k} W_i(X, Y)$

$$W_i(X,Y) = \begin{cases} |h_i(X,Y)|, & i = 1,2,\cdots,k_i \\ \max\{o, g_i(X,Y)\}, & i = k_1 + 1, k_1 + 2, \cdots, k. \end{cases}$$

(4) The penalty factor r is usually fixed. However, some people use it as a variable, such as Cello[7], who employed a self-adaptive penalty function, but his procedure was rather complex (using two populations). We also make r a variable namely $r = r\ (t)$, where t is the iteration count. It can self-adjust according to the reflection information, so we label it a "self-adaptive penalty operator". Since the constraints have been normalized, r is relative only to the range of the objective function, which ensures a balance between the errors of the fitness function and the objective function, in order of magnitude.

(5) Guo's algorithm can deal only with continuous optimization problems. It cannot deal directly with integer or mixed integer NLP problems. In our algorithm, when we are confronted with such problems, we need only replace the integer variables derived from the

range of the float of the fitness function with "*integer function*" *int*(Y^*), where *int*(Y^*) is defined as the integer part of Y^*. No other changes to the algorithm are needed.

(6) The only genetic operator used in Guo's algorithm was crossover. However, we can add self –adaptive mutations in it, we introduce a better of Gaussian and Cauchy mutation operator into the subspace search. For Gaussian density function f_G with expectation 0; and variance σ^2 is

$$f_G = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} , \qquad -\infty < x < +\infty$$

For Cauchy density function f_C with scale parameter $t>0$ is,

$$f_C = \frac{1}{\pi} \frac{1}{t^2 + x^2} , \qquad -\infty < x < +\infty$$

2.4 A Self-adaptive mutations with multi-parent crossover evolutionary algorithm

Considering the above points, we introduce a new algorithm as follows:

Denoting $Z = (X, Y^*)$, where $Z \in D^*$, and

$D^* = \{(X, Y^*) \mid X_{lower} \le X \le X_{upper},\ Y_{lower} \le Y^* \le Y_u,\ X \in R^p,\ Y^* \in R^q\}$, we define integer vector Y=int(Y^*), where $Y_u = Y_{upper}+0.999\cdots9I$

Denoting $W(Z) = W(X, \text{int}(Y^*))$,

we define the Boolean function "*better*" as follows:

$$better(Z_1, Z_2) = \begin{cases} W(X_1) < W(X_2) & \text{TRUE} \\ W(X_1) > W(X_2) & \text{FALSE} \\ (W(X_1) = W(X_2)) \wedge (f(X_1) \le f(X_2)) & \text{TRUE} \\ (W(X_1) = W(X_2)) \wedge (f(X_1) > f(X_2)) & \text{FALSE} \end{cases}$$

The general NLP problem (1) can be expressed as follows:

$$\text{Minimize } f(X, \text{int}(Y^*)) \quad \text{in } D^* \quad \text{S.t.} \qquad (3)$$

$$W(Z)=0 , \qquad Z \in D^*$$

The new algorithm can now be described as follows:

SMMCEA :

```
    Begin
        initialize P = {$Z_1, Z_2, \ldots, Z_N$}; $Z_i \in D^*$;
        $t$ := 0;
        $Z_{best} = \arg\min_{1\le i\le N} f(Z_i)$;
        $Z_{worst} = \arg\underset{1\le i\le N}{Max} f(Z_i)$;
        while not abs ( $F(Z_{best}) - F(Z_{worst})$) $\le\varepsilon$ do
            select randomly $M$ points $Z_1', Z_2',\dots, Z_M'$ from $P$ to form the subspace $V$;
            select $s$ points randomly $Z_1^*, Z_2^* \dots Z_s^*$ from $V$;
            for i=1,…s do
                for j=1,…p+q do
                    $Z_{Gi}^*(j) := Z_i^*(j) + \sigma_i(j) N_j(0,1)$
                    $Z_{Ci}^*(j) := Z_i^*(j) + \sigma_i(j) C_j(1)$
                    $\sigma_i(j) := \sigma_i(j)\exp(\tau N(0,1) + \tau' N_j(0,1))$
                endfor
        if better($Z_{Gi}^*, Z_{Ci}^*$) then $Z_i'^* := Z_{Gi}^*$ else $Z_i'^* := Z_{Ci}^*$ ;
        endfor
        $Z' = \arg\underset{1\le i\le N}{Min} f(Z_i)$;
        if better ($Z', Z_{worst}$) then $Z_{worst} := Z'$;
        $t := t + 1$;
        $Z_{best} = \arg\underset{1\le i\le N}{Min} f(Z_i)$;
        $Z_{worst} = \arg\underset{1\le i\le N}{Max} f(Z_i)$;
        if abs ($f(Z_{best}) - f(Z_{worst})$) $\le\eta$ .and. $M\ge 3$ then
            $M := M - 1$;
    endwhile
    output $t$, $Z_{best}$, $f(Z_{best})$;
end
```

Where Z_{Gi}^*(j), Z_{Ci}^*(j) and σ_i (j) denote the j-th component of the vectors Z_{Gi}^*, Z_{Ci}^* and σ_i, respectively. N(0,1) denotes a normally distributed one-dimensional random number with mean zero and standard deviation one. $N_j(0, 1)$ indicates that the Gaussian random number is generated anew for each value of j. $C_j(1)$ denotes a Cauchy distributed one-dimensional random number with *t=1.*

The factors τ and τ' have commonly set to $\left(\sqrt{2\sqrt{(p+q)}}\right)^{-1}$ and $\left(\sqrt{2(p+q)}\right)^{-1}$.

The new algorithm has the two important features:

1. This algorithm is an ergodicity search. During the random search of the subspace, we employ a "non-convex combination" approach, that is, the coefficients a_i of $Z' = \sum_{i=1}^{m} a_i Z_i'$ are random numbers in the interval [-0.5, 1.5] This ensures a non-zero probability that any point in the solution space is searched. This ergodicity of the algorithm ensures that the optimum is not ignored.

2. The monotonic fitness decrease of the population (when the minimum is required). Each iteration ($t \to t+1$) of the algorithm discards only the individual having the worst fitness in the population. This ensures a monotonically decreasing trend of the values of objective function of the population, which ensures that each individual of the population will reach the optimum.

When we consider the population $P(0)$, $P(1)$, $P(2)$,…, $P(t)$,… as a Markov chain, we can prove the convergence of our new algorithm. See [12].

3. Introduction of DDEA

Experiments indicate that if SMMCEA is directly applied to the optimization of multi-modal function, it is easy to encounter the following two conditions:

1. If keep searching with relatively large population size and crossover size, the individuals of the population will spread around near different modals, but it's difficult for population to get any more improvement and to reach all the modals exactly.
2. If keep searching with relatively small population size and crossover size, the individuals of the population will converge rapidly and reach a few modals, but lose many other modals.

To adopt it to the optimization of multi-modal functions, we combine the above two conditions together and forms two-phase evolutionary algorithm. we divide the optimization procedure into two phases: the first phase is called global optimization, which keeps searching with relatively large population size and crossover size in order to determine the neighborhood of all modals; the second phase is called local optimization, which begins search from each of the neighborhoods which is determined by the global optimization and then keep searching with relatively small subpopulation size and crossover size in order to converge rapidly and reach the modals respectively.

In addition, we introduce the following strategies to make the algorithm suitable to the different tasks of the two phases:

1. During the phase of global optimization, in order to avoid the loss of some obtained modals we introduce the strategy of good individuals isolation: before each evolvement all the individuals in the current population are sorted by their fitness value and then some of the good individuals are limited not to be parents in the next multi-parent crossover.
2. During the phase of local optimization, in order to make all the subpopulations converge to their modals respectively more quickly, we introduce the strategy of best individual exemplar: the best individual of the current population will be compelled to be one of the parents in the next multi-parent crossover.
3. During the phase of local optimization, in order to begin search based on the result of the global optimization and to keep the search around the neighborhood of all the modals, to each modal we will construct a local feasible area η, which is to be modified during the evolvement.

The detailed procedures of the optimization DDEA are as the following:

Phase 1: Global optimization (using SMMCEA)

```
Randomly initialize population $P(0)= \{P_1,P_2,\ldots, P_{N1}\}$,Evaluate $P(0)$,$t_1=0$
while $t_1 < MAXT_1$ do
  randomly select $m_1$ parents from $P(t_1)$ with the strategy of good individuals isolation
  produce a child by multi-parent crossover and self-adaptive Gaussian and Cauchy
  mutation
    if the child is better than the worst individual of $P(t_1)$ then
          replace the worst individual of $P(t_1)$ with the child
          end if
          $t_1= t_1+1$
end while
```

Phase 2: Local optimization

```
      for k= 1 to $N_1$ do
         initialize local feasible area $\eta$, which is the rectangle area around $P_k$ with the
         radium r
         Randomly initialize subpopulation SUBP(0) within the area of $\eta$
                        $SUBP(0)=\{ SUBP_1, SUBP_2,\ldots, SUBP_{N2} \}$
         $t_2=0$
         while ($t_2 < MAXT_2$ and individuals of $SUBP(t_2)$ are different )do
               randomly select $m_2$ parents from $SUBP(t_2)$ with the strategy of the best
           individual exemplar
                  produce a child by multi-parent crossover
                  if the child $\in \eta$ and it is better than the worst individual of $SUBP(t_2)$
                  then replace the worst individual of $SUBP(t_2)$ with the child
                  end if
                        evaluate the best individual of $SUBP(t_2)$, which is named as
                        $SUBP_{best}$
                  modify local feasible area $\eta$, make it as the rectangle area around
                  $SUBP_{best}$
            with the radium r
                        $t_2= t_2+1$
    end while
    output the best individual of $SUBP(t_2)$
end for
```

The new algorithm employs a zoomed (global to local) dual strategy (two steps) approach. The first (global) step employs a global search, *i.e.* it divides the (chromosome) population into *L* (*L* ≤ *k*) niches, each of which includes at least one of the *k* optimal solutions (if the objective function is continuous in *D**). This step uses a SMMCEA [4]. If the number of parents *M* in the multi-parent recombination operator is large enough, for example, *M* ≥ 8, then after sufficient large generations the population is decomposed into subpopulations (each of which approaches to an optimal solution), else it will converge to only one solution [11].

The second (local) step employs an evolution strategy [13] to search for the local optima in the chosen *L* subspaces determined by the subpopulations. Since the *L* optimal solutions are located in separate subspaces, the local strategy consists of two sub-steps:

a). Rank the individuals of the population obtained from the first (global) step according to their fitness values. Then choose the best *L* individuals from the population, ensuring that they are not close to each other like hedgehogs.

b). Generate *L* subspaces with the chosen individual at the center of each. Search these niches locally until each subspace converges to an optimal solution. If one does not know how many optimal solutions a given problem has, one can predict the number *k*, for example, by using the number of individuals whose fitness values are larger than the average fitness value.

The algorithm has different limiting behaviors for different problems, namely:

a). When the problem has only $k = 1$ solution, *i.e.* the only globally optimal solution. Following the nature of population descent, all of the individuals will descend together to the bottom of the valley.

b). When the problem has $k > 1$ solutions, *i.e.* if $k \le N$, where *N* is the size of the population, *k* solutions may be generated in the population. The algorithm will then find multi-solutions in a single run.

4. Numerical experiments and analysis

Example 1 Humpback function (the function has six local optimal solutions, two of which are global optimal solutions)

$$\min f(x_1, x_2) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$$

where $x_1 \in [-3,3], x_2 \in [-2,2]$

Example 2 Typical function with many global optimal solutions(the function has increasing number of global optimal solutions while *j* is increased)

$$\min f(x_1, x_2) = 3 - (\sin(jx_1))^2 - (\sin(jx_2))^2,$$

where $x_1, x_2 \in [0,6], j = 1,2,\cdots$

Example 3 Absolute value function(the function has a plenty of local optimal solutions, 16 of which are global optimal solutions)

$$\min f(x_1, x_2) = \prod_{i=1}^{4} |x_1 - 3i| + \prod_{j=1}^{4} |x_2 - 4j|,$$

where $x_1 \in [0,13], x_2 \in [0,17]$

Example 4 N-dimension Shubert function[8](when n=2, the function has 720 local optimal solutions, 18 of which are global optimal solutions)

$$\min f(x_1, x_2, \cdots, x_n) = \prod_{i=1}^{n} \sum_{j=1}^{5} j \cos((j+1)x_i + j)$$

where $x_i \in [-10, 10], i = 1, 2, \cdots, n$

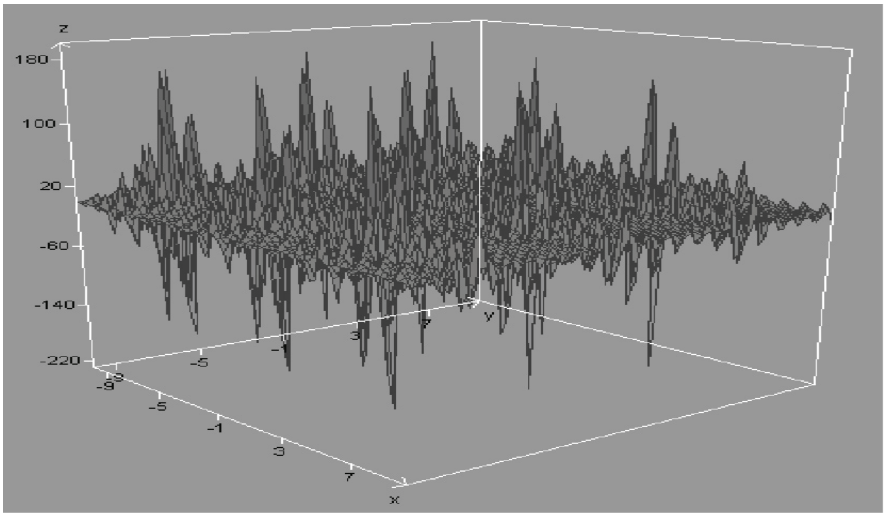


Fig. 1. Shubert function

All the examples mentioned above are representatives of different kinds of functions. ***Example 1***, ***Example 2*** and ***Example 4*** are cited from [9]. ***Example 1*** is the representative of glossy function with only a few modals, ***Example 2*** is the representative of glossy function with many modals, ***Example 3*** is the representative of non-glossy function, and example 4 is the representative of high-dimension function. Generally we can get satisfying optimal solutions when we set the parameters according to the following principle:

The phase of global optimization: $N_1 \approx 10$*the number of actual optimal solutions

$$2000 < MAXT_1 < 100 * N_1$$

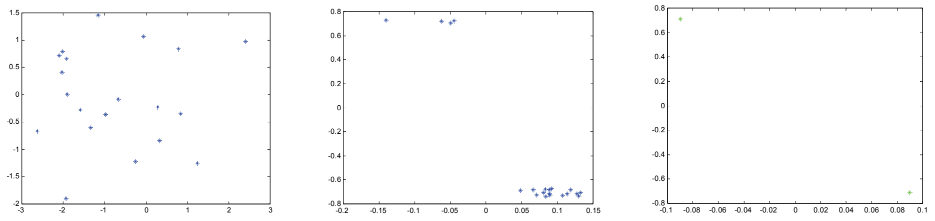
$$6 \le m_1 \le 10$$

The phase of local optimization: $10 < N_2 < 20, r = 2.0$

$$2000 < MAXT_2 < 5000$$

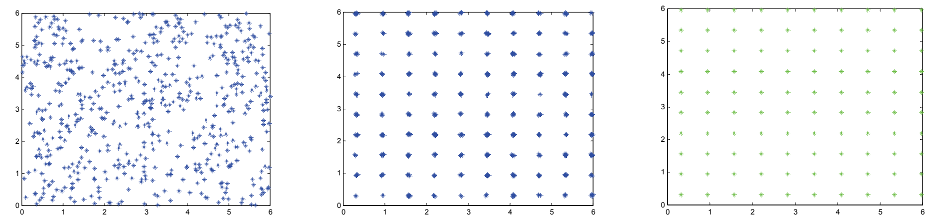
$$3 \le m_2 \le 5$$

The following figures show population distribution in different phases for each example, which indicate the optimization procedures of different examples. Each figure has three parts: (a) is the distribution of population after randomly initialization; (b) is the distribution of population after global optimization; (c) is the distribution of the found modals after local optimization. The horizontal coordinate is the value of x_1 and the vertical coordinate is the value of x_2.



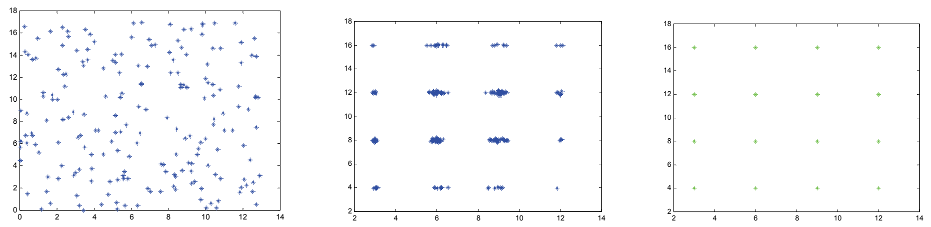
(a) after randomly initialization (b) after global optimization (c) after local optimization

Fig. 2. Population distribution for example 1



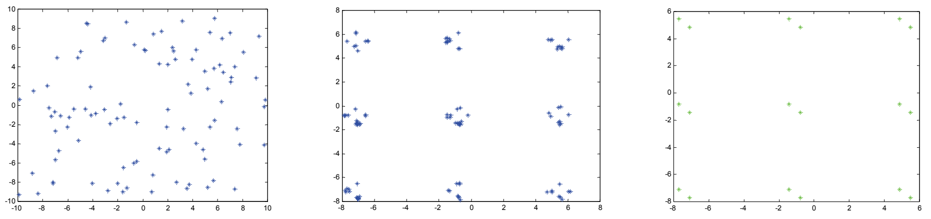
(a) after randomly initialization (b) after global optimization (c) after local optimization

Fig. 3. Population distribution for example 2 when j=5



(a) after randomly initialization (b) after global optimization (c) after local optimization

Fig. 4. Population distribution for example 3



(a) after randomly initialization (b) after global optimization (c) after local optimization

Fig. 5. Population distribution for example 4 when n=2

Additionally, the following tables list parameters and results for different experiments:

Example No.	Parameters		results		
	N_1	$MAXT_1$	Actual modals	Found modals	fitness of

					all modals
Example 1	20	2000	2[9]	2	-1.031628
Example 2（j=5）	600	60000	100[9]	100	1.000000
Example 3	200	20000	16	16	0.000000
Example 4（n=2）	100	10000	18[9]	18	-186.730909

Table 1. Experiment parameters and results for each example (other parameters are: m_1=7,m_2=5,N_2=10,$MAXT_2$=2000)

The value of j	2	3	4	5	6	7	8	9	10
Actual modals	16	36	64	100	121	169	225	289	361
Found modals	16	36	64	100	121	169	225	289	361

Table 2. Experiment results for example 2 with different value of j (The fitness of all modals is 1.000000)

Example No.	Parameters		Results	
	N_1	$MAXT_1$	Found modals	fitness of all modals
Example 4（n=3）	800	500000~1000000	81	-2709.09350

Table 3. Parameters and experiment results for example 4 when n=3 (other parameters are: m_1=7,m_2=5,N_2=10,$MAXT_2$=10000).

From population distribution of the optimization procedures showed in Fig2, Fig3, Fig4 and Fig5, as well as the experiment results showed in Tables 1 and Table 2, we can see that DDEA is very efficient for the optimization of low- dimension multi-modal function, usually we can reach all the modals exactly. But Table 3 indicates that when the dimension of the function is increased to higher than two, the efficiency is decreased because of the search space is expanded sharply.

5. Conclusion

We here proposed some self-adaptive methods to choose the results of Gaussian and Cauchy mutation, and the dimension of subspace. We used the better of Gaussian and Cauchy mutation to do local search in subspace, and used multi-parents crossover to exchange their information to do global search, and used the worst individual eliminated selection strategy to keep population more diversity.

Judging by the results obtained from the above numerical experiments, we conclude that our new algorithm is both universal and robust. It can be used to solve function optimization problems with complex constraints, such as NLP problems with inequality and (or) equality constraints, or without constraints. It can solve 0-1 NLP problems, integer NLP problems and mixed integer NLP problems. When confronted with different types of problems, we don’t need to change our algorithm. All that is needed is to input the fitness function, the constraint expressions, and the upper and lower limits of the variables of the problem. Our algorithm usually finds the global optimal value.

In the paper we analyze the character of the multi-parent genetic algorithm, when applied to solve the optimization of multi-modal function, MPGA works in different forms during different phases and then forms two-phase genetic algorithm. The experiments indicate that DDEA is effective to solve the optimization of multi-modal function whose dimension is no higher than two, but to high-dimension function, the efficiency is not eminent and it needs to be improved much more.

6. Acknowledgements

This work was supported by the National Natural Science Foundation of China (No.60772163) and the Natural Science Foundation of Hubei Province (No. 2005ABA234). Thanks especially give to the anonymous reviewers for their valuable comments.

7. References

Holland J H. Adaptation in Natural and Artificial System. *Ann Arbor: The University of Michigan Press*, 1975.

De Jong, K. A. An analysis of the behavior of a class of genetic adaptive systems. *Ph.D. Thesis*, Ann Arbor, MI:University of Michigan. Dissertation Abstracts International, Vol. 36(10), 5410B (University Microfilms No.76-9381).

Goldberg, D. E. and Richardson, J. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. 41–49.

Guangming Lin, Lishan Kang, Yuping Chen, Bob McKay and Ruhul Sarker, *A self-adaptive Mutations with Multi-paretn Crossover Evolutionary Algorithm for Solving Function Optimization Parbolems Lecture Notes in Computer Science* 4683, pp.157-168, Springer-Verlag Berlin Heidelberg.

Tao Guo, Evolutionary Computation and Optimization. PhD thesis, Wuhan University, Wuhan,1999.

Tao Guo, Kang Lishan. A New Evolutionary Algorithm for Function Optimization. *Wuhan University Journal of Natural Sciences*, 1999, 4(4): 404-419.

Carlos A. Coello: Self-adaptive penalties for GA-based optimization, in Proceedings of the Congress on Evolutionary Computation, Washington, D.C USA, IEEE Press, 1999,537~580

Toyoo Fukuda, Kazuyuki Mori and Makoto Tsukiyama. (1999). Parallel search for multi-modal function optimization with diversity and learning of immune algorithm. In: D. Dasgupta (Ed.) *Artificial Immune Systems and Their Applications*. Springer-Verlag, Berlin, Heidelberg, 210–220.

Zhai Hai-feng. Zhao Ming-wang. A Cell Excluse Genetic Algorithm for Finding All Globally Optimal Solutions of Multimodal Function. *Control and Decision*. 1998,13(2):131-135.

Yan Li, Lishan Kang, Hugo de Garis, Zhuo Kang and Pu Liu. (2002).A robust algorithm for solving the nonlinear programming problems. *International Journal of Computer Mathematics*, 79(5), 523–536.

Lishan Kang,Yan Li, Zhuo Kang, Pu Liu andYuping Chen. (2000).Asynchronous parallel evolutionary algorithm for function optimization. 16th world computer congress. *Proceedings of Conference on Software: Theory and Practice*. Electronic Technology Press, Aug., Beijing, 737–742.

Jun He, Lishan Kang. On the convergence rates of genetic algorithms. Theoretical Computer Science, 229 (1999) 23~29

Yan Liexiang and Ma Dexian. (1999). The sequence competition algorithm for global optimization of continuous variables function. *Journal of Hubei Technology College,* 14(1–2).

Part III:

Dynamic Environment and Multi-Objective Optimization

10

Evolutionary Algorithms with Dissortative Mating on Static and Dynamic Environments

Carlos M. Fernandes[1,2] and Agostinho C. Rosa[1]
[1]*Laseeb – Instituto de Sistemas e Robótica – Instituto Superior Técnico*
[2]*Depart. de Arquitectura y Tecnología de Computadores – University of Granada*
[1]*Portugal*
[2]*Spain*

1. Introduction

Evolutionary Algorithms (EAs) (Bäck, 1996) mimic the process of natural selection by recombining the most promising solutions to a problem from a population of individuals, each one representing a possible solution. There are several methods to select the individuals, but all of them follow the same general rule: good (or partially good) solutions must be chosen more often for recombination events than poorer solutions. In traditional Genetic Algorithms (GAs), for instance, the chromosomes are recombined via a crossover operator over a certain number of generations until a stop criterion is reached. The parents are selected according to their fitness values, that is, better solutions have larger probability to be chosen to generate offspring. By considering merely the quality of solutions represented in the chromosomes when selecting individuals for mating purposes, the traditional GAs emulate what, in nature, is called *random mating* (Roughgarden, 1979; Russel, 1998), that is, mating chance is independent of genotypic or phenotypic distance between individuals.

However, random mating is not the sole mechanism of sexual reproduction observed in nature. *Non-random mating*, which encloses different kinds of strategies based on parenthood or likeness of the agents involved in the reproduction game, is frequently found in natural species, and it is believed to be predominant among vertebrates. Humans, for instance, mate preferentially outside their family tree: this non-random mating scheme is called *outbreeding* and has its opposite in *inbreeding*, a selection strategy where individuals mate preferentially with their relatives (Roughgarden, 1979; Russel, 1998). It is often stated that inbreeding decreases the genetic diversity in a population while outbreeding increases that same diversity (Russel, 1998). In addition, inbreeding will increase the normal rate of a harmful allele present in the family. If inbreeding is extensive and intensive, homozygosity will increase in frequency and the family experiences a growth in the genetic load (measure of all of the harmful recessive alleles in a population or family line) of the harmful allele.

Assortative mating is another non-random mating mechanism, in which individuals choose their mates according to phenotypic similarities (Roughgarden, 1979; Russel, 1998). When similar individuals mate more often than expected by chance, we are in presence of positive assortative mating (or assortative mating in the strict sense). When dissimilar individuals mate more often, the scheme is called negative assortative mating (or *dissortative mating*). In humans, assortative mating is well exemplified by the correlation between heights or intelligence in partners. On the other hand, humans do not mate assortatively with respect to blood groups. This kind of behavior, which selects assortatively for some traits and not others, makes it difficult to unmask the effects of assortative mating in the population. In fact, human assortative mating is not completely positive except for some small and isolated communities (the Old Order Amish, for instance).

Positive assortative mating results in an average increase in homozygosity and in an increase in population variance. However, this does not mean that genetic diversity is increasing. In fact, this type of mating may result in highly distinct cluster of similar genotypes, thus playing a crucial role when speciation without geographic barriers occurs (sympatric speciation) (Todd & Miller, 1991). Dissortative mating, on the other hand, has the primary consequence of a progressive increase in the frequency of heterozygous genotypes; the increase in the diversity of the population is a direct consequence of these changes in the genotype frequencies. Evidences show that mating is very unlikely to be random in nature and may have the potential to act as an evolutionary agent, although its effects are very complex and hard to model and analyze (Jaffe, 1999). Even so, artificial life models presented by Jaffe (1999) and Ochoa et al. (1999) shed some light into the subject, and gave empirical support to the hypothesis that mating is not likely to be random in nature and that assortative and dissortative mating may produce higher survival rates among individuals evolving in, static and dynamic environment, respectively. While in dynamic landscapes genetic variability is fundamental to a quick and effective response to changes, in static environments diversity is not so important. In fact, natural organisms move towards an optimal degree of genetic variability that depends on the environment, via some mating scheme. Environment itself appears to guide the evolution of mating strategies.

In *Evolutionary Computation* (Bäck, 1996), selective pressure and genetic diversity are two major topics, probably those of primary importance (Whitley, 1988). Pressure and diversity are closely related to the delicate equilibrium between exploration and exploitation needed in order to have "safe" search in EAs. Therefore, non-random mating naturally came out in EAs research field in order to deal with the problem of genetic diversity and premature convergence: some efficient algorithms appeared, especially when applied to problems where the genetic diversity is needed in order to maintain exploration high and avoid local optima traps. In addition, diverse search stages usually call for different balance between exploration and exploitation mechanisms. To an initial strong explorative stage, the algorithm gradually must enter a more exploitive phase, where the neighborhood of good solutions found so far is inspected in order to reach the global optimum. When the problem's environment change over time, that is, when dealing with dynamic optimization, genetic diversity becomes even more important, since full convergence must be avoided: the algorithm must maintain sufficient diversity to readapt itself to a change in the fitness function, even if it has converged to the current optimum. In dynamic environments, it is often more important to track the best solution than to converge, that is, it may be sufficient to keep the population near the optimum, even if returning only near-optimal solution, thus avoiding the risk of a full convergence in a specific period of the search, which would reduce the possibilities of readaptation after a change.

Very often recombination is associated with exploitation while mutation is said to play a determinant role in exploration by preventing alleles becoming extinct. While this appears to be true, it may have misled some researches towards assortative mating instead of dissortative, because of the higher exploitation performed by the first strategy. If similar individuals tend to mate, it is more likely that their neighboring space is closely inspected. On the other hand, several studies on dissortative mating showed empirical evidence that this scheme is more adapted to a wide range of problems, both static and dynamic (Craighurst & Martin, 1995; Eschelman, 1991; Eschelman & Schaffer, 1991; Fernandes et al., 2000, 2001; Fernandes & Rosa 2001; Fernandes, 2002; García-Martínez et al., 2007; Matsui, 1999; Ochoa et al., 2005) − see next section for a state-of-the-art review.

This chapter proposes a review and an empirical study on EAs with dissortative mating strategies and their application to static and dynamic problems. Dissortative mating will be discussed within a biological framework and some Artificial Life models will be analyzed; a detailed description of several methods found in EAs literature will be also given. The empirical study will be centered on the Variable Dissortative Mating GA (VDMGA), which was recently presented in (Fernandes & Rosa, 2008) by the authors of this chapter. VDMGA holds a mechanism that varies GA's mating restrictions during the run, by means of a simple rule based on the number of chromosomes created in each generation and indirectly influenced by the genetic diversity of the population. The empirical study presented in (Fernandes & Rosa, 2008) shows that VDMGA performs well when applied to a wide range of problems: it consistently outperforms traditional GAs and assortative mating GAs, and it is faster and more robust than some previously proposed dissortative mating GAs. Results suggest that VDMGA's ability to escape local optima and converge more often to the global solution may come from maintaining the genetic diversity at a higher level when compared with traditional GAs. VDMGA's genetic diversity naturally leads the research towards the application of the algorithm on Dynamic Optimization Problems (DOPs). Due to their specific characteristics, DOPs require additional tools, many of them different from those widely studied by EAs researchers on static problems. Memory schemes and niching (Branke & Schmeck, 2002) are some of the techniques used to tackle DOPs. Strategies for maintaining genetic diversity and/or introducing novelty in the EAs populations are also very efficient strategies when solving dynamic problems (Branke & Schmeck, 2002). In this chapter, the original VDMGA is subject to minor modifications, and then applied to DOPs benchmarks and compared to other GAs. The results confirm the predictions and show that VDMGA may improve other GAs' performance on changing environments. As already been observed when tackling static fitness functions (Fernandes & Rosa, 2008), dissortative mating, via a simple and easily tunable algorithm with diversity preservation, reveals interesting skills when evolving in dynamic environments.

2. Non-random mating evolutionary algorithms

This section describes some EAs with outbreeding, assortative and dissortative mating strategies found in the literature. A special emphasis is given to the ones that, to the extent of the authors of this chapter knowledge, were seminal in their line of work, and to those that preceded (or are, at some level, related to) VDMGA.

In the GA with outbreeding described in (Craighurst, 1995), individuals with a certain degree of parenthood are not allowed to recombine and generate offspring. An incest prevention degree is defined in the beginning of the run and remains unchanged until the convergence criterion is fulfilled. This degree defines how far back in the family tree of an individual the GA must inspect in order to prevent the recombination events. This policy does not completely restrict mating between similar individuals, but it sure decreases its frequency since related individuals tend to share a large amount of common alleles. Tests (Craighurst, 1995) compare the outbreeding GA with a standard GA when applied to the Traveling Salesman Problem. The non-random mating algorithm outperformed the standard GA but the differences in the algorithms' performances were noticed mainly with low mutation rates. This is not surprising since incest prohibition is supposed to maintain the genetic diversity of the population at a higher level for longer periods, thus reducing the need for mutation to introduce genetic novelty into converging populations. Fernandes et al. (2000) combined the outbreeding strategy proposed in (Craighurst, 1995) with a varying population size GA (Arabas, 1994) to create the *non-incest Genetic Algorithm with Varying Population Size* (niGAVaPS). The results showed that the two mechanisms worked together well in order to find the optimum of Four Peaks and Royal Road R4 functions. Tests made with the algorithm ranging through different degrees of incest prohibition showed improvements in the capability of escaping local optima when the individuals are not allowed to mate with their parents and siblings.

There are several studies indicating that dissortative mating may improve EAs performance by maintaining the genetic diversity of the population at a higher level during the search process. For instance, CHC (Eschelman, 1991; Eschelman & Schaffer, 1991) which stands for *Cross generational elitist selection, Heterogeneous Recombination and Cataclysmic Mutation*, is a variation of the standard GA that holds a simple mechanism of dissortative mating which has given proofs of being rather effective in a wide range of problems. Although the title in (Eschelman & Schaffer, 1991) may suggest that CHC is an outbreeding GA, a closer look reveal that the algorithm uses a dissortative mating strategy in order to prevent premature convergence. CHC uses no mutation in the classical sense of the concept, but instead it goes through a process of macro-mutation (or hyper-mutation) when the best fitness of the population does not change after a certain number of generations. The genetic diversity is assured by a highly disruptive crossover operator, the Half Uniform Crossover (HUX) (Eschelman & Schaffer, 1991), and a reproduction restriction that assures that selected pairs of chromosomes will not generate offspring unless their Hamming Distance is above a certain threshold. CHC search process goes as follows. In each generation, $p/2$ pairs of chromosomes are randomly selected from the population with size p. All pairs are submitted to the reproduction process. First, their Hamming distance is computed. If the value is found to be above the threshold then the chromosomes generate two children with the HUX operator. When the process is concluded, the newly generated population of p' offspring replaces the worst chromosomes in the main population, therefore maintaining the size of the population. The threshold is usually set in the beginning of the runs to ¼ of the chromosome length, and decremented when no offspring is generated. When the algorithm is stuck in local optima, a cataclysmic mutation is applied by replacing the entire population, except the best chromosome, with mutated copies of that individual.

The *Assortative Mating GA* (AMGA) was introduced in (Fernandes et al., 2001). The only difference between AMGA and a standard GA is the way parents are selected for recombination. In each recombination event one parent (*first parent*) is select by any traditional method. Then, a set of n individuals is selected by the same method. After computing the similarity between the first parent and all the n individuals in the set, the second parent is chosen according to the type of assortative mating in progress. If the algorithm is the *positive Assortative Mating* GA (pAMGA) the individual more similar to the

first parent is chosen. With the *negative Assortative Mating* GA (nAMGA) the individual less similar is chosen as the second parent (please remember that negative assortative is the same as dissortative). The intensity of the non-random mating scheme may be controlled by the size of the set of candidates to the second parent position. Increasing *n* increases the frequency of mating between dissimilar (if negative assortative) or similar (if positive) individuals. Experiments with the algorithm solving a vector quantization problem showed pAMGA and standard GA performed similarly, while nAMGA outperformed both (Fernandes et al., 2001). Increasing the size of the candidates set resulted in higher success rates (number of runs in which the global optima was found) of nAMGA. In (Fernandes & Rosa, 2001), the algorithm was combined with a varying population size mechanism, tested with a Royal Road (R4) function (Mitchell, 1994) and compared with a standard GA and the niGAVaPS (Fernandes et al., 2000). The negative assortative mating (or dissortative mating) strategy has proven to be more able in escaping Royal Road's local optima traps. pAMGA was also tested under the same conditions but its performance was clearly inferior to standard GA.
A similar idea was tested by Ochoa et al. (2005) on dynamic environments. The authors tested haploid and diploid GAs with assortative mating (where parents are selected as in AMGA) on a knapsack problem with moving extrema, and nAMGA was more able to track dynamic optima. Standard GA often failed to track the optima but the worst performance was attained by pAMGA. In general, the haploid algorithms produced better results than the diploid ones. The authors also discuss the optimal mutation rate for different strategies. By means of exhaustive tests, they concluded that the optimal mutation rate increases when the mating strategy goes from negative (dissortative) to positive assortative. These results were predictable: dissortative mating is supposed to maintain the population diversity at a higher level, reducing the amount of mutation needed in order to prevent the premature convergence of the population. In this line of work, the same authors proposed a study on the error threshold of replication in GAs with different mating strategies (Ochoa, 2006; Ochoa & Jaffe, 2006). The error threshold is a critical mutation rate beyond which structures obtained by an evolutionary process are destroyed more frequently than selection can reproduce them. By evolving a GA on four different fitness landscapes, the authors first conclude that recombination shifts the error threshold toward lower values. Then, the tests show that assortative mating overcomes this effect by increasing the error threshold, while the dissortative strategy pushes the error into lower values. The authors argue that this study may have effects on both natural and artificial systems since it supports the hypothesis that assortative mating overcomes some of the disadvantages inherent to sex. They also intend to shed some light into the relation between mutation rates and mating strategies in EAs. This last issue is directly related with the idea that assortative mating increases the optimal mutation rate of an EA, while dissortative strategies decreases it. This behavior has already been observed in (Fernandes, 2002) and (Ochoa et al., 2005).
Fernandes & Rosa (2006) proposed the *Self-Regulated Evolutionary Algorithm* (SRPEA). SRPEA is an algorithm with a dynamic on-the-fly variation of the population size. Selected individuals are recombined to generate offspring only if their Hamming distance is above a threshold value. That value changes over time, depending on the number of newborn individuals and deaths in each generation. Individuals die (that is, are removed from the population) only when their lifetime (which is set to specific value in the beginning of the search depending on the individual's fitness) reaches zero, which means that parents and

children may belong to the same population. An empirical study demonstrated that the algorithm self-regulates its population size: there are neither uncontrolled demographic explosions nor quasi-extinction long stages, as it is observed in the dynamics of other varying population EAs (Arabas, 1994). VDMGA, the main algorithm in this chapter's study, is directly related to SRPEA.

In (García-Martinez et al., 2006), an assortative mating strategy is used to implement a local search genetic algorithm. The approach is consistent with the fact that crossover is the main mechanism of a GA generating local search, and assortative mating, by its own characteristics, tends to increase the strength of exploitation, thus leading to a more intensive local search. On the other hand, Gárcia-Martinez et al. (2007) introduced a real-coded genetic algorithm with dissortative mating. The authors show that the inclusion of that mating strategy increases the performance of the GA on a set of proposed problems. In addition, empirical analysis indicates that the merits of dissortative mating are clearer with lower values of a parameter of the PBX-a crossover (Lozano et al., 2004). This observation is closely related with the optimal mutation rate issue described above, since a determines the spread of the probability distribution used to create offspring with PBX-a. This way, parameter a acts as genetic diversity controller, with higher values leading to GAs with higher exploratory capabilities, as it happens with mutation rate values. Therefore, if dissortative mating is expected to decrease optimal mutation rates, optimal values of a may also be dependent on the mating strategy chosen for the GA, being lower when dissimilar individuals have more chance to generate offspring.

A large number of other GAs with non-random mating may be found in Evolutionary Computation literature. A few are briefly described in the following paragraph.

Mauldin (1984) proposed a method to avoid similar individuals in the population based on a Hamming distance restriction. CHC is in some way a descendent of Mauldin's method, and, as a result, so is VDMGA. Hillis (1992) described a co-evolutionary computation paradigm with assortative mating applied to a sorting network problem. The author does not provide results comparing the proposed strategy and random mating but it states that the choice on assortative mating was inspired by some problem characteristics rather than genetic diversity concerns. Ronald (1995) introduced the concept of seduction in GAs, which consists in selecting the second parent according to the preferences of the first parent. After the first chromosome involved in a recombination event is selected, all other individuals in the population are provided with a secondary fitness according to certain rules that reflects the preferences of the first parent. Then, the second parent is chosen according to the secondary fitness. Petrowski proposes (1997) speciation in order to restrict mating. De et al. (1998) proposed genotypic and phenotypic assortative mating. The new approaches are compared with standard GA and CHC on some well-known test functions and on the problem of selecting the optimal set of weights in a multilayer *perceptron*. Phenotypic assortative mating revealed to be the best strategy, outperforming standard GA and CHC on the range of proposed problems. Matsui (1999) incorporated dissortative mating within the tournament selection strategy. After the first parent is selected, the second parent is chosen according to a function that depends on the individual fitness and the Hamming distance to the first parent (all individuals in the population are inspected in order to determine the distance to the first parent). In addition, the author incorporates a family-based selection mechanism that, by applying selection and replacement at family level (two parents and two offspring), maintains the genetic diversity of the population. Ting et al. (2003) introduced the Tabu Genetic Algorithm (TGA). TGA combines the characteristics of GAs and Tabu Search (Glover, 1986), by incorporating a taboo list in a traditional GA that prevents inbreeding and maintains genetic diversity. An aspiration criterion is also used by TGA in order to allow some crossovers even if they violate the taboo. Since incest prevention efficiency is sensitive to mutation rate, the authors include a self-adaptive mutation in TGA. The process is somehow similar to the cataclysmic mutation that occurs in CHC, since mutation in TGA occurs in presence of a deadlock situation, that is, when the genetic diversity of the population as decreased down to a level were allowed recombination is almost or even impossible to occur. Finally, Wagner & Affenzeller (2005) introduced the SexualGA, which simulates sexual selection within the frame of a GA and uses two different selection schemes in the same population.

3. Dynamic optimization problems

A problem is said to be a Dynamic Optimization Problem (DOP) when there is a change in the fitness function, problem instance or restrictions, thus making the optimum change as well. When changes occur, solutions already found may be no longer valuable and the process must engage in a new search effort. Traditional EAs, for instance, may encounter some difficulties while solving dynamic problems: if the first convergence stage reduces population diversity, then the algorithm may not be able to react to sudden changes. The crucial and delicate equilibrium needed between exploration and exploitation in static environments becomes even more important and complex when dealing with DOPs. In addition, if the change is detectable (which not always possible), it is hard to decide if it is better to continue the search with same population, after a shift in the environment, or if a restart is more efficient. The extent of the change is of crucial importance in that decision. This problem was stated by Branke & Schmek (2002), which suggested a classification of DOPs and a classification of the most widespread EAs that deal with changing environments. One standard approach to deal with DOPs is to regard each change as the arrival of a new optimization problem that has to be solved from scratch. However, this simple approach is often impractical since solving a problem from scratch without reusing information from the past might be time consuming, a change might not be identifiable directly, or the solution to the new problem should not differ too much from the solution of the old problem. Thus, as in the on-line tracking process suggested in (Angeline, 1997), it has been recommended in (Branke, 1999; Branke, 2002; Branke & Schmeck 2002) to have an optimization algorithm that is capable of continuously adapting the solution to a changing environment, reusing the information gained in the past. Since natural adaptation is a continuous and continuing process and EAs have much in common with natural evolution, they seem to be a suitable candidate for this task. However, evolutionary approaches typically converge to an optimum and thereby lose the diversity necessary for efficiently exploring the search space and consequently also the ability to adapt to a change in the environment (Branke, 2002; Branke & Schmeck 2002). The problem here can be stated as seeking an appropriate balance between two contradictory characters of the search procedure, those between the exploring (ideal for gathering new solutions) and exploiting (making the best use of past solutions) nature of the algorithm. Over the past few years, a number of authors have addressed the problem of convergence and subsequent loss of adaptability in many different ways. According to (Branke e Schmeck 2002), most of these approaches could be grouped into one of the following three categories established by them:

1. React on Changes: The EA is run in standard fashion, but as soon as a change in the environment is detected, explicit actions are taken to increase diversity and thus facilitating the shift to the new optimum.
2. Maintaining Diversity throughout the run: Convergence is avoided all the time and it is hoped that a spread-out population can adapt to changes more easily.
3. Memory-based Approaches: The EA is supplied with a memory to recall useful information from past generations, which seems especially useful when the optimum repeatedly returns to previous locations.

Techniques such as *Hypermutation* (Cobb, 1990) pursue the first category, keeping the whole population after a change but increasing population diversity by drastically increasing the mutation rate for some number of generations. Please note that reacting to changes assumes that changes are detectable, a condition, as already stated, that is not always fulfilled (Branke, 2002).

The *Random Immigrants Genetic Algorithm* (RIGA) (Grefenstette, 1992) is an example of a strategy that falls in the second category. In RIGA the population is partly replaced by r_r randomly generated individuals in every generation. This guarantees the introduction of new genetic material in every time step and avoids the convergence of the whole population to a narrow region of the search space. The performance is affected by the parameter r_r. RIGA is used in the following sections to evaluate VDMGA's performance on DOPs; therefore, its pseudo-code is presented here:

Algorithm 1: Random Immigrants Genetic Algorithm

```
initialize Population(P)
evaluate Population(P)
while (not termination condition) do
    P ← Replace Fraction of Population (P, $r_r$)
    create P.new by selection, crossover and mutation of P
    P ← P.new
end while
```

The following algorithms may also be classified in category 2. As described in the previous section, the *negative Assortative Mating Genetic Algorithm* (nAMGA) (Fernandes & Rosa 2001) is used in (Ochoa et al., 2005) to solve a knapsack DOP. Negative assortative mating (or dissortative mating), by preventing the recombination of similar individuals, slows down the expected diversity loss of traditional GAs thus having the proper characteristics to be classified whitin category 2. The *co-evolutionary agent based model of genotype editing* (ABMGE) (Huang et al., 2007) use several genetic editing characteristics that are gleaned from the RNA editing system as observed in several organisms. Their results outperformed traditional EAs via obtaining greater phenotypic plasticity. In (Tinós & Yang, 2007), a RIGA associated with the Bak-Sneppen model is presented and tested on DOPs: the *Self-Organized Random Immigrants Genetic Algorithm* (SORIGA). Bak-Sneppen (Bak & Sneppen, 1993) is known as a Self-Organized Critically model, a phenomenon that was detected in 1987 by Bak, Tang and Wiesenfield (Bak et al., 1987), and which characterized by displaying scale invariant behavior. When associated with EAs it may periodically insert large amounts of new material in the population or completely reorganize a solution to a problem. For those reasons, it soon was adopted by EA researchers in order to provide new means to control parameter values or maintain population diversity, thus avoiding premature convergence to local optima. DOPs research field was a logical following step. Besides SORIGA, another approach has been recently proposed by Fernandes et al. (2008a), in which the Sandpile model (Bak et al., 1987) is attached to a GA is order to solve DOPs.

Another kind of approach is to supply the algorithm with some sort of memory, storing good partial solutions in order to reuse them later (category 3). This can be advantageous in cases where the environment is changing periodically, and repeated situations occur. However, they also could be counterproductive if the environment changes dramatically with open-ended novelty. Memory may be provided in two general ways: *implicitly* by using redundant representations, or *explicitly* by introducing an extra memory and formulating strategies to deposit and retrieve solutions later. Generally, the most prominent approach to implicit memory and redundant representation is multiploidy (Goldberg & Smith, 1987). On the other hand, while redundant representations allow the EA to implicitly store some useful information during the run, it is not clear that the algorithm actually uses this memory in an efficient way. As an alternative, some approaches use an explicit memory in which specific information is stored and reintroduced into the population at later generations, as in (Louis & Xu, 1996). Branke (1999) compared a number of replacement strategies for inserting new individuals into a memory stressing the importance of diversity for memory-based approaches.

Estimation of Distribution Algorithms (EDAs) (Pelikan, Goldberg & Lobo, 1999; Lorrañga & Lozano, 2002) is a class of EAs where a probability model replaces an explicit representation of the population. In the last decade, research on EDAs has experienced a continuous and consistent growth. However, only recently the DOP issue has started to raise a strong interest on EDAs' researchers. For instance, the Population Based Incremental Learning (PBIL) (Baluja, 1994) - one of the first EDAs - is used in (Yang & Xao, 2005) to solve DOPs created by a problem generator proposed by the same authors. The authors compare several versions of PBIL with GAs and RIGAs. In (Yang, 2005), the author proposes the Univariate Marginal Distribution Algorithm (UMDA) with enhanced memory and the results of the experiments show that the memory is efficient in dynamic environments. In addition, a combination of memory and random immigrants for the UMDA is studied. Lima et al. (2008) investigates the incorporation of restricted tournament replacement (RTR) in the extended compact genetic algorithm (ECGA) (Harik et al., 1999) for solving problems with non-stationary optima. (RTR is a simple yet efficient niching method used to maintain diversity in a population of individuals.) Finally, Fernandes et al. (2008) proposed a new update strategy for UMDA based on Swarm Intelligence.

Some recent proposals have been made using a Swarm Intelligence (Bonabeau, Dorigo & Threraulaz, 1999) approach to attempt to solve dynamic problems. Swarm Intelligence is the property of a system whereby the collective behaviors of simple entities interacting locally with their environment cause global patterns to emerge. In (Guntsch & Middendorf, 2002) the authors applied population based ACO algorithms for tracking extrema in dynamic environments. Others, like (Ramos et al., 2005) developed distributed pheromone layering over the dynamic environment itself, in order to track different peaks. Finally, Fernandes et al. (2007) developed the *Binary Ant Algorithm* (BAA), based on the ACO framework, to take advantage of ACO's ability to solve combinatorial DOPs and generalize it to binary DOPs. However, BAA may also be regarded as a kind of EDA, since, like this class of algorithms, BAA creates the possible solutions to a problem via a transition probability model. Actually, there have been recent attempts to unify ACO and EDAs into the same framework (Zlochin, et al., 2004).

4. The variable dissortative mating genetic algorithm

To model dissortative mating in EAs, some kind of relaxation policy may be needed in order to avoid a freezing population, since evolution eventually leads the search process into a stage of low diversity, where all the individuals are almost identical. In addition, the population usually searches for an optimal degree of genetic variability according to the landscape were it evolves. It is possible that the population movement towards the optimal regions of the landscape also requires different levels of genetic diversity along the way, in order to maintain a robust search. Therefore, the degree of assortative or dissortative mating should vary along the run in order to deal with the inevitable decrease in diversity and to follow the search path of the population. Some methods try to maintain the diversity in a permanent high level, but that may be incompatible with the desirable convergence of the algorithm. For instance, a constant macro-mutation certainly maintains the diversity of the population, but the expected success of an EA based on such premises is not high. Diversity by itself is not a guarantee of a successful search through the landscape.

The *Variable Dissortative Mating Genetic Algorithm* (VDMGA) (Fernandes & Rosa, 2008) is a non-random mating GA, which incorporates an adaptive Hamming distance mating restriction that tends to relax as the search process advances, but may be occasionally reinforced. The algorithm works in the following way. When the first population is randomly created, a threshold value is set to an initial level equal to *L-1*, where *L* is the chromosome length. Then, offspring may be created by selecting pairs of parents (by any method), followed by recombination and mutation. However, recombination only occurs if the genetic distance (Hamming distance in implementation made for this chapter) between the two parents is found to be above the threshold. If not, the recombination event is considered as "failed" and another pair of chromosomes is selected until $N/2$ pairs have tried to recombine (where *N* is the size of the population). When this process ends, the amount of successful and failed recombination events is compared, and the threshold is incremented if successful mating exceeds failed mating. Otherwise, threshold is decremented (the process repeats if no mating succeeded). This way, the threshold is indirectly controlled by the diversity of the population. After the reproduction cycle is completed, a new population is created by selecting the *N* best members from the parents' population and newly generated offspring (if a parent and a child have the same fitness then the child is chosen). Parents and children compete together for survival, conducing to a highly selective algorithm (VDMGA belongs to the class of steady-state GAs). The process repeats until a stop criterion is reached.

VDMGA's threshold value evolves in conformity with the genetic diversity of the population. When diversity decreases, threshold tends to be decremented since the frequency of unsuccessful mating will necessarily increase. However, the mutation operator introduces some variability in the population which may result in occasional increments of the threshold that moves it away from zero (if threshold reaches zero, all individuals are allowed to crossover, like in random mating GAs). Tests performed on several functions confirmed this predicted behavior (Fernandes & Rosa, 2008).

Two changes must be made on the original VDMGA presented in (Fernandes & Rosa, 2008) in order to solve DOPs with an enhanced performance.

Algorithm 2: Variable Dissortative Mating Genetic Algorithm

```
initialize Population(P) with size(P) = N
evaluate Population(P)
set initial threshold(iT)            /* iT ← L-1 for static problems; iT ← L/4 for DOPs*/
threshold(T) ← iT
   while (not termination condition)
      create new individuals P.new
      evaluate new individuals P.new
      if (static problem)
         P ← P+P.new
         remove worst individuals from population(P) until size(P) reaches initial size N
      end if
      if (DOP)
         replace size(P.new) worst individuals from population(P) by P.new
      end if
   end while
```

Procedure: **create new individuals**

```
  matingEvents ← N/2                 /* N is the population size */
  successfulMatings ← 0
  failedMatings ← 0
  while (successfulMatings < 1) do
     for (i ← 1 to matingEvents) do
        select two chromosomes ($c_1$, $c_2$)          /* Any method may be used here */
        compute Hamming distance H($c_1$, $c_2$)
        if (H($c_1$, $c_2$) >= T)
           crossover and mutate
           successfulMatings ← successfulMatings+1
        end if
        if (H($c_1$, $c_2$) < T)
           failedMatings ← failedMatings +1
        end if
     end for
     if (failedMatings > successfulMatings)  T ← T-1
     else                               T ← T+1
  end while
```

(1) In each time step, VDMGA builds an auxiliary pool of chromosomes, with parents and offspring, and then creates the new population by selecting the best chromosomes from the pool. This means that all newly created (and evaluated) individuals may be excluded from the population (considering the "worst" case scenario). Since the study on DOPs performed for this chapter assumes that changes not are detectable − and this is the most general assumption, since changes are not always detectable (Branke, 2002) −, all individuals in the population must be (re)evaluated in each generation, even if they have been created in a previous generation. Individuals with fitness values corresponding to previous shapes of the search space will mislead the search and modify performance metrics in a wrong

manner. (If changes were detectable, reevaluations would only be necessary when detecting a change.) Therefore, when dealing with DOPs, it is better to introduce a larger number of new individuals in VDMGA's population, not only to diminish reevaluations, but also to bring a larger amount of genetic material into the population. For that purpose, original VDMGA replacement strategy is substituted by the following process: all new individuals *N'* are introduced in the new population, replacing the worst *N'* old chromosomes − see pseudo-code for details.
(2) The original initial threshold value was set to *L-1,* such that a strong exploratory behavior is guaranteed to take place in the beginning of the search. Starting with *L*-1, results showed that VDMGA self-regulates the threshold in the first generation according to the conditions of the problem. The adaptive characteristic of the threshold and the robustness of VDMGA to its initial value suggested that it might be convenient to treat threshold's initial value as a constant and let the algorithm self-tune the parameter, thus reducing the complexity of the parameter's space. Since the expected ratio of dissimilar alleles in two random binary chromosomes is equal to 0.5 (considering infinite strings) it is likely that during the first generation ($t = 1$) the threshold decreases to values around $0.5\times L$. On the other hand, experimental results showed that the threshold value in the following generations depends on population size (*N*) and length of the chromosome (*L*): tests performed in (Fernandes & Rosa, 2008) show that the threshold value at the end of the first generation varies from 49.4% and 67.5% of the chromosome length *L* depending on *N* and *L*. As stated before, VDMGA needs to (re)evaluate all the old chromosomes in the population in order to deal with DOPs. If the algorithm passes through an initial stage, during which few new chromosomes are created, until it reaches a more stable threshold value, then a prohibitive number of reevaluations are performed, delaying the algorithm and compromising the first stage of optimization, especially when the changes occur fast. For that reason, initial threshold value is set to a lower value when the problem is dynamic. A value bellow $0.5\times L$ is sufficient. In the tests performed for this study and described in the following sections, initial threshold was set to $0.25\times L$.

5. Performance and scalability on static environments

In (Fernandes & Rosa, 2008), VDMGA was subject to a wide range of experiments on some optimization functions frequently found in EAs literature. The test suite included unimodal and multimodal functions (with and without regular arrangement of local optima), a step function without local gradient information, scalable functions, high dimensional functions and complex combinatorial functions. VDMGA was compared with traditional GAs, CHC (Eschelman, 1991) and nAMGA (Fernandes et al., 2000). pAMGA (Fernandes et al., 2000) was also included in the tests in order to compare analogous dissortative and assortative mating strategies and demonstrate that the former are more efficient in solving the proposed optimization problems. Overall results displayed VDMGA's superior performance when compared to other GAs (while statistically equivalent to nAMGA in some functions, VDMGA proved to be more efficient when facing the harder problems). Please refer to (Fernandes & Rosa, 2008) for a detailed description of the test set and results.
A simple scalability test is also provided in (Fernandes & Rosa, 2008). Using the 4-bit fully deceptive function (Whitley, 1991), results confirm the assumption that VDMGA's optimal population sizes are smaller than standard GA's. Consequently, the slope of the scalability log-log curve is reduced in VDMGA when compared with a generational GA and a steady-

state GA, even if only by a small amount. For this chapter, VDMGA's scalability is investigated in *l*-trap function. The main interest is to perceive how VDMGA reacts to increasing the number of *l*-traps that are juxtaposed and summed together.

5.1 Trap functions and VDMGA's scalability

To investigate how VDMGA's scales on landscapes with different characteristics, experiments were conducted with trap functions, which were used as subproblems to construct larger problems. A trap function is a piecewise-linear function defined on unitation (the number of ones in a binary string). There are two distinct regions in search space, one leading to a global optimum and the other leading to the local optimum (see figure 2). In general, a trap function is defined as in equation 1.

$$trap\left(u\left(\vec{x}\right)\right)=\begin{cases}\frac{a}{z}\left(z-u\left(\vec{x}\right)\right), if \quad u\left(\vec{x}\right)\leq z\\ \frac{b}{l-z}\left(u\left(\vec{x}\right)-z\right), otherwise\end{cases} \tag{1}$$

where $u\left(\vec{x}\right)$ is the unitation function, defined as:

$$u\left(\vec{x}\right)=u\left(x_1,\ \dots\ ,x_L\right)=x_1+\dots x_1=\sum_{i=0}^{L}x_i \tag{2}$$

and *a* is the local optimum, *b* is the global optimum, *l* is the problem size (*l*-bit trap function) and *z* is slope-change location separating the attraction basin of the two optima as depicted in figure 1.

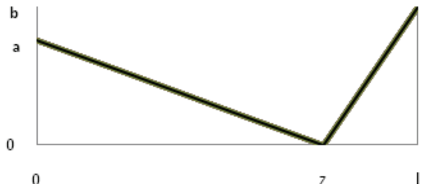


Fig. 1. Generalized *l*-trap function.

Depending on the parameter setting, trap functions may be deceptive or not. Deceptive problems are functions where low-order building-blocks do not combine to form higher order building-blocks. Instead, low-order building-blocks may mislead the search towards local optima, thus challenging GA's search mechanisms. For a trap function to be deceptive, the ratio *r* between the local (*a*) and global (*b*) optimum must be so that:

$$r\geq\frac{2-\frac{1}{L-z}}{2-\frac{1}{z}} \tag{3}$$

In the experiments, 2-bit, 3-bit and 4-bit trap functions were defined with the following parameters: a = *l*-1; b = *l*; z = *l*-1. This way, equation 1 may be simplified:

$$F(u(\vec{x}) = \begin{cases} l, & if\ u(\vec{x}) = l \\ l - 1 - u(\vec{x}), & otherwise \end{cases} \tag{4}$$

Please note that with these settings, the ratio r of the 2-trap function is bellow the deception threshold, while 4-trap is fully deceptive since the condition of equation 3 is satisfied. The ratio of the 3-trap function is equal to the threshold, which means that the function lies in the region between deceptive and non-deceptive. Under these conditions, it is possible to investigate not only how standard GAs and VDMGA scale on l-trap functions, but also to observe how that scaling varies when moving from non-deceptive to fully deceptive search spaces. For that purpose, L-bit decomposable functions were constructed by juxtaposing m trap functions and summing the fitness of each sub-function to obtain the total fitness:

$$f(\vec{x}) = \sum_{i=1}^{m} trap(\vec{x}_i) \tag{5}$$

For each trap and each size m, a standard generational GA (GGA) and VDMGA were run with several values of population size N. Starting from N = 4, optimal population size was determined by the bisection method (Sastry, 2001). The success rate (percentage of runs in which the global optimum was attained) and the average evaluations needed to find the solution (Average Evaluations to a Solution - AES) were measured. Each configuration was executed for 50 times and the results are averaged over those runs. The best configuration was defined as the one with 98% success rate and lower AES. Then, AES optimal population size values corresponding to the best run were plotted and the resulting log-log graphics are depicted in figure 2. The algorithms were tested with uniform crossover and no mutation. Crossover probability, p_c, was set to 1.0. Selection method is binary tournament (k_{ts} = 1.0). (Please note that without mutation it is simply required that one bit is set to 0 or 1 in the entire population for the run to be declared not successful.)

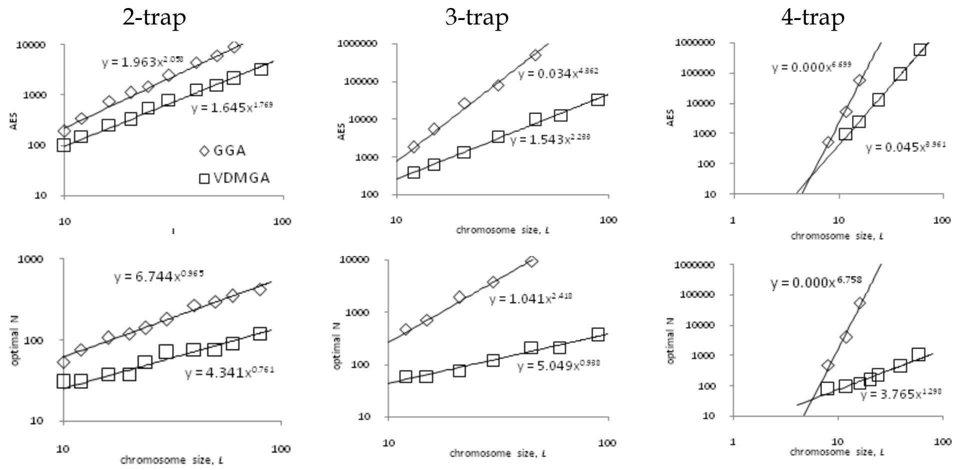


Fig. 2. Scalability with trap functions. Optimal population size and AES values for different problem size $L = l \times m$.

When solving 2-trap functions the algorithms behave similarly, but when the trap dimension increases to 3 and 4, the differences in the scalability is much more noticeable. The difficulty that deceptive trap functions pose to GAs is rather clear when noticing that GGA optimal population size values are close to search space size, that is, 2^L, when solving 4-trap functions. VDMGA significantly reduces the slope of the scalability curve, revealing a good ability to maintain diversity and recombine information in order to achieve the higher order building-blocks. Since VDMGA maintains genetic diversity at a higher level, smaller populations are sufficient to find the global optimum, and thus fewer evaluations are required to converge to that same optimal solution.

VDMGA is a steady-state algorithm, and due to its structure, most of the generations keep the best solutions in the population. When compared to a GGA, which holds no elitism and the offspring completely replaces the parents' population, it is expected that it scales better. To avoid any misinterpretation of the results provided by VDMGA, another scalability test was performed to compare the algorithm with a GGA with 2-elitism (2-e), and a steady-state GA (SSGA) in which half of the population is replaced by the offspring (the worst chromosomes in the current population are replaced by $N/2$ newly generated chromosomes). Results, presented in figure 3, show that both SSGA and GGA 2-e maintain a better scalability than GGA when raising the size of the trap from 2 to 4. In addition, SSGA keeps its performance very close to VDMGA when solving not only 2-traps, but also 3-traps.

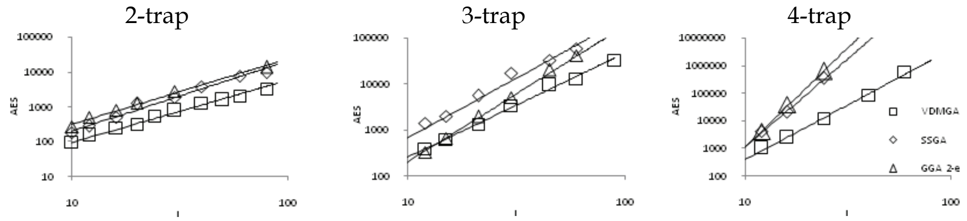


Fig. 3. Scalability with trap functions. Comparing VDMGA with an elitist generational GA (GGA 2-e) and a steady-state GA (SSGA).

However, when reaching 4-trap functions, it is clear that VDMGA scales better than elitist GGA and SSGA. It may be assumed now that this improved scalability is to a great extent due to the fact that VDMGA maintains a higher diversity during the run (Fernandes & Rosa, 2008), and not to its steady-state nature. Scalability tests are very important and useful because when tackling real-world problems, the algorithm may be requested to codify solutions in extremely large binary strings. If the GA does not scale well, optimization becomes practically impossible above a certain problem size. Scalability issues have been increasingly raising the interest of EAs research community, especially amongst EDAs (Pelikan, Goldberg & Lobo, 1999; Lorrañga & Lozano, 2002) researchers.

6. VDMGA on dynamic optimization problems

The test environment proposed in (Yang & Xao, 2005) was used to create an experimental setup for VDMGA on DOPs. Given a stationary problem $f(x)\left(x \in \{0,1\}^L\right)$ where L is the chromosome length, the dynamic environments may be constructed by applying a binary mask $\mathrm{M} \in \{0,1\}^L$ to each solution before its evaluation in the following manner:

$$f(x,t) = f(x \text{ XOR } \mathrm{M}(k)) \tag{6}$$

Where t is the generation index, $k = \frac{t}{\tau}$ is the period index and $f(x,t)$ is the fitness of solution x. M(k) can be incrementally generated as follows:

$$\mathrm{M}(k) = \mathrm{M}(k\text{-}1) \text{ XOR } \mathrm{T}(k) \tag{7}$$

where T(k) is an intermediate binary mask for every period k. This mask T(k) has $\rho \times$ L ones, where ρ is a value between 0 and 1.0 which controls the intensity or severity of change. Notice that $\rho = 0$ corresponds to a stationary problem since T vectors will carry only 0's and no change will occur in the environment. On the other hand, $\rho = 1$ will guarantee the highest degree of change, that is, for instance, if a solution to a problem is a vector of 1's, then the dynamic solution will oscillate between a vector of 1's and a vector of 0's. Therefore, by changing ρ and τ in the previous set of equations it is possible to control two of the most important features when testing algorithms on DOPs: severity (ρ) and speed (τ) of change (Angeline, 1997).

The generator was applied to trap functions. GGA 2-e and SSGA were tested in order to compare them with VDMGA. It is not the aim of this study to compare VDMGA with the best GAs in solving DOPs, even because there is hardly any evidence of a GA that consistently outperforms any other in a wide range of problems and dynamics. Instead, the study of the effects of VDMGA's diversity maintenance on its behavior on dynamic environments is the main aim of this section: the way dissortative mating may be used in order to improve GAs performance and on which kind of DOPs that improvement is more noticeable. Nevertheless, a commonly used algorithm on dynamic optimization studies was added to the test bench: RIGA (see section 3). Two variations were tested. In RIGA 1, the immigrants replace randomly selected individuals from the population, while in RIGA 2 the r_r immigrants replace the worst r_r individuals in the population. (Both RIGA were implemented with 2-elitism. Non-elitist GAs were tested but the performance on trap DOPs was very poor. VDMGA outperformed non-elitist GAs on every problem and (ρ, τ) configuration, but such a test is clearly unfair to standard and Random Immigrants GAs.

All the algorithms were tested with N = 240, p_c = 1.0, uniform crossover and binary tournament (k_{ts} = 1). Performance was measured by comparing the mean *best_of_generation*:

$$mean\ best\_of\_generation = \frac{\sum_{i=0}^{R} \frac{\sum_{j=0}^{T} bestfitness_{i,j}}{T}}{R} \tag{8}$$

where T is the number of generations and R is the number of runs (30 in all the experiments). Several tests were conducted by varying severity (ρ) and speed (τ) of change: ρ was set to 0.05, 0.6 and 0.95; speed of change τ was set to 10, 100 and 200 generations. This means that 9 kinds of environmental changes were tested for each function and algorithm. Every environment was tested with 10 periods of change, thus making T = 100 for τ = 10, T = 1000 for τ = 100 and T = 2000 for τ = 200. Since it is expected that the

optimal mutation rate is not equal for every GA in the test bench, it is of extreme importance to test the algorithms with different p_m. Values ranged from 0.5/L to 5/L and the results displayed on tables 1-3 correspond to best configurations (best configurations were determined by averaging the nine performance values). In order to properly compare the algorithms it is imperative that each GA performs the same number of function evaluations in each generation. Otherwise, during each period between changes, different GAs may be requiring different computation effort. For that reason, RIGAs population size must be set to N-r_r, because RIGA performs extra r_r evaluation in each time step. For RIGA's tests in this section, r_r was set to 24, therefore, N is equal to 216. In addition, since this study assumes that changes in the environment are not detectable, all chromosomes must be evaluated in each generation, even those that have already been evaluated in a previous generation, as in VDMGA (please remember that VDMGA is a steady-state algorithm, that is, parents and children may belong to the same population). For the same reason, SSGA also reevaluates the fraction (half) of the population that has not been replaced by children. (GGA, due to its 2-elitism, must also reevaluate, in each generation, the two best chromosomes in the population.) This way, VDMGA always performs N fitness calculations in each generation but only a fraction of those evaluations are performed on new individuals. This feature is expected to penalize VDMGA's performance on DOPs with very fast changes (low τ), since it may happen that for some periods of time only a small amount of new genetic material (new individuals) are inserted in the population in each generation. Actually, this outcome is confirmed on the first test, performed on 3-trap functions.
Table 1 shows the results obtained by the various GAs on a function constructed by juxtaposing ten 3-trap subfunctions (L = 30). A statistical comparison was carried out by *t-tests* with 58 degrees of freedom at a 0.05 level of significance. The (+) signs means that the corresponding algorithm is significantly better than VDMGA on that particular configuration of ρ and τ. A (~) sign means that the performance is statistically equivalent and (−) sign means that the GA performs worst than VDMGA. A general observation of table 1 shows that only when τ = 10 the GAs consistently outperform VDMGA. With lower speed, VDMGA has always a better performance than the other algorithms in the test bench when ρ = 0.6 and ρ = 0.95, being statistically equivalent when ρ = 0.05. This was an expected outcome, due to what was stated above about VDMGA's ration between function evaluations in each time step and new chromosomes inserted in the population.
Table 2 shows the results with 4-trap functions (L = 12). Values appear to be more balanced in this case: all the algorithms perform similarly, but when increasing the size of the problem to L = 24 – see table 3 −, VDMGA improves its performance when compared to other GAs when τ = 100 and τ = 200 (please remember that VDMGA is expected to face some difficulties when facing fast changing environments. However, the algorithm performs well in 12-bit and 24-bit 4-trap function when ρ = 0.05 and τ = 10.)
An unexpected result occurs when speed of change is slow and ρ = 0.95. For instance, when τ = 200, RIGAs outperforms VDMGA. But when looking at the dynamic behavior of the algorithms, in figure 4, a possible explanation arises for this particular result. Figure 4 shows the dynamics of VDMGA, SSGA and RIGA 2 when tracking the extrema of 4-trap functions (L = 24) by plotting the *best_of_generation* values over all generations. When ρ = 0.6 and τ = 200 the graphs shows that VDMGA becomes closer to the optimum (and results on table 3 confirm that VDMGA outperforms other algorithms). However, when increasing ρ to 0.95,

τ	3-trap (L = 30)				
ρ	GGA	SSGA	RIGA 1	RIGA 2	**VDMGA**
	($p_m = 1/L$)	($p_m = 1/2L$)	($p_m = 1/L$)	($p_m = 1/L$)	**($p_m = 1/L$)**
10	26.06	26.020	25.938	26.144	**25.319**
0.05	±0.978 (+)	±0.506 (+)	±0.661 (+)	±0.819 (+)	**±0.556**
10	22.078	21.467	21.934	21.952	**21.227**
0.60	±0.266 (+)	±0.289 (+)	±0.305 (+)	±0.362 (+)	**±0.280**
10	23.937	23.638	23.832	23.978	**22.877**
0.95	±0.278 (+)	±0.326 (+)	±0.221 (+)	±0.237 (+)	**±0.404**
100	29.712	29.656	29.674	29.664	**29.622**
0.05	±0.090 (~)	±0.082 (~)	±0.145 (~)	±0.175 (~)	**±0.103**
100	26.293	26.095	26.322	26.258	**26.444**
0.60	±0.186 (−)	±0.257 (−)	±0.292 (−)	±0.300 (−)	**±0.250**
100	25.605	25.730	25.628	25.597	**25.999**
0.95	±0.137 (−)	±0.098 (−)	±0.163 (−)	±0.121 (−)	**±0.242**
200	29.851	29.822	29.849	29.852	**29.821**
0.05	±0.051 (~)	±0.075 (~)	±0.526 (~)	±0.056 (~)	**±0.050**
200	27.120	26.972	27.350	27.314	**27.826**
0.60	±0.200 (−)	±0.261 (−)	±0.225 (−)	±0.272 (−)	**±0.170**
200	25.838	25.978	25.802	25.818	**26.211**
0.95	±0.129 (−)	±0.096 (−)	±0.122 (−)	±0.180 (−)	**±0.185**

Table 1. Results on 3-traps (L = 30). Mean *best_of_generation* and corresponding standard deviation values (results averaged over 30 runs).

τ	4-trap (L = 12)				
ρ	GGA	SSGA	RIGA 1	RIGA 2	**VDMGA**
	($p_m = 3/L$)	($p_m = 3/L$)	($p_m = 4/L$)	($p_m = 4/L$)	**($p_m = 2/L$)**
10	10.712	11.307	10.800	10.782	**11.283**
0.05	±0.215 (−)	±0.271 (~)	±0.171 (−)	±0.162 (−)	**±0.254**
10	10.800	10.484	10.783	10.833	**10.585**
0.60	±0.177 (+)	±0.176 (~)	±0.178 (+)	±0.179 (+)	**±0.175**
10	10.914	11.360	10.798	10.825	**11.436**
0.95	±0.213 (−)	±0.193 (~)	±0.174 (+)	±0.191 (+)	**±0.204**
100	11.653	11.948	11.700	11.705	**11.957**
0.05	±0.109 (−)	±0.031 (~)	±0.010 (−)	±0.088 (−)	**±0.020**
100	11.687	11.661	11.713	11.725	**11.672**
0.60	±0.089 (~)	±0.074 (~)	±0.020 (~)	±0.075 (~)	**±0.060**
100	11.710	11.622	11.735	11.688	**11.696**
0.95	±0.080 (~)	±0.076 (−)	±0.016 (~)	±0.068 (~)	**±0.062**
200	11.823	11.981	11.842	11.843	**11.981**
0.05	±0.066 (−)	±0.010 (~)	±0.012 (−)	±0.034 (−)	**±0.012**
200	11.842	11.823	11.847	11.873	**11.831**
0.60	±0.051 (~)	±0.027 (~)	±0.017 (~)	±0.035 (+)	**±0.036**
200	11.852	11.691	11.863	11.864	**11.742**
0.95	±0.049 (+)	±0.055 (−)	±0.014 (+)	±0.037 (+)	**±0.028**

Table 2. Results on 4-traps (L = 12). Mean *best_of_generation* and standard deviation values, averaged over 30 runs.

τ	4-trap (L = 24)				
ρ	SGA ($p_m = 1/L$)	SSGA ($p_m = 2/L$)	RIGA 1 ($p_m = 1/L$)	RIGA 2 ($p_m = 1/L$)	**VDMGA ($p_m = 3/L$)**
10	18.394	19.710	18.301	18.417	**19.544**
0.05	±0.355 (−)	±0.648 (~)	±0.340 (−)	±0.391 (−)	**±0.345**
10	18.270	17.777	18.142	18.066	**17.703**
0.60	±0.185 (+)	±0.311 (~)	±0.282 (+)	±0.282 (+)	**±0.289**
10	20.807	20.489	20.682	20.747	**19.724**
0.95	±0.200 (+)	±0.347 (+)	±0.230 (+)	±0.161 (+)	**±0.284**
100	19.136	22.370	19.091	19.125	**23.421**
0.05	±0.408 (−)	±0.629 (−)	±0.434 (−)	±0.583 (−)	**±0.141**
100	20.570	20.630	20.474	20.593	**20.518**
0.60	±0.242 (~)	±0.293 (~)	±0.233 (~)	±0.274 (~)	**±0.323**
100	21.465	21.308	21.418	21.452	**21.472**
0.95	±0.099 (~)	±0.103 (−)	±0.116 (~)	±0.076 (~)	**±0.176**
200	19.065	23.385	19.220	19.430	**23.746**
0.05	±0.140 (−)	±0.347 (−)	±0.760 (−)	±0.895 (−)	**±0.0726**
200	20.947	21.058	20.851	20.800	**21.670**
0.60	±0.267 (−)	±0.228 (−)	±0.294 (−)	±0.231 (−)	**±0.175**
200	21.509	21.332	21.503	21.495	**21.354**
0.95	±0.065 (+)	±0.136 (~)	±0.087 (+)	±0.082 (+)	**±0.166**

Table 3. Results on 4-traps. Mean *best_of_generation* and standard deviation values, averaged over 30 runs.

the curves become much different. The shape of RIGA's curve may be easily explained by the characteristics of the trap functions used in this study: the global optimum of the functions is the string with all 1's and the local optimum is the string with all 0's. RIGA is stuck in a region of the search space and, when the environment changes dramatically (ρ = 0.95), what were once chromosomes near the global optimum local then become (nearly) local optimum solutions. With 4-trap functions and L = 24, global optimum is 24 and local optimum value is 18. Please note how RIGA oscillates between values near 24 and 18. The algorithm is not able to track the optimum; it just "waits", for the global optimum to pass by every other period of change. Exclusively looking at mean *best_of_generation* values may conduce to a misinterpretation of GAs abilities to solve DOPs.

Another aspect is worth notice. It is clear that RIGA 2 is not able to track the optima when changes are small (ρ = 0.05), at least not as able as VDMGA and SSGA. Random material inserted in the population is not an appropriate strategy to deal with an environment that shifts only by a small amount. For ρ = 0.05 and low speed of change (τ = 100 and τ = 200) VDMGA tracks the optima with much more ability than SSGA, even if it is slower in the first stage of search: only three periods of τ generations are needed for VDMGA to track the optima and remain close to it in the following periods. RIGA 2 appeared to perform well on 24-bit 4-trap when compared to other algorithms (table 3). However, a closer inspection, by plotting the evolution of the tracking process, reveals that the algorithm fails when changes are both small (ρ = 0.05) and severe (ρ = 0.95). VDMGA, on the other hand, maintains a more stable performance trough all the different combinations of speed and severity of change, being particular able to track the optimum when ρ = 0.05.

7. Genetic diversity and threshold dynamics

As described above, assortative and dissortative mating have effects on the frequency of heterozygous and homozygous genotypes. Consequently, population diversity may also be affected: dissortative tends to increase genetic diversity while assortative decreases it. This may also be true when dealing with artificial systems such as GAs. Previous reports

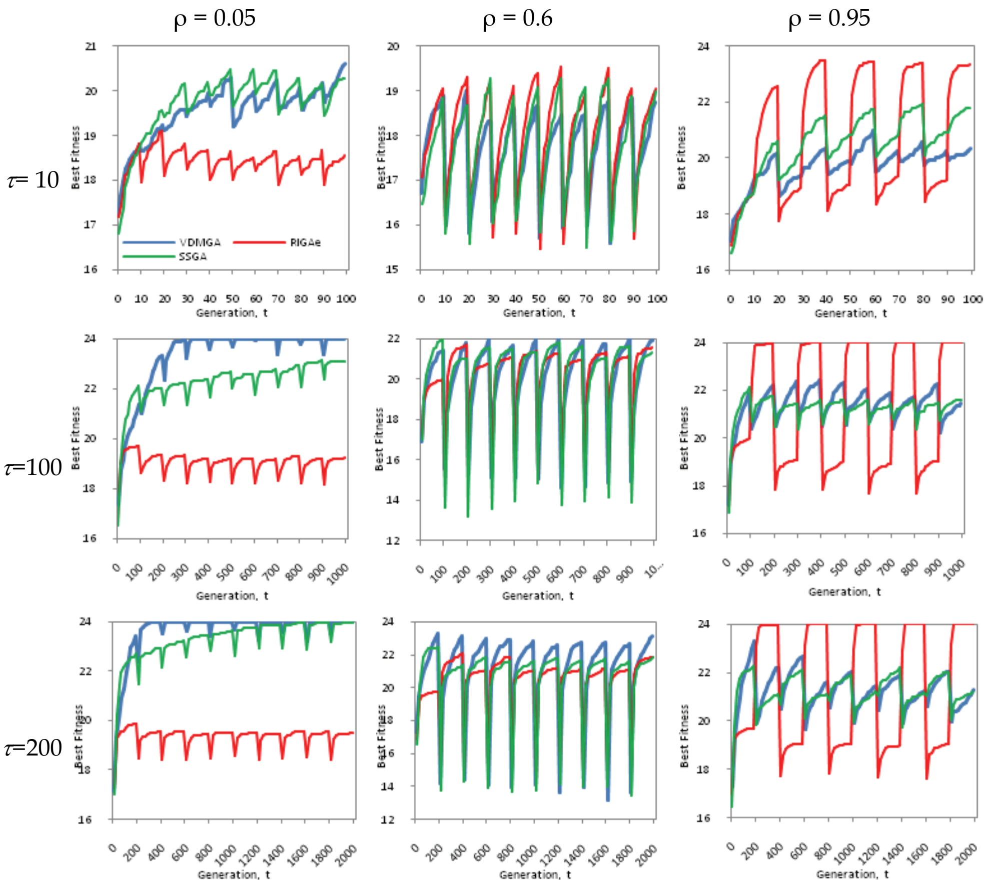


Fig. 4. Dynamics when tracking 4-trap functions (L = 24). *Best_of_generation* curves.

(Fernandes & Rosa, 2001; Fernandes, 2002) show that the variation of diversity in GAs populations is influenced by the chosen mating strategy. In (Fernandes & Rosa, 2008), a study on genetic diversity also confirmed this assumption. To measure diversity, the following equation was used:

$$d(P) = \frac{\sum_{i=1}^{L} \min(F_i, 1 - F_i)}{L/2} \tag{9}$$

where $F_i = \sum_{j=1}^{N} P_i(j)$ and $P_i(j) = \begin{cases} 1, & \textit{if } j^{th} \textit{ gene of chromossome } i \textit{ has allele } 1 \\ 0, & \textit{if } j^{th} \textit{ gene of chromossome } i \textit{ has allele } 0 \end{cases}$

Diversity was inspected on VDMGA, SSGA and RIGA 2. For that purpose, a problem with ten 4-trap subfunctions was used ($L = l \times m = 4 \times 10 = 40$). Population size was set to 100, p_c was set to 1.0 (binary tournament selection and uniform crossover) and different mutation rates p_m were tested. The algorithms were run for 100 generations. Each run was repeated for 30 times and the results are the average over those runs.

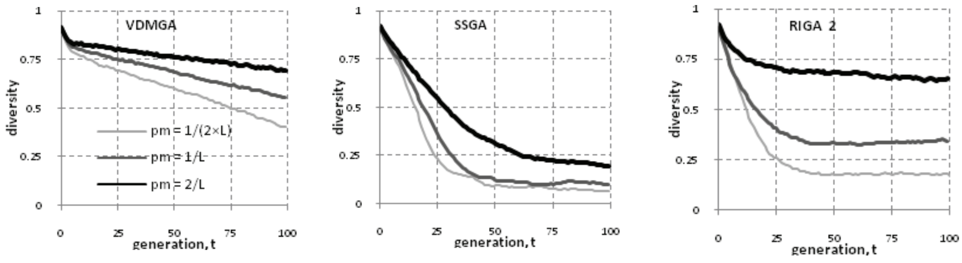


Fig. 5. Genetic diversity.

Graphics in figure 5 show similar results as in previous reports: VDMGA maintains a higher diversity than SSGA, even when comparing different mutation rates. RIGA gets closer to VDMGA's diversity but, as depicted in figure 6, performance is much lower. By observing the growth of the best fitness in the population, it is clear that RIGA 2 is outperformed by SSGA, converging to a lower local optimum. On the other hand, VDMGA, although being slower in a first stage of search, attains higher fitness values, which are still growing when $t = 100$. These results illustrate how important are the genetic diversity maintenance schemes, and not diversity maintenance itself. RIGA, although maintaining the diversity for a longer period, is outperformed by SSGA on this particular test.

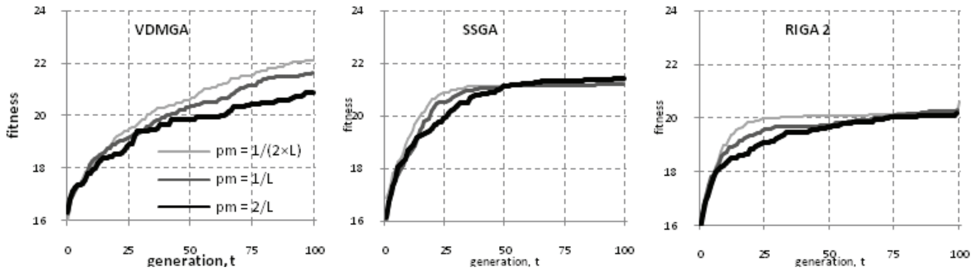


Fig. 6. Best fitness on 4-traps (L = 24).

A final test was conducted with the aim of investigating diversity when the environment changes. For that purpose, a 4-trap DOP with L = 4 was used. GAs parameters were set as in previous experiment. VDMGA's diversity is compared with SSGA in figure 7 (only five periods of change are shown in the graphs), for two configurations of (ρ, τ). As expected, VDMGA maintains a higher diversity throughout the successive search periods, even with lower mutation rates.

VDMGA's threshold values during a run of each one of the previous experiments (ρ = 0.05 and τ = 100; ρ = 0.6 and τ = 200) may be seen in figure 8 (with $p_m = 1/L$). The graphics indicate that the threshold reacts to the changes in the environment when it is close to 0: when the environment shifts, the threshold tends to increase. The explanation for this outcome is simple and resides in the fact that after a change occurs, new genetic material enters in a previously converged population, allowing the threshold to increase because successful matings have also increase. An amplified threshold will then prevent mating between similar individuals and continue to guarantee higher genetic diversity.

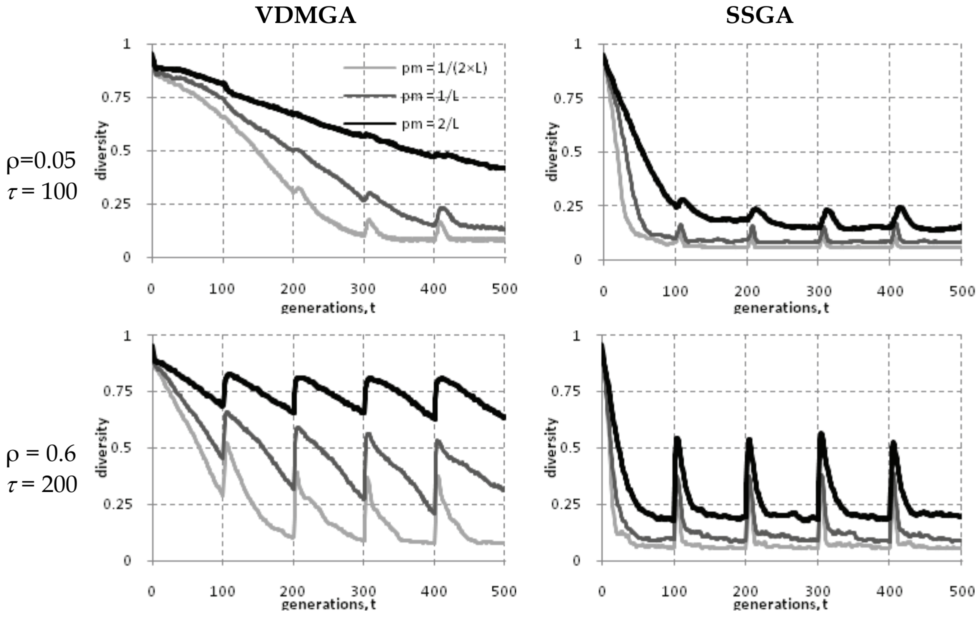


Fig. 7. SSGA and VDMGA's diversity on dynamic 4-trap functions.

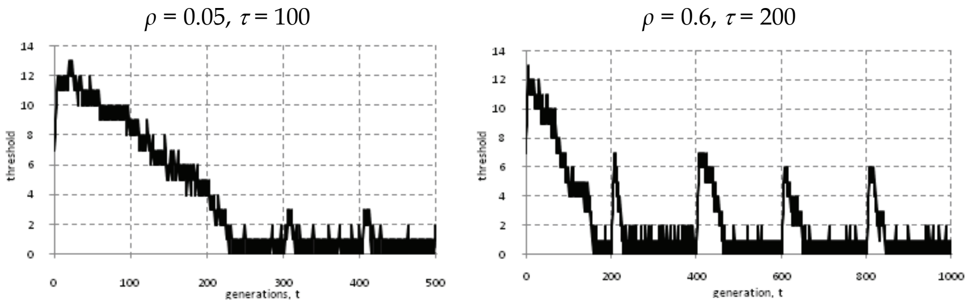


Fig. 8. VDMGA's threshold value. Mutation rate, $p_m = 1/L$

8. Conclusions

This chapter presented a study on Genetic Algorithms (GA) with dissortative mating. A survey on non-random mating was given, in which the most prominent techniques in Evolutionary Computation literature were presented and described. In addition, a survey on Bio-inspired Computation applied to Dynamic Optimization Problems (DOPs) was also given, since DOPs was one of the main aims of the experimental study performed for this chapter. The experiments were performed with the aim of checking the ability of Variable Dissortative Mating GA (VDMGA) on tracking the extrema in dynamic problems. VDMGA, presented in a recent work (Fernandes & Rosa, 2008), inhibits crossover when the Hamming distance between the chromosomes is below a threshold value. The threshold is updated (incremented or decremented) by a simple rule which is indirectly influenced by the genetic diversity of the population: it tends to decrease when the amount of successful crossovers is superior to the number of failed attempts in a generation; when the ratio of successful recombination events rises, the threshold will have a tendency to increase. VDMGA holds this mechanism without the need for further parameters than traditional GAs. In fact, the parameters that need to be tuned are reduced to population size and mutation rate. In addition, no replacement strategy has to be chosen: VDMGA is a steady-state GA in which the number of new chromosomes entering the population in each generation is controlled by the threshold value, genetic diversity and population's stage of convergence.

Scalability tests were performed in order to investigate how VDMGA reacts to growing problem size. Deceptive and non-deceptive trap functions were used for that purpose. The algorithm was tested and compared with traditional GAs. Results showed that VDMGA scales clearly better than other traditional GAs when the trap function is deceptive.

DOPs experiments demonstrated that in most of the cases, VDMGA is able to perform equally or better than other GAs, except when the speed of change is high. In particular, VDMGA outperformed, in general, the Random Immigrants GA, which a typical algorithm used in DOPs studies to compare other methods performance. Statistical t-tests were performed, giving stronger reliability to the conclusions.

A study on the genetic diversity was also performed. As expected, VDMGA maintains a higher diversity throughout the run. The speed of the algorithm may be reduced in a first stage of search (and that is one of the reasons VDMGA is not so able to solve fast DOPs), but the diversity of its population gives it the ability to converge more often to the global optimum.

VDMGA is a simple yet effective algorithm to deal with static and dynamic environments. It holds no more parameters than a standard GA. When regarding DOPs, VDMGA may be classified in the category of methods that preserve diversity in order to tackle DOPs (see section 3). Thus, it avoids the complexity of methods that hold memory schemes (which in general need rules and parameters to determine how to deal with memory), and the lower range of problems in which algorithms that react to changes may be applied. Changes in DOPs are not always detectable and a reaction to changes assumes that it is possible to detect when the environment shifts.

9. Acknowledgments

First author wishes to thank FCT, *Ministério da Ciência e Tecnologia*, his Research Fellowship SFRH/BD/18868/2004, partially supported by *Fundação para a Ciência e a Tecnologia* (*ISR/IST* plurianual funding) through POS_Conhecimento Program that includes FEDER funds.

10. References

Angeline, P. (1997). Tracking Extrema in Dynamic Environments. *Proceedings of the 6th International Conference on Evolutionary Programming*, Springer, pp. 335-345.

Arabas, J.; Michalewicz, Z.; Mulawka, J. (1994). GAVaPS – A genetic algorithm with varying population size, *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, IEEE, Vol. 1: pp. 73-78.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*, Oxford University, New York.

Bak; P.; Tang, C.; K. Wiesenfeld, K. (1987). Self-organized criticality: an explanation of 1/f noise. *Physical Review of Letters*, vol. 59, pp. 381-384.

Bak, P.; K. Sneppen (1993). Punctuated equilibrium and criticality in a simple model of evolution. *Physical Review of Letters*, vol. 71, pp. 4083-4086.

Baluja, S. (1994). Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, *Technical Report CMU-CS-94-163*, Carnegie Mellon University, USA.

Branke, J. (1999). Memory enhanced evolutionary algorithms for changing optimization problems. *Proceedings of the 1999 Congress on Evolutionary Computation*, IEEE, pp. 1875-1882.

Branke, J. (2002), *Evolutionary optimization in dynamic environments*. Kluwer Academic Publishers.

Branke, J.; Schmeck, H. (2002), Designing evolutionary algorithms for dynamic optimization problems. *Theory and Application of Evolutionary Computation: Recent Trends*, A. Ghosh and S. Tsutsui (editors), pp. 239-262.

Cobb, H.G. (1990). An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments, *Technical Report AIC-90-001*, Naval Research Laboratory, Washington, USA.

Craighurst R, Martin W (1995) Enhancing GA performance through crossover prohibitions based on ancestry, *Proceedings of the Sixth International Conference on Genetic Algorithms*, Morgan Kauffman, pp. 130-135.

De, S.; Pal, S.K.; Ghosh, A. (1998). Genotypic and phenotypic assortative mating in genetic algorithm. *Information Science* 105: pp. 209-225.

Eschelman, L.J. (1991). The CHC algorithm: How to have safe search when engaging in non-traditional genetic recombination, *Proceedings of Foundations of Genetic Algorithms*, Academic Press, 1: pp. 70-79.

Eschelman, L.J.; Schaffer, J.D. (1991). Preventing premature convergence in genetic algorithms by preventing incest. *Proceedings of the fourth International Conference on Genetic Algorithms*, Morgan Kauffman, pp. 115-122.

Fernandes, C.M.; Tavares, R.; Rosa, A.C. (2000). NiGAVaPS – Outbreeding in genetic algorithms, *Proceedings of 2000 Symposium on Applied Computing*, ACM, pp. 477-482.

Fernandes, C.M.; Tavares, T.; Munteanu, C.; Rosa, A.C. (2001). Using Assortative Mating in Genetic Algorithms for Vector Quantization Problems, *Proceedings of 2001 Symposium on Applied Computing*, ACM, pp. 361-365.

Fernandes, C.M.; Rosa, A.C. (2001). A Study on Non-Random Mating in Evolutionary Algorithms Using a Royal Road Function. *Proceedings of the 2001 Congress on Evolutionary Computation*, IEEE, pp. 60-66.

Fernandes, C.M. (2002) *Algoritmos Genéticos e Acasalamento não-aleatório*, Msc dissertation thesis, IST, Universidade Técnica de Lisboa, in Portuguese.

Fernandes, C.M., Rosa, A.C. (2006). Self-Regulated Population Size in Evolutionary Algorithms, *Proceedings of 9th International Conference on Parallel Problem Solving from Nature*, LNCS 4193, 920-929.

Fernandes, C.M.; Rosa, A.C.; Ramos, V. (2007). Binary ant algorithm. *Proceedings of the 2007 Genetic and Evolutionary Computation Conference*, ACM, pp. 41-48.

Fernandes, C.; Rosa, A.C. (2008). Self-adjusting the intensity of dissortative mating of genetic algorithms, *Journal of Soft Computing*, in press.

Fernandes. C.; Merelo J.J.; Ramos, V.; Rosa, A.C. (2008a). A self-organized criticality mutation operator for dynamic optimization problems. to appear in *Proceedings of the 2008 Genetic and Evolutionary Computation Conference,* ACM.

Fernandes C, Lima C, Rosa AC (2008b), UMDAs for Dynamic Optimization. to appear in *Proceedings of the 2008 Genetic and Evolutionary Computation Conference,* ACM Press.

García-Martínez, C.; Lozano, M.; Molina, D. (2006). A Local Genetic Algorithm for Binary-Coded problems, In T. Runarsson et al. (eds.), *Proceedings of 9th International Conference on Parallel Problem Solving from Nature*, LNCS 4193, 192-201.

García-Martínez C, Lozano M, Herrera F, Molina D, Sánchez AM (2008). Global and local real-coded genetic algorithms based on parent-centric crossover operators, *European Journal of Operational Research*, 185(3): pp. 1088-1113.

Glover, F. (1986). Future paths for Integer Programming and Links to Artificial Intelligence, *Computers and Operations Research*, 5: pp. 533-549.

Grefenstette, J.J. (1992). Genetic algorithms for changing environments, *Proceedings of Parallel Problem Solving from Nature II,* North-Holland, pp. 137-144.

Goldberg, D.E.; Smith, R.E. (1987). Nonstationary function optimization using genetic algorithms with dominance and diploidy, *Proceedings of the 2nd International Conference on Genetic Algorithms*, ACM, pp. 59-68.

Guntsch, M.; Middendorf, M. (2002). Applying population based ACO to dynamic optimization problems. *Proceedings of 3rd International Workshop ANTS 2002*, Springer, pp. 111-122.

Harik, G. R. (1999). Linkage learning via probabilistic modeling in the ECGA. *IlliGAL Report No. 99010*, Illinois Genetic Algorithms Laboratory.

Hillis, W. (1992). Co-evolving parasites improve simulated evolution as an optimization procedure, *Artificial Life II*, Addison-Wesley, pp. 313-324.

Huang, C.; J. Kaur, A.; Maguitman, L.; Rocha, L. (2007). Agent-based model of genotype editing, *Evolutionary Computation*, MIT Press, pp. 253-289.

Jaffe, K. (1999). On the adaptive value of some mate selection techniques, *Acta Biotheoretica*, 47: pp. 29-40.

Lorrañga, P.; Lozano, J.A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation.* Boston: Kluwer Academic Publishers, Boston.

Louis, S.J.; Xu, Z. (1996). Genetic Algorithms for open shop scheduling and rescheduling. *Proceedings ofg the 11_{th} International Conference on Computers and their Applications*, pp. 99-102.

Lozano, M.; Herrera, F.; Krasnogor, N.; Molina, D. (2004). Real coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation Journal*, 12(3): pp. 273-302.

Matsui K (1999) New selection method to improve the population diversity in genetic algorithms, In: *Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics.*

Mauldin, M. (1984). Maintaining genetic diversity in genetic search, *National Conference on Artificial Intelligence*, AAAI, pp. 247-250.

Mitchell, M. (1994). When will a GA outperform hillclimbing? *Advances in Neural Information Processing Systems*, 6: pp. 51-58.

Ochoa, G.; Madler-Kron, C.; Rodriguez, R.; Jaffe, K. (1999). On sex, selection and the Red Queen. *Journal of Theoretical Biology* 199: pp. 1-9.

Ochoa, G.; Madler-Kron, C.; Rodriguez, R.; Jaffe, K. (2005). Assortative mating in genetic algorithms for dynamic problems. *Proceedings of the 2005 EvoWorkshops*, LNCS 3449, pp. 617-622.

Ochoa, G. (2006). Error Thresholds in Genetic Algorithms. *Evolutionary Computation*, 14(2): pp. 157-182.

Ochoa, G.; Jaffe, K. (2006). Assortative Mating Drastically Alters the Magnitude of Error Thresholds. *Proceedings of 9th International Conference on Parallel Problem Solving from Nature*, LNCS 4193, pp. 890-899.

Pelikan, M.; Goldberg D.; Lobo, F. (1999). A Survey of Optimization by Building and Using Probabilistic Models. *Technial Report 99018*, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory (IlliGAL), IL, USA.

Sastry, K. (2001). *Evaluation-relaxation schemes for genetic and evolutionary algorithms*. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA.

Petrowski, A. (1997). A new selection operator dedicated to speciation, *Proceedings of the 7th International Conference on Genetic Algorithms*, Morgan Kauffman, pp. 144-151.

Ramos, V.; Fernandes, C.; Rosa, A.C. (2005). On self-regulated swarms, societal memory, speed and dynamics. *Proceedings of ALifeX*, MIT Press, pp. 393-399.

Ronald, E. (1995). When selection meets seduction. *Proceedings of the 6th International Conference on Genetic Algorithms*, Morgan Kauffman, pp. 167-173.

Roughgarden, J. (1979). *Theory of population genetics and evolutionary ecology*, Prentice-Hall.

Russel, P.J. (1998). *Genetics*. Benjamin/Cummings.

Ting, C; Sheng-Tu, L.; Chungnan, L. (2003). On the harmounious mating strategy through tabu search. *Journal of Information Sciences*, 156(3-4): pp. 189-214.

Tinós, R; Yang, S. (2007). A self-organizing random immigrants genetic algorithm for dynamic optimization problems. *Genetic Programming and Evolvable Machines*, 8: pp. 255-286.

Todd, P.M.; Miller, G.F. (1991). On the sympatric origin of species: Mercurian mating in the quicksilver model. *Proceedings of the IV International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 547-554.

Wagner, S.; Affenzeller, M. (2005). SexualGA: Gender-specific selection for genetic algorithms. *Proceedings of the 9th World Multiconference on Systemics, Cybernetics and Informatics*, vol.4, pp. 76-81.

Whitley, D. (1988). GENITOR: a different genetic algorithm. *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, pp. 118-130.

Whitley, D. (1991). Fundamental principles of deception in genetic search. *Foundations of Genetic Algorithms*, 1: pp. 221-241.

Yang, S.; Yao, X. (2005). Experimental Study on population-based incremental learning algorithms for dynamic optimization problems. *Journal of Soft Computing*, 9(11): pp. 815-834.

Yang, S. (2005). Memory-enhanced univariate marginal distribution algorithms. *Proceedings of the 2005 Congress on Evolutionary Computation*, ACM, pp. 2560-2567.

Zlochin, M.; Birattari, M.; Meuleau, N.; Dorigo, M. (2004). Modelbased search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 131: pp. 373-395.

11

Adapting Genetic Algorithms for Combinatorial Optimization Problems in Dynamic Environments

Abdunnaser Younes, Shawki Areibi*, Paul Calamai and Otman Basir
*University of Waterloo, *University of Guelph*
Canada

1. Introduction

Combinatorial optimization problems (COPs) have a wide range of applications in engineering, operation research, and social sciences. Moreover, as real-time information and communication systems become increasingly available and the processing of real-time data becomes increasingly affordable, new versions of highly dynamic real-world applications are created. In such applications, information on the problem is not completely known a priori, but instead is revealed to the decision maker progressively with time. Consequently, solutions to different instances of a typical dynamic problem have to be found as time proceeds, concurrently with the incoming information.
Given that the overwhelming majority of COPs are NP-hard, the presence of time and the associated uncertainty in their dynamic versions increases their complexity, making their dynamic versions even harder to solve than its static counterpart. However, environmental changes in real life typically do not alter the problem completely but affect only some part of the problem at a time. For example, not all vehicles break down at once, not all pre-made assignments are cancelled, weather changes affect only parts of roads, any other events like sickness of employees and machine breakdown do not happen concurrently. Thus, after an environmental change, there remains some information from the past that can be used for the future. Such problems call for a methodology to track their optimal solutions through time. The required algorithm should not only be capable of tackling combinatorial problems but should also be adaptive to changes in the environment.
Evolutionary Algorithms (EAs) have been successfully applied to most COPs. Moreover, the ability of EAs to sample the search space, their ability to simultaneously manipulate a group of solutions, and their potential for adaptability increase their potential for dynamic problems. However, their tendency to converge prematurely in static problems and their lack of diversity in tracking optima that shift in dynamic environments are deficiencies that need to be addressed.
Although many real world problems can be viewed as dynamic we are interested only in those problems where the decision maker does not have prior knowledge of the complete problem, and hence the problem can not be solved in advance. This article presents strategies to improve the ability of an algorithm to adapt to environmental changes, and more importantly to improve its efficiency at finding quality solutions. The first constructed

model controls genetic parameters during static and dynamic phases of the environment; and a second model uses multiple populations to improve the performance of the first model and increases its potential for parallel implementation. Experimental results on dynamic versions of *flexible manufacturing systems* (FMS) and the *travelling salesman problem* (TSP) are presented to demonstrate the effectiveness of these models in improving solution quality with limited increase in computation time.

The remainder of this article is organized as follows: Section 2 defines the dynamic problems of interest, and gives the mathematical formulation of the TSP and FMS problems. Section 3 contains a survey of how dynamic environments are tackled by EAs. Section 4 presents adaptive dynamic solvers that include a diversity controlling EA model and an island-based model. The main goal of Section 5 is to demonstrate that the adaptive models presented in this article can be applied to realistic problems by comparing the developed dynamic solvers on the TSP and FMS benchmarks respectively.

2. Background

Dynamism in real-world problems can be attributed to several factors: Some are natural like wear and weather conditions; some can be related to human behaviour like variation in aptitude of different individuals, inefficiency, absence and sickness; and others are business related, such as the addition of new orders and the cancellation of old ones.

However, the mere existence of a time dimension in a problem does not mean that the problem is dynamic. Problems that can be solved in advance are not dynamic and not considered in this article even though they might be time dependent.

If future demands are either known in advance or predictable with sufficient accuracy, then the whole problem can be solved in advance.

According to Psaraftis (1995), Bianchi (1990), and Branke (2001), the following features can be found in most real-world dynamic problems:

- Time dependency: the problem can change with time in such a way that future instances are not completely known, yet the problem is completely known up to the current moment without any ambiguity about past information.
- A solution that is optimal or near optimal at a certain instance may lose its quality in the next instance, or may even become infeasible.
- The goal of the optimization algorithm is to track the shifting optima through time as closely as possible.
- Solutions cannot be determined in advance but should be computed to the incoming information.
- Solving the problem entails setting up a strategy that specifies how the algorithm should react to environmental changes, e.g. to resolve the problem from scratch at every change or to adapt some parameters of the algorithm to the changes.
- The problem is often associated with advances in information systems and communication technologies which enable the processing of information as soon as received. In fact, many dynamic problems have come to exist as a direct result of advances in communication and real-time systems.

Techniques that work for static problems may therefore not be effective for dynamic problems which require algorithms that make use of old information to find new optima quickly.

2.1 Representative dynamic combinatorial problems

Combinatorial problems typically assume distinct structures (for example vehicle routing versus job shop scheduling). Consequently, benchmark problems for COPs tend to be very specific to the application at hand. The test problems used for dynamic scheduling and sequencing with evolutionary algorithms are typical examples (Bierwirth & Kopfer 1994; Bierwirth et al. 1995; Bierwirth & Mattfeld 1999; Lin et al. 1997; Reeves & Karatza 1993). However, the *travelling salesman problem* has often been considered representative of various combinatorial problems. In this article, we use the dynamic TSP and a dynamic FMS to compare the performance of several dynamic solvers.

2.2 Travelling salesman problem

Although the TSP problem finds applications in science and engineering, its real importance stems from the fact that it is typical of many COPs. Furthermore, it has often been the case that progress on the TSP has led to progress on other COPs. The TSP is modelled to answer the following question: if a travelling salesman wishes to visit exactly once each of a list of cities and then return to the city from which he started his tour, what is the shortest route the travelling salesman should take?

As an easy to describe but a hard to solve problem, the TSP has fascinated many researchers, and some have developed time-dependent variants as dynamic benchmarks. For example, Guntsch et al. (2001) introduced a dynamic TSP where environmental change takes place by exchanging a number of cities from the actual problem with the same number from a spare pool of cities. They use this problem to test an adaptive ant colony algorithm. Eyckelhof and Snoek (2002) tested a new ants system approach on another version of the dynamic problem. In their benchmark, they vary edge length by a constant increment/decrement to imitate the appearance and the removal of traffic jams on roads. Younes et al. (2005) introduced a scheme to generate a dynamic TSP in a more comprehensive way. In their benchmarks, environmental changes take place in the form of variations in the edge length, number of cities, and city-swap changes.

2.2.1 Mathematical formulation

There are many different formulations for the travelling salesman problem. One common formulation is the integer programming formulation, which is given in (Rardin 1998) as follows:

$$min \quad \sum_{i}\sum_{j>i} d_{ij}x_{ij} \tag{1}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{j<i} x_{ji} + \sum_{j>i} x_{ij} && = 2 \quad && \text{for all } i \\ & \sum_{i\in S}\sum_{j\notin S, j>i} x_{ij} + \sum_{i\notin S}\sum_{j\in S, j>i} x_{ij} && \geq 2 && \text{for all proper point subsets } S, |S| \geq 3 \\ & x_{ij} = 0 \text{ or } 1 && && \text{for all } i;\ \ j > i \end{aligned}$$

where x_{ij}= 1 if link $(i; j)$ is part of the solution, and d_{ij} is the distance from point i to point j. The first set of constraints ensures that each city is visited once, and the second set of constraints ensures that no sub-tours are formed.

2.2.2 Solution representation

In this article a possible TSP solution is represented in a straight forward manner by a chromosome; where values of the genes are the city numbers, and the relative position of the genes represent city order in the tour. An example of a chromosome that represents a 10 city tour is shown in Figure 1. With this simple representation, however, individuals cannot undergo standard mutation and crossover operators.

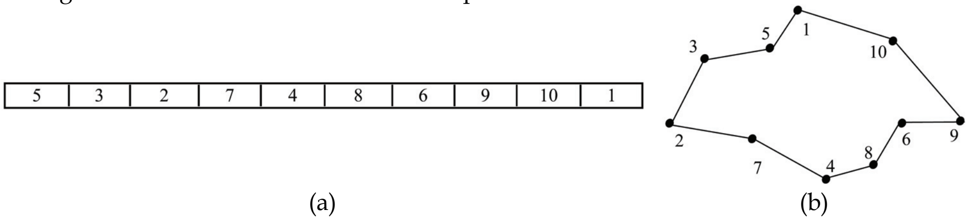


Fig. 1. Chromosome representation (a) of a 10 city tour (b) that starts and ends at city 5.

2.3 Flexible manufacturing systems

The large number of combinatorial problems associated with manufacturing optimization (Dimopoulos & Zalzala 2000) is behind the growth in the use of intelligent techniques, such as flexible manufacturing systems (FMS), in the manufacturing field during the last decade. An FMS produces a variety of part types that are flexibly routed through machines instead of the conventional straight assembly-line routing (Chen & Ho 2002). The flexibility associated with this system enables it to cope with unforeseen events such as machine failures, erratic demands, and changes in product mix.

A typical FMS is a production system that consists of a heterogeneous group of numerically controlled machines (machines, robots, and computers) connected through an automated guided vehicle system. Each machine can perform a specific set of operations that may intersect with operation sets of the other machines. Production planning and scheduling is more complicated in an FMS than it is in traditional manufacturing (Wang et al. 2005). One source of additional complexity is associated with machine-operation versatility, since each machine can perform different operations and an operation can be performed on different alternative machines. Another source of complexity is associated with unexpected events, such as machine breakdown, change of demand, or introduction of new products. A fundamental goal that is gaining importance is the ability to handle such unforeseen events. To illustrate the kind of FMS we are focusing on, we give the following example.

2.3.1 Example

A simple flexible manufacturing system consists of three machines, M_1, M_2 and M_3. The three respective sets of operations for these machines are $\{O_1, O_6\}$, $\{O_1, O_2, O_5\}$, and $\{O_4, O_6\}$, where O_i denotes operation i. This system is to be used to process three part types P_1, P_2, and P_3, each of which requires a set of operations, respectively, given as $\{O_1, O_4, O_6\}$, $\{O_1, O_2, O_5, O_6\}$, and $\{O_4, O_6\}$. There are several processing choices for this setting; here are two of them:

Choice (a) For part P_1: $(O_1 \rightarrow M_2;\ O_4 \rightarrow M_3;\ O_6 \rightarrow M_3)$; i.e, assign machine M_2 to perform operation O_1, and assign M_3 to process O_4 and O_6. For part P_2: $(O_1 \rightarrow M_1;\ O_2 \rightarrow M_2;\ O_5 \rightarrow M_2;\ O_6 \rightarrow M_1)$. For part P_3: $(O_4 \rightarrow M_3;\ O_6 \rightarrow M_3)$.

Choice (b) For part P_1: $(O_1 \rightarrow M_2;\ O_4 \rightarrow M_3;\ O_6 \rightarrow M_1)$. For part P_2: $(O_1 \rightarrow M_1;\ O_2 \rightarrow M_2;\ O_5 \rightarrow M_2;\ O_6 \rightarrow M_3)$. For part P_3: $(O_4 \rightarrow M_3;\ O_6 \rightarrow M_1)$.

By comparing both choices, one notices that the first solution tends to minimize the transfer of parts between machines. On the other hand the second solution is biased towards balancing the operations on the machines. However, we need to consider both objectives at the same time, which may not be easy since the objectives are conflicting.

2.3.2 Mathematical formulation

The assignment problem considered in this section is given in Younes et al. (2002) using the following notations:
i,l are machine indices ($i,l = 1,2,3,...,n_m$);
j is part index ($j = 1,2,3,...,n_p$);
$\hat{k}_j$ is processing choice for part j ($j = 1,2,3,....,n_p$);
k_j is the number of processing choices of P_j;
$n_{ij\hat{k}_j}$ is the number of necessary operations required by Pj on M_i in processing choice $\hat{k}_j$, $1 \leq \hat{k}_j \leq k_j$
$t_{ij\hat{k}_j}$ is the work-load of machine M_i to process part P_j in processing choice $\hat{k}_j$;

$$x_{ji\hat{k}_j} = \begin{cases} 1 & \text{if } P_j \text{ requires } M_i \text{ in processing choice } \hat{k}_j \\ 0 & \text{otherwise;} \end{cases}$$

$$q_{j\hat{k}_j} = \begin{cases} 1 & \text{if processing choice } \hat{k}_j \text{ is selected for part } P_j \\ 0 & \text{otherwise.} \end{cases}$$

Using this notation, the three objective functions of the problem (f_1, f_2, and f_3) are given as follows:

1. Minimization of part transfer (by minimizing the number of machines required to process each part):

$$f_1 = min_{\hat{k}_j} \sum_{i=1}^{n_m} q_{j\hat{k}_j} x_{ji\hat{k}_j}, \forall_j \tag{2}$$

2. Load Balancing by minimizing the cardinality distance (measured in number of operations) between the workload of any pair of machines:

$$f_2 = min_{\hat{k}_j} \sum_{j=1}^{n_p} q_{j\hat{k}_j} \sum_{i=1}^{n_m} \sum_{l=(i+1)}^{n_m} |x_{ji\hat{k}_j} t_{ji\hat{k}_j} - x_{jl\hat{k}_j} t_{jl\hat{k}_j}| \tag{3}$$

3. Minimization of the number of necessary operations required from each machine over the possible processing choices:

$$f_3 = min_{\hat{k}_j} \sum_{i=1}^{n_m} q_{j\hat{k}_j} x_{ji\hat{k}_j} n_{ji\hat{k}_j}, \forall_j \tag{4}$$

An overall multi-objective mathematical model of FMS can then be formulated as follows:

$$Optimize(f_1, f_2, f_3)$$

s.t.

$$\sum_{\hat{k}_j=1}^{k_j} q_{j\hat{k}_j} = 1$$

$$x_{ji\hat{k}_j} \in \{0,1\};\ q_{j\hat{k}_j} \in \{0,1\};\ n_{ji\hat{k}_j} \geq 1;\ t_{ji\hat{k}_j} \geq 0$$

The first set of constraints ensures that only one processing choice can be selected for each part. The complexity and the specifics of the problem require revising several components of the conventional evolutionary algorithm to obtain an effective implementation on the FMS problem. In particular, we need to devise problem-oriented methods for encoding solutions, crossover, fitness assignment, and constraint handling.

2.3.3 Solution representation

An individual solution is represented by a series of operations for all parts involved. Each gene in the chromosome represents a machine type that can possibly process a specific operation. Figure 2 illustrates a chromosome representation of a possible solution to the example given in Section 2.3.1. The advantages of this representation scheme are the simplicity and the capability of undergoing standard operators without producing infeasible solutions (as long as parent solutions are feasible).

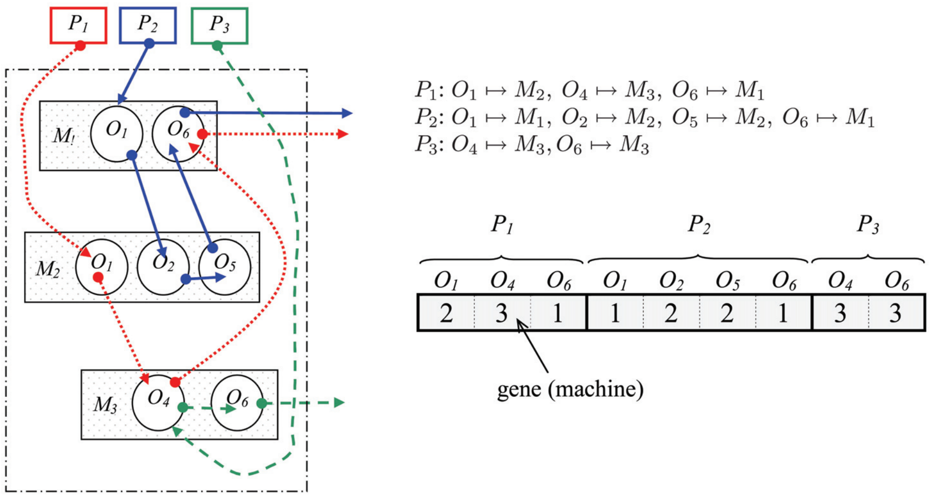

P_1
P_2
P_3
M_1
O_1
O_6
M_2
O_1
O_2
O_5
M_3
O_4
O_6
P_1: $O_1 \mapsto M_2,\ O_4 \mapsto M_3,\ O_6 \mapsto M_1$
P_2: $O_1 \mapsto M_1,\ O_2 \mapsto M_2,\ O_5 \mapsto M_2,\ O_6 \mapsto M_1$
P_3: $O_4 \mapsto M_3, O_6 \mapsto M_3$
P_1 P_2 P_3
O_1 O_4 O_6 O_1 O_2 O_5 O_6 O_4 O_6
2 3 1 1 2 2 1 3 3
gene (machine)
(a)
(b)

Fig. 2. Chromosome representation. A schematic diagram of the possible choice of part routing in (a) is represented by the chromosome in (b)

3. Techniques for dynamic environments

The limitation on computation time imposed on dynamic problems calls for algorithms that adapt quickly to environmental changes. We discuss some of the techniques that have been used to enhance the performance of the standard *genetic algorithm* (GA) in dynamic environments in the following paragraphs (we direct the interested reader to Jin and Branke (2005) for an extensive survey).

3.1 Restart

The most straightforward approach to increase diversity of a GA search is to restart the algorithm completely by reinitializing the population after each environmental change. However, any information gained in the past search will be discarded with the old population after every environmental change. Thus, if changes in the problem are frequent, this time consuming method will likely produce results of low quality. Furthermore, successive instances in the typical dynamic problem do not differ completely from each other. Hence, some researchers use partial restart: Rather than reinitializing the entire population randomly, a fraction of the new population is seeded with old solutions (Louis and Xu 1996; Louis and Johnson 1997). It should be noted here that for environmental changes that affect the problem constraints, old solutions may become infeasible and hence not be directly reusable. However, repairing infeasible solutions can be an effective approach that leads to suboptimal solutions.

3.2 Adapting genetic parameters

Many researchers have explored the use of adaptive genetic operators in stationary environments (see Eiben et al. (1999) for an extensive survey of parameter control in evolutionary algorithms). In fact, the general view today is that there is no fixed set of parameters that remain optimal throughout the search process even for a static problem.
With variable parameters (self adapting or otherwise) finding some success on static problems, it would be natural to investigate them on dynamic problems.
Cobb (1990) proposed *hyper-mutation* to track optima in continuously-changing environments, by increasing the mutation rate drastically when the quality of the best individuals deteriorates. Grefenstette (1992) proposed *random immigrants* to increase the population diversity by replacing a fraction of the population at every generation. Grefenstette (1999) compared *genetically-controlled mutation* with fixed mutation and hyper-mutation, and reported that genetically controlled mutation performed slightly worse than the hypermutation whereas fixed mutation produced the worst results.

3.3 Memory

When the problem exhibits periodic behaviour, old solutions might be used to bias the search in their vicinity and reduce computational time. Ng & Wong (1995) and Lewis et al. (1998) are among the first who used memory-based approaches in dynamic problems. However, if used at all, memory should be used with care as it may have the negative effect of misleading the GA and preventing it from exploring new promising regions (Branke 1999). This should be expected in dynamic environments where information stored in memory becomes more and more obsolete as time proceeds.

3.4 Multiple population genetic algorithms

The inherent parallel structure of GAs makes them ideal candidates for parallelization. Since the GA modules work on the individuals of the population independently, it is straightforward to parallelize several aspects of a GA including the creation of initial populations, individual evaluation, crossover, and mutation. Communication between the processors will be needed only in the selection module since individuals are selected according to global information distributed among all the processors.
Island genetic algorithms (IGA) (Tanese 1989; Whitley & Starkweather 1990) alleviate the communication load, and lead to better solution quality at the expense of slightly slower

convergence. They have showed a speedup in computation time. Even when an IGA was implemented in a serial manner (i.e., using a single processor), it was faster than the standard GA in reaching the same solutions.

Several multi-population implementations were specifically developed for dynamic environments, for example the *shifting balance genetic algorithm* (SBGA) by Wineberg and Oppacher (2000); the *multinational genetic algorithm* (MGA) by Ursem (2000); and the *self-organizing scouts* (SOS) by Branke et al. (2000).

In SBGA there is a single large *core* population that contains the best found individual, and several small colony populations that keep searching for new optima. The main function of the core population is to track the shifting optimal solution. The colonies update the core population by sending immigrants from time to time.

The SOS approach adopts an opposite approach to SBGA by allocating the task of searching for new optima to the *base* (main) population and the tracking to the *scout* (satellite) populations. The idea in SOS is that once a peak is discovered there is no need to have many individuals around it; a fraction of the base population is sufficient to perform the task of tracking that particular peak over time. By keeping one large base population, SOS behaves more like a standard GA—rather than an IGA—since the main search is allocated to one population. This suggests that the method will be more effective when the environment is dynamic (many different optima arise through time) and hence the use of scouts will be warranted. SOS is more adaptive than SBGA, which basically maintains only one good solution in its base.

MGA uses several populations of comparable sizes, each containing one good individual (the peak of the neighbourhood). MGA is also self-organizing since it structures the population into subpopulations using an interesting procedure called *hill-valley detection*, which causes the immigration of an individual that is not located on the same peak with the rest of its population and the merging of two populations that represent the same peak. The main disadvantage of MGA is the frequent evaluations done for valley detection.

3.5 Adapting search to population diversity

There is a growing trend of using population diversity to guide evolutionary algorithms. Zhu (2003) presents a *diversity-controlling adaptive genetic algorithm* (DCAGA) for the vehicle routing problem. In this model, the population diversity is maintained at pre-defined levels by adapting rates of GA operators to the problem dynamics. However, it may be difficult to set a single value as a target as there is no agreed upon accurate measure for diversity (Burke et al. 2004). Moreover, the contemporary notion that the best set of genetic parameters changes during the run can be used to reason that the value of the best (target) diversity also changes during the run.

Ursem (2002) proposes *diversity-guided evolutionary algorithms* (DGEA) which measures population diversity as the sum of distances to an average point and uses it to alter the search between an exploration phase and an exploitation phase. Riget & Vesterstroem (2002) use a similar approach but with particle swarm optimization. However, the limitation on runtime in dynamic problems may not permit alternate phases.

4. Efficient solvers for dynamic COPs

From the foregoing discussion, techniques based on parameter adaptation and multiple populations seem to be the most promising for tackling dynamic optimization problems.

These techniques, however, were designed for either static problems or dynamic continuous optimization problems, thus none can be used without modification for dynamic COPs. This section introduces two models that are specifically designed for dynamic COPs: the first model uses measured population diversity to control the search process, and the second model extends the first model using multiple populations.

4.1 Adaptive diversity model

The *adaptive diversity model* (ADM) is comparable in many ways to other diversity controlled models. ADM, like DCAGA, controls the genetic parameters. However, unlike DCAGA ADM controls the parameter during environmental changes, and without specifying a single target for diversity. ADM, like DGEA, uses two diversity limits to control the search process, however, it does not reduce the search to the distinct pure exploitation and pure exploration phases, and it does not rely on the continuity of chromosome representation.

In deciding on the best measure for population diversity, it is important to keep in mind that the purpose of measuring diversity is to assess the explorative state of the search process to update the algorithm parameters, rather than precisely determining variety in the population as a goal in itself. For this goal, diversity measures that are based on genotopic distances are convenient since genetic operators act directly on genotype.

Costs of computing diversity of a population of size n can be reduced by a factor of n by using an average point to represent the whole population. However, arithmetic averages can be used only with real-valued representations. Moreover, an arithmetic average does not reflect the convergence point of a population, since evolutionary algorithms are designed to converge around the population-best. Hence, it is more appropriate to measure the population diversity in terms of distances from the population-best rather than distances from an average point. By reserving individual v_n for the population-best, the aggregated genotypic measure (d) of the population can be expressed as

$$d = \sum_{i=1}^{n-1} \frac{dist(v_i, v_n)}{n-1}. \tag{5}$$

Considering the mutation operator for a start, ADM can be described as follows. When an environmental change is detected (at $t = t_m$), the mutation rate is set to an upper limit $\overline{\mu}$. While the environment is static ($t_m \leq t < t_{m+1}$), population diversity $d(t)$ is continually measured and compared to two reference values, an upper limit d_h and a lower limit d_l, and the mutation rate $\mu(t)$ is adjusted using the following scheme:

$$\mu(t) = \begin{cases} \overline{\mu}, & t = t_m \\ Z_l \cdot \big(\overline{\mu} - \mu(t-1)\big) + \mu(t-1), & t \neq t_m,\ d(t) < d_l \\ \mu(t-1) - Z_h \cdot \big(\mu(t-1) - \underline{\mu}\big), & t \neq t_m,\ d(t) > d_h \\ \mu(t-1), & t \neq t_m,\ d_l \leq d(t) \leq d_h \end{cases} \tag{6}$$

where $Z_l = min\left\{\frac{d_l - d(t)}{D}, 1\right\}$, $Z_h = min\left\{\frac{d(t) - d_h}{D}, 1\right\}$, $D = d_h - d_l$

The formula for adaptive crossover rate $\hat{A}(t)$ is similar to that of mutation. However, since high selection pressures reduce population diversity the selection probability $s(t)$ is adapted in an opposite manner to that used for mutation in Equation 6, as follows:

$$s(t) = \begin{cases} \underline{s}, & t = t_m \\ s(t-1) - Z_l \cdot \big(s(t-1) - \underline{s}\big), & t \neq t_m, d(t) < d_l \\ Z_h \cdot \big(\overline{s} - s(t-1)\big) + s(t-1), & t \neq t_m, d(t) > d_h \\ s(t-1), & t \neq t_m,\ d_l < d(t) < d_h \end{cases} \tag{7}$$

where $\underline{s}$ and $\overline{s}$ are the lower and the upper limits of selection probability respectively; and Z_l, and Z_h are as given earlier in the mutation formula 6.
Figure 3 illustrates the general principle of the ADM, and how it drives genetic parameters toward exploration or exploiting in response to measured diversity. In this figure, *P* can be the value of any of the controlled genetic parameters μ, χ or s. P_r corresponds to maximum exploration values; i.e., $\overline{\mu}$, $\overline{\chi}$ or $\underline{s}$, whereas P_t corresponds to maximum exploitation values ($\overline{\mu}$, $\overline{\chi}$, or $\overline{s}$).
The pseudo code for a dynamic solver using ADM can be obtained from Figure 5, by setting the number of islands to one and cancelling the call to PerformMigration().

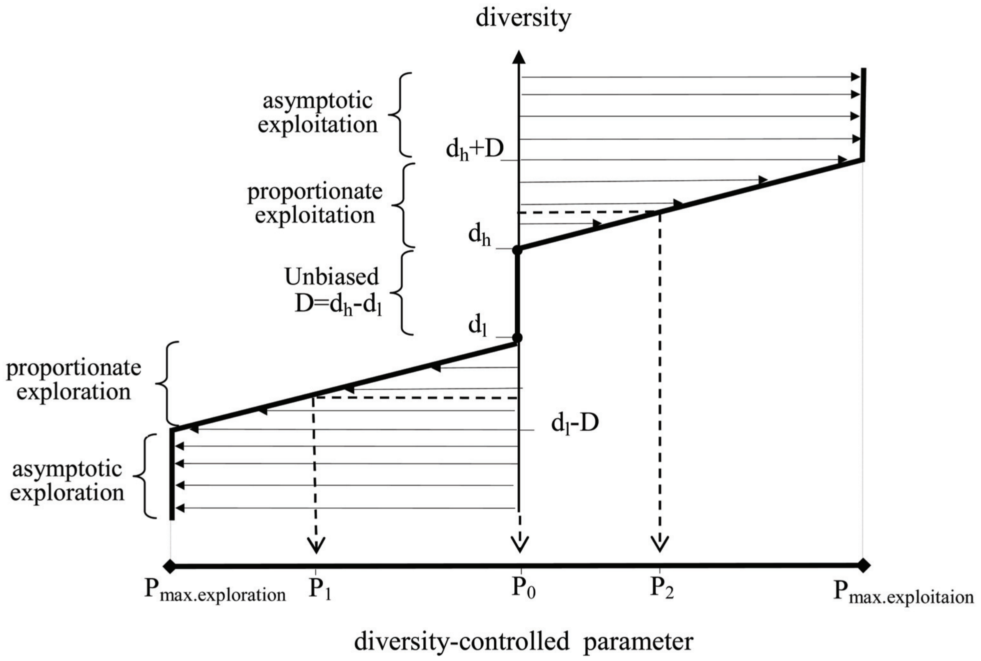


Fig. 3. Diversity range is divided into five regions.

Low diversity maps the genetic parameter into a more explorative value (e.g., P_1) and high diversity maps it into a less explorative value (e.g., P_2). Diversity values between d_l and d_h do not change the current values of the genetic parameters (the parameter is mapped into its original value P_0). The farther the diversity is from the unbiased range, the more change to the genetic parameter. Diversity in the asymptotic regions maps the parameter into one of its extreme values ($P_{max.exploration}$ or $P_{max.exploitation}$) .

4.2 Adaptive island model

The *adaptive island model* (AIM) shares many features with other multiple population evolutionary algorithms that have been mentioned previously. However, unlike SBGA and SOS, AIM uses a fixed number of equal-size islands. In addition, no specific island is given the role of base or core island in AIM: the island that contains population-best is considered the current base island. AIM maintains several good solutions at any time, each of which is the center of an island. Accordingly, all islands participate in exploring the search space and at the same time exploit good individuals. AIM is more like MGA, but still does not rely on the continuity nature of the variables to guide the search process. As well, AIM uses diversity-controlled genetic operators, in a way similar to that of ADM.

AIM extends the function of ADM to control a number of islands. Thus, two measures of diversity are used to guide the search: an island diversity measure and a population diversity measure. Island diversity is measured as the sum of distances from individuals in the island to the island-best, and population diversity is measured as the sum of the distances from each island best to the best individual in all islands.

Each island is basically a small population of individuals close to each other. It evolves under the control of its own diversity independently from other islands. The best individual in the island is used as an aggregate point for measuring island diversity and as a representative of the island in measuring inter-island diversity (or simply population diversity).

With the islands charged with maintaining population diversity, the algorithm becomes less reliant on the usual (destructive) high rates of mutation. Furthermore, mutation now is required to maintain diversity within individual islands (not within entire population), thus lower rates of mutation are needed. Therefore, mutation rate in AIM, though still diversity dependent, has a lower upper limit.

In order to avoid premature convergence due to islands being isolated from each other, individuals are forced to migrate from one island to another at pre-defined intervals in a ring-like scheme, as illustrated in Figure 4. This scheme helps impart new genetic material to destination islands and increase survival probability of high fitness individuals.

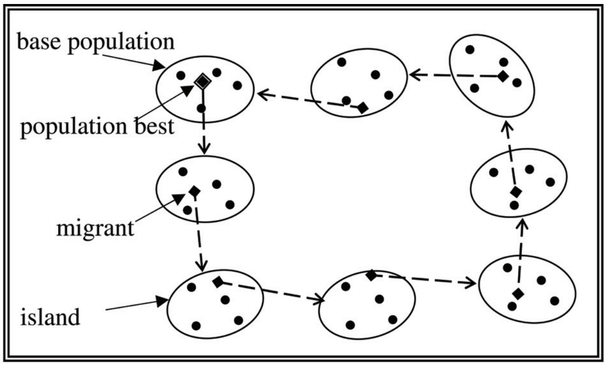


Fig. 4. Ring migration scheme, with the best individuals migrating among islands

On the global level, AIM is required to keep islands in different parts of the search space. This requirement is achieved by measuring inter-island diversity before migration and by mutating duplicate islands. If two islands are found very close to each other, one of them is

considered a duplicate, and consequently its individuals are mutated to cover a different region of the search space. Elite solutions consisting of the best individual from each island are retained throughout the isolation period. During migration, elite solutions are not lost since best individuals are forced to migrate to new islands.
At environmental changes, each island is re-evaluated and its genetic parameters are reset to their respective maximum exploration limits. During quiescent phases of the environment, genetic parameters are changed in response to individual island diversity measures. A pseudo code for AIM is given in Figure 5.

```
Procedure AIM
    i = 0;   // initiate instances counter
    g = 0;   // initiate generations counter
    pop = Generate(isl[1], isl[2], . . . , isl[n_islands]);
    Evaluate( pop );
    repeat
        while quiescent environment
            EvolveIslands( pop , g );
            PerformMigration( pop );
        endwhile
        i = i+1;
        AdaptPopulation( pop , strategy );
    until terminating condition

Procedure EvolveIslands( pop , g )
 for  (k=1 to n_islands)
    repeat
      g = g+1;
      Select( isl[k] );
      Cross( isl[k] );
      Mutate( isl[k] );
      best_of_isl[k] = Evaluate( isl[k] );
      isl_div = MeasureDiversity( isl[k] );
        // use diversity to update genetic parameters
      params[k] = DiversityAdaptParam( params[k] , isl_div );
    until end of isolation period
      pop_best = Best(pop_best,  best_of_isl[k] );
 endfor

Procedure AdaptPopulation( pop , strategy )
 Evaluate(pop);
 Repair(pop);
 ApplyDynamicStrategy(pop);     // set parameters to max explorative limits

Procedure PerformMigration( pop )
 pop_div = MeasureDiversity( isl[1], isl[2], . . . , isl[n_islands] );
 for  (k=1 to n_islands)
      //mutate current island if it is too close to another island
    MutateIsland( isl[k], pop-div);
    Migrate( isl[k] );
 endfor
```

Fig. 5. Pseudo code for AIM. The model can be reduced to ADM by setting the number of islands to one, and cancelling the call to PerformMigration().

5. Empirical study and analysis

The main purpose of this section is to demonstrate the applicability of the adaptive models to realistic problems. First, this section describes the performance measure and the strategies under comparison. Benchmarks and modes of dynamics are then given for each problem together with the results of comparison. Statistical analysis of the significance of the comparisons is given in an appendix at the end of this article.

5.1 Standard strategies and measures of performance

The dynamic test problems are used to compare the proposed techniques against three standard models: a *fixed model* (FM) that uses a GA with fixed operator rates and does not apply any specific measures to tackle dynamism in the problem, a *restart model* (RM) that randomly re-generates the population at each environmental change, and a *random immigrants model* (RIM) that replaces a fraction (10%) of the population with random immigrants (randomly generated individuals) at each environmental change.
Since the problems considered in this article are minimization of cost functions, the related performance measures are directly based on the solution cost rather than on the fitness. First, a mean best of generation (MBG) is defined after G generations of the r_{th} run as:

$$MBG(G,r) = \frac{1}{G}\sum_{g=1}^{G}\left(\frac{\varepsilon_g^r}{\hat{c}_g}\right), \qquad \varepsilon_g^r = min\left\{e_\theta^r \mid t_g \le \theta < t_{g+1}\right\} \tag{8}$$

where e_θ^r is the cost associated with the individual evaluated at time step θ and run r, t_g is the time step at which generation g started, and $\hat{c}_g$ is the optimal cost (or the best known cost) to the problem instance at generation g. The algorithm's performance on the benchmark over R runs can then be abstracted as

$$RunsAverageMBG(G,R) = \frac{1}{R}\sum_{r=1}^{R} MBG(G,r). \tag{9}$$

With these definitions, smaller values of the performance measure indicate improved performance. Moreover, since *MBG* is measured relative to the value of the best solutions found during benchmark construction, it will in general exceed unity. Less than unity values, if encountered, indicate superior performance of the corresponding model in that the dynamic solver with limited (time per instance) budget outperforms a static solver with virtually unlimited budget.

5.2 Algorithm parameter settings

In all tested models, the underlying GA is generational with tournament selection in which selection pressure can be altered by changing a selection probability parameter. A population of fifty individuals is used throughout. The population is divided into five islands in the AIM model (i.e., ten individuals per island).
The FM, RM and RIM models use a crossover rate of 0.9 and a selection probability of 1.0. The mutation rate is set to the inverse of the chromosome length (Reeves & Rowe 2002). For the ADM and AIM models, the previous values represent the exploitation limits of their

corresponding operators, with the exploration limits being 1.0 for crossover, 0.9 for selection, and twice the exploitation limit for mutation.

For TSP, edge crossover (Whitley et al. 1991) and pair-wise node swap mutation are used throughout. The mutation operator sweeps down the list of bits in the chromosome, swapping each with a randomly selected bit if a probability test is passed.

For FMS, a simple single-point crossover operator and a standard mutation operator are used throughout (Younes et al. 2002).

5.3 TSP experimentation

5.3.1 TSP benchmark problems

Static problems of sizes comparable to those reported in the literature (Guntsch et al. 2001; Eyckelhof & Snoek 2002) are used in the comparative experiments of this section. These problems are given in the TSP library (Reinelt 1991) as berlin52, kroA100, and pcb442. In this article they are referred to as *be52*, *k100*, and *p442* respectively. Dynamic versions are constructed from these problems in three ways (modes): an edge change mode (ECM), an insert/delete mode (IDM) and a vertex swap mode (VSM).

Edge change mode The ECM mode reflects one of the real-world scenarios, a traffic jam. Here, the distance between the cities is viewed as a time period or cost that may change over time, hence the introduction and the removal of a traffic jam, respectively, can be simulated by the increase or decrease in the distance between cities. The change step of the traffic jam is the increase in the cost of a single edge. The strategy is as follows: If the edge cost is to be increased then that edge should be selected from the best tour. However, if the cost were to be reduced then the selected edge should not be part of the best tour.

The BG starts from one known instance and solves it to find the best or the near best tour. An edge is then selected randomly from the best tour, and its cost is increased by a user defined factor creating a new instance which will likely have a different best tour.

Insert/delete mode The IDM mode reflects the addition and deletion of new assignments (cities). This mode works in a manner similar to the ECM mode. The step of the change in this mode is the addition or the deletion of a single city. This mode generates the most difficult problems to solve dynamically since they require variable chromosome length to reflect the increase or decrease in the number of cities from one instance to the next.

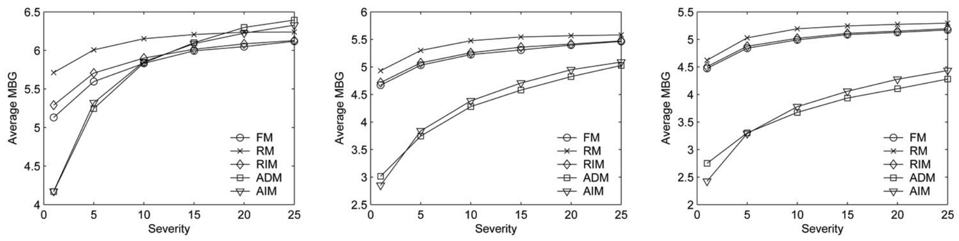
Vertex swap mode The VSM mode is another way to create a dynamic TSP by interchanging city locations. This mode offers a simple, quick and easy way to test and analyze the dynamic algorithm. The locations of two randomly selected cities are interchanged; this does not change the length of the optimal tour but does change the solution (this is analogous to shifting the independent variable(s) of a continuous function by a predetermined amount). The change step (the smallest possible change) in this mode is an interchange of costs between a pair of cities; this can be very large in comparison with the change steps of the previous two modes.

In the experiments conducted, each benchmark problem is created from an initial sequence of 1000 static problems inter-separated by single elementary steps. Depending on the specified severity, a number of intermediate static problems will be skipped to construct one test problem.

Each sequence of static problems is translated into 21 dynamic test problems by combining seven degrees of severity (1, 5, 10, 15, 20, 25 steps per shift, and random) and three periods of change (500, 2500, and 5000 evaluations per shift, which correspond to 10, 50, and 100 generations per shift based on a population of 50 individuals).

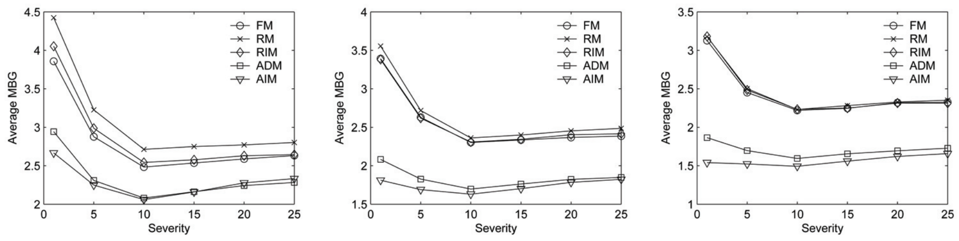
5.3.2 TSP results

Experimental results on the dynamic k100 problem in the VSM mode under three different periods of change are given in Figure 6, where the mean best of generation (averaged over ten runs) is plotted against severity of change. The ADM and AIM models outperform the other models in almost all cases. The other three models give comparable results to each other in general, with differences in solution quality tending to decrease as the severity of change increases. Only when the change severity is 10 steps per shift or more, may the other models give slightly better performance than ADM and AIM. Keep in mind that in this 100 vertex problem, a severity of 10 in the VSM mode amounts to changing (4 × 10) edges; that is, about 40% of the edges in an individual are replaced, which constitutes a substantial amount of change. As we are interested in small environmental changes (which are the norm in practice), we can safely conclude that the experiments attest to the superiority of the ADM and AIM over the other three models in the range of change of interest.



Period = 10 generations Period = 50 generations Period = 100 generations

Fig. 6. Comparison of evolutionary models (k100_VSM)

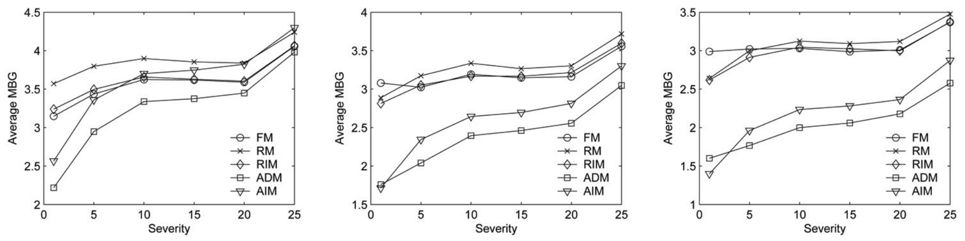


Period = 10 generations Period = 50 generations Period = 100 generations

Fig. 7. Comparison of evolutionary models (k100_ECM)

Running the benchmark generator in either the ECM mode or the IDM mode gives similar results as illustrated in Figure 7 and Figure 8 respectively. It can be seen that ADM and AIM outperform the other models in most considered dynamics.

The RM model produces the worst results in all conducted experiments (even though this model has been modified to retain the best solution in the hope of obtaining better results than those obtainable using a conventional restart).



Period = 10 generations Period = 50 generations Period = 100 generations

Fig. 8. Comparison of evolutionary models (k100_IDM)

It is not easy to conclude from previous results the superiority of either model (ADM or AIM), since both give very comparable results in almost all cases. However, when more than one processor can be used, AIM is the best of the two models since it can be easily parallelized by allocating different islands to different processors and consequently reduce computation time drastically.

5.4 FMS experimentation

5.4.1 FMS benchmark problems

Four instances of sizes comparable to those used in the literature (Younes et al. 2002) are used in the comparative experiments of this section.

Three of these instances (20 agents, 200 jobs), (20 agents, 100 jobs) and (10 agents, 100 jobs) were used in Chu and Beasley (1997). The data describing these problems can be found in the *gapd* file in the OR-library (Beasley 1990). In this article they are referred to as *gap1*, *gap2*, and *gap3* respectively. As described in Chen & Ho (2002), agents are considered as machines, jobs are considered as operations, and each part is assumed to consist of five operations. In these instances, a machine is assumed capable of performing all the required operations. However, in general machines may have limited capabilities; that is, each machine can perform a specific set of operations that may or may not overlap with those of the other machines. To enable this feature, a machine-operation incidence matrix is generated for each instance as follows: If the cost of allocating a job to an agent is below a certain level, the corresponding entry in the new incidence matrix is equal to one to indicate that the machine is capable of performing the corresponding operation. Alternatively, if the cost is above this level, the corresponding entry in the incidence matrix is zero to indicate that the job is not applicable to the machine. The final lists that associate parts with operations and machines with operations are used to construct the dynamic problems.

The fourth problem instance is randomly generated. It was specifically designed and used to test FMS systems with overlapping capabilities in Younes et al. (2002). This instance consists of 11 machines, 20 parts, and 9 operations. In this article, it is referred to as *rnd1*.

In terms of the number of part operations (chromosome length) and the number of machines (alleles), the dimensions of these problems are 200×20, 100×20, 100×10,and 62×11 for *gap1*, *gap2*, *gap3*, and *rnd1* respectively.

Dynamic problems are constructed from these instances in three ways (modes): a machine delete mode (MDM), a part add mode (PAM), and a machine swap mode (MSM).

Machine delete mode The MDM mode reflects the real-world scenarios in which a machine suddenly breaks down. The change step of this mode is the deletion of a single machine.

Part add mode The PAM mode reflects the addition and deletion of new assignments (parts). The step of change in this mode is the addition or the deletion of a single part. This mode requires variable representation to reflect the increase or decrease in the number of operations associated with the changing parts.

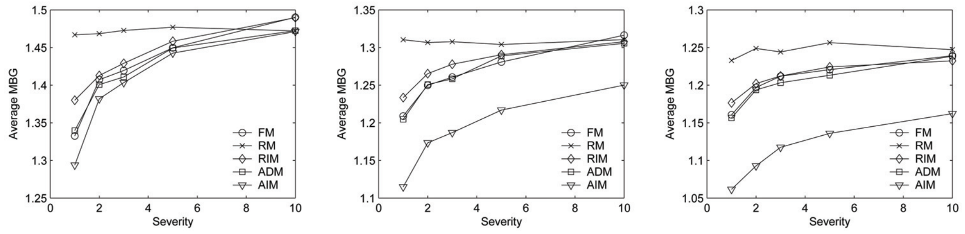
Machine swap mode The MSM mode is a direct application of the mapping-based benchmark generation scheme (Younes et al. 2005). By interchanging machine labels, a dynamic FMS can be generated easily and quickly. The change step in this mode is an interchange of a single pair of machines. As a mapping change scheme, this mode does not require computing a new solution after each change. We only need to swap the machines of the current optimal solution to determine the optimum of the next instance.

In the current experimentation, each benchmark problem is created from an initial sequence of 100 static problems inter-separated by single elementary steps. Depending on the specified severity, a number of intermediate static problems will be skipped to construct one test problem.

Each sequence of static problems is translated into 18 dynamic test problems by combining seven degrees of severity (1, 2, 3, 5, 10 steps per shift, and random) and three periods of change (500, 2500, and 5000 evaluations per shift, which correspond to 10, 50, and 100 generations per shift based on a population of 50 individuals).

5.4.2 FMS results

Experiments were conducted on the rnd1, gap1, gap2, and gap3 problems in the three modes of environmental change. In this section, we focus on the gap1 problem, the largest and presumably the hardest, and on the rnd1 problem, the most distinct. Results of comparisons in the MSM mode are shown in Figure 9, where the average MBG (over ten runs) is plotted against different values of severity. First, we notice that results of the RM model are inferior to those of the other models when the change severity is small. As severity increases, RM results become comparatively better, and at extreme severities RM outperforms the other models. This trend is consistent over different periods of environmental change confirming our notion that restart strategies are best used when the problem changes completely; i.e., when no benefits are expected from re-using old information.

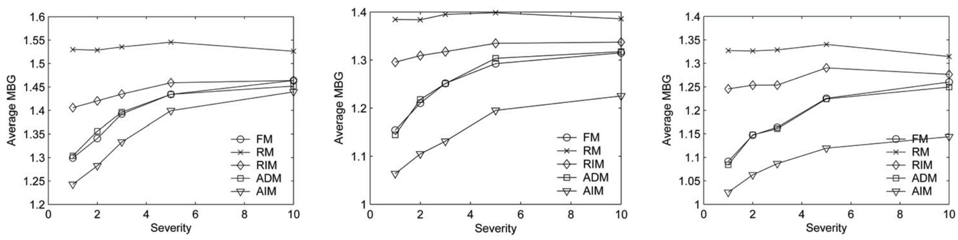


Period = 10 generations Period = 50 generations Period = 100 generations

Fig. 9. Comparison of evolutionary models (rnd1_MSM)

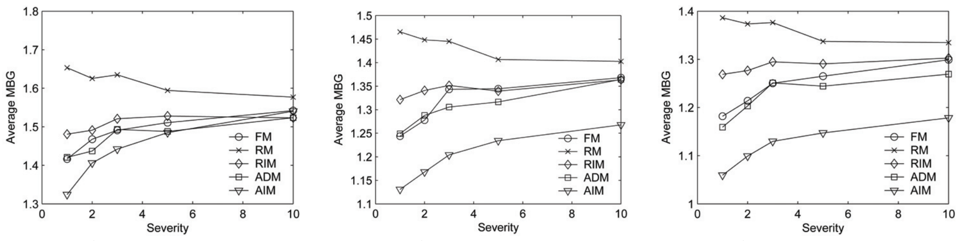
Starting with the ten generation period, we notice that models that reuse old information (all models except for RM) give comparable performance. However, as the period of change increases, differences between their performance become more apparent. This trend can be explained as follows: when the environmental change is fast, the models do not have sufficient time to converge, and hence they give nearly the same results. When allowed more time, the models start to converge, and those using the best approach to persevere after obsolete convergence produce the best results. The AIM model clearly stands out as the best model.

Comparing the five models on the PAM and MDM modes confirms the results obtained on the MSM mode. The inferiority of the RM model and the superiority of the AIM model persist, as can be seen in Figure 10 and Figure 11.



Period = 10 generations Period = 50 generations Period = 100 generations

Fig. 10. Comparison of evolutionary models (rnd1_PAM)

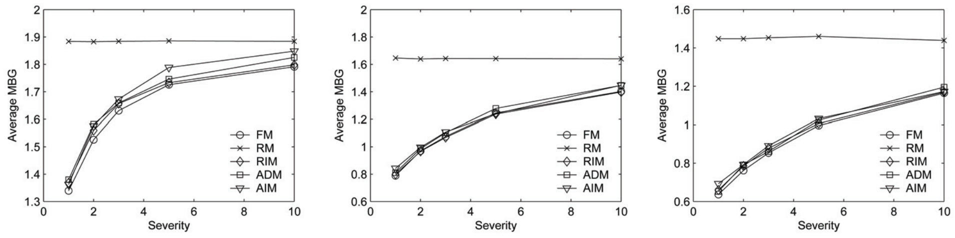


Period = 10 generations Period = 50 generations Period = 100 generations

Fig. 11. Comparison of evolutionary models (rnd1_MDM)

The inferior performance of the RM model is more apparent in the other, large, test problems: the performance of the RM model is consistently poor across the problem dynamics whereas the performance of the other models deteriorates as the severity of environmental change increases. Figure 12 shows the case of gap1 in the MSM mode (other modes show similar behaviour). Comparing the gap1 results to those of rnd1, the apparent deterioration of RM (relative to the other models) in the case of gap1 can be explained by examining change severity. Although values of severity are numerically the same in both cases, relative to problem size they are different, since gap1 is larger than rnd1. In other words, the severity range used in the experiments on gap1 is virtually less than that used on rnd1.

In summary, we can conclude that AIM is the best of the five models, as illustrated clearly in the rnd1 experiments. For other problems in which AIM seems to produce comparable results to those of the other models, we can still opt for the AIM model as it offers the additional advantage of being easy to parallelize, as mentioned in the TSP results section.



Period = 10 generations Period = 50 generations Period = 100 generations

Fig. 12. Comparison of evolutionary models (gap1 MSM)

6. Conclusions and future work

The island based model proves to be effective under different dynamics. Although statistical analysis suggest that these benefits are not significant under some problem dynamics, this model can be more rewarding if several processors are employed. With each island allocated to a different processor, the per processor computational costs are reduced significantly.

The problem of parameter tuning is aggravated with dynamic environments, as a result of the increased problem complexity and the increased number of algorithm parameters; however, by using diversity to control the EA parameters, the models developed in this article had significantly reduced tuning efforts.

There are several ways in which the developed models can be applied and improved:

- The effectiveness of the developed methods on the TSP and FMS problems encourages their application to other problems, such as intelligent transportation systems, engine parameter control, scheduling of airline maintenance, and dynamic network routing.
- Diversity controlled models can use operator-specific diversity measures so that each operator is controlled by its respective diversity measure, i.e., based on algorithmic distance. Future work that is worth exploring involves using adaptive limits of diversity for the models presented in this article.

7. Appendix. Statistical analysis

Statistical t-tests that are used to compare the means of two samples can be used to compare the performance of two algorithms. The typical t-test is performed to build a confidence interval that is used to either accept or reject a null hypothesis that both sample means are equal. In applying this test to compare the performance of two algorithms, the measures of performance are treated as sample means, the required replicates of each sample mean are obtained by performing several independent runs of each algorithm, and the null hypothesis is that there is no significant difference in the performance of both algorithms. However, when more than two samples are compared, the probability of multiple t-tests incorrectly finding a significant difference between a pair of samples increases with the number of comparisons. Analysis of variance (ANOVA) overcomes this problem by testing the samples as a whole for significant differences. Therefore, in this article, ANOVA is performed to test the hypothesis that measures of performance of all the models under considerations are equal. Then, a multiple post ANOVA comparison test, known as Tukey's test, is carried out to produce 95% confidence intervals for the difference in the mean best of generation of each pair of models.

Statistical results reported here are obtained using a significance level of 5% to construct 95% confidence intervals on the difference in the mean best of generation. Tables in this section summarize the statistical computations of the results reported in Section 5: Table 1, Table 2, and Table 3 are for TSP K100 problem in the three modes of change (respectively, ECM, IDM, and VSM); Table 4 and Table 5 are for the FMS rnd1 and gap1 problems in the MSM mode.

period	10							100							1000						
severity	1	5	10	15	20	25	r	1	5	10	15	20	25	r	1	5	10	15	20	25	r
FM−RM	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	0	-1	0	-1	-1	-1	-1	-1	-1
FM−RIM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FM−ADM	1	1	0	0	-1	-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FM−AIM	1	1	0	0	-1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RM−RIM	1	1	1	1	0	0	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1
RM−ADM	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RM−AIM	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RIM−ADM	1	1	0	0	-1	-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RIM−AIM	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ADM−AIM	0	0	0	0	0	0	0	0	-1	0	-1	0	0	0	1	0	-1	-1	-1	-1	0

Table 1. Multiple comparison test of evolutionary models (k100-VSM)

period	10							100							1000						
severity	1	5	10	15	20	25	r	1	5	10	15	20	25	r	1	5	10	15	20	25	r
FM−RM	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
FM−RIM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FM−ADM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FM−AIM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RM−RIM	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RM−ADM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RM−AIM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RIM−ADM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RIM−AIM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ADM−AIM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1

Table 2. Multiple comparison test of evolutionary models (k100-ECM)

period	10							100							1000						
severity	1	5	10	15	20	25	r	1	5	10	15	20	25	r	1	5	10	15	20	25	r
FM−RM	-1	-1	0	0	0	0	-1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
FM−RIM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
FM−ADM	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FM−AIM	1	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	0	1
RM−RIM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RM−ADM	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RM−AIM	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RIM−ADM	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RIM−AIM	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ADM−AIM	0	-1	0	-1	-1	0	-1	0	-1	0	0	-1	0	0	0	0	0	0	0	0	0

Table 3. Multiple comparison test of evolutionary models (k100-IDM)

Each table covers the combinations of problem dynamics (periods of change and levels of severity of change) described earlier, and an additional column for a random severity) The entries in these tables are interpreted as follows. An entry of 1 signifies that the confidence interval for the difference in performance measures of the corresponding pair consists entirely of positive values, which indicates that the first model is inferior to the second model. Conversely, an entry of -1 signifies that the confidence interval for the corresponding pair consists entirely of negative values, which indicates that the first model is superior to the second one. An entry of 0 indicates that there is no significant difference between the two models.

period	10						100						1000					
severity	1	2	3	5	10	r	1	2	3	5	10	r	1	2	3	5	10	r
FM−RM	-1	-1	-1	0	0	-1	-1	-1	-1	0	0	-1	-1	-1	0	0	0	-1
FM−RIM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FM−ADM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FM−AIM	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
RM−RIM	1	1	1	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1
RM−ADM	1	1	1	0	0	1	1	1	1	0	0	1	1	1	0	0	0	1
RM−AIM	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
RIM−ADM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RIM−AIM	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	1
ADM−AIM	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Table 4. Multiple comparison test of evolutionary models (rnd1-MSM)

Statistical analysis confirms the arguments made on the graphical comparisons in the previous section. As can be seen in Table 1, 2, and 3, there are significant differences between the performance of the adaptive models (ADM and AIM) and the other three models (FM, RM, and RIM), while there is no significant difference between ADM and AIM. Collectively, the statistical tables confirm the graphical comparisons presented in the previous section. As can be seen in Table 4, and 5, there are significant differences between the performance of the RM model and all others.

period	10						100						1000					
severity	1	2	3	5	10	r	1	2	3	5	10	r	1	2	3	5	10	r
FM−RM	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
FM−RIM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FM−ADM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FM−AIM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RM−RIM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RM−ADM	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
RM−AIM	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
RIM−ADM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RIM−AIM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADM−AIM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5. Multiple comparison test of evolutionary models (gap1-MSM)

8. References

Beasley, J. E. 1990. Or-library: distributing test problems by electronic mail. *Journal of the Operational Research Society 41(11)*, 1069–1072.

Bianchi, L. 1990. Notes on dynamic vehicle routing - the state of the art. Tech. Rep. idsia 05-01, Italy.

Bierwirth, C. and Kopfer, H. 1994. Dynamic task scheduling with genetic algorithms in manufacturing systems. Tech. rep., Department of Economics, University of Bremen, Germany.

Bierwirth, C., Kopfer, H., Mattfeld, D. C., and Rixen, I. 1995. Genetic algorithm based scheduling in a dynamic manufacturing environment. In *Proc. of IEEE Conference on Evolutionary Computation*. IEEE Press.

Bierwirth, C. and Mattfeld, D. C. 1999. Production scheduling and rescheduling with genetic algorithms. *Evolutionary Computation 7,* 1, 1–18.

Branke, J. 1999. Memory enhanced evolutionary algorithms for changing optimization problems. In *1999 Congress on Evolutionary Computation*. IEEE Service Center, Piscataway, NJ, 1875–1882.

Branke, J. 2001. *Evolutionary Optimization in Dynamic Environments*. Environments. Kluwer Academic Publishers.

Branke, J., Kaussler, T., Schmidt, C., and Schmeck, H. 2000. A multi-population approach to dynamic optimization problaqw2 nh ems. In *Adaptive Computing in Design and Manufacturing 2000.* Springer.

Burke, E. K., Gustafson, S., and Kendall, G. 2004. Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation 8,* 1, 47–62.

Chen, J.-H. and Ho, S.-Y. 2002. Multi-objective evolutionary optimization of flexible manufacturing systems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2002).* Morgan Koffman, New York, New York, 1260–1267.

Chu, P. C. and Beasley, J. E. 1997. A genetic algorithm for the generalised assignment problem. *Computers and Operations Research 24*, 17–23.

Cobb, H. G. 1990. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Tech. Rep. 6760 (NLR Memorandum), Navy Center for Applied Research in Artificial Intelligence,Washington, D.C.

Dimopoulos, C. and Zalzala, A. 2000. Recent developments in evolutionary computation for manufacturing optimization: Problems, solutions, and comparisons. *IEE Transactions on Evolutionary Computation 4,* 2, 93–113.

Eiben, A. E., Hinterding, R., and Michalewicz, Z. 1999. Parameter control in evolutionary algorithms. *IEEE Trans. on Evolutionary Computation 3,* 2, 124–141.

Eyckelhof, C. J. and Snoek, M. 2002. Ant systems for a dynamic tsp. In *ANTS '02: Proceedings of the Third InternationalWorkshop on Ant Algorithms*. Springer Verlag, London, UK, 88–99.

Grefenstette, J. J. 1992. Genetic algorithms for changing environments. In *Parallel Problem Solving from Nature 2 (Proc. 2nd Int. Conf. on Parallel Problem Solving from Nature, Brussels 1992)*, R. M¨anner and B. Manderick, Eds. Elsevier, Amsterdam, 137–144.

Grefenstette, J. J. 1999. Evolvability in dynamic fitness landscapes: a genetic algorithm approach. In *1999 Congress on Evolutionary Computation*. IEEE Service Center, Piscataway, NJ, 2031–2038.

Guntsch, M., Middendorf, M., and Schmeck, H. 2001. An ant colony optimization approach to dynamic tsp. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, L. Spector, E. D. Goodman, A.Wu,W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds. Morgan Kaufmann, San Francisco, California, USA, 860–867.

Jin, Y. and Branke, J. 2005. Evolutionary optimization in uncertain environments-a survey. *IEEE Trans. Evolutionary Computation 9,* 3, 303–317.

Lewis, J., Hart, E., and Ritchie, G. 1998. A comparison of dominance mechanisms and simple mutation on non-stationary problems. In *Parallel Problem Solving from Nature –*

PPSN V, A. E. Eiben, T. B¨ack, M. Schoenauer, and H.-P. Schwefel, Eds. Springer, Berlin, 139–148. Lecture Notes in Computer Science 1498.

Lin, S.-C., Goodman, E. D., and Punch, W. F. 1997. A genetic algorithm approach to dynamic job shop scheduling problems. In *Seventh International Conference on Genetic Algorithms*, T. B¨ack, Ed. Morgan Kaufmann, 481–488.

Louis, S. J. and Johnson, J. 1997. Solving similar problems using genetic algorithms and case-based memory. In *Proc. of The Seventh Int. Conf. on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA, 283–290.

Louis, S. J. and Xu, Z. 1996. Genetic algorithms for open shop scheduling and rescheduling. In *ISCA 11th Int. Conf. on Computers and their Applications*, M. E. Cohen and D. L. Hudson, Eds. 99–102.

Ng, K. P. and Wong, K. C. 1995. A new diploid scheme and dominance change mechanism for non-stationary function optimization. In *Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann, 159–166.

Psaraftis, H. 1995. Dynamic vehicle routing: Status and prospects. *Annals Operations Research 61*, 143–164.

Rardin, R. 1998. *Optimization In Operation Research*. Prentice-Hall, Inc.

Reeves, C. and Karatza, H. 1993. Dynamic sequencing of a multi-processor system: a genetic algorithm approach. In *Artificial Neural Nets and Genetic Algorithms*, R. F. Albrecht, C. R. Reeves, and N. C. Steele, Eds. Springer, 491–495.

Reeves, C. R. and Rowe, J. E. 2002. *Genetic Algorithms: Principles and Perspectives: A Guide to GA Theory*. Kluwer Academic Publishers, Norwell, MA, USA.

Reinelt, G. 1991. TSPLIB — a traveling salesman problem library. *ORSA Journal on Computing 3*, 376 – 384.

Riget, J. and Vesterstroem, J. 2002. A diversity-guided particle swarm optimizer – the arpso.

Tanese, R. 1989. Distributed genetic algorithm. In *Proc. of the Third Int. Conf. on Genetic Algorithms*, J. D. Schaffer, Ed. Morgan Kaufmann, San Mateo, CA, 434–439.

Ursem, R. K. 2000. Multinational GAs: Multimodal optimization techniques in dynamic environments. In *Proc. of the Genetic and Evolutionary Computation Conf. (GECCO-00)*, D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds. Morga Kaufmann, San Francisco, CA, 19–26.

Ursem, R. K. 2002. Diversity-guided evolutionary algorithms. In *Proceedings of Paralle Problem Solving from Nature VII (PPSN-2002)*. Springer Verlag, 462–471.

Wang, C., Ghenniwa, H., and Shen, W. 2005. Heuristic scheduling algorithm for flexibl manufacturing systems with partially overlapping machine capabilities. In *Proc. Of 2005 IEEE International Conference on Mechatronics and Automation,* IEEE Press, Niagara Falls, Canada, 1139 1144.

Whitley, D. and Starkweather, T. 1990. Genitor ii.: a distributed geneti algorithm. *J. Exp. Theor. Artif. Intell. 2,* 3, 189–214.

Whitley, D., Starkweather, T., and Shaner, D. 1991. The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination. In *Handbook o Genetic Algorithms*, L. Davis, Ed. Van Nostrand Reinhold, New York, 350–372.

Wineberg, M. and Oppacher, F. 2000. Enhancing the ga's ability to cope with dynamic environments. In *GECCO*. 3–10.

Younes, A., Calamai, P., and Basir, O. 2005. Generalized benchmark generation for dynamic combinatorial problems. In *Genetic and Evolutionary Computation Conference (GECCO2005) workshop program*. ACM Press,Washington, D.C., USA, 25–31.

Younes, A., Ghenniwa, H., and Areibi, S. 2002. An adaptive genetic algorithm for multi objective flexible manufacturing systems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2002)*. Morgan Koffman, New York, New York, 1241–1249.

Zhu, K. Q. 2003. A diversity-controlling adaptive genetic algorithm for the vehicle routing problem with time windows. In *ICTAI*. 176–183.

Agent-Based Co-Evolutionary Techniques for Solving Multi-Objective Optimization Problems

Rafał Dreżewski and Leszek Siwik
AGH University of Science and Technology
Poland

1. Introduction

Evolutionary algorithms (EAs) are optimization and search techniques inspired by the Darwinian model of biological evolutionary processes (Bäck et al., 1997). EAs are robust and efficient techniques, which find approximate solutions to many problems which are difficult or even impossible to solve with the use of "classical" techniques. There are many different types of evolutionary algorithms developed during over 40 years of research.
One of the branches of EAs are *co-evolutionary algorithms (CEAs)* (Paredis, 1998). The main difference between EAs and CEAs is the way in which the fitness of an individual is evaluated in each approach. In the case of evolutionary algorithms each individual has the solution of the given problem encoded within its genotype and its fitness depends only on how "good" is that solution. In the case of co-evolutionary algorithms of course there is also obviously solution to the given problem encoded within the individual's genotype but the fitness is estimated on the basis of interactions of the given individual with other individuals present in the population. Thus co-evolutionary algorithms are applicable in the case of problems for which it is difficult or even impossible to formulate explicit fitness function—in such cases we can just encode the solutions within the individuals' genotypes and individuals compete—or co-operate—with each other, and such process of interactions leads to the fitness estimation. Co-evolutionary interactions between individuals have also other positive effects. One of them is maintaining the population diversity, another one are "arms races"—continuous "progress" toward better and better solutions to the given problem via competition between species.
Co-evolutionary algorithms are classified into two general categories: competitive and cooperative (Paredis, 1998). The main difference between these two types of co-evolutionary algorithms is the way in which the individuals interact during the fitness estimation. In the case of competitive co-evolutionary algorithms the value of fitness is estimated as a result of the series of tournaments, in which the individual for which the fitness is estimated and some other individuals from the population are engaged. The way of choosing the competitors for tournaments may vary in different versions of algorithms—for example it may be the competition with the best individual from the other species or competition with several randomly chosen individuals, etc.
On the other hand, co-operative co-evolutionary algorithms (CCEAs) are CEAs in which there exist several sub-populations (species) (Potter & De Jong, 2000). Each of them solves

only one sub- problem of the given problem. In such a case the whole solution is the group of individuals composed of the representants of all sub-populations. Individuals interact only during the fitness estimation process. In order to evaluate the given individual, representants from the other sub-populations are chosen (different ways of choosing such representants may be found in (Potter & De Jong, 2000)). Within the group the given individual is evaluated in such a way that the fitness value of the whole solution (group) becomes the fitness value of the given individual. Individuals coming from the same species are evaluated within the group composed of the same representants of other species.

Sexual selection is another mechanism used for maintaining population diversity in EAs. Sexual selection results from the co-evolution of female mate choice and male displayed trait (Gavrilets & Waxman, 2002). Sexual selection is considered to be one of the ecological mechanisms responsible for biodiversity and sympatric speciation (Gavrilets &Waxman, 2002; Todd & Miller, 1997). The research on sexual selection mechanism generally concentrated on two aspects. The first one was modeling and simulation of sexual selection as speciation mechanism and population diversity mechanism (for example see (Gavrilets &Waxman, 2002; Todd & Miller, 1997)). The second one was the application of sexual selection in evolutionary algorithms as a mechanism for maintaining population diversity. The applications of sexual selection include multi-objective optimization (Allenson, 1992; Lis & Eiben, 1996) and multimodal optimization (Ratford et al., 1997).

In the case of evolutionary multi-objective optimization (Deb, 1999), high quality approximation of *Pareto frontier* (basic ideas of multi-objective optimization are introduced in Section 2) should fulfill at least three distinguishing features. First of all, the population should be "located" as close to the ideal Pareto frontier as possible. Secondly it should include as many alternatives (individuals) as possible and, last but not least, all proposed non-dominated alternatives should be evenly distributed over the whole true Pareto set. In the case of multi-objective optimization maintaining of population diversity plays the crucial role. Premature loss of population diversity can result not only in lack of drifting to the true Pareto frontier but also in obtaining approximation of Pareto set that is focused around its selected area(s), what is very undesirable. In the case of multi-objective problems with many local Pareto frontiers (so called "multi-modal multi-objective problems" defined by Deb in (Deb, 1999)) the loss of population diversity may result in locating only a local Pareto frontier instead of a global one.

Co-evolutionary multi-agent systems (CoEMAS) are the result of research on decentralized models of co-evolutionary computations. CoEMAS model is the extension of "basic" model of evolution in multi-agent system—*evolutionary multi-agent systems (EMAS)* (Cetnarowicz et al., 1996). The basic idea of such an approach is the realization of evolutionary processes in multi-agent system—the population of agents evolves, agents live within the environment, they can reproduce, die, compete for resources, observe the environment, communicate with other agents, and make autonomously all their decisions concerning reproduction, choosing partner for reproduction, and so on. Co-evolutionary multi-agent systems additionally allow us to define many species and sexes of agents and to introduce interactions between them (Dreżewski, 2003).

All these features lead to completely decentralized evolutionary processes and to the class of systems that have very interesting features. It seems that the most important of them are the following:

- synchronization constraints of the computations are relaxed because the evolutionary processes are decentralized—individuals are agents, which act independently and do not need synchronization,
- there exists the possibility of constructing hybrid systems using many different computational intelligence techniques within one single, coherent multi-agent architecture,
- there are possibilities of introducing new evolutionary and social mechanisms, which were hard or even impossible to introduce in the case of classical evolutionary algorithms.

The possible areas of application of CoEMAS include multi-modal optimization (for example see (Dreżewski, 2006)), multi-objective optimization (the review of selected results is presented in this chapter), and modeling and simulation of social and economical phenomena.

This chapter starts with the overview of multi-objective optimization problems. Next, introduction to the basic ideas of CoEMAS systems—the general model of co-evolution in multi-agent system—is presented. In the following parts of the chapter the agent-based co-evolutionary systems for multi-objective optimization are presented. Each system is described with the use of notions and formalisms introduced in the general model of coevolution in multi-agent system. Each of the presented systems uses different coevolutionary interactions and mechanisms: sexual selection mechanism, and host-parasite co-evolution. For all the systems results of experiments with commonly used multi-objective test problems are presented. The results obtained during the experiments are the basis for comparisons of agent-based co-evolutionary techniques with “classical” evolutionary approaches.

2. An introduction to multi-objective optimization

During most real-life decision processes many different (often contradictory) factors have to be considered, and the decision maker has to deal with an ambiguous situation: the solutions which optimize one criterion may prove insufficiently good considering the others. From the mathematical point of view such multi-objective (or multi-criteria) problem can be formulated as follows (Coello Coello et al., 2007; Abraham et al., 2005; Zitzler, 1999; Van Veldhuizen, 1999).

Let the problem variables be represented by a real-valued vector:

$$\vec{x} = [x_1, x_2, \ldots, x_m]^T \in \mathbb{R}^m \tag{1}$$

where m is the number of variables. Then a subset of $\mathbb{R}^m$ of all possible (feasible) decision alternatives (options) can be defined by a system of:

- inequalities (constraints): $g_k(\vec{x}) \geq 0$ and $k = 1, 2, \ldots, K$
- equalities (bounds): $h_l(\vec{x}) = 0$, $l = 1, 2, \ldots, L$

and denoted by $\mathcal{D}$. The alternatives are evaluated by a system of n functions (objectives) denoted here by vector $F = [f_1, f_2, \ldots, f_n]^T$:

$$f_i : \mathbb{R}^m \to \mathbb{R}, \quad i = 1, 2, \ldots, n \tag{2}$$

Because there are many criteria–to indicate which solution is better than the other–specialized ordering relation has to be introduced. To avoid problems with converting minimization to maximization problems (and vice versa of course) additional operator $\lhd$ can be defined. Then, notation $\bar{x}_1 \lhd \bar{x}_2$ indicates that solution $\bar{x}_1$ is simply better than solution $\bar{x}_2$ for particular objective. Now, the crucial concept of Pareto optimality (what is the subject of our research) i.e. so called dominance relation can be defined. It is said that solution $\bar{x}_A$ dominates solution $\bar{x}_B$ ($\bar{x}_A \prec \bar{x}_B$) if and only if:

$$\bar{x}_A \prec \bar{x}_B \Leftrightarrow \begin{cases} f_j(\bar{x}_A) \ntriangleright f_j(\bar{x}_B) \quad for \quad j = 1, 2 \ldots, n \\ \exists i \in \{1, 2, \ldots, n\} : \quad f_i(\bar{x}_A) \lhd f_i(\bar{x}_B) \end{cases}$$

A solution in the Pareto sense of the multi-objective optimization problem means determination of all non-dominated alternatives from the set $\mathcal{D}$. The Pareto-optimal set consists of globally optimal solutions and is defined as follows. The set $\mathcal{P} \subseteq D$ is global Pareto-optimal set if (Zitzler, 1999):

$$\forall \vec{x}^a \in \mathcal{P} : \ \nexists \vec{x}^b \in D \text{ such that } \vec{x}^b \geq \vec{x}^a \tag{3}$$

There may also exist locally optimal solutions, which constitute locally non-dominated set (*local Pareto-optimal set*) (Deb, 2001). The set $\mathcal{P}_{local} \subseteq D$ is local Pareto-optimal set if (Zitzler, 1999):

$$\forall \vec{x}^a \in \mathcal{P}_{local} : \ \nexists \vec{x}^b \in D \text{ such that}$$
$$\vec{x}^b \geq \vec{x}^a \wedge \|\vec{x}^b - \vec{x}^a\| < \varepsilon \wedge \|F(\vec{x}^b) - F(\vec{x}^a)\| < \delta$$

where $\|\cdot\|$ is a distance metric and $\varepsilon > 0$, $\delta > 0$.

These locally or globally non-dominated solutions define in the criteria space so-called local ($\mathcal{PF}_{local}$) or global ($\mathcal{PF}$) Pareto frontiers that can be defined as follows:

$$\mathcal{PF}_{local} = \{\vec{y} = F(\vec{x}) \in \mathbb{R}^n \mid \vec{x} \in \mathcal{P}_{local}\} \tag{4a}$$

$$\mathcal{PF} = \{\vec{y} = F(\vec{x}) \in \mathbb{R}^n \mid \vec{x} \in \mathcal{P}\} \tag{4b}$$

Multi-objective problems with one global and many local Pareto frontiers are called *multimodal multi-objective problems* (Deb, 2001).

3. General model of co-evolution in multi-agent system

As it was said, co-evolutionary multi-agent systems are the result of research on decentralized models of evolutionary computations which resulted in the realization of evolutionary processes in multi-agent system and the formulation of model of co-evolution in such system. The basic elements of CoEMAS are environment with some topography, agents (which are located and can migrate within the environment, which are able to reproduce, die, compete for limited resources, and communicate with each other), the selection mechanism based on competition for limited resources, and some agent-agent and agent-environment relations defined (see Fig. 1).

The selection mechanism in such systems is based on the resources defined in the system. Agents collect such resources, which are given to them by the environment in such a way that "better" agents (i.e. which have "better" solutions encoded within their genotypes) are given more resources and "worse" agents are given less resources. Agents then use such resources for every activity (like reproduction and migration) and base all their decisions on the possessed amount of resources.

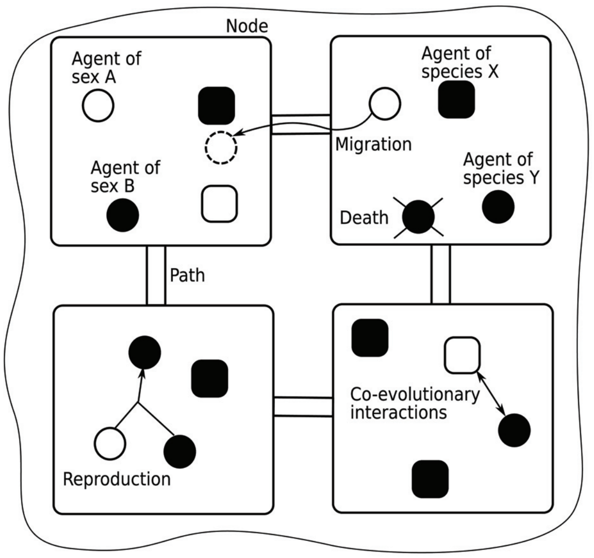


Fig. 1. The idea of co-evolutionary multi-agent system

In this section the general model of co-evolution in multi-agent system (CoEMAS) is presented. We will formally describe the basic elements of such systems and present the algorithm of agent's basic activities.

3.1 The co-evolutionary multi-agent system

The *CoEMAS* is described as 4-tuple:

$$CoEMAS = \langle E, S, \Gamma, \Omega \rangle \tag{5}$$

where E is the environment of the *CoEMAS* , S is the set of species ($s \in S$) that co-evolve in *CoEMAS*, Γ is the set of resource types that exist in the system, the amount of type γ resource will be denoted by r^{γ}, Ω is the set of information types that exist in the system, the information of type ω will be denoted by i^{ω}.

3.2 The environment

The environment of *CoEMAS* may be described as 3-tuple:

$$E = \left\langle T^E, \Gamma^E, \Omega^E \right\rangle \tag{6}$$

where T^E is the topography of environment E, Γ^E is the set of resource types that exist in the environment, Ω^E is the set of information types that exist in the environment. The topography of the environment is given by:

$$T^E = \langle H, l \rangle \tag{7}$$

where H is directed graph with the cost function c defined: $H = \langle V, B, c \rangle$, V is the set of vertices, B is the set of arches. The distance between two nodes is defined as the length of the shortest path between them in graph H.
The l function makes it possible to locate particular agent in the environment space:

$$l: \quad A \to V \tag{8}$$

where A is the set of agents, that exist in *CoEMAS* .
Vertice v is given by:

$$v = \langle A^v, \Gamma^v, \Omega^v, \varphi \rangle \tag{9}$$

A^v is the set of agents that are located in the vertice v, Γ^v is the set of resource types that exist within the v ($\Gamma^v \subseteq \Gamma^E$), Ω^v is the set of information types that exist within the v ($\Omega^v \subseteq \Omega^E$), φ is the fitness function.

3.3 The species

Species $s \in S$ is defined as follows:

$$s = \langle A^s, SX^s, Z^s, C^s \rangle \tag{10}$$

where:

- A^s is the set of agents of species s (by a^s we will denote the agent, which is of species s, $a^s \in A^s$);
- SX^s is the set of sexes within the s;
- Z^s is the set of actions, which can be performed by the agents of species s ($Z^s = \bigcup_{a \in A^s} Z^a$, where Z^a is the set of actions, which can be performed by the agent a);
- C^s is the set of relations with other species that exist within *CoEMAS*.

The set of relations of s_i with other species (C^{s_i}) is the sum of the following sets of relations:

$$C^{s_i} = \left\{ \xrightarrow{s_i, z-} : z \in Z^{s_i} \right\} \cup \left\{ \xrightarrow{s_i, z+} : z \in Z^{s_i} \right\} \tag{11}$$

where $\xrightarrow{s_i, z-}$ and $\xrightarrow{s_i, z+}$ are relations between species, based on some actions $z \in Z^{s_i}$, which can be performed by the agents of species s_i:

$$\xrightarrow{s_i, z-} = \{ \langle s_i, s_j \rangle \in S \times S \text{: agents of species } s_i \text{ can decrease the fitness of agents of species } s_j \text{ by performing the action } z \in Z^{s_i} \} \tag{12}$$

$$\xrightarrow{s_i, z+} = \{ \langle s_i, s_j \rangle \in S \times S \text{: agents of species } s_i \text{ can increase the fitness of agents of species } s_j \text{ by performing the action } z \in Z^{s_i} \} \tag{13}$$

If $s_i \xrightarrow{s_i, z_k -} s_i$ then we are dealing with the intra-species competition, for example the competition for limited resources, and if $s_i \xrightarrow{s_i, z_l +} s_i$ then there is some form of co-operation within the species s_i.

With the use of the above relations we can define many different co-evolutionary interactions e.g.: predator-prey, host-parasite, mutualism, etc. For example, host-parasite interactions between two species, s_i (parasites) and s_j (hosts) ($i \neq j$) take place if and only if $\exists z_k \in Z^{s_i} \wedge \exists z_l \in Z^{s_i}$, such that $s_i \xrightarrow{s_i, z_k-} s_j$ and $s_j \xrightarrow{s_j, z_l+} s_i$, and parasite can only live in tight co-existence with the host.

3.4 The sex

The sex $sx \in SX^s$ which is within the species s is defined as follows:

$$sx = \langle A^{sx}, Z^{sx}, C^{sx} \rangle \tag{14}$$

where A^{sx} is the set of agents of sex sx and species s ($A^{sx} \subseteq A^s$):

$$A^{sx} = \left\{ a : a \in A^s \wedge a \text{ is the agent of sex } sx \right\} \tag{15}$$

With a^{sx} we will denote the agent of sex sx ($a^{sx} \in A^{sx}$). Z^{sx} is the set of actions which can be performed by the agents of sex sx, $Z^{sx} = \bigcup_{a \in A^{SX}} Z^a$, where Z^a is the set of actions which can be performed by the agent a. And finally C^{sx} is the set of relations between the sx and other sexes of the species s.

Analogically as in the case of species, we can define the relations between the sexes of the same species. The set of all relations of the sex $sx_i \in S\,X^s$ with other sexes of species s (C^{sx_i}) is the sum of the following sets of relations:

$$C^{sx_i} = \left\{ \xrightarrow{sx_i, z-} : z \in Z^{sx_i} \right\} \cup \left\{ \xrightarrow{sx_i, z+} : z \in Z^{sx_i} \right\} \tag{16}$$

where $\xrightarrow{sx_i, z-}$ and $\xrightarrow{sx_i, z+}$ are the relations between sexes, in which some actions $z \in Z^{sx_i}$ are used:

$$\begin{aligned}\xrightarrow{sx_i, z-} = \{ \langle sx_i, sx_j \rangle \in SX^s \times SX^s : & \text{ agents of sex } sx_i \text{ can decrease the fitness of} \\ & \text{agents of sex } sx_j \text{ by performing the action } z \in Z^{sx_i} \}\end{aligned} \tag{17}$$

$$\begin{aligned}\xrightarrow{sx_i, z+} = \{ \langle sx_i, sx_j \rangle \in SX^s \times SX^s : & \text{ agents of sex } sx_i \text{ can increase the fitness of} \\ & \text{agents of sex } sx_j \text{ by performing the action } z \in Z^{sx_i} \}\end{aligned} \tag{18}$$

If performing the action $z_k \in Z^{sx_i}$ (which permanently or temporally increases the fitness of the agent a^{sx_j} of sex $sx_j \in SX^s$) by the agent a^{sx_i} of sex $sx_i \in SX^s$ results in performing the action $z_l \in Z^{sx_i}$ by the agent a^{sx_i} and performing the action $z_m \in Z^{sx_j}$ by the agent a^{sx_j}, what results in decreasing of the fitness of agents a^{sx_i} and a^{sx_j} then such relation $\xrightarrow[z_l-, z_m-]{sx_i, z_k+}$ will be defined in the following way:

$$\xrightarrow[z_l-,z_m-]{sx_i,z_k+} = \{ \langle sx_i, sx_j \rangle \in SX^s \times SX^s : \text{agents of sex } sx_i \text{ can increase (permanently or temporally) the fitness of the agents of sex } sx_j \text{, by performing the action } z_k \in Z^{sx_i} \text{, which results in performing the action } z_l \in Z^{sx_i} \text{ and the action } z_m \in Z^{sx_j} \text{, which decrease the fitness of the agents of sex } sx_i \text{ and } sx_j \} \quad (19)$$

Such relation represents the sexual selection mechanism, where the action $z_k \in Z^{sx_i}$ is the action of choosing the partner for reproduction, the action $z_l \in Z^{sx_i}$ is the action of reproduction performed by the agent of sex sx_i (with high costs associated with it) and the $z_m \in Z^{sx_j}$ is the action of reproduction performed by the agent of sex sx_j (with lower costs than in the case of z_l action).

3.5 Agent

Agent a (see Fig. 2) of sex sx and species s (in order to simplify the notation we assume that $a \equiv a^{sx,s}$) is defined as follows:

$$a = \langle gn^a, Z^a, \Gamma^a, \Omega^a, PR^a \rangle \quad (20)$$

where:

- gn^a is the genotype of agent a, which may be composed of any number of chromosomes (for example: $gn^a = \langle (x_1, x_2, \ldots, x_k) \rangle$, where $x_i \in \mathbb{R}$, $gn^a \in \mathbb{R}^k$
- Z^a is the set of actions, which agent a can perform;
- Γ^a is the set of resource types, which are used by agent a ($\Gamma^a \subseteq \Gamma$);
- Ω^a is the set of informations, which agent a can possess and use ($\Omega^a \subseteq \Omega$);
- PR^a is partially ordered set of profiles of agent a ($PR^a \equiv \langle PR^a, \trianglelefteq \rangle$) with defined partial order relation $\trianglelefteq$.

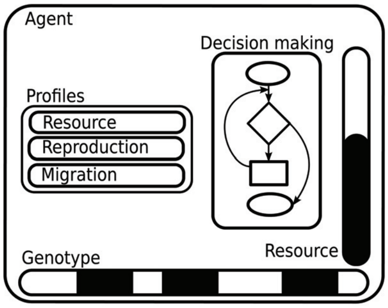


Fig. 2. Agent in the *CoEMAS*

Relation $\trianglelefteq$ is defined in the following way:

$$\trianglelefteq = \{ \langle pr_i, pr_j \rangle \in PR^a \times PR^a : \text{realization of active goals of profile } pr_i \text{ has equal or higher priority than the realization of active goals of profile } pr_j \} \quad (21)$$

The active goal (which is denoted as gl^*) is the goal gl, which should be realized in the given time. The relation $\trianglelefteq$ is reflexive, transitive and antisymmetric and partially orders the set PR^a:

$$pr \trianglelefteq pr \qquad \text{for every } pr \in PR^a \tag{22a}$$

$$(pr_i \trianglelefteq pr_j \wedge pr_j \trianglelefteq pr_k) \Rightarrow pr_i \trianglelefteq pr_k \qquad \text{for every } pr_i, pr_j, pr_k \in PR^a \tag{22b}$$

$$(pr_i \trianglelefteq pr_j \wedge pr_j \trianglelefteq pr_i) \Rightarrow pr_i = pr_k \qquad \text{for every } pr_i, pr_j \in PR^a \tag{22c}$$

The set of profiles PR^a is defined in the following way:

$$PR^a = \{pr_1, pr_2, \ldots, pr_n\} \tag{23a}$$

$$pr_1 \trianglelefteq pr_2 \trianglelefteq \cdots \trianglelefteq pr_n \tag{23b}$$

Profile pr_1 is the basic profile—it means that the realization of its goals has the highest priority and they will be realized before the goals of other profiles.
Profile pr of agent a ($pr \in PR^a$) can be the profile in which only resources are used:

$$pr = \langle \Gamma^{pr}, ST^{pr}, RST^{pr}, GL^{pr} \rangle \tag{25}$$

in which only informations are used:

$$pr = \langle \Omega^{pr}, M^{pr}, ST^{pr}, RST^{pr}, GL^{pr} \rangle \tag{26}$$

or resources and informations are used:

$$pr = \langle \Gamma^{pr}, \Omega^{pr}, M^{pr}, ST^{pr}, RST^{pr}, GL^{pr} \rangle \tag{27}$$

where:

- Γ^{pr} is the set of resource types, which are used within the profile pr ($\Gamma^{pr} \subseteq \Gamma^a$);
- Ω^{pr} is the set of information types, which are used within the profile pr ($\Omega^{pr} \subseteq \Omega^a$);
- M^{pr} is the set of informations, which represent the agent's knowledge about the environment and other agents (it is the model of the environment of agent a);
- ST^{pr} is the partially ordered set of strategies ($ST^{pr} \equiv \langle ST^{pr}, \preccurlyeq \rangle$),which can be used by agent within the profile pr in order to realize an active goal of this profile;
- RST^{pr} is the set of strategies that are realized within the profile pr—generally, not all of the strategies from the set ST^{pr} have to be realized within the profile pr, some of them may be realized within other profiles;
- GL^{pr} is partially ordered set of goals ($GL^{pr} \equiv \langle GL^{pr}, \preccurlyeq \rangle$), which agent has to realize within the profile pr.

The relation $\preccurlyeq$ is defined in the following way:

$$\preccurlyeq = \{\langle st_i, st_j \rangle \in ST^{pr} \times ST^{pr} : \text{ strategy } st_i \text{ has equal or higher priority than strategy } st_j\} \tag{27}$$

This relation is reflexive, transitive and antisymmetric and partially orders the set ST^{pr}. Every single strategy $st \in ST^{pr}$ is consisted of actions, which ordered performance leads to the realization of some active goal of the profile pr:

$$st = \langle z_1, z_2, \ldots, z_k \rangle, \quad st \in ST^{pr}, \quad z_i \in Z^a \tag{28}$$

The relation $\preccurlyeq$ is defined in the following way:

$$\preccurlyeq = \{\langle gl_i, gl_j \rangle \in GL^{pr} \times GL^{pr} : \text{ goal } gl_i \text{ has equal or higher priority than the goal } gl_j\} \tag{29}$$

This relation is reflexive, transitive and antisymmetric and partially orders the set GL^{pr}.
The partially ordered sets of profiles PR^a, goals GL^{pr} and strategies ST^{pr} are used by the agent in order to make decisions about the realized goal and to choose the appropriate strategy in order to realize that goal. The basic activities of the agent a are shown in Algorithm 1.

Algorithm 1. Basic activities of agent a in $CoEMAS$

```
1  $r^\gamma \leftarrow r^\gamma_{init}$ ;                /* $r^\gamma_{init}$ is the initial amount of resource given to the agent */
2  while $r^\gamma > 0$ do
3      activate the profile $pr_i \in PR^a$ with the highest priority and with the active goal
       $gl^*_j \in GL^{pr_i}$;
4      if $pr_i$ is the resource profile then
5          if $0 < r^\gamma < r^\gamma_{min}$ then ;   /* $r^\gamma_{min}$ is the minimal amount of resource needed by the
           agent to realize its activities */
6
7              choose the strategy $st_k \in ST^{pr_i}$ with the highest priority that can be used to
               take some resources from the environment or other agent;
8              perform actions contained within the $st_k$;
9          else if $r^\gamma = 0$ then
10             execute $\langle die \rangle$ strategy;
11         end
12     else if $pr_i$ is the reproduction profile then
13         if $r^\gamma > r^{rep,\gamma}_{min}$ then ;          /* $r^{rep,\gamma}_{min}$ is the minimal amount of resource needed for
           reproduction */
14
15             choose the strategy $st_k \in ST^{pr_i}$ with the highest priority that can be used to
               reproduce;
16             perform actions contained within the $st_k$;
17         end
18     else if $pr_i$ is the migration profile then
19         if $r^\gamma > r^{mig,\gamma}_{min}$ then ;          /* $r^{mig,\gamma}_{min}$ is the minimal amount of resource needed for
           migration */
20
21             choose the strategy $st_k \in ST^{pr_i}$ with the highest priority that can be used to
               migrate;
22             perform actions contained within the $st_k$;
23             give $r^{mig,\gamma}_{min}$ amount of resource to the environment;
24         end
25     end
26 end
```

In *CoEMAS* systems the set of profiles is usually composed of resource profile (pr_1), reproduction profile (pr_2), and migration profile (pr_3):

$$PR^a = \{pr_1, pr_2, pr_3\} \tag{30a}$$

$$pr_1 \trianglelefteq pr_2 \trianglelefteq pr_3 \tag{30b}$$

The highest priority has the resource profile, then there is reproduction profile, and finally migration profile.

4. Co-evolutionary multi-agent systems for multi-objective optimization

In this section we will describe two co-evolutionary multi-agent systems used in the experiments. Each of these systems uses different co-evolutionary mechanism: sexual selection, and host-parasite interactions. All of the systems are based on general model of co-evolution in multi-agent system described in Section 3—in this section only such elements of the systems will be described that are specific for these instantiations of the general model. In all the systems presented below, real-valued vectors are used as agents' genotypes. Mutation with self-adaptation and intermediate recombination are used as evolutionary operators (Bäck et al., 1997).

4.1 Co-evolutionary multi-agent system with sexual selection mechanism (SCoEMAS)

The co-evolutionary multi-agent system with sexual selection mechanism is described as 4-tuple (see Eq. (5)):

$$CoEMAS = \langle E, S, \Gamma = \{\gamma\}, \Omega = \{\omega_1, \omega_2\}\rangle \tag{31}$$

The informations of type ω_1 represent all nodes connected with the given node. The informations of type ω_2 represent all agents located within the given node.

4.1.1 Species

The set of species $S = \{s\}$. The only species s is defined as follows:

$$s = \langle A^s, SX^s, Z^s, C^s\rangle \tag{32}$$

where SX^s is the set of sexes which exist within the s species, Z^s is the set of actions that agents of species s can perform, and C^s is the set of relations of s species with other species that exist in the *SCoEMAS*.

Actions The set of actions Z^s is defined as follows:

$$\begin{aligned} Z^s = \{&die, searchDominated, get, giveDominating, searchPartner, choose, \\ &clone, rec, mut, give, accept, selNode, migr\} \end{aligned} \tag{33}$$

where:

- *die* is the action of death (agent dies when it is out of resources);
- *searchDominated* finds the agents that are dominated by the given agent;
- *get* is used to get the resources from a dominated agent;

- *giveDominating* gives some resources to the dominating agent;
- *searchPartner* is used to find candidates for reproduction partners;
- *choose* realizes the mechanism of sexual selection—the partner is chosen on the basis of individual preferences;
- *clone* is used to make the new agent—offspring;
- *rec* realizes the recombination (intermediate recombination is used (Bäck et al., 1997));
- *mut* realizes the mutation (mutation with self-adaptation is used (Bäck et al., 1997));
- *give* is used to give the offspring some amount of the parent's resources;
- *accept* action accepts the agent performing *choose* action as the partner for reproduction;
- *selNode* chooses the node (from the nodes connected with the current node) to which the agent will migrate;
- *migr* allows the agent to migrate from the given node to another node of the environment. The migration causes the lose of some amount of the agent's resources.

Relations The set of relations is defined as follows:

$$C^s = \left\{ \xrightarrow{s,get-} \right\} \tag{34}$$

The relation models intra species competition for limited resources ("-" denotes that as a result of performing *get* action the fitness of another agent of species *s* is decreased):

$$\xrightarrow{s,get-} = \{\langle s, s\rangle\} \tag{35}$$

4.1.2 The sexes

The number of sexes within the *s* species corresponds with the number of criteria (*n*) of the multi-objective problem being solved:

$$SX^s = \{sx_1, \ldots, sx_n\} \tag{36}$$

Actions The set of actions of sex *sx* is defined in the following way: $Z^{sx} = Z^s$.
Relations The set of relations of sex sx_i is defined as follows:

$$C^{sx_i} = \left\{ \xrightarrow[give-,give-]{sx_i,choose+} \right\} \tag{37}$$

The relation $\xrightarrow[give-,give-]{sx_i,choose+}$ realizes the sexual selection mechanism (see Eq. (19)). Each agent has its own preferences, which are composed of the vector of weights (each weight for one of the criteria of the problem being solved). These individual preferences are used during the selection of partner for reproduction (*choose* action).

4.1.3 The agent

Agent *a* of sex *sx* and species *s* (in order to simplify the notation we assume that $a \equiv a^{sx,s}$) is defined as follows:

$$a = \langle gn^a, Z^a = Z^s, \Gamma^a = \Gamma, \Omega^a = \Omega, PR^a \rangle \tag{38}$$

In the case of *SCoEMAS* system the genotype of each agent is composed of three vectors (chromosomes): $\vec{x}$ of real-coded decision parameters' values, $\vec{\sigma}$ of standard deviations' values, which are used during mutation with self-adaptation, and $\vec{w}$ of weights used during selecting partner for reproduction ($gn^a = \langle \vec{x}, \vec{\sigma}, \vec{w} \rangle$). Basic activities of agent *a* with the use of profiles are presented in Alg. 2.

Algorithm 2. Basic activities of agent *a* in *SCoEMAS*

```
$r^\gamma \leftarrow r^\gamma_{init}$;
while $r^\gamma > 0$ do
    activate the profile $pr_i \in PR^a$ with the highest priority and with the active goal
    $gl^*_j \in GL^{pr_i}$;
    if $pr_1$ is activated then
        if $0 < r^\gamma < r^\gamma_{min}$ then
            $\langle searchDominated, get \rangle$;
            $r^\gamma \leftarrow \left(r^\gamma + r^\gamma_{get}\right)$;
        else if $r^\gamma = 0$ then
            $\langle die \rangle$;
        end
        if $\langle giveDominating \rangle$ is executed then
            $r^\gamma \leftarrow \left(r^\gamma - r^\gamma_{get}\right)$;
        end
    else if $pr_2$ is activated then
        if $r^\gamma > r^{rep,\gamma}_{min}$ then
            if $\langle searchPartner, choose, clone, rec, mut, give \rangle$ is activated then
                $r^\gamma \leftarrow \left(r^\gamma - r^{clone,\gamma}_{give}\right)$;
            else if $\langle accept, give \rangle$ is activated then
                $r^\gamma \leftarrow \left(r^\gamma - r^{accept,\gamma}_{give}\right)$;          /* $r^{clone,\gamma}_{give} \gg r^{accept,\gamma}_{give}$ */
            end
        end
    else if $pr_3$ is activated then
        if $r^\gamma > r^{mig,\gamma}_{min}$ then
            $\langle selNode, migr \rangle$;
            $r^\gamma \leftarrow \left(r^\gamma - r^{mig,\gamma}_{min}\right)$;
        end
    end
end
```

Profiles The set of profiles $PR^a = \{pr_1, pr_2, pr_3\}$, where pr_1 is the resource profile, pr_2 is the reproduction profile, and pr_3 is the migration profile. The resource profile is defined in the following way:

$$pr_1 = \langle \Gamma^{pr_1} = \Gamma, \Omega^{pr_1} = \{\omega_2\}, M^{pr_1} = \{i^{\omega_2}\}, ST^{pr_1}, RST^{pr_1} = ST^{pr_1}, GL^{pr_1} \rangle \quad (39)$$

The set of strategies includes two strategies:

$$ST^{pr_1} = \{\langle die \rangle, \langle searchDominated, get \rangle, \langle giveDominating \rangle\} \quad (40)$$

The goal of the profile is to keep the amount of resource above the minimal level.

The reproduction profile is defined as follows:

$$pr_2 = \langle \Gamma^{pr_2} = \Gamma, \Omega^{pr_2} = \{\omega_2\}, M^{pr_2} = \{i^{\omega_2}\}, ST^{pr_2}, RST^{pr_2} = ST^{pr_2}, GL^{pr_2} \rangle \tag{41}$$

The set of strategies includes two strategies:

$$ST^{pr_2} = \{\langle searchPartner, choose, clone, rec, mut, give \rangle, \langle accept, give \rangle\} \tag{42}$$

The goal of the profile is to reproduce when the amount of resource is above the minimal level needed for reproduction.
The migration profile is defined as follows:

$$pr_3 = \langle \Gamma^{pr_3} = \Gamma, \Omega^{pr_3} = \{\omega_1\}, M^{pr_3} = \{i^{\omega_1}\}, ST^{pr_3} = \{\langle selNode, migr \rangle\}, RST^{pr_3} = ST^{pr_3}, GL^{pr_3} \rangle \tag{43}$$

The goal of the profile is to migrate to another node when the amount of resource is above the minimal level needed for migration.

4.2 Co-evolutionary multi-agent system with host-parasite interactions (HPCoEMAS)

The co-evolutionary multi-agent system with host-parasite interactions is defined as follows (see Eq. (5)):

$$HPCoEMAS = \langle E, S, \Gamma, \Omega \rangle \tag{44}$$

The set of species includes two species, hosts and parasites: $S = \{host, par\}$. One resource type exists within the system ($\Gamma = \{\gamma\}$). Three information types ($\Omega = \{\omega_1, \omega_2, \omega_3\}$) are used. Information of type ω_1 denotes nodes to which each agent can migrate when it is located within particular node. Information of type ω_2 denotes such host-agents that are located within the particular node in time t. Information of type ω_3 denotes the host of the given parasite.

4.2.1 Host species

The host species is defined as follows:

$$host = \left\langle A^{host}, SX^{host} = \{sx\}, Z^{host}, C^{host} \right\rangle \tag{45}$$

where SX^{host} is the set of sexes which exist within the *host* species, Z^{host} is the set of actions that agents of species *host* can perform, and C^{host} is the set of relations of *host* species with other species that exist in the *HPCoEMAS*.
Actions The set of actions Z^{host} is defined as follows:

$$Z^{host} = \{die, get, give, accept, seek, clone, rec, mut, giveChild, migr\} \tag{46}$$

where:
- *die* is the action of death (host dies when it is out of resources);
- *get* action gets some resource from the environment;
- *give* action gives some resource to the parasite;
- *accept* action accepts other agent as a reproduction partner;
- *seek* action seeks for another host agent that is able to reproduce;

- *clone* is the action of producing offspring (parents give some of their resources to the offspring during this action);
- *rec* is the recombination operator (intermediate recombination is used (Bäck et al., 1997));
- *mut* is the mutation operator (mutation with self-adaptation is used (Bäck et al., 1997));
- *giveChild* action gives some resource to the offspring;
- *migr* is the action of migrating from one node to another. During this action agent loses some of its resource.

Relations The set of relations of *host* species with other species that exist within the system is defined as follows:

$$C^{host} = \left\{ \xrightarrow{host,get-}, \xrightarrow{host,give+} \right\} \tag{47}$$

The first relation models intra species competition for limited resources given by the environment:

$$\xrightarrow{host,get-} = \{\langle host, host \rangle\} \tag{48}$$

The second one models host-parasite interactions:

$$\xrightarrow{host,give+} = \{\langle host, par \rangle\} \tag{49}$$

4.2.2 Parasite species

The parasite species is defined as follows:

$$par = \langle A^{par}, SX^{par} = \{sx\}, Z^{par}, C^{par} \rangle \tag{50}$$

Actions The set of actions Z^{par} is defined as follows:

$$Z^{par} = \{die, seekHost, get, clone, mut, giveChild, migr\} \tag{51}$$

where:

- *die* is the action of death;
- *seekHost* is the action used in order to find the host. Test that is being performed by parasite-agent on host-agent before infection consists in comparing—in the sense of Pareto domination relation—solutions represented by assaulting parasite-agent and host-agents that is being assaulted. The more solution represented by host-agent is dominated by parasite-agent the higher is the probability of infection.
- *get* action gets some resource from the host;
- *clone* is the action of producing two offspring;
- *mut* is the mutation operator (mutation with self-adaptation is used (Bäck et al., 1997));
- *giveChild* action gives all the resources to the offspring—after the reproduction parasite agent dies;
- *migr* is the action of migrating from one node to another. During this action agent loses some of its resource.

Relations The set of relations of *par* species with other species that exist within the system are defined as follows:

$$C^{par} = \left\{ \xrightarrow{par,get-} \right\} \tag{52}$$

This relation models host-parasite interactions:

$$\xrightarrow{par,get-} = \{\langle par, host \rangle\} \tag{53}$$

As a result of performing *get* action some amount of the resources is taken from the host.

4.2.3 Host agent

Agent a of species *host* ($a \equiv a^{host}$) is defined as follows:

$$a = \left\langle gn^a, Z^a = Z^{host}, \Gamma^a = \Gamma, \Omega^a = \{\omega_1, \omega_2\}, PR^a \right\rangle \tag{54}$$

Genotype of agent a is consisted of two vectors (chromosomes): $\vec{x}$ of real-coded decision parameters' values and $\vec{\sigma}$ of standard deviations' values, which are used during mutation with self-adaptation. $Z^a = Z^{host}$ (see Eq. (46)) is the set of actions which agent a can perform. Γ^a is the set of resource types used by the agent, and Ω^a is the set of information types. Basic activities of the agent a are presented in Alg. 3.

Profiles The partially ordered set of profiles includes resource profile (pr_1), reproduction profile (pr_2), interaction profile (pr_3), and migration profile (pr_4):

$$PR^a = \{pr_1, pr_2, pr_3, pr_4\} \tag{55a}$$

$$pr_1 \trianglelefteq pr_2 \trianglelefteq pr_3 \trianglelefteq pr_4 \tag{55b}$$

The resource profile is defined in the following way:

$$pr_1 = \langle \Gamma^{pr_1} = \Gamma, \Omega^{pr_1} = \emptyset, M^{pr_1} = \emptyset, ST^{pr_1}, RST^{pr_1} = ST^{pr_1}, GL^{pr_1} \rangle \tag{56}$$

The set of strategies includes two strategies:

$$ST^{pr_1} = \{\langle die \rangle, \langle get \rangle\} \tag{57}$$

The goal of the pr_1 profile is to keep the amount of resources above the minimal level or to die when the amount of resources falls to zero.

The reproduction profile is defined as follows:

$$pr_2 = \langle \Gamma^{pr_2} = \Gamma, \Omega^{pr_2} = \{\omega_2\}, M^{pr_2} = \{i^{\omega_2}\}, ST^{pr_2}, RST^{pr_2} = ST^{pr_2}, GL^{pr_2} \rangle \tag{58}$$

The set of strategies includes two strategies:

$$ST^{pr_2} = \{\langle seek, clone, rec, mut, giveChild \rangle, \langle accept, giveChild \rangle\} \tag{59}$$

The only goal of the pr_2 profile is to reproduce. In order to realize this goal agent can use strategy of reproduction ⟨*seek, clone, rec, mut, giveChild*⟩ or can accept other agent as a reproduction partner ⟨*accept, giveChild*⟩.

The interaction profile is defined as follows:

$$pr_3 = \langle \Gamma^{pr_3} = \Gamma, \Omega^{pr_3} = \emptyset, M^{pr_3} = \emptyset, ST^{pr_3} = \{\langle give \rangle\}, RST^{pr_3} = ST^{pr_3}, GL^{pr_3} \rangle \tag{60}$$

The goal of the pr_3 profile is to interact with parasites with the use of strategy $\langle give \rangle$, which gives some of the host's resources to the parasite.
The migration profile is defined as follows:

$$pr_4 = \langle \Gamma^{pr_4} = \Gamma, \Omega^{pr_4} = \{\omega_1\}, M^{pr_4} = \{i^{\omega_1}\}, ST^{pr_4} = \{\langle migr \rangle\}, RST^{pr_4} = ST^{pr_4}, GL^{pr_4} \rangle \tag{61}$$

The goal of the pr_4 profile is to migrate within the environment. In order to realize such a goal the migration strategy is used, which firstly chooses the node and then realizes the migration. Agent loses some of its resources in order to migrate.

Algorithm 3. Basic activities of agent $a \equiv a^{host}$ in $HPCoEMAS$

```
$r^\gamma \leftarrow r^\gamma_{init}$;
while $r^\gamma > 0$ do
    activate the profile $pr_i \in PR^a$ with the highest priority and with the active goal
    $gl^*_j \in GL^{pr_i}$;
    if $pr_1$ is activated then
        if $0 < r^\gamma < r^\gamma_{min}$ then
            $\langle get \rangle$;
            $r^\gamma \leftarrow \left(r^\gamma + r^{env,\gamma}_{get}\right)$;        /* $r^{env,\gamma}_{get}$ is the amount of resource given by the
            environment */
        else if $r^\gamma = 0$ then
            $\langle die \rangle$;
        end
    else if $pr_2$ is activated then
        if $r^\gamma > r^{rep,\gamma}_{min}$ then
            if $\langle seek, clone, rec, mut, giveChild \rangle$ is performed then
                $r^\gamma \leftarrow \left(r^\gamma - r^\gamma_{giveChild}\right)$;
            else if $\langle accept, giveChild \rangle$ is performed then
                $r^\gamma \leftarrow \left(r^\gamma - r^\gamma_{giveChild}\right)$;
            end
        end
    else if $pr_3$ is activated then
        $\langle give \rangle$;
        $r^\gamma \leftarrow \left(r^\gamma - r^\gamma_{give}\right)$;
    else if $pr_4$ is activated then
        if $r^\gamma > r^{mig,\gamma}_{min}$ then
            $\langle migr \rangle$;
            $r^\gamma \leftarrow \left(r^\gamma - r^{mig,\gamma}_{min}\right)$;
        end
    end
end
```

4.2.4 Parasite agent

Agent a of species par ($a \equiv a^{par}$) is defined as follows:

$$a = \langle gn^a, Z^a = Z^{par}, \Gamma^a = \Gamma, \Omega^a = \Omega, PR^a \rangle \tag{62}$$

Genotype of agent a is consisted of two vectors (chromosomes): $\vec{x}$ of real-coded decision parameters' values and $\vec{\sigma}$ of standard deviations' values. $Z^a = Z^{par}$ (see eq. (51)) is the set of actions which agent a can perform. Γ^a is the set of resource types used by the agent, and Ω^a is the set of information types. Basic activities of the agent a are presented in Alg. 4.

Algorithm 4. Basic activities of agent $a \equiv a^{par}$ in *HPCoEMAS*

```
1  $r^\gamma \leftarrow r^\gamma_{init}$;
2  while $r^\gamma > 0$ do
3      activate the profile $pr_i \in PR^a$ with the highest priority and with the active goal
       $gl^*_j \in GL^{pr_i}$;
4      if $pr_1$ is activated then
5          if $0 < r^\gamma < r^\gamma_{min}$ then
6              if $i^{\omega_3} = \emptyset$ then
7                  $\langle seekHost, get \rangle$;
8                  $r^\gamma \leftarrow \left(r^\gamma + r^\gamma_{get}\right)$;     /* $r^\gamma_{get}$ is the amount of resource taken from host */
9              else
10                 $\langle get \rangle$;
11                 $r^\gamma \leftarrow \left(r^\gamma + r^\gamma_{get}\right)$;
12             end
13         else if $r^\gamma = 0$ then
14             $\langle die \rangle$;
15         end
16     else if $pr_2$ is activated then
17         if $r^\gamma > r^{rep,\gamma}_{min}$ then
18             $\langle clone, mut, giveChild \rangle$;
19             $r^\gamma \leftarrow 0$;     /* All the resources are given to the offspring */
20         end
21     else if $pr_3$ is activated then
22         if $r^\gamma > r^{mig,\gamma}_{min}$ then
23             $\langle migr \rangle$;
24             $r^\gamma \leftarrow \left(r^\gamma - r^{mig,\gamma}_{min}\right)$;
25         end
26     end
27 end
```

Profiles The partially ordered set of profiles includes resource profile (pr_1), reproduction profile (pr_2), and migration profile (pr_3):

$$PR^a = \{pr_1, pr_2, pr_3\} \tag{63a}$$

$$pr_1 \trianglelefteq pr_2 \trianglelefteq pr_3 \tag{63b}$$

The resource profile is defined in the following way:

$$pr_1 = \langle \Gamma^{pr_1} = \Gamma, \Omega^{pr_1} = \{\omega_2, \omega_3\}, M^{pr_1} = \{i^{\omega_2}, i^{\omega_3}\}, ST^{pr_1}, RST^{pr_1} = ST^{pr_1}, GL^{pr_1} \rangle \tag{64}$$

The set of strategies includes three strategies:

$$ST^{pr_1} = \{\langle die\rangle, \langle get\rangle, \langle seekHost, get\rangle\} \tag{65}$$

The goal of the pr_1 profile is to keep the amount of resources above the minimal level or to die when the amount of resources falls to zero. When the parasite has not infected any host (information i^{ω_3} is used), it uses strategy ⟨*seekHost*, *get*⟩ in order to find and infect some host and get its resources. If the parasite has already infected a host it can use ⟨*get*⟩ strategy in order to take some resources.
The reproduction profile is defined as follows:

$$pr_2 = \langle \Gamma^{pr_2} = \Gamma, \Omega^{pr_2} = \emptyset, M^{pr_2} = \emptyset, ST^{pr_2}, RST^{pr_2} = ST^{pr_2}, GL^{pr_2}\rangle \tag{66}$$

The set of strategies includes one strategy:

$$ST^{pr_2} = \{\langle clone, mut, giveChild\rangle\} \tag{67}$$

The only goal of the pr_2 profile is to reproduce. In order to realize this goal agent can use strategy of reproduction: ⟨*clone*,*mut*, *giveChild*⟩. Two offsprings are produced and the parent gives them all its resources and then dies.
The migration profile is defined as follows:

$$pr_3 = \langle \Gamma^{pr_3} = \Gamma, \Omega^{pr_3} = \{\omega_1\}, M^{pr_3} = \{i^{\omega_1}\}, ST^{pr_3} = \{\langle migr\rangle\}, RST^{pr_3} = ST^{pr_3}, GL^{pr_3}\rangle \tag{68}$$

The goal of the pr_3 profile is to migrate within the environment. In order to realize such a goal the migration strategy is used, which firstly chooses the node and then realizes the migration. During this some amount of the resource is given back to the environment.

5. Experimental results

Presented formally in section 4 agent-based co-evolutionary approaches for multi-objective optimization have been tentatively assessed. Obtained during experiments preliminary results were presented in some of our previous papers and in this section they are shortly summarized.

5.1 Performance metrics

Using only one single measure during assessing the effectiveness of (evolutionary) algorithms for multi-objective optimization is not enough (Zitzler et al., 2003) however it is impossible to present all obtained results (metrics as well as obtained Pareto frontiers and Pareto sets) discussing simultaneously (a lot of) ideas and issues related to the proposed new approach for evolutionary multi-objective optimization in one single article especially that the main goal of this chapter is to present coherent formal models of innovative agent-based co-evolutionary systems dedicated for multi-objective optimization rather than indepth results′ analysis. Since hypervolume (HV) or hypervolume ratio (HVR) metrics allow to estimate both: the convergence to the true Pareto frontier as well as distribution of solutions over the whole approximation of the Pareto frontier, despite of its shortcomings it is one of the most commonly and most frequently used measure as the main metric for comparing the quality of obtained result sets—that is why results and comparisons presented in this paper are based mainly on this very measure.

Hypervolume or hypervolume ratio (Zitzler & Thiele, 1998) describes the area covered by solutions of obtained approximation of the Pareto frontier (PF). For each found nondominated solution, hypercube is evaluated with respect to the fixed reference point. In order to evaluate hypervolume ratio, value of hypervolume for obtained set is normalized with hypervolume value computed for true Pareto frontier. HV and HVR are defined as follows:

$$\mathrm{HV} = v\left(\bigcup_{i=1}^{N} v_i\right) \tag{69a}$$

$$\mathrm{HVR} = \frac{\mathrm{HV}(PF^{*})}{\mathrm{HV}(PF)} \tag{69b}$$

where v_i is hypercube computed for $i{-}th$ found non-dominated solution, PF^* represents obtained approximation of the Pareto frontier and PF is the true Pareto frontier.

Assuming the following meaning of used below symbols: $\mathbb{P}$—Pareto set, $A, B \subseteq D$—two sets of decision vectors, $\sigma \geq 0$—appropriately chosen neighborhood parameter and $\|\cdot\|$—the given distance metric, then the following (used also in some of our experiments) measures can be defined (Zitzler, 1999):

- $\sigma(A, B)$—the coverage of two sets maps the ordered pair (A, B) to the interval [0, 1] in the following way:

$$\delta(A, B) = \frac{|\{b \in B \mid \exists a \in A : a \geq b\}|}{|B|} \tag{70}$$

- $\xi(A, B)$—the coverage difference of two sets ($\wp$ denotes value of the *size of dominated space* measure):

$$\xi(A, B) = \wp(A + B) - \wp(B) \tag{71}$$

- M_1—the average distance to the Pareto optimal set $\mathcal{P}$:

$$M_1(\mathcal{P}) = \frac{1}{|\mathcal{P}|}\sum_{p \in \mathcal{P}} min\{\|p - x\| \mid x \in \mathcal{P}\} \tag{72}$$

- M_2—the distribution in combination with the number of non-dominated solutions found:

$$M_2(\mathcal{P}) = \frac{1}{|\mathcal{P} - 1|}\sum_{p \in \mathcal{P}} |\{r \in \mathcal{P} \mid \|p - r\| > \sigma\}| \tag{73}$$

- M_3—the spread of non-dominated solutions over the set A:

$$M_3(\mathcal{P}) = \sqrt{\sum_{i=1}^{N} max\{\|p_i - r_i\| \mid p, r \in \mathcal{P}\}} \tag{74}$$

5.2 Test problems

Firstly, *Binh* (Binh & Korn, 1996; Binh & Korn, 1997) as well as *Schaffer* (Schaffer, 1985) problems were used. Binh problem is defined as follows:

$$Binh = \begin{cases} f_1(x,y) = x^2 + y^2 \\ f_2(x,y) = (x-5)^2 + (y-5)^2 \\ where \quad -5 \le x, y \le 10 \end{cases} \tag{75}$$

whereas used *modified Schaffer* problem is defined as follows:

$$Modified\,Schaffer = \begin{cases} f_1(x) = x^2 \\ f_2(x) = (x-2)^2 \\ where \quad -32 \le x \le 32 \end{cases} \tag{76}$$

Obviously during our experiments also well known and commonly used test suites were used. Inter alia such problems as ZDT test suite was used but because of its importance it is discussed wider in section 5.2.1.

5.2.1 ZDT (Zitzler-Deb-Thiele) test suite

One of test suites used during experiments presented and shortly discussed in the course of this section is Zitzler-Deb-Thiele test suite which in the literature it is known and identified as the set of test problems ZDT1-ZDT6 ((Zitzler, 1999, p. 57–63), (Zitzler et al., 2000), (Deb, 2001, p. 356–362), (Coello Coello et al., 2007, p. 194–199)). K. Deb in his work (Deb, 1998) tried to identify and systematize factors that can heighten difficulties in identifying by optimizing algorithm the true (model) Pareto frontier of multi-objective optimization problem that is being solved. The two main issues regarding the quality of obtained approximation of the Pareto frontier are: closeness to the true Pareto frontier as well as even dispersion of found non-dominated solution over the whole (approximation) of the Pareto frontier. Drifting to the Pareto frontier can be disturbed by such features of the problem as its multi-modality or isolated optima, what is known and can be observed also in the case of single-objective optimization. The other features that can (negatively) influence the ability of optimization algorithm for obtaining the high-quality Pareto frontier approximation are convex or concave character of the frontier or its discontinuity as well. Taking such observations into consideration the set of six test functions (ZDT1-ZDT6) was proposed. Each of them addresses and makes it possible to assess if algorithm that is being tested is able to overcome difficulties caused by each of mentioned feature. The whole ZDT test suite is constructed according to the following schema:

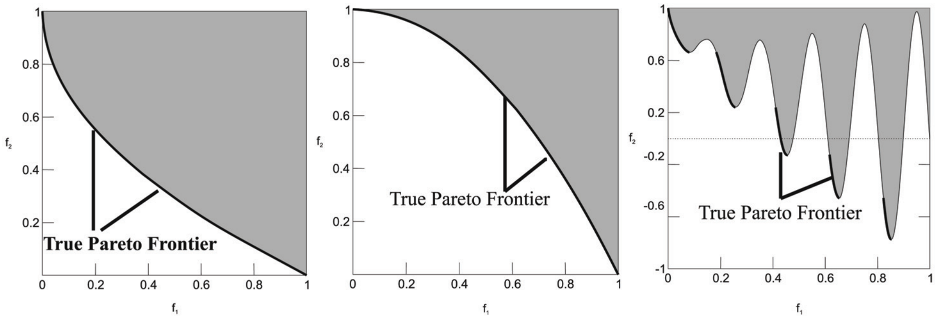
$$ZDT = \begin{cases} Minimize & F(x) & = & (f_1(x_1), f_2(x)) \\ On\ condition & f_2(x) & = & g(x_2, \ldots, x_n) \cdot h(f_1(x_1), g(x_2, \ldots, x_n)) \end{cases} \tag{77}$$

where: $x = (x_1, \ldots, x_n)$. Well, as one may see, ZDT1-ZDT6 problems are constructed on the basis of functions f_1, g and h as well, where f_1 is a function of one single (first) decision variable (x_1), function g is a function of the rest $n-1$ decision variables, and finally, function h is a function depending on values of functions f_1 and g. Particular problems ZDT1-ZDT6 assume different definitions of f_1, g and h functions as well as the number of decision variables n and the range of values of decision variables.

ZDT1 problem is the simplest (with continuous and convex true Pareto frontier) multi-objective optimization problem within the ZDT test-suite. The visualization of the true

Pareto frontier for ZDT1 problem (with $g(x) = 1$) is presented in Fig. 3a. Definitions of f_1, g and h functions in the case of ZDT1 problem are as follows:

$$ZDT1 = \begin{cases} f_1(x) = x_1 \\ g(x_2, ..., x_n) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i \\ h(f_1, g) = 1 - \sqrt{f_1/g(x)} \\ where\ n = 30, x_i \in [0, 1] \end{cases} \tag{78}$$



(a) (b) (c)

Fig. 3. Visualization of objective space and the true Pareto frontiers for problems ZDT1 (a) ZDT2 (b) and ZDT3 (c)

ZDT2 problem introduces the first potential difficulty for optimizing algorithm i.e. it is a problem with continuous but concave true Pareto frontier. The visualization of the true Pareto frontier for ZDT2 problem (with $g(x) = 1$) is presented in Fig. 3b. Definitions of f_1, g and h in this case are as follows:

$$ZDT2 = \begin{cases} f_1(x) = x_1 \\ g(x_2, ..., x_n) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i \\ h(f_1, g) = 1 - (f_1/g(x))^2 \\ where\ n = 30, x_i \in [0, 1] \end{cases} \tag{79}$$

ZDT3 problem introduces the next difficulty for optimization algorithm, this time it is discontinuity of the Pareto frontier. In the case of ZDT3 problem (defined obviously according to the (77) schema) the formulation of functions f_1, g and h are as follows:

$$ZDT3 = \begin{cases} f_1(x) = x_1 \\ g(x_2, ..., x_n) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i \\ h(f_1, g) = 1 - \sqrt{f_1/g(x)} - \frac{f_1}{g(x)} sin(10\pi f_1) \\ where\ n = 30, x_i \in [0, 1] \end{cases} \tag{80}$$

Using *sinus* function in the case of ZDT3 problem in the definition of function h causes discontinuity in the Pareto frontier and simultaneously it does not cause discontinuity in the space of decision variables. The visualization of the true Pareto frontier for ZDT3 problem is presented in Fig. 3c.

ZDT4 problem makes it possible to assess the optimization algorithm in the case of solving multi-objective but simultaneously multi-modal optimization problem. The visualization of the true Pareto frontier for ZDT4 problem obtained with $g(x) = 1$) is presented in Fig. 4a.

ZDT4 problem introduces 21^9 local Pareto frontiers and the formulations of f_1, g and h in this case are as follows:

$$ZDT4 = \begin{cases} f_1(x) = x_1 \\ g(x_2, \ldots, x_n) = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10cos(4\pi x_i)) \\ h(f_1, g) = 1 - \sqrt{f_1/g(x)} \\ where\ n = 10,\ x_1 \in [0,1],\ x_i \in [-5,5] \end{cases} \tag{81}$$

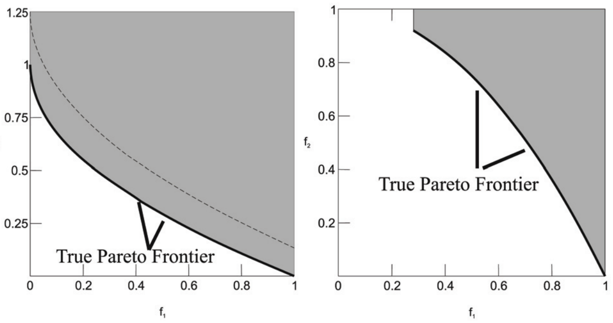


Fig. 4. Visualization of objective space and the true Pareto frontiers for problems ZDT4 (a) and ZDT6 (b)

ZDT6 problem is a multi-objective optimization problem introducing several potential difficulties for optimization algorithm. It is a problem with non-convex Pareto frontier. Additionally, non-dominated solutions are dispersed not evenly. Next, in the space of decision variables, the "density" of solutions is less and less in the vicinity of the true Pareto frontier.
The visualization of the true Pareto frontier for ZDT6 problem is presented in Fig. 4b. Functions f_1, g and h defined obviously according to the schema (77) in the case of ZDT6 problem are formulated as follows:

$$ZDT6 = \begin{cases} f_1(x) = 1 - exp(-4x_1)sin^6(6\pi x_1) \\ g(x_2, \ldots, x_n) = 1 + 9(\frac{(\sum_{i=2}^{n} x_i)}{(n-1)})^{0.25} \\ h(f_1, g) = 1 - (\frac{f_1(x)}{g(x)})^2 \\ where\ n = 10,\ x_i \in [0,1] \end{cases} \tag{82}$$

5.3 A glance at assessing sexual-selection based approach (SCoEMAS)

Sexual-selection co-evolutionary multi-agent system (SCoEMAS) presented in section 4.1 was preliminary assessed using inter alia presented in section 5.2.1 ZDT test suite. Also this time, SCoEMAS approach was compared among others with the state-of-the-art evolutionary algorithms for multi-objective optimization i.e. NSGA-II (Deb et al., 2002; Deb et al., 2000) and SPEA2 (Zitzler et al., 2001; Zitzler et al., 2002).
The size of population of SCoEMAS is 100, and the size of population of benchmarking algorithms are as follows: NSGA-II—300 and SPEA2—100. Selected parameters and their values assumed during presented experiments are as follows: $r^{\gamma}_{init} = 50$ (it represents the

level of resources possessed initially by individual just after its creation), r^{γ}_{get} = 30 (it represents resources transferred in the case of domination), $r^{rep,\gamma}_{min}$ = 30 (it represents the level of resources required for reproduction), p_{mut} = 0.5 (mutation probability).
In Figure 5, Figure 6 and Figure 7 there are presented values of HVR measure obtained with time by SCoEMAS for ZDT1 (Figure 5a), ZDT2 (Figure 5b), ZDT3 (Figure 6a), ZDT4 (Figure 6b) and ZDT6 (Figure 7) problems. For comparison there are presented also results obtained by NSGA-II and SPEA2 algorithms.

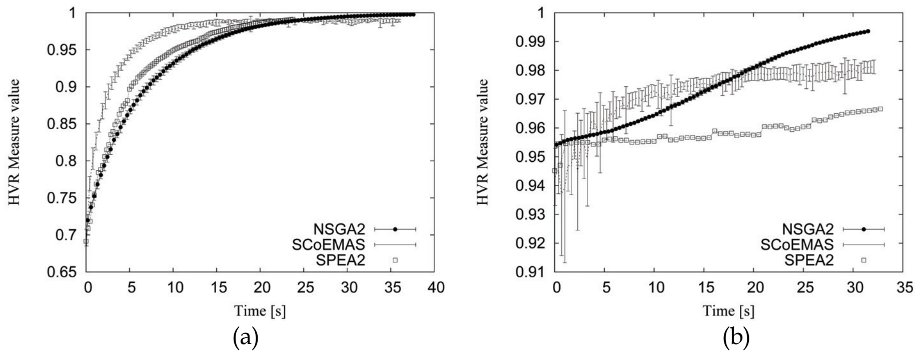


Fig. 5. HVR values obtained by SCoEMAS, NSGA-II and SPEA2 run against Zitzler's problems ZDT1 (a), and ZDT2 (b) (Siwik & Dreżewski, 2008)

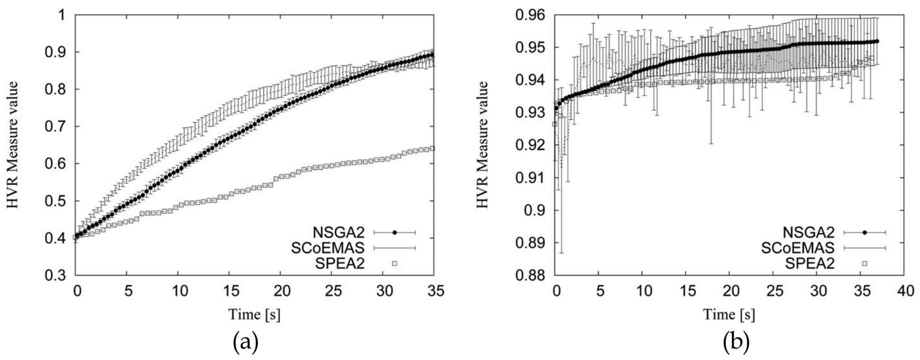


Fig. 6. HVR values obtained by SCoEMAS, NSGA-II and SPEA2 run against Zitzler's problems ZDT3 (a), and ZDT4 (b) (Siwik & Dreżewski, 2008)

On the basis of presented characteristics it can be said that initially co-evolutionary multi-agent system with sexual selection is faster than two other algorithms, it allows for obtaining better solutions—what can be observed as higher values of HVR(t) metrics but finally, the best results are obtained by NSGA-II algorithm. A little bit worse alternative than NSGA-II is SCoEMAS and finally SPEA2 is the third alternative—but obviously it depends on the problem that is being solved and differences between analyzed algorithms are not very distinctive.
Deeper analysis of obtained results can be found in (Dreżewski & Siwik, 2007; Dreżewski & Siwik, 2006*a*; Siwik & Dreżewski, 2008).

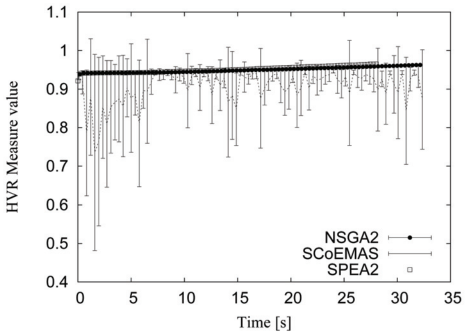


Fig. 7. HVR values obtained by SCoEMAS, NSGA-II and SPEA2 run against Zitzler's ZDT6 problem (Siwik & Dreżewski, 2008)

5.4 A glance at assessing host-parasite based approach (HPCoEMAS)

Discussed in section 4.2 co-evolutionary multi-agent system with host-parasite mechanism was tested using, inter alia, *Binh* and slightly modified *Schaffer* test functions that are defined as in equations (75) and (76).

Coverage of two sets $\delta(A, B)$				
	SPEA	**VEGA**	**NPGA**	**HPCoEMAS**
SPEA	✓	0.08	0.00	0.04
VEGA	0.92	✓	0.30	0.32
NPGA	1.00	0.62	✓	0.40
HPCoEMAS	0.96	0.70	0.58	✓

Table 1. Comparison of proposed HPCoEMAS approach with selected classical EMOAs according to the *Coverage of two sets* metrics (Dreżewski & Siwik, 2006*b*)

Coverage difference of two sets $\xi(A, B)$				
	SPEA	**VEGA**	**NPGA**	**HPCoEMAS**
SPEA	✓	8	0	6
VEGA	116	✓	3	13
NPGA	154	42	✓	25
HPCoEMAS	197	27	7	✓

Table 2. Comparison of proposed HPCoEMAS approach with selected classical EMOAs according to the *Coverage difference of two sets* metrics (Dreżewski & Siwik, 2006*b*)

	Size of dominated space ($\wp$)	**Average distance to the model Pareto set (M_1)**	**Distribution (M_2)**	**Spread (M_3)**
SPEA	39521	0.8	0.21	10.2
VEGA	39405	2.3	0.11	10.3
NPGA	39368	3.2	0.18	10.1
HPCoEMAS	39324	3.7	0.15	9.9

Table 3. Comparison of proposed HPCoEMAS approach with selected classical EMOAs according to other four metrics (Dreżewski & Siwik, 2006*b*)

This time, the following benchmarking algorithms were used: vector evaluated genetic algorithm (VEGA) (Schaffer, 1984; Schaffer, 1985), niched-pareto genetic algorithm (NPGA) (Horn et al., 1994) and strength Pareto evolutionary algorithm (SPEA) (Zitzler, 1999).
To compare proposed approach with implemented classical algorithms metrics defined in equations (70), (71), (72), (73) and (74) have been used. Obtained values of these metrics are presented in Table 1, Table 2 and Table 3.
Basing on defined above test functions and measures, some comparative studies of proposed co-evolutionary agent-based system with host-parasite interactions and well known and commonly used algorithms (i.e. VEGA, NPGA and SPEA) could be performed and the most important conclusion from such experiments can be formulated as follows: proposed HPCoEMAS system has turned out to be comparable to the *classical algorithms* according almost all considered metrics except for *Average distance to the model Pareto set* (see. Table 3). More conclusions and deeper analysis can be found in (Dreżewski & Siwik, 2006*b*).

6. Summary and conclusions

During last 25 years multi-objective optimization has been in the limelight of researchers. Because of practical importance and applications of multi-objective optimization as the most natural way of decision making and real-life optimizing method—growing interests of researchers in this very field of science was a natural consequence and extension of previous research on single-objective optimization techniques. Unfortunately, when searching for the approximation of the Pareto frontier, classical computational methods often prove ineffective for many (real) decision problems. The corresponding models are too complex or the formulas applied too complicated, or it can even occur that some formulations must be rejected in the face of numerical instability of available solvers. Also, because of such a specificity of multi-objective optimization (especially when—as in our case—we are considering multi-objective optimization in the Pareto sense) that we are looking for the whole set of nondominated solutions rather than one single solution—the special attention has been paid on population-based optimization techniques and if so, the most important techniques turned out here to be evolutionary-based methods. Research on applying evolutionary-based methods for solving multi-objective optimization tasks resulted in developing a completely new (and now commonly and very well known) science field: evolutionary multi-objective optimization (EMOO). To confirm above sentences, it is enough to mention statistics regarding at least the number of conference and journal articles, PhD thesis, conferences, books etc. devoted to EMOO and available at http://delta.cs.cinvestav.mx/~coello/EMOO.
After the first stage of research on EMOO when plenty of algorithms were proposed[1], simultaneously with introducing in early 2000s two the most important EMOO algorithms

[1] It is enough to mention such algorithms as: Rudolph's algorithm (Rudolph, 2001), distance-based Pareto GA (Osyczka & Kundu, 1995), strength Pareto EA (Zitzler & Thiele, 1998), multi-objective micro GA (Coello Coello & Toscano, 2005), Pareto-archived evolution strategy (Knowles & Corne, 2000), multi-objective messy GA (Van Veldhuizen, 1999), vector-optimized evolution strategy (Kursawe, 1991), random weighted GA (Murata & Ishibuchi, 1995), weight-based GA (Hajela et al., 1993), niched-pareto GA (Horn et al., 1994), non-dominated sorting GA (Srinivas & Deb, 1994), multiple objective GA (Fonseca & Fleming, 1993), distributed sharing GA (Hiroyasu et al., 1999)

i.e. NSGA-II and SPEA2 it seemed that no further research regarding new optimization techniques is needed. Unfortunately, in the case of really challenging problems (for instance in the case of multi-objective optimization in noisy environments, in the case of solving constrained problems, in the case of modeling market-related interactions etc.) mentioned algorithm turned out to be not efficient enough.

In this context, techniques with a kind of "soft selection" such as evolutionary multi-agent systems (EMAS), where in the population there can exist even not very strong individuals—seem to be very attractive alternatives. It turns out that "basic" EMAS model applied for multi-objective optimization can be improved significantly with the use of additional mechanisms and interactions among agents that can be introduced into such a system. In particular, as it is presented in the course of this chapter, some co-evolutionary interactions, mechanisms and techniques can be there successfully introduced. In section 5 there are presented results obtained with the use of two different co-evolutionary multi-agent systems. As one may see, presented results are not always significantly better than results obtained by "referenced" algorithms (in particular by state-of-the-art algorithms) but both, this chapter as well as presented results should be perceived as a kind of summary of the first stage of research on possibilities of developing co-evolutionary multi-agent systems for multi-objective optimization.

The most important conclusion of this very first stage of our research is as follows: on the basis of CoEMAS approach it is possible to model a wide range of co-evolutionary interactions. It is possible to develop such models as a distributed, decentralized and autonomous agent system. All proposed approaches can be modeled in a coherent way and can be derived from a basic CoEMAS model in a smooth and elegant way. So, in spite of not so high-quality results presented in previous section—after mentioned first stage of our research we know that both formal modeling as well as implementation of co-evolutionary multi-agent systems is possible in general. Because of their potential possibilities for modeling of (extremely) complex environments, problems, interactions, markets—further research on CoEMASes should result in plenty of their successful applications for solving real-life multi-objective optimization problems.

7. References

Abraham, A., Jain, L. C. & Goldberg, R. (2005). *Evolutionary Multiobjective Optimization Theoretical Advances and Applications*, Springer

Allenson, R. (1992). Genetic algorithms with gender for multi-function optimisation, Technical Report EPCC-SS92-01, Edinburgh Parallel Computing Centre, Edinburgh, Scotland

Bäck, T., Fogel, D. & Michalewicz, Z., (Ed.) (1997). *Handbook of Evolutionary Computation,* IOP Publishing and Oxford University Press

Binh, T. T. & Korn, U. (1996). An evolution strategy for the multiobjective optimization, In: *Proceedings of the Second International Conference on Genetic Algorithms Mendel96*, Brno, Czech Republic, pp. 176–182

Binh, T. T. & Korn, U. (1997). Multicriteria control system design using an intelligent evolution strategy, In: *Proceedings of Conference for Control of Industrial Systems* (CIS97), Vol. 2, Belfort, France, pp. 242–247

Cetnarowicz, K., Kisiel-Dorohinicki,M. & Nawarecki, E. (1996). The application of evolution process in multi-agent world to the prediction system, In: M. Tokoro, (Ed.), *Proceedings of the 2nd International Conference on Multi-Agent Systems* (ICMAS 1996), AAAI Press, Menlo Park, CA

Coello Coello, C. A. & Toscano, G. (2005). Multiobjective structural optimization using a micro-genetic algorithm, *Structural and Multidisciplinary Optimization* 30(5), 388– 403

Coello Coello, C. A., Van Veldhuizen, D. A. & Lamont, G. B. (2007). *Evolutionary algorithms for solving multi-objective problems*, Genetic and evolutionary computation, second edn, Springer Verlag

Deb, K. (1998). Multi-objective genetic algorithms: Problem difficulties and construction of test functions, Technical Report CI-49/98, Department of Computer Science, University of Dortmund

Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test problems, *Evolutionary Computation* 7(3), 205–230

Deb, K. (2001). Multi-Objective Optimization using Evolutionary Algorithms, JohnWiley & Sons

Deb, K., Agrawal, S., Pratap, A. & Meyarivan, T. (2000). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II, In: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J.Merelo & H.-P. Schwefel, (Ed.), *Proceed ings of the Parallel Problem Solving from Nature VI Conference*, Springer. Lecture Notes in Computer Science No. 1917, Paris, France, pp. 849–858

Deb, K., Pratap, A., Agrawal, S. & Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II, IEEE *Transactions on Evolutionary Computation* 6(2), 181– 197

Dreżewski, R. (2003). A model of co-evolution in multi-agent system, In: V. Mařík, J. Müller &M. Pěchouček, (Ed.), *Multi-Agent Systems and Applications* III, Vol. 2691 of LNCS, Springer-Verlag, Berlin, Heidelberg, pp. 314–323

Dreżewski, R. (2006). Co-evolutionary multi-agent system with speciation and resource sharing mechanisms, *Computing and Informatics* 25(4), 305–331

Dreżewski, R. & Siwik, L. (2006a). Co-evolutionary multi-agent system with sexual selection mechanism for multi-objective optimization, In: *Proceedings of the IEEE World Congress on Computational Intelligence* (WCCI 2006), IEEE

Dreżewski, R. & Siwik, L. (2006b). Multi-objective optimization using co-evolutionary multiagent system with host-parasite mechanism, In: V. N. Alexandrov, G. D. van Albada, P. M. A. Sloot & J. Dongarra, (Ed.), *Computational Science – ICCS 2006*, Vol. 3993 of LNCS, Springer-Verlag, Berlin, Heidelberg, pp. 871–878

Dreżewski, R. & Siwik, L. (2007). Techniques for maintaining population diversity in classical and agent-based multi-objective evolutionary algorithms, In: Y. Shi, G. D. van Albada, J. Dongarra & P. M. A. Sloot, (Ed.), *Computational Science – ICCS 2007*, Vol. 4488 of LNCS, Springer-Verlag, Berlin, Heidelberg, pp. 904–911

Fonseca, C. M. & Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization, In: *Genetic Algorithms: Proceedings of the Fifth International Conference*,Morgan Kaufmann, pp. 416–423

Gavrilets, S. & Waxman, D. (2002). Sympatric speciation by sexual conflict, *Proceedings of the National Academy of Sciences of the USA* 99, 10533–10538

Hajela, P., Lee, E. & Lin, C. Y. (1993). Genetic algorithms in structural topology optimization, In: *Proceedings of the NATO Advanced Research Workshop on Topology Design of Structures, Vol. 1*, pp. 117–133

Hiroyasu, T., Miki,M. & Watanabe, S. (1999). Distributed genetic algorithms with a new sharing approach, In: *Proceedings of the Conference on Evolutionary Computation,* Vol. 1, IEEE Service Center

Horn, J., Nafpliotis, N. & Goldberg, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization, In: Proceedings of the First IEEE Conference *on Evolutionary Computation, IEEEWorld Congress on Computational Intelligence,* Vol. 1, IEEE Service Center, Piscataway, New Jersey, pp. 82–87

Knowles, J. D. & Corne, D. (2000). Approximating the nondominated front using the pareto archived evolution strategy, *Evolutionary Computation* 8(2), 149–172

Kursawe, F. (1991). A variant of evolution strategies for vector optimization, In: H. Schwefel & R. Manner, (Ed.), *Parallel Problem Solving from Nature.* 1st Workshop, PPSN I, Vol. 496, Springer-Verlag, Berlin, Germany, pp. 193–197

Lis, J. & Eiben, A. E. (1996). A multi-sexual genetic algorithm for multiobjective optimization, In: T. Fukuda & T. Furuhashi, (Ed.), *Proceedings of the Third IEEE Conference on Evolutionary Computation*, IEEE Press, Piscataway NJ, pp. 59–64

Murata, T. & Ishibuchi, H. (1995). Moga: multi-objective genetic algorithms, In: *Proceedings of the IEEE International Conference on Evolutionary Computation*, Vol. 1, IEEE, IEEE Service Center, pp. 289–294

Osyczka, A. & Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm, *Structural and Multidisciplinary Optimization* 10(2), 94–99

Paredis, J. (1998). Coevolutionary algorithms, In: T. Bäck, D. Fogel & Z. Michalewicz, (Ed.), *Handbook of Evolutionary Computation*, 1st supplement, IOP Publishing and Oxford University Press

Potter, M. A. & De Jong, K. A. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents, *Evolutionary Computation* 8(1), 1–29

Ratford, M., Tuson, A. L. & Thompson, H. (1997). An investigation of sexual selection as a mechanism for obtaining multiple distinct solutions, Technical Report 879, Department of Artificial Intelligence, University of Edinburgh

Rudolph, G. (2001). Evolutionary search under partially ordered finite sets, In: *M. F. Sebaaly, (Ed.), Proceedings of the International NAISO Congress on Information Science Innovations* (ISI 2001), ICSC Academic Press, Dubai, U. A. E., pp. 818–822

Schaffer, J. D. (1984). Some experiments in machine learning using vector evaluated genetic algorithms, PhD thesis, Vanderbilt University

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms, In: J. Grefenstette, (Ed.), *Proceedings of the First International Conference on Genetic Algorithms,* Lawrence Erlbaum Associates Publishers, Hillsdale, New Jersey, pp. 93–100

Siwik, L. & Dreżewski, R. (2008). Agent-based multi-objective evolutionary algorithm with sexual selection, In: *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI 2008*), IEEE.

Srinivas, N. & Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms, *Evolutionary Computation* 2(3), 221–248

Todd, P. M. & Miller, G. F. (1997). Biodiversity through sexual selection, In: C. G. Langton & T. Shimohara, (Ed.), Artificial Life V: Proceedings of the Fifth

InternationalWorkshop on the Synthesis and Simulation of Living Systems (Complex Adaptive Systems), Bradford Books, pp. 289–299

Van Veldhuizen, D. A. (1999). Multiobjective Evolutionary Algorithms: Classifications, Analyses and New Innovations, PhD thesis, Graduate School of Engineering of the Air Force Institute of Technology Air University

Zitzler, E. (1999). Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, PhD thesis, ETH Zurich, Switzerland

Zitzler, E., Deb, K.& Thiele, L. (2000). Comparison of Multiobjective Evolutionary Algorithms: Empirical Results, *Evolutionary Computation* 8(2), 173–195

Zitzler, E., Laumanns, M. & Thiele, L. (2001). SPEA2: Improving the strength pareto evolutionary algorithm, Technical Report TIK-Report 103, Computer Engineering and Networks Laboratory (TIK), Department of Electrical Engineering, Swiss Federal Institute of Technology (ETH) Zurich, ETH Zentrum, Gloriastrasse 35, CH-8092 Zurich, Switzerland

Zitzler, E., Laumanns, M. & Thiele, L. (2002). SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization, In: K. Giannakoglou et al., (Ed.), *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems* (EUROGEN 2001), International Center for Numerical Methods in Engineering (CIMNE), pp. 95–100

Zitzler, E. & Thiele, L. (1998). An evolutionary algorithm formultiobjective optimization: The strength pareto approach, Technical Report 43, Swiss Federal Institute of Technology, Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M. & da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: An analysis and review, *IEEE Transactions on Evolutionary Computation* 7(2), 117–132

Evolutionary Multi-Objective Robust Optimization

J. Ferreira*, C. M. Fonseca**, J. A. Covas* and A. Gaspar-Cunha*
** I3N, University of Minho, Guimarães,*
*** Centre for Intelligent Systems, University of Algarve, Faro, Portugal*

1. Introduction

Most practical engineering optimization problems are multi-objective, *i.e.*, their solution must consider simultaneously various performance criteria, which are often conflicting. Multi-Objective Evolutionary Algorithms (MOEAs) are particularly adequate for solving these problems, as they work with a population (of vectors or solutions) rather than with a single point (Schaffer, 1984; Fonseca & Fleming, 1993; Srinivas & Deb, 1995; Horn et al., 1994; Deb et al., 2002; Zitzler et al., 2001; Knowles & Corne, 2000; Gaspar-Cunha et al. 2004). This feature enables the creation of Pareto frontiers representing the trade-off between the criteria, simultaneously providing a link with the decision variables (Deb, 2001, Coello et al., 2002). Moreover, since in real applications small changes of the design variables or of environmental parameters may frequently occur, the performance of the optimal solution (or solutions) should be only slightly affected by these, *i.e.*, the solutions should also be robust (Ray, 2002; Jin & Branke, 2005). The optimization problems involving unmanageable stochastic factors can be typified as (Jin & Branke, 2005): i) those where the performance is affected by noise originated by sources such as sensor measurements and/or environmental parameters (Wiesmann et al., 1998; Das, 1997); ii) those where the design variables change after the optimal solution has been found (Ray, 2002; Tsutsui & Ghosh, 1997; Chen et al., 1999); iii) problems where the process performance is estimated by an approximation to the real value; iv) and those where the performance changes with time, which implies that the optimization algorithm must be updated continuously. This text focuses exclusively problems of the second category.
Given the above, optimization algorithms should determine the solutions that simultaneously maximize performance and guarantee satisfactory robustness, but the latter is rarely included in traditional algorithms. As robustness and performance can be conflicting, it is important to know their interdependency for each optimization problem. A robustness analysis should be performed as the search proceeds and not after, by introducing a robustness measure during the optimization. Robustness can be studied either by replacing the original objective function by an expression measuring both the performance and the expectation of each criterion in the vicinity of a specific solution, or by inserting an additional optimization criterion assessing robustness in addition to the original

criteria. As will be demonstrated in the next sections, in the first situation the role of the optimization algorithm is to find the solution that optimizes the expectation (in the vicinity of the solutions considered) of the original criterion (or criteria), while in the second case a trade-off between the original criteria and the robustness measure is obtained (Jin & Sendhoff, 2003).

In single objective (or criterion) optimization, the best solution is the one that satisfies simultaneously performance and robustness. Robust single objective optimization has been applied to various engineering fields and using different optimization methodologies (Ribeiro & Elsayed, 1995; Tsutsui & Ghosh, 1997; Das, 1997; Wiesmann et al., 1998; Du & Chen, 1998; Chen et al. 1999; Ray, 2002; Arnold & Beyer, 2003; Sorensen, 2004). However, only recently robustness analysis has been extended to Multi-Objective Optimization Problems (MOOP) (Kouvelis & Sayin 2002; Bagchi, 2003; Jin & Sendhoff, 2003; Kazancioglu et al., 2003; Gaspar-Cunha & Covas, 2005; Ölvander, 2005; Guanawan & Azarm, 2005; Deb & Gupta, 2006; Paenke et al., 2006; Barrico & Antunes, 2006; Moshaiov & Avigrad, 2006; Gaspar-Cunha & Covas, 2008). Depending on the type of Pareto frontier, the aim can be: i) to locate the optimal Pareto front's most robust section (Deb & Gupta, 2006; Gaspar-Cunha & Covas, 2008) and/or ii) in the case of a multimodal problem, to find the most robust Pareto frontier, and not only the most robust region of the optimal Pareto frontier (Guanawan & Azarm, 2005; Deb & Gupta, 2006).

An important question arising from MOOP is the choice of the (single) solution to be used on the real problem under study (Ferreira et al., 2008). Generally, to select a solution from the pool of the available ones, the Decision Maker (DM) characterizes the relative importance of the criteria and subsequently applies a decision methodology. The use of a weighted stress function approach (Ferreira et al., 2008) is advantageous, as it enables the DM to define the extension of the optimal Pareto frontier to be obtained, via the use of a dispersion parameter. This concept could be adapted by taking into account robustness and not the relative criteria importance.

Consequently, this work aims to discuss robustness assessment during multi-objective optimization using a MOEA, namely in terms of the identification of the robust region (or regions) of the optimal Pareto frontier. The text is organized as follows. In section 2, robustness concepts will be presented and extended to multi-objective optimization. The multi-objective evolutionary algorithm used and the corresponding modifications required to take robustness into account will be described and discussed in section 3. The performance of the robustness measures will be evaluated in section 4 via their application to several benchmark multi-objective optimization problems. Finally, the main conclusions are summarized in section 5.

2. Robustness concepts

2.1 Single objective optimization

A single objective optimization can be formulated as follows:

$$\begin{array}{lll} \underset{x_l}{max} & f(x_l) & l = 1,\cdots,L \\ \text{subject to} & g_j(x_l) = 0 & j = 1,\cdots,J \\ & h_k(x_l) \geq 0 & k = 1,\cdots,K \\ & x_{l,min} \leq x_l \leq xl,_{max} & \end{array} \tag{1}$$

where x_l are the L parameters (or design vectors) $x_1, x_2, \ldots, x_L$, g_j and h_k are the J equality ($J \geq 0$) and K inequality ($K \geq 0$) constraints, respectively, and $x_{l,min}$ and $x_{l,max}$ are the lower and upper limits of the parameters.

The most robust solution is that for which the objective function f is less sensitive to variations of the design parameters x_l. Figure 1 shows the evolution of the objective function $f(x_1,x_2)$ (to be maximized) against the design parameter x_1, when another factor and/or the design parameter x_2 changes slightly from x_2' to x_2''. Solution S_2 is less sensitive than solution S_1 to variations of x_2, since the changes in the objective function are less significant (Δf_2 and Δf_1 for S_2 and S_1, respectively) and, consequently, it can be considered as the most robust solution (taking into consideration that here robustness is measured only as a function of changes occurring in the objective function). On the other hand, since S_1 is more performing than S_2, a balance between performance (or fitness) of a solution and its robustness has to be done. In spite of its lower fitness, solution S_2 is the most robust and would be the selected one by an optimization algorithm (Guanawan & Azarm, 2005; Gaspar-Cunha & Covas, 2005; Deb & Gupta, 2006; Paenke et al., 2006; Gaspar-Cunha & Covas, 2008).

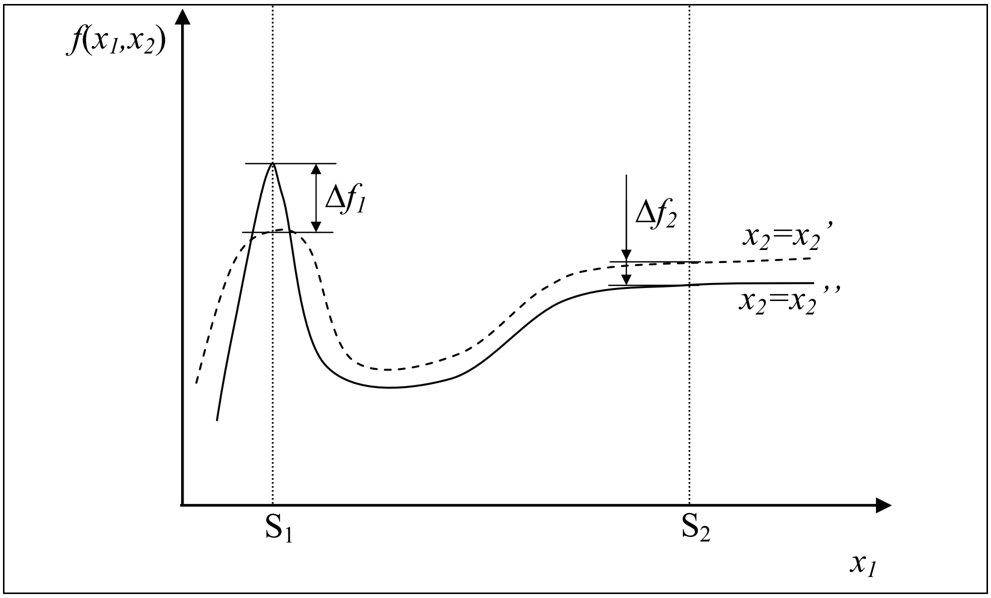


Fig. 1. Concept of robustness in the case of a single objective function

Two major approaches have been developed in order to deal with robustness in an optimization process (Ray, 2002; Jin & Sendhoff, 2003; Gaspar-Cunha & Covas, 2005; Deb & Gupta, 2006; Gaspar-Cunha & Covas, 2008):

- **Expectation measure**: the original objective function is replaced by a measure of both its performance and expectation in the vicinity of the solution considered. Figure 2 illustrates this method. Figure 2-A shows that in function $f(x)$, having five different peaks, the third is the most robust, since fitness fluctuations around its maximum are smaller. However, most probably, an optimization algorithm would select the first peak. An expectation measure takes this fact into account by replacing the original function by another such as that illustrated in Figure 2-B. Now, if a conventional optimization is performed using this new function, the peak selected (peak three) will be the most robust. Various types of expectation measures have been proposed in the literature (Tsutsui & Ghosh, 1997; Das, 1997; Wiesmann et al., 1998; Jin & Sendhoff, 2003; Gaspar-Cunha & Covas, 2005; Deb & Gupta, 2006; Gaspar-Cunha & Covas, 2008).

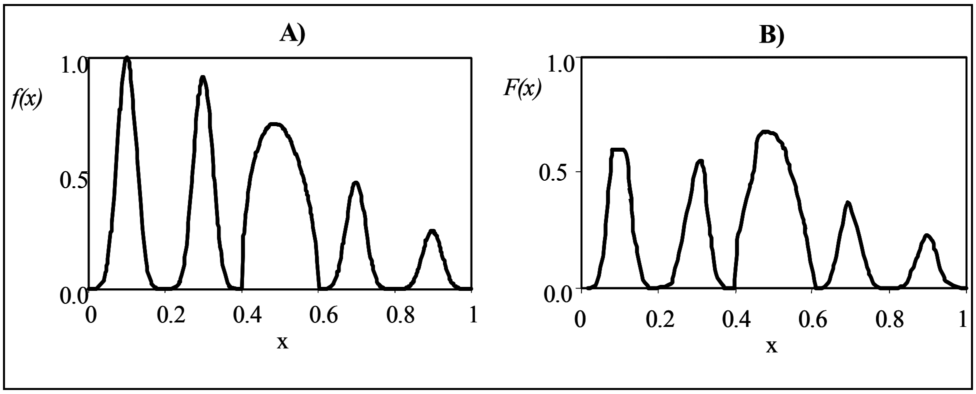


Fig. 2. Expectation measure for a single objective function

- **Variance measure**: An additional criterion is appended to the objective function to measure the deviation of the latter around the vicinity of the design point. Variance measures take only into account function deviations, ignoring the associated performance. Thus, in the case of a single objective function, the optimization algorithm must perform a two-criterion optimization, one concerning performance and the other robustness (Jin & Sendhoff, 2003; Gaspar-Cunha & Covas, 2005; Deb & Gupta, 2006; Gaspar-Cunha & Covas, 2008).

Deb & Gupta (2006) denoted the above two approaches as type I and II, respectively. The performance of selected expectation and variance measures was evaluated in terms of their capacity to detect robust peaks (Gaspar-Cunha & Covas, 2008), by assessing such features as: i) easy application to problems where the shape of the objective function is not known a priori, ii) capacity to define robustness regardless of that shape, iii) independence of the algorithm parameters, iv) clear definition of the function maxima in the Fitness versus Robustness Pareto representation, and v) efficiency. The best performance was attained when the following variance measure was used:

$$f_i^R = \frac{1}{N'} \sum_{j=0}^{N} \left| \frac{\tilde{f}(x_j) - \tilde{f}(x_i)}{x_j - x_i} \right|, \qquad d_{i,j} < d_{\max} \tag{2}$$

where the robustness of individual i is defined as the average value of the ratio of the difference between the normalized fitness of individual i, $\tilde{f}(x_i)$, and that of its neighbours (j), over the distance separating them. In this expression, $\tilde{f}(x_i) = \frac{f(x_i) - f_{\min}}{f_{\max} - f_{\min}}$ for maximization and $\tilde{f}(x_i) = 1 - \frac{f(x_i) - f_{\min}}{f_{\max} - f_{\min}}$ for minimization of the objective function $f(x_i)$, with f_{max} and f_{min} representing the limits of its range of variation, N' is the number of population individuals whose Euclidian distance between points i and j ($d_{i,j}$) is lower than d_{max} (*i.e.*, $d_{i,j} < d_{max}$):

$$d_{i,j} = \sqrt{\sum_{m=1}^{M} \left(x_{m,j} - x_{m,i} \right)^2} \tag{3}$$

and M is the number of criteria. The smaller f^R_i, the more robust the solution is.

2.2 Extending robustness to multiple objectives

In a multi-objective optimization various objectives, often conflicting, co-exist:

$$\begin{array}{llll} \underset{x_l}{max} & f_m(x_l) & l = 1, \cdots, L & m = 1, \cdots, M \\ \text{subject to} & g_j(x_l) = 0 & j = 1, \cdots, J & \\ & h_k(x_l) \geq 0 & k = 1, \cdots, K & \\ & x_{l,min} \leq x_l \leq xl,_{max} & & \end{array} \tag{4}$$

where f_m are the M objective functions of the L parameters (or design vectors) x_1, x_2, …, x_L and g_j and h_k are the J equality ($J \geq 0$) and K inequality ($K \geq 0$) constraints, respectively.

The application of a robustness analysis to MOOPs must consider all the criteria simultaneously. As for single objective, a multi-objective robust solution must be less sensitive to variations of the design parameters, as illustrated in Figure 3. The figure shows that the same local perturbation on the parameters space (x_1, x_2) causes different behaviours of solutions I and II. Solution I is more robust, as the same perturbations on the parameters space causes lower changes on the objective space. Each of the Pareto optimal solutions must be analysed in what concerns robustness, *i.e.*, its sensitivity to changes on the design parameters. Since robustness must be assessed for every criterion, the combined effect of changes in all the objectives must be considered simultaneously and used as a measure of robustness.

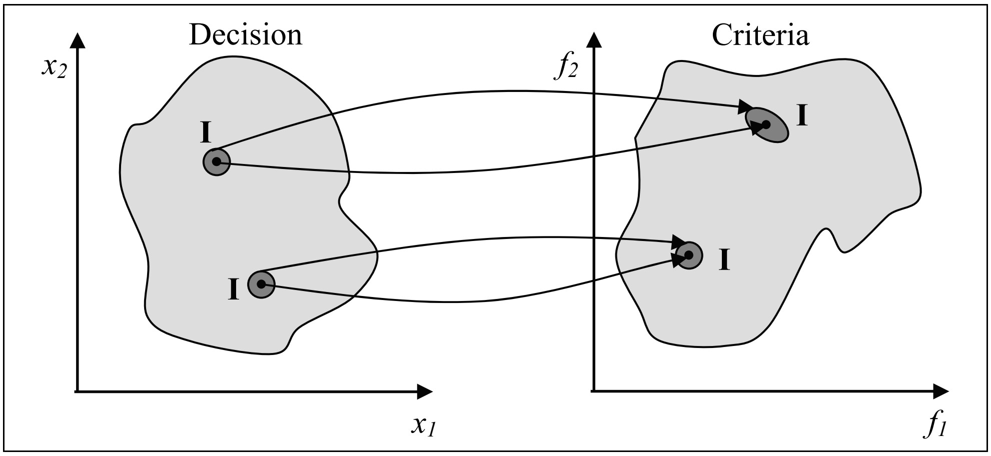


Fig. 3. Concept of robustness for multi-objective functions

In multi-objective robust optimization the aim is to obtain a set of Pareto solutions that are, at the same time, multi-objectively robust and Pareto optimal. As shown in Figure 4, different situations may arise (Guanawan & Azarm, 2005; Deb & Gupta, 2006):

1. All the solutions on the Pareto-optimal frontier are robust (Figure 4-A);
2. Only some of the solutions belonging to the Pareto-optimal frontier are robust (Figure 4-B);
3. The solutions belonging to the Pareto-optimal frontier are not robust, but a robust Pareto frontier exists (Figure 4-C);
4. Some of the robust solutions belong to the Pareto-optimal frontier, but others do not (Figure 4-D).

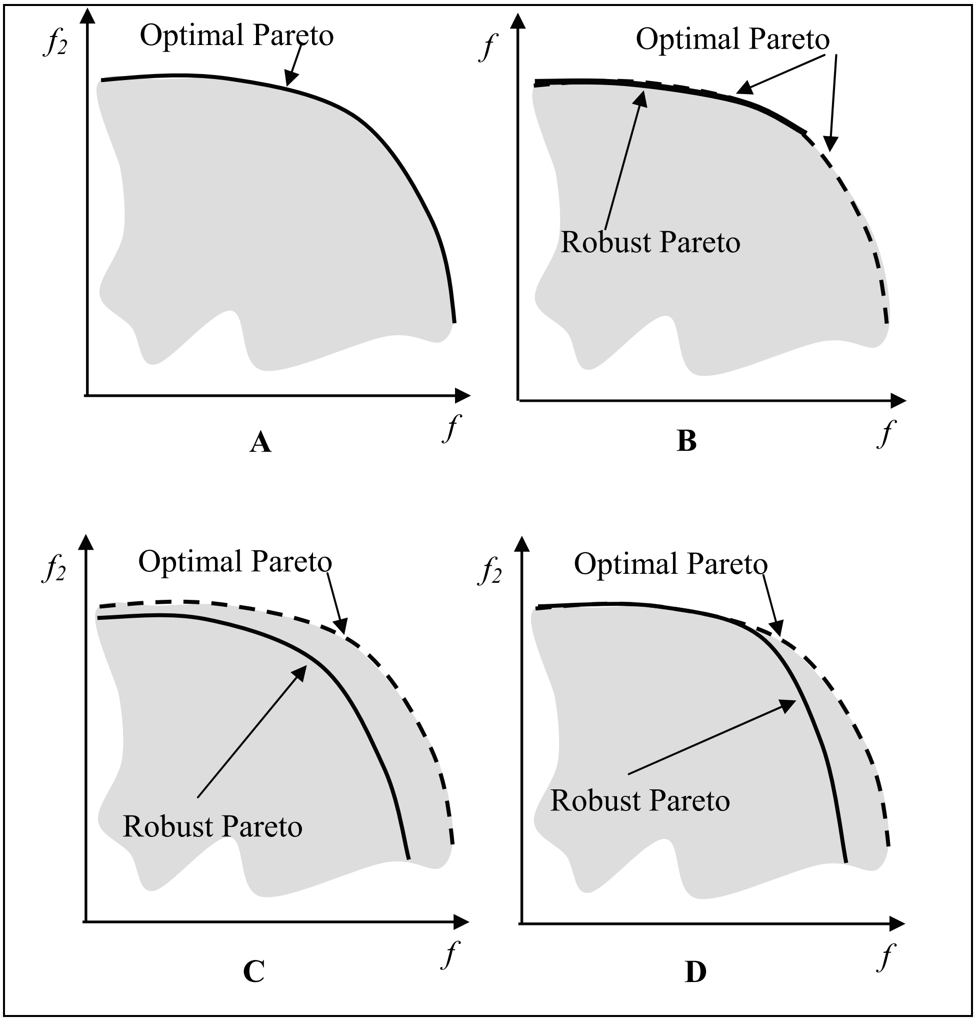


Fig. 4. Optimal Pareto frontier versus robust Pareto frontier

All the above situations should be taken into consideration by a resourceful optimization algorithm. When the DM is only interested in the most robust section of the optimal Pareto frontier (see Figure 5), this can be done by using, for example, the dispersion parameter referred above.

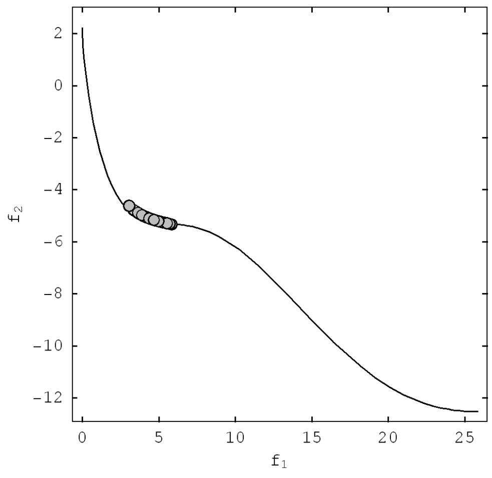


Fig. 5. Robust region of the optimal Pareto frontier (Test Problem 1, see below)

3. Multi-objective optimization

3.1 Multi-Objective Evolutionary Algorithms (MOEAs)

Multi-Objective Evolutionary Algorithms (MOEAs) are an efficient tool to deal with the above type of problems, since they are able to determine in a single run the optimal Pareto front. For that reason, they have been intensively used in the last decade (Fonseca & Fleming, 1998; Deb, 2001, Coello et al., 2002; Gaspar-Cunha & Covas, 2004).

A MOEA must provide the homogeneous distribution of the population along the Pareto frontier, together with improving the solutions along successive generations. Usually, a fitness assignment operator is applied to guide the population towards the Pareto frontier using a robust and efficient multi-objective selection method, as well as a density estimation operator to maintain the solutions dispersed along the Pareto frontier, as it is able to take into account the proximity of the solutions. Moreover, in order to prevent fitness deterioration along the successive generations, an archiving process is introduced by maintaining an external population where the best solutions found sequentially are kept and periodically incorporated into the main population.

The Reduced Pareto Set Genetic Algorithm with elitism (RPSGAe) will be adopted in this chapter (Gaspar-Cunha et al., 1997), although some changes in its working mode have to be implemented in order to take into account the robustness procedure proposed. RPSGAe is able to distribute the solutions uniformly along the Pareto frontier, its performance having been assessed using benchmark problems and statistical comparison techniques. The method starts by sorting the population individuals in a number of pre-defined ranks using a clustering technique, thus reducing the number of solutions on the efficient frontier while maintaining intact its characteristics (Gaspar-Cunha & Covas, 2004). Then, the individuals' fitness is calculated through a ranking function. With the aim of incorporating this technique, the traditional GA was modified as follows (Gaspar-Cunha & Covas, 2004):

```
1. Random initial population (internal)
2. Empty external population
3. while not Stop-Condition do
        a- Evaluate internal population
        b- **Calculate expectation and/or robustness measures**
        c- **Calculate niche count (mi)**
        d- Calculate the Ranking of the individuals using the RPSGAe
        e- **Calculate the global Fitness** ($\widetilde{F}(i)$)
        f- Copy the best individuals to the external population
        g- if the external population becomes full
                Apply the RPSGAe to this population
                Copy the best individuals to the internal population
        end if
        h- Select the individuals for reproduction
        i- Crossover
        j- Mutation
end while
```

As described above, the calculations start with the random definition of an internal population of size *N* and of an empty external population of size *Ne*. At each generation, a fixed number of the best individuals (that was obtained by reducing the internal population with the clustering algorithm), is copied to an external population (Gaspar-Cunha et al., 1997). The process is repeated until the external population becomes complete. Then, the RPSGAe is applied to sort the individuals of this population, and a pre-defined number of the best individuals is incorporated in the internal population, by replacing the lowest fitness individuals. Detailed information on this algorithm can be found elsewhere (Gaspar-Cunha & Covas, 2004; Gaspar-Cunha, 2000).

3.2 Introducing robustness in MOEAs

Three additional steps must be added on to the RPSGAe presented above, to comprise robustness estimation. They consist of a computation of robustness measures (taking into account the dispersion parameter), a niche count and the determination of the global fitness, yielding the general flowchart of Figure 7. The dispersion parameter (ε') quantifies the extension of the robust section to be obtained (see Figure 5). This parameter can be defined by the DM and ranges between 0, when a single solution is to be obtained, and 1, when the entire optimal Pareto frontier is to be obtained. In order to consider the influence of the dispersion parameter (ε'), the way how the indifference limits ($\widetilde{L}_j$) and the distances between the solutions ($\widetilde{D}_{j,k}$) are defined in the RPSGAe algorithm was also changed (see Gaspar-Cunha & Covas, 2004), the following equations being used:

$$\widetilde{L}_j = L_j \times \left(1 + \frac{2}{(\max R - \min R)}\right)^{\frac{1-\varepsilon'}{\varepsilon'}} \tag{5}$$

$$\widetilde{D}_{j,k} = D_{j,k} \times \left(1 + \frac{1}{R(\text{ind}_{k+1})}\right)^{\frac{1-\varepsilon'}{\varepsilon'}} \tag{6}$$

Here, max R and min R are the maximum and the minimum values of the robustness found for each generation, respectively, L_i are the indifference limits for criterion i, $D_{j,k}$ is the difference between the criterion value of solutions j and k, $R(ind_{k+1})$ is the robustness measure of the individual located in position k+1 after the population was ordered by criterion j. The robustness measure is calculated by Equation 2, thus when R increases the robustness of the solution decreases. In these equations, the dispersion parameter (ε') plays an important role. If ε'=1, equations 5 and 6 are reduced to L_i and $D_{i,j}$, respectively, and the algorithm will converge for the entire robust Pareto frontier. Otherwise, when ε' decreases, the size of the robust Pareto frontier decreases as well. In a limiting situation, *i.e.*, when ε' is approximately nil, a single point is obtained. Figure 8 shows curves of $\widetilde{L}_j / L_j$ and $\widetilde{D}_{j,k} / D_{j,k}$ ratios against the dispersion parameter, for different values of R (2.0, 0.5 and 0.1).

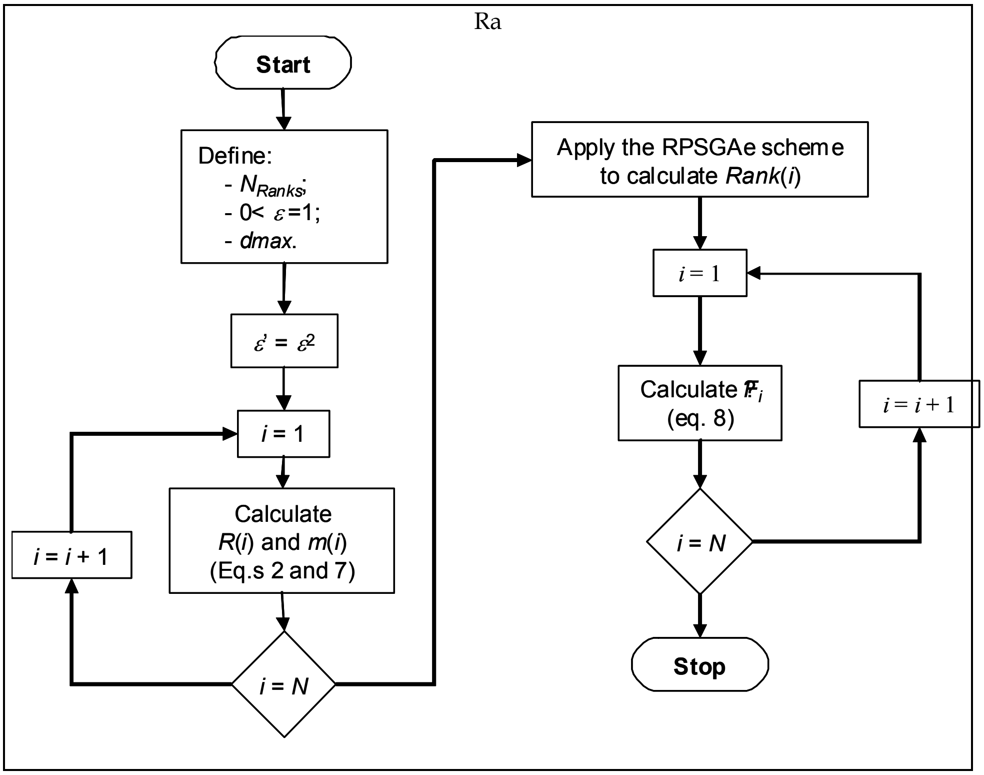


Fig. 7. Flowchart of the robustness routine

Ratio $\widetilde{D}_{j,k} / D_{j,k}$ is given by $\left(1 + \frac{1}{R(\text{ind}_{k+1})}\right)^{\frac{1-\varepsilon'}{\varepsilon'}}$ (see equation 6). Thus, at constant R, when ε' decreases means that influence of the difference between the value of solutions j and k (*i.e.*,

$D_{j,k}$) on $\tilde{D}_{j,k}$ diminishes. Therefore, for small values of the dispersion parameter, the attribution of the fitness by the RPSGAe algorithm is made almost exclusively by the value of the robustness of the solutions and not by taking into account the distance between them. This procedure avoids that robust solutions are eliminated during the consecutive generations in case they are next to each other. An identical analysis can be made for different robustness values (R in Figure 8). When R increases (*i.e.*, when the robustness decreases) the value of $\tilde{D}_{j,k} / D_{j,k}$ must decreases in order to produce the same result. The same reasoning applies to the $\tilde{L}_j / L_j$ ratio.

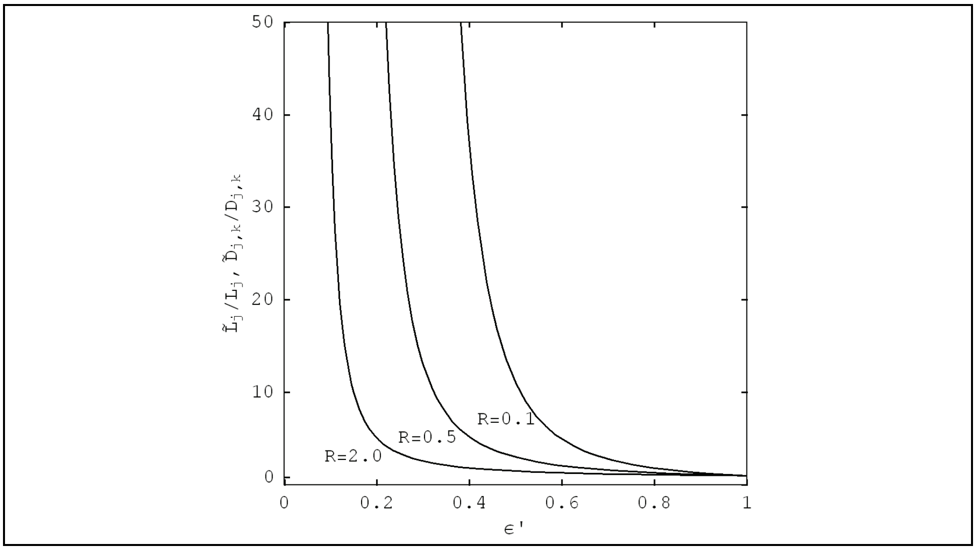


Fig. 8. Shape of the curves of $\tilde{L}_j / L_j$ and $\tilde{D}_{j,k} / D_{j,k}$ rates as a function of the dispersion parameter for different R values

The niche count was considered using a sharing function (Goldberg & Richardson, 1987):

$$m(i) = \sum_{j=1}^{N} sh(d_{ij}) \tag{7}$$

where $sh(d_{ij})$ is related to individual i and takes into account its distance to all its neighbours j (d_{ij}).

Finally, the global fitness was calculated using the following equation:

$$\tilde{F}(i) = Rank(i) + (1-\varepsilon')\frac{R(i)}{R(i)+1} + \varepsilon'\frac{m(i)}{m(i)+1} \tag{8}$$

In conclusion, the following calculation steps must be carried out (see Figure 7):

1. The robustness routine starts with the definition of the number of ranks (N_{ranks}), the span of the Pareto frontier to be obtained ($\varepsilon \in [0,1]$) and the maximum radial distance to each solution to be considered in the robustness calculation (d_{max});

2. To reduce the sensitivity of the algorithm to small values of the objective functions, the dispersion parameter is changed as $\varepsilon' = \varepsilon^2$;
3. For each individual, *i*, robustness , *R*(*i*), and niche count, *m*(*i*) , are determined using equations 2 and 7, respectively;
4. The RPSGAe algorithm is applied, with the modifications introduced by equations 5 and 6, to calculate *Rank*(*i*);
5. For each solution, *i*, the new fitness is calculated using equation 8.

4. Results and discussion

4.1 Test problems

The robustness methodology presented in the previous sections will be tested using the 7 Test Problems (TP) listed below, each of different type and with distinctive Pareto frontier characteristics. Each TP is presented in terms of its creator, aim, number of decision parameters, criteria and range of variation of the decision parameter.

TP 1 and 2 are simple one parameter problems, the first having one region with higher robustness, while the second contains three such regions. TP 3 to TP5 are complex MOOPs with 30 parameters each, and two criteria. TP3 and TP4 have a single region with higher robustness and the Pareto frontier is convex and concave, respectively. TP5 has a discontinuous Pareto frontier with a single region with higher robustness. TP 6 and TP7 are the three criteria version of TP1 and TP4, respectively.

Three studies will be performed, to determine: i) the effect of the RPSGAe algorithm, *i.e.*, N_{ranks}, and d_{max}; ii) the effect of the value of the dispersion parameter and iii) the performance of the robustness methodology for different type of problems.

The RPSGAe algorithm parameters utilized are the following: N_{ranks} = 20 (the values of 10 and 30 were also used for the first study), d_{max} = 0.008 (0.005 and 0.03 were also tried in the first study), indifference limits equal to 0.1 for all criteria, SBX real crossover operator with an index of 10 and real polynomial mutation operator with and index of 20.

TP 1: $x \in [-2;6]$; Minimize; *L*=1; *M*=2.

$$\begin{aligned} f_1(x) &= x^2 \\ f_2(x) &= e^{|x|-5} + (6/5)\cos(2x) - 2{,}7x + 1 \end{aligned} \tag{9}$$

TP 2: $x \in [0;5]$; Maximize; *L*=1; *M*=2.

$$\begin{aligned} f_1(x) &= x \\ f_2(x) &= -5x + \cos(4x) \end{aligned} \tag{10}$$

TP 3 (ZDT1): $x_i \in [0;1]$; Minimize; *L*=30; *M*=2; Deb, Pratapat et al., 2002.

$$\begin{aligned} & f_1(x_1) = x_1 \\ & f_2(x_2, \cdots, x_L) = g(x) \times \left(1 - \sqrt{f_1(x_1)/g(x)}\right) \\ & \text{with, } g(x) = 1 + 9\frac{\sum_{l=2}^{L} x_l}{L-1} \end{aligned} \tag{11}$$

TP 4 (ZDT2): x$_i$ ∈[0;1]; Minimize; *L*=30; *M*=2; Deb, Pratapat et al., 2002.

$$f_1(x_1) = x_1$$
$$f_2(x_2,\cdots,x_L) = g(x)\times\left(1-\left(f_1(x_1)\big/ g(x)\right)^2\right) \tag{12}$$
$$\text{with, } g(x) = 1 + 9\frac{\sum_{l=2}^{L} x_l}{L-1}$$

TP 5 (ZDT3): x$_i$ ∈[0;1]; Minimize; *L*=30; *M*=2; Deb, Pratapat et al., 2002.

$$f_1(x_1) = x_1$$
$$f_2(x_2,\cdots,x_L) = g(x)\times\left(1-\sqrt{x_1\big/ g(x)} - \frac{x_1}{g(x)}\sin(10\pi x_1)\right) \tag{13}$$
$$\text{with, } g(x) = 1 + 9\frac{\sum_{l=2}^{L} x_l}{L-1}$$

TP 6: x$_1$ ∈[0;2π]; x$_2$ ∈[0;5]; Minimize; *L*=2; *M*=3.

$$\begin{aligned} &f_1(x) = sin(x_1).g(x_2) \\ &f_2(x) = cos(x_1).g(x_2) \\ &f_3(x) = x_2^2 \\ &\text{with, } g(x_2) = e^{x_2(x_2-5)} - \frac{6}{5}sin(2x_2) - 2{,}7x_2 - 1 \end{aligned} \tag{14}$$

TP 7 (DTLZ2): x$_i$ ∈[0;1]; Minimize; *L*=12; *M*=3; Deb, Thiele et al., 2002.

$$\begin{aligned} &f_1(x) = (1+g(x)).cos\left(x_1\frac{\pi}{2}\right).cos\left(x_2\frac{\pi}{2}\right) \\ &f_2(x) = (1+g(x)).cos\left(x_1\frac{\pi}{2}\right).sin\left(x_2\frac{\pi}{2}\right) \\ &f_3(x) = (1+g(x)).sin\left(x_1\frac{\pi}{2}\right) \\ &\text{with, } g(x) = \sum_{i=3}^{L}(x_i - 0.5)^2 \end{aligned} \tag{15}$$

4.2 Effect of the RPSGAe parameters

Figure 9 compares the results obtained with the robustness procedure for TP 1 and TP4, using different values of the parameter. The line indicates the optimal Pareto frontier and the dots identify the solutions obtained with the new procedure. As shown, the algorithm is able to produce good results independently of the value of N_{ranks} (hence, in the remaining of this study N_{ranks} was set as 20).

Similar conclusions were obtained for d_{max} parameter - Figure 10, so d_{max} was kept equal to 0.008.

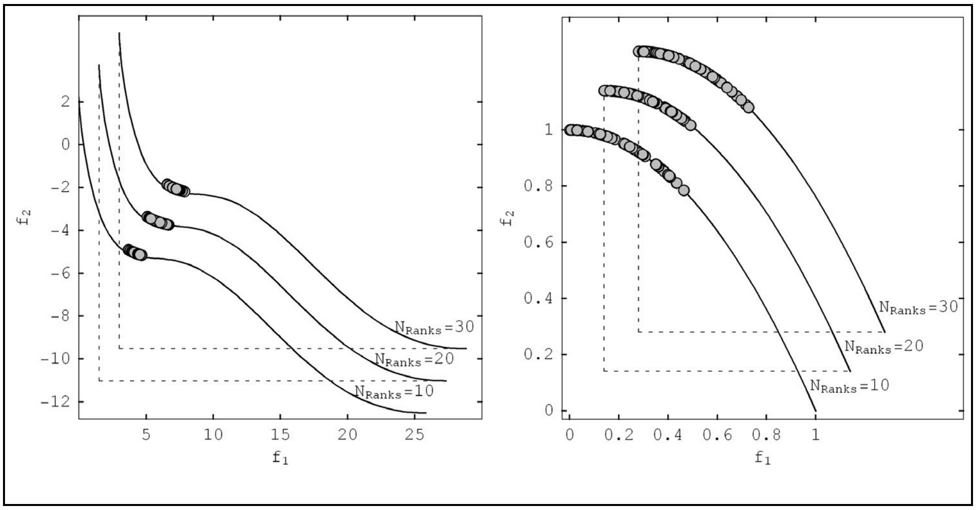


Fig. 9. Influence of N_{ranks} parameter for TP1 and TP4

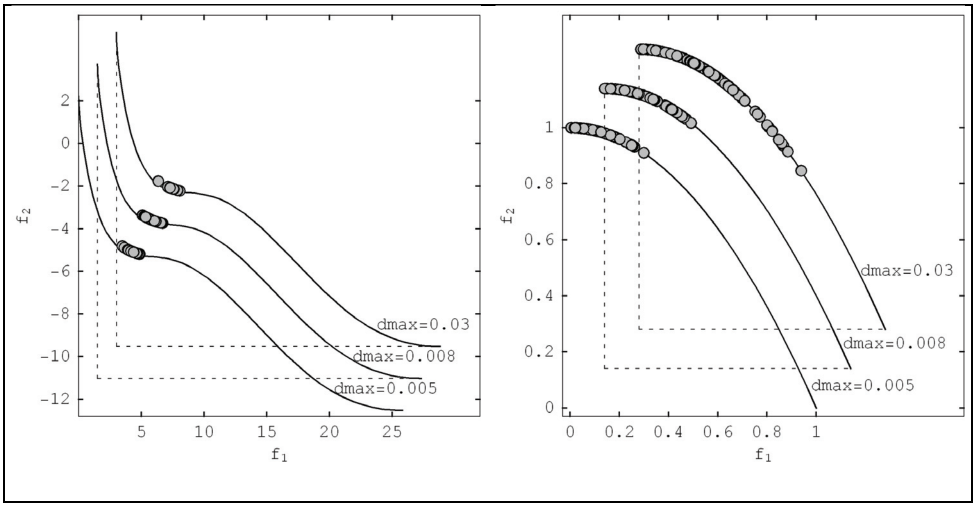


Fig. 10. Influence of d_{max} parameter for TP1 and TP4

4.3 Effect of the dispersion parameter

The aim of the dispersion parameter is to provide the Decision Maker with the possibility of choosing different sizes of the optimal/robustness Pareto frontier. Figure 11 shows the results obtained for TP1 using different values of that parameter, identical outcomes having been observed for the remaining test problems. The methodology seems to be sensitive to the variation on the dispersion parameter, which is a very positive feature.

4.4 Effect of the type of problem

The results obtained for TP2 to TP7, using $\varepsilon = 0.1$, are presented in Figure 12. The algorithm is able to deal with the various types of test problems proposed. TP2 is a difficult test problem due to the need to converge to the three different sections with the same robustness. TP3 and TP4 show that the algorithm proposed can converge to the most robust region even for problems with 30 parameters or of discontinuous nature. Finally, TP6 and TP7 show that the methodology proposed is able to deal with more than two dimensions with a good convergence, which is not generally the case for current optimization algorithms available.

5. Conclusions

This work presented and tested an optimization procedure that takes into account robustness in multi-objective optimization. It was shown that the method is able to deal with different types of problems and with different degrees of complexity.
The extension of the robust Pareto frontier can be controlled by the Decision Maker by making use of a dispersion parameter. The effectiveness of this parameter was demonstrated in a number of test problems.

6. References

Arrold, D.V. & Beyer, H.-G. (2003). A Comparison of Evolution Strategies with Other Direct Search Methods in the Presence of Noise, *Computational Optimization and Applications*, Vol. 24, No. 1 (2003) 135-159

Bagchi, T.P. (2003). Multiobjective Robust Design by Genetic Algorithms, *Materials and Manufacturing Processes*, Vol. 18, No. 3 (2003) 341-354

Barrico, C. & Antunes, C.H. (2006). Robustness Analysis in Multi-Objective Optimization Using a Degree of Robustness Concept, *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 6778-6783, Vancouver, Canada, July 2006, IEEE

Chen, W.; Sahai, A.; Messac, A. & Sundararaj, G. (1999). Physical Programming for Robust Design, *Proceedings of 40th Structures, Structural Dynamics and Materials Conference*, St. Louis, USA, April 1999

Coello, C.; Veldhuizen, D. & Lamont, G. (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer, ISBN 0306467623, Norwell

Das, I. (1997). *Nonlinear Multicriteria Optimization and Robust Optimality*, Rice University, PhD Thesis, Houston

Deb K. (2001). *Multi-Objective Optimisation Using Evolutionary Algorithms*, Wiley, ISBN 0-471-87339-X, Chichester

Deb, K.; Pratap, A.; Agrawal, S. & Meyarivan, T. (2002). A Fast and Elitist Multi-Objective Genetic Algorithm: NSGAII, *IEE Transactions on Evolutionary Computation*, Vol. 6, No. 2 (April 2002) 182-197, ISBN 1089-778X.

Ded, K.; Thiele, L.,; Laumanns, M. & Zitzler E. (2002). Scalable Multi-Objective Optimization Test Problems, *IEEE Transactions on Evolutionary Computation*, Vol. 1, (May 2002) 825-830, ISBN 0-7803-7282-4

Deb, K. & Gupta, H. (2005). Searching for robust Pareto-optimal solutions in multi-objective optimization, *Proceedings of the Third International Conference on Evolutionary Multi-*

Criterion Optimization, pp. 150-164, ISBN 978-3-540-24983-2, Guanajuato, Mexico, January 2005, Springer, Berlin

Deb, K. & Gupta, H. (2006). Introducing Robustness in Multi-objective Optimization. *Evolutionary Computation*, Vol. 14, No. 4, (December 2006) 463-494, 1063-6560.

Du, X. & Chen, W. (1998). Towards a Better Understanding of Modelling Feasibility Robustness in Engineering Design, *Proceedings of DETC 99*, Las Vegas, USA, September 1999, ASM

Ferreira, J.C.; Fonseca, C.M. & Gaspar-Cunha, A. (2008) Methodology to Select Solutions for Multi-Objective Optimization Problems: Weight Stress Function Method, *Applied Intelligence*, Accepted for publication (2008)

Fonseca, C.; Fleming, P. (1993). Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, *Proceedings of Fifth International Conference on Genetic Algorithms*, pp. 416-423, University of Illinois, July 1993, Morgan Kauffman, Urbana-Champaign

Fonseca, C. & Fleming, P. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms, part I: A unified formulation, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 28, No. 1 (1998) 26-37

Gaspar-Cunha, A.; Oliveira, P. & Covas, J. (1997). Use of Genetic Algorithms in Multicriteria Optimization to Solve Industrial Problems, *Proceedings of Seventh Int. Conf. on Genetic Algorithms*, pp. 682-688, Michigan, USA.

Gaspar-Cunha, A. (2000). *Modelling and Optimisation of Single Screw Extrusion*, University of Minho, PhD Thesis, Guimarães, Portugal

Gaspar-Cunha, A. & Covas, J. (2004). RPSGAe - A Multiobjective Genetic Algorithm with Elitism: Application to Polymer Extrusion, In: *Metaheuristics for Multiobjective Optimisation, Lecture Notes in Economics and Mathematical Systems*, Gandibleux, X.; Sevaux, M.; Sörensen, K.; T'kindt, V. (Eds.), 221-249, Springer, ISBN 3-540-20637-X, Berlin

Gaspar-Cunha, A. & Covas, J. (2005). Robustness using Multi-Objective Evolutionary Algorithms, *Proceedings of 10th Online World Conference in Soft Computing in Industrial Applications*, pp. 189-193, ISBN 3-540-29123-7 (http://www.cranfield.ac.uk/wsc10/), September 2005, Springer, Berlin

Gaspar-Cunha, A. & Covas, J. (2008). Robustness in Multi-Objective Optimization using Evolutionary Algorithms, *Computational Optimization and Applications*, Vol. 39, No. 1, (January 2008) 75-96, ISBN 0926-6003

Goldberg, D. & Richardson, J. (1987). Genetic Algorithms with Sharing for Multimodal Function Optimization, *Proceedings of Second Int. Conf. on Genetic Algorithms*, pp. 41-49, 0-8058-0158-8, Cambridge, July 1985, Lawrence Erlbaum Associates, Mahwah

Goldberg, D. (1989). Genetic Algorithms in Search, Optimisation and Machine Learning, Addison-Wesley, 0201157675, Reading

Guanawan, S. & Azarm, S. (2005). Multi-Objective Robust Optimization using a Sensitivity Region Concept, *Struct. Multidisciplinar Optimization,* Vol. 29, No. 1 (2005) 50-60

Horn, J.; Nafpliotis, N. & Goldberg, D. (1994), A Niched Pareto Genetic Algorithm for Multiobjective Optimization, *Proceedings of First IEEE Conference on Evolutionary Computation*, pp. 82-87, Jun 1994.

Jin, Y. & Sendhoff, B. (2003). Trade-Off between Performance and Robustness: An Evolutionary Multiobjective Approach, *Proceedings of Second Int. Conf. on Evol.*

Multi-Objective Optimization, pp. 237-251, ISBN 3540018697, Faro, Portugal, April 2003, Springer

Jin, Y. & Branke, J. (2005). Evolutionary Optimization in Uncertain Environments – A Survey, *IEEE Transactions on Evolutionary Computation*, Vol. 9, No. 3, (June 2005) 303-317, 1089-778X

Kazancioglu, E.; Wu, G.; Ko, J.; Bohac, S.; Filipi, Z.; Hu, S.; Assanis, D. & Saitou, K. (2003). Robust Optimization of an Automobile Valvetrain using a Multiobjective Genetic Algorithm, *Proceedings of DETC'03*, pp. 1-12, Chicago, USA, September 2003, ASME

Knowles, J. & Corne, D. (2000). Approximating the Non-dominated Front using the Pareto Archived Evolutionary Strategy, *Evolutionary Computation*, Vol. 8, No. 2, (June 2000) 149-172, 1063-6560

Moshaiov, A. & Avigrad, G. (2006). Concept-Based IEC for Multi-Objective Search with Robustness to Human Preference Uncertainty, *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 6785-6791, Vancouver, Canada, July 2006, IEEE

Olvander, J. (2005). Robustness Considerations in Multi-Objective Optimal Design, *J. of Engineering Design*, Vol. 16, No. 5 (October 2005) 511-523

Paenk I., Branke, J. & Jin, Y. (2006). Efficient Search for Robust Solutions by Means of Evolutionary Algorithms and Fitness Approximation, *IEEE Transations on Evolutionary Computation*, Vol. 10, No. 4 (August 2006) 405-420

Kouvelis, P. & Sayin, S. (2002). Algorithm Robust for the Bicriteria Discrete Optimization Problem, *Annals of Operational Research*, Vol. 147, No. 1, (October 2006) 71–85

Ray, T. (2002). Constrained Robust Optimal Design using a Multiobjective Evolutionary Algorithm, *Proceedings of the 2002 Congress on Evolutionary Computation*, 2002, pp. 419-424, ISBN 0-7803-7282-4, Honolulu, May 2002, IEEE

Ribeiro, J.L. & Elsayed, E.A. (1995). A case Study on Process Optimization using the Gradient Loss Funstion, *Int. J. Prod. Res.*, Vol. 33, No. 12, 3233-3248

Roseman, M. & Gero, J. (1985). Reducing the Pareto Optimal Set in Multicriteria Optimization, *Engineering Optimization*, Vol. 8, No. 3, 189-206, 0305-215X

Schaffer, J. (1984). *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms*, Vanderbilt University, Ph. D. Thesis, Nashville

Sörensen, K. (2004). Finding Robust Solutions Using Local Search, *J. of Mathematical Modelling and Algorithms*, Vol. 3, No. 1, 89-103

Srinivas, N. & Deb, K. (1995). Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, *Evolutionary Computation*, Vol. 2, No.3, 221-248

Tsutsui, S. & Ghosh, A. (1997). Genetic Algorithms with a Robust Solution Scheme, *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 3, (September 1997) 201-208, 1089-778X

Wiesmann, D.; Hammel, U. & Bäck, T. (1998). Robust Design of Multilayer Optical Coatings by Means of Evolutionary Algorithms, *IEEE Transactions on Evolutionary Computation*, Vol. 2, No. 4, (November 1998) 162-167, 4235.738986

Zitzler, E.; Deb K. & Thiele, L. (2000). Comparison of Multiobjective Evolutionary Algorithms: Empirical Results, *Evolutionary Computation*, Vol. 8, No. 2, (June 2000) 173-195, 1063-6560.

Zitzler, E.; Laumanns, M. & Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm, TIK report, No. 103, Swiss Federal Institute of Technology, Zurich, Switzerland.

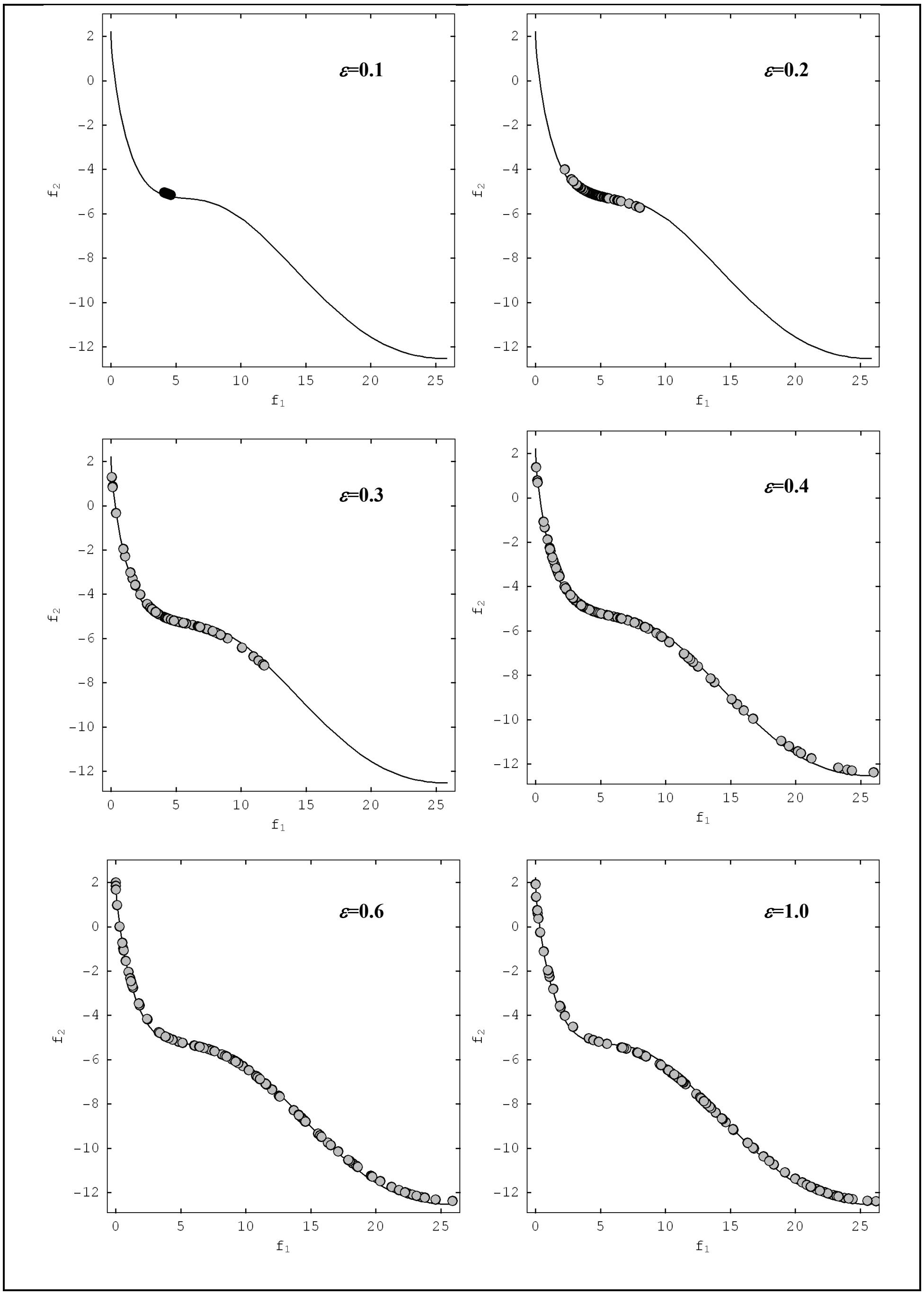


Fig. 11. Influence of dispersion parameter for TP1

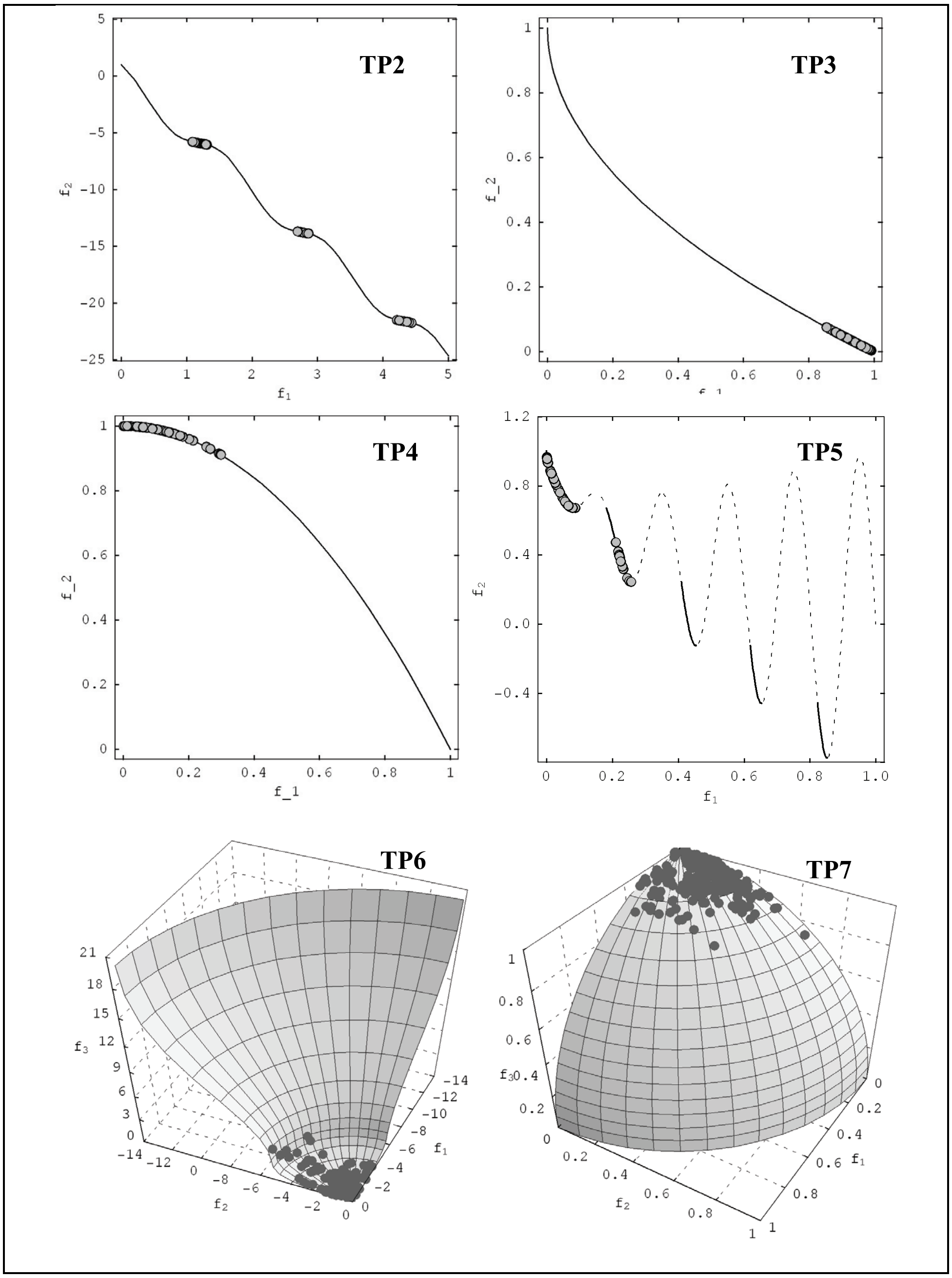


Fig. 12. Results for TP2 to TP7 (ε=0.1)

14

Improving Interpretability of Fuzzy Models Using Multi-Objective Neuro-Evolutionary Algorithms

Gracia Sánchez Carpena, José Francisco Sánchez Ruiz,
José Manuel Alcaraz Muñoz and Fernando Jiménez
University of Murcia
Spain

1. Introduction

Evolutionary Algorithms (EA) (Goldberg, 1989) have been successfully applied to learn fuzzy models (Ishibuchi et al., 1999). EAs have been also combined with other techniques like fuzzy clustering (Gómez-Skarmeta & Jiménez 1999) and neural networks (Russo, 1998). This has resulted in many complex algorithms and, as recognized in (Valente de Oliveira, 1999) and in (Setnes et al., 1998), often interpretability of the resulting rule base is not considered to be of importance. In such cases, the fuzzy model becomes a black-box, and one can question the rationale for applying fuzzy modeling instead of other techniques.
On the other hand, EAs have been recognized as appropriate techniques for multi-objective optimization because they perform a search for multiple solutions in parallel (Coello et al., 2002) (Deb, 2001). Current evolutionary approaches for multi-objective optimization consist of multi-objective EAs based on the Pareto optimality notion, in which all objective are simultaneously optimized to find multiple non-dominated solutions in a single run of the EA. The decision maker can then choose the most appropriate solution according to the current decision environment at the end of the EA run. Moreover, if the decision environment changes, it is not always necessary to run the EA again. Another solution may be chosen out of the set of non-dominated solutions that has already been obtained.
The multi-objective evolutionary approach can also be considered from the fuzzy modeling perspective (Ishibuchi et al., 1997). Current research lines in fuzzy modeling mostly tackle improving accuracy in descriptive models, and improving interpretability in approximative models (Casillas et al., 2003). This chapter deals with the second issue approaching the problem by means of multi-objective optimization in which accuracy and interpretability criteria are simultaneously considered.
In this chapter, we propose a multi-objective neuro-evolutionary optimization approach to generate TSK fuzzy models considering accuracy and interpretability criteria. This approach allows a linguistic approximation of the fuzzy models. The rule-based fuzzy model and criteria taken into account for fuzzy modeling are explained in the text, where a multi-objective constrained optimization model is proposed.
Two different multi-objective evolutionary algorithms (MONEA, ENORA-II) are proposed and compared with the well-known algotithm NSGA-II (Deb et al., 2000) for the

approximation of a non linear system (studied by Wang & Yen, 1998, 1999). The results of the experiments performed for this standard test problem show a real ability and effectiveness of the proposed approach to find accurate and interpretable TSK fuzzy models.

2. Improving interpretability in TSK fuzzy models

2.1 Fuzzy models identification

We consider Takagi-Sugeno-Kang (TSK) type rule-based models (Takagi & Sugeno, 1985) where rule consequents are taken to be linear functions of the inputs. The rules have, therefore, the following expression:

$$\begin{aligned} R_i: \ & \textit{If } x_1 \textit{ is } A_{i1} \textit{ and } \ldots \textit{ and } x_n \textit{ is } A_{in} \\ & \textit{then } y_i = \theta_{i1}x_1 + \ldots + \theta_{in}x_n + \theta_{i(n+1)} \end{aligned}$$

where:

$i = 1,\ldots,M$, M is the number of rules,

$\mathbf{x} = (x_1,\ldots,x_n)$, $x_j \in [l_j\ u_j] \subset \Re$ is the input vector ($j = 1,\ldots,n$),

$\theta_{ij} \in [l,\ u] \subset \Re$ are the consequent parameters ($j = 1,\ldots,n+1$),

y_i is the output of the ith rule, and

A_{ij} are fuzzy sets defined in the antecedent space by membership functions $\mu_{A_{ij}} : \mathsf{X}_j \to [0,1]$,

being X_j the domain of the input variable x_j ($j = 1,\ldots,n$).

The total output of the model is computed by aggregating the individual contributions of each rule:

$$y = \frac{\sum_{i=1}^{M} \mu_i(\mathbf{x}) f_i(\mathbf{x})}{\sum_{i=1}^{M} \mu_i(\mathbf{x})} \tag{1}$$

where $\mu_i(\mathbf{x})$ is the normalized firing strength of the ith rule:

$$\mu_i(\mathbf{x}) = \prod_{j=1}^{n} \mu_{A_{ij}}(x_j) \tag{2}$$

and $f_i(\mathbf{x})$ is the function defined in the consequent of the ith rule:

$$f_i(\mathbf{x}) = \theta_{i1}x_1 + \ldots + \theta_{in}x_n + \theta_{i(n+1)} \tag{3}$$

Each fuzzy set A_{ij} is described by a symmetric gaussian membership function:

$$\mu_{A_{ij}}(x_j) = \exp\left[-\frac{1}{2}\left(\frac{x_j - c_{ij}}{\sigma_{ij}}\right)^2\right] \tag{4}$$

where:

$c_{ij} \in \lfloor l_j, u_j \rfloor$ is the center,

$\sigma_{ij} > 0$ is the variance,

$i = 1,\ldots,M$ and

$j = 1,\ldots,n$.

This fuzzy model can be defined by a radial basis function neural network. The number of neurons in the hidden layer of an RBF neural network is equal to the number of rules in the fuzzy model. The firing strength of the *i*th neuron in the hidden layer matches the firing strength of the *i*th rule in the fuzzy model. We apply a symmetric gaussian membership function defined by two parameters, the center *c* and the variance σ. Therefore, each neuron in the hidden layer has these two parameters that define its firing strength value.
The neurons in the output layer perform the computations for the first order linear function described in the consequents of the fuzzy model, therefore, the *i*th neuron of the output layer has the parameters $\boldsymbol{\theta}_i = \left(\theta_{i1},\ldots,\theta_{i(n+1)}\right)$ that correspond to the linear function defined in the *i*th rule of the fuzzy model.

2.2 Criteria for fuzzy modeling

We consider three main criteria: (i) accuracy, (ii) transparency, and (iii) compactness. It is necessary to define quantitative measures for these criteria by means of appropriate objective functions which define the complete fuzzy model identification.
Accuracy.
The accuracy of a model can be measured with the *mean squared error*:

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(y_k - t_k)^2 \tag{5}$$

where:
y_k is the model output for the *k*th input vector,
t_k is the desired output for the *k*th input vector, and
N is the number of data samples.
Transparency.
For the second criterion, transparency, there are many possible measures, however we consider one of the most used, the *similarity* (Setnes, 1995). The similarity S among distinct fuzzy sets in each variable can be expressed as follows:

$$S = \max_{\substack{i=1,\ldots,M \\ j=1,\ldots,n \\ k=1,\ldots,M \\ A_{ij} \neq A_{kj}}} S\left(A_{ij}, A_{kj}\right) \tag{6}$$

Similarity between two different fuzzy sets A and B can be measured using different criteria. In our case we use the following measure:

$$S(A,B) = \max\left\{\frac{|A \cap B|}{|A|}, \frac{|A \cap B|}{|B|}\right\} \tag{7}$$

The value of S is, therefore, an aggregated similarity measure for the fuzzy rule-based model with the objective to minimize the maximum similarity between the fuzzy sets in each input domain.

Compactness.

Finally, measures for the third criterion, the compactness, are the number of rules, (M) and the number of different fuzzy sets (L) of the fuzzy model. It is assumed that models with a small number of rules and fuzzy sets are compact.

Table 1 summarizes the three criteria considered for the fuzzy models and the measures defined for each criterion.

Criteria	Measures
Accuracy	MSE
Transparency	S
Compactness	M, L

Table 1. Criteria for the fuzzy models and their measures

2.3 An optimization model for fuzzy modeling

According to the previous remarks, we propose the following multi-objective constrained optimization model:

$$\begin{array}{ll} \textit{Minimize} & f_1 = MSE \\ \textit{Minimize} & f_2 = M \\ \textit{Subject to} & g_1 : S - g_s \leq 0 \end{array} \tag{8}$$

where $g_s \in [0, 1]$ is a threshold for similarity defined by the decision maker (we use $g_s = 0{,}25$). An "a posteriori" articulation of preferences applied to the non-dominated solutions of the problem is used to obtain the final compromise solution.

3. Multi-objective neuro-evolutionary algorithms

We propose a hybrid learning system to find multiple Pareto-optimal solutions simultaneously, considering accuracy, transparency and compactness criteria. We study different multi-objective evolutionary algorithms to evolve the structure and parameters of TSK-type rule sets, together with gradient-based learning to train rule consequents. Additionally, a rule set simplification operator is used to encourage rule base transparency and compactness. This method may be applied to a wide variety of classification and control problems.

Considering the multi-objective constrained optimization model (8), we use three Pareto-based multi-objective evolutionary algorithms: MONEA, ENORA-II and NSGA-II. MONEA and ENORA-II are algorithms proposed by authors in (Gómez-Skarmeta et al., 2007), and (Sánchez et al., 2007) respectively, while NSGA-II is the well-known multi-objective EA proposed by Deb in (Deb, 2001).

The main common characteristics are the following:

- The algorithms are Pareto-based multi-objective EAs for fuzzy modeling; that is, they have been designed to find, in a single run, multiple non-dominated solutions according to the Pareto decision strategy. There is no dependence between the objective

functions and the design of the EAs; thus, any objective function can easily be incorporated.
- Constraints with respect to the fuzzy model structure are satisfied by incorporating specific knowledge about the problem. The initialization procedure and variation operators always generate individuals that satisfy these constraints.
- The EAs have a variable-length, real-coded representation. Each individual of a population contains a variable number of rules between *1* and *max*, where *max* is defined by a decision maker. Fuzzy numbers in the antecedents and the parameters in the consequent are coded by floating-point numbers.
- The initial population is generated randomly with a uniform distribution within the boundaries of the search space, defined by the learning data and model constraints.
- The EAs search among rule sets treated with the technique described in Section 3.6 and trained as defined in Section 3.3, which is an added ad hoc technique for transparency, compactness, and accuracy.

Table 2 summarizes common and specific characteristics of the algorithms MONEA, NSGA-II and ENORA-II.

Common characteristics	
Pittsburgh approach, real-coded representation.	
Training of the RBF network consequents.	
Constraint-handling technique.	
Variation operators.	
Rule-set simplification technique.	
Elitist generational replacement strategy.	
Specific characteristics	
MONEA:	Preselection over 10 children, steady-state replacement (n = 2).
ENORA-II:	Non-dominated radial slots sorting.
NSGA-II:	Non-dominated crowded sorting.

Table 2. Common and specific characteristics of MONEA, ENORA-II and NSGA-II.

3.1 Representation of solutions

The EAs have a variable-length, real-coded representation using a Pittsburgh approach. An individual I for this problem is a rule set of M (between 1 and *max*, where *max* is defined by a decision maker) rules defined by the weights of the RBF neural network. With n input variables, we have for each individual the following parameters:

- Parameters of the fuzzy sets A_{ij}:

$$\text{centers } c_{ij} \text{ and variances } \sigma_{ij}\,,\ i=1,\dots,M\,,\ j=1,\dots,n$$

- Coefficients for the linear function of the consequents:

$$\theta_{ij}\,,\ i=1,\dots,M\,,\ j=1,\dots,n+1$$

3.2 Initial population

The population is initialized by generating individuals with different numbers of rules. Each individual is generated randomly with a uniform distribution within the boundaries of the search space, defined by the learning data and trained with the gradient technique described in subsection 3.3.

An individual with M rules is generated with the following procedure:

1. For each fuzzy set A_{ij} ($i=1,\ldots,M$, $j=1,\ldots,n$), generate two real values: c_{ij} in the interval $\lfloor l_j, u_j \rfloor$ and the parameter of the gaussian fuzzy set , σ_{ij} .
2. Parameters θ_{ij} ($i=1,\ldots,M$, $j=1,\ldots,n+1$) are random real values in the interval $[l,u]$.
3. The individual is treated with the technique to improve transparency and compactness describe in subsection 3.6.
4. The individual is trained using the gradient technique described in subsection 3.3.

3.3 Training of the RBF neural networks

In RBF neural networks, each neuron in the hidden layer can be associated with a fuzzy rule; therefore RBF neural networks are suitable to describe fuzzy models. The RBF neural networks associated with the fuzzy models can be trained with a gradient method to obtain more accuracy. However, in order to maintain the transparency and compactness of the fuzzy sets, only the consequent parameters are trained. The training algorithm incrementally updates the parameters based on the currently presented training pattern. The network parameters are updated by applying the gradient descent method to the *MSE* error function. The error function for the *i*th training pattern is given by the *MSE* function error defined in equation (5). The updating rule is the following:

$$\theta_{ij} \leftarrow \theta_{ij} + \eta \Delta \theta_{ij} = \theta_{ij} - \eta \frac{\partial MSE}{\partial \theta_{ij}}$$

where:

$i=1,\ldots,M$,

$j=1,\ldots,n+1$, and

η is the learning rate.

This rule is applied during a number of iterations (epochs). We use a value $\eta = 0.01$ and a number of 10 epochs. The negative gradients of MSE with respect to each parameter are calculated in the following way:

$$\Delta \theta_{ij} = -\frac{\partial MSE}{\partial \theta_{ij}} = (t_k - y_k)\frac{1}{z}\mu_i(\mathrm{x})x_j, j = 1,\ldots,n$$

$$\Delta \theta_{i(n+1)} = -\frac{\partial MSE}{\partial \theta_{i(n+1)}} = (t_k - y_k)\frac{1}{z}\mu_i(\mathbf{x})$$

where:

$i=1,\ldots,M$,

$\mu_i(\mathbf{x})$ is the firing strength for the *i*th rule defined in equation (2), and

$z = \sum_{i=1}^{M} \mu_i(\mathbf{x})$.

3.4 Constraint-handling

The EAs use the following constraint handling rule proposed in (Jiménez et al., 2002). This rule considers that an individual *I* is better than an individual *J* if any of the following conditions is true:

- *I* is feasible and *J* is not
- *I* and *J* are both unfeasible, but $S_I < S_J$
  (S_I and S_J are similarity of *I* and *J*)
- *I* and *J* are feasible and *I* dominates *J*

3.5 Variation operators

As already said, an individual is a set of M rules. A rule is a collection of n fuzzy numbers (antecedent) plus $n+1$ real parameters (consequent), and a fuzzy number is composed of two real numbers. In order to achieve an appropriate exploitation and exploration of the potential solutions in the search space, variation operators working in the different levels of the individuals are necessary. In this way, we consider three levels of variation operators: rule set level, rule level and parameter level.

Rule Set Level Variation Operators

Rule Set Crossover

This operator exchanges a random number of rules. Given two parents $I_1 = \left(R_1^1 \ldots R_{M_1}^1\right)$and $I_2 = \left(R_1^2 \ldots R_{M_2}^2\right)$generate two children:

$$I_3 = \left(R_1^1 \ldots R_a^1 R_1^2 \ldots R_b^2\right)$$

$$I_4 = \left(R_{a+1}^1 \ldots R_{M_1}^1 R_{b+1}^2 \ldots R_{M_2}^2\right)$$

with:

$$a = round(\alpha M_1)$$

$$b = round((1-\alpha)M_2)$$

where α is a random real number in $[0,1]$. The number of rules of the children is therefore in $[M_1, M_2]$.

Rule Set Increase Crossover

This operator increases the number of each child rules adding a random number of rules of the other parent. Given two parents $I_1 = \left(R_1^1 \ldots R_{M_1}^1\right)$ and $I_2 = \left(R_1^2 \ldots R_{M_2}^2\right)$ generate two children:

$$I_3 = \left(R_1^1 \ldots R_{M_1}^1 R_1^2 \ldots R_a^2\right)$$

$$I_4 = \left(R_1^2 \dots R_{M_2}^2 R_1^1 \dots R_b^1\right)$$

with:

$$a = \min\{\max - M_1, M_2\}$$

$$b = \min\{\max - M_2, M_1\}$$

<u>Rule Set Mutation</u>
This operator adds or deletes, with the same probability, a rule. Given an individual $I = (R_1 \dots R_M)$ generates other individual I':

$$\begin{array}{ll} I' = (R_1, \dots R_{a-1} R_{a+1} \dots R_M), & \textit{if } \alpha \le 0.5 \\ I' = (R_1, \dots R_M R_{M+1}), & \textit{in other case} \end{array}$$

where:
α is a random real number in $[0,1]$,
a a random index in $[1, M]$, and
R_{M+1} a new random rule generated with the initialization procedure.

Rule Level Variation Operators
<u>Rule Arithmetic Crossover</u>
It performs an arithmetic crossover of two random rules. Given two parents $I_1 = \left(R_1^1 \dots R_{M_1}^1\right)$ and $I_2 = \left(R_1^2 \dots R_{M_2}^2\right)$ generates two children:

$$I_3 = \left(R_1^1 \dots R_i^3 \dots R_{M_1}^1\right)$$

$$I_4 = \left(R_1^2 \dots R_j^4 \dots R_{M_2}^2\right)$$

with R_i^3 and R_j^4 obtained by arithmetic crossover:

$$R_i^3 = \alpha R_i^1 + (1-\alpha) R_j^2$$

$$R_j^4 = \alpha R_j^2 + (1-\alpha) R_i^1$$

where:
α is a random real number in $[0,1]$,
i, j are random index in $[1, M_1]$ and $[1, M_2]$, respectively.
The product αR_i is defined as follows:

$$\alpha R_i: \quad \alpha A_{i1} \dots \alpha A_{in} \quad \alpha\theta_{i1} \dots \alpha\theta_{in} \; \alpha\theta_{i(n+1)}$$

The fuzzy set αA_{ij} is defined as follows:

$$\alpha A_{ij} = \{\alpha a_{ij}, \alpha b_{ij}, \alpha c_{ij}, \alpha d_{ij}\}$$

Rule Uniform Crossover

It performs a uniform crossover of two random rules. Given two parents $I_1 = \left(R_1^1 \ldots R_{M_1}^1\right)$ and $I_2 = \left(R_1^2 \ldots R_{M_2}^2\right)$ generates two children:

$$I_3 = \left(R_1^1 \ldots R_i^3 \ldots R_{M_1}^1\right)$$

$$I_4 = \left(R_1^2 \ldots R_j^4 \ldots R_{M_2}^2\right)$$

where:

R_i^3 and R_j^4 are obtained by uniform crossover,

i, j are random index in $[1, M_1]$ and $[1, M_2]$.

Parameter Level Variation Operators

The operators considered at this level are arithmetic crossover, uniform crossover, non-uniform mutation, uniform mutation and small mutation. These operators excluding the last one have been studied and described by other authors (Goldberg, 1989). The small mutation produces a small change in the individual and it is suitable for fine tuning of the real parameters.

3.6 Rule set simplification technique

Automated approaches to fuzzy modeling often introduce redundancy in terms of several similar fuzzy sets and fuzzy rules that describe almost the same region in the domain of some variable. According to some similarity measure, two similar fuzzy sets can be merged or separated. The merging-separation process is repeated until fuzzy sets for each model variable are not similar. This simplification may results in several identical rules, which must be removed from the rule set. The proposed algorithm is the following:

```
1 While there be $i, j, k$ such that $S(A_{ij}, A_{kj}) > \eta_2$
      If $S(A_{ij}, A_{kj}) > \eta_1$ then
            Calculate $C$ as the merging of $A_{ij}$ and $A_{kj}$
            Substitute $A_{ij}$ and $A_{kj}$ by $C$
      in other case
            Split $A_{ij}$ and $A_{kj}$
2 While there be $i, k$ such that the antecedents of rules $R_i$ and $R_k$ are the same
      Calculate a new consequent with the average of the parameters of the consequents
      of $R_i$ and $R_k$
      Substitute the consequent of $R_i$ by the new consequent
      Eliminate $R_k$
```

Similarity between two fuzzy sets, $S(A, B)$, is measured using the expression in equation (7). The values η_1 and η_2 are the threshold to perform the merging or the separation and must be $0 < \eta_2 < \eta_1 < 1$. (we use $\eta_1 = 0.9$ and $\eta_2 = 0.6$)

If $S(A,B) > \eta_1$, fuzzy sets A and B are merged in a new fuzzy set C as follows:

$$c_C = \alpha c_A + (1-\alpha)c_B$$

$$\sigma = \max\{c_C - \min\{c_A - \sigma_A, c_B - \sigma_B\}, \max\{c_A + \sigma_A, c_B + \sigma_B\} - c_C\}$$

where $\alpha \in [0,1]$ determines the influence of A and B in the new fuzzy set C:

$$\alpha = \frac{c_A^r - c_A^l}{c_A^r - c_A^l + c_B^r - c_B^l}$$

If $\eta_2 < S(A,B) < \eta_1$, fuzzy sets A and B are splitted as follows:

$$\begin{array}{rl} \textit{If } \sigma_A < \sigma_B \textit{ then} & \sigma_A \leftarrow \sigma_A(1-\beta) \\ \textit{in other case} & \sigma_B \leftarrow \sigma_B(1-\beta) \end{array}$$

where $\beta \in [0,1]$ indicates the amount of separation between A and B (we use $\beta = 0.1$).

3.7 Algorithm descriptions

In order to describe the algorithms, we consider the following formulation as a general form of the multi-objective constrained optimization model (8):

$$\begin{array}{rl} \textit{Minimize} \quad f_k & k = 1,..,n \\ \textit{Subject to } g_i \le 0 & i = 1,..,m \end{array} \tag{9}$$

Where f_k, g_i are arbitrary functions.

Multi-objective neuro-evolutionary algorithm (MONEA)

The main characteristic of MONEA is that Chromosome selection and replacement are achieved by means of a variant of the Preselection scheme. This technique is, implicitly, a niche formation technique and an elitist strategy. Moreover, an explicit niche formation technique has been added to maintain diversity with respect to the number of rules of the individuals.

```
Algorithm MONEA
1.   t ← 0
2.   Initialize P (t)
4.   while t < T do
5.       parent$_1$,parent$_2$ ← Random selection from P(t)
6.       Generate a new individual best$_1$ ← parent$_1$
7.       Generate a new individual best$_2$ ← parent$_2$
8.       Repeat *nChildren* times
9.           child$_1$,child$_2$ ← Crossing and Mutation of parent$_1$ and parent$_2$
10.          Improve transparency and compactness in child$_1$ and child$_2$
11.          Train child$_1$and child$_2$ by the gradient technique
12.          For i=1 to 2
```

```
13.                 If child_i is better than best_i and
                    (the number of rules of child_i is equal to the number of rules of parent_i) or
                    (the niche count of parent_i is greater than minNS and the niche count of the
                    child_i is smaller than maxNS) then
14.                     best_i ← child_i
15.         P (t + 1) ← P(t) – {parent_1, parent_2} ∪ {best_1, best_2}
16.         t ← t + 1
17.     end while
```

The preselection scheme is an implicit niche formation technique to maintain diversity in the population because an offspring replaces an individual similar to itself (one of its parents). Implicit niche formation techniques are more appropriate for fuzzy modeling than explicit techniques, such as the sharing function, which can provoke excessive computational time. However, we need an additional mechanism for diversity with respect to the number of rules of the individuals in the population. The added explicit niche formation technique ensures that the number of individuals with *M* rules, for all *M* Є *[1, max]*, is greater or equal to *minNS* and smaller or equal to *maxNS*. Moreover, the preselection scheme is also an elitist strategy because the best individual in the population is replaced only by a better one.

The better function

Given two individuals *k* and *l*, *k* is better than *l* if:

- *k* is feasible and *l* is unfeasible, or
- *k* and *l* are unfeasible and $\max_{j=1\ldots m}\left\{g_j^k\right\} \le \max_{j=1\ldots m}\left\{g_j^l\right\}$, or (10)
- *k* and *l* are feasible and *k* dominates *l*, or

ENORA-II: An Elitist Pareto-Based Multi-Objective Evolutionary Algorithm

ENORA-II uses a real-coded representation, uniform and arithmetical cross, and uniform and non-uniform mutation. Diversity among individuals is maintained by using an ad-hoc elitist generational replacement technique.

ENORA-II has a population *P* of *N* individuals. The following algorithm shows the pseudocode of ENORA-II.

Algorithm ENORA-II

```
1.  t ← 0
2.  Initialize P (t)
3.  Evaluate P (t)
4.  while t < T do
5.      Q (t) ← Random Selection, Crossing and Mutation of N individuals from P (t)
6.      Improve transparency and compactness in Q(t)
7.      Train all individuals in Q(t) by the gradient technique
8.      Evaluate Q(t);
9.      P (t + 1) ← Best individuals from P (t) ∪ Q(t);
10.     t ← t + 1;
11.     end while;
12. return the non dominated individuals from P(t);
```

Given a population *P* of *N* individuals, *N* children are generated by random selection, crossing and mutation. The new population is obtained selecting the *N* best individuals from the union of parents and children.

Better individuals
The better individuals are obtained by using the ranking established by the operator *best*. It assumes that every individual *i* has two attributes:

- a ranking in its slot (r_i), and
- a crowding distance (d_i).

Based on these attributes, an individual *i* is better than an individual *j* if:

- $r_i < r_j$ or
- $r_i = r_j$ and $d_i > d_j$.

Crowding distance
Quantity d_i is a measure of the search space around individual *i* which is not occupied by any other individual in the population. This quantity d_i serves as an estimate of the perimeter of the cuboid formed by using the nearest neighbors as the vertices.

$$d_i = \begin{cases} \infty & , \ \textit{if } f_j^i = f_j^{\max} \textit{ or } f_j^i = f_j^{\min} \textit{ for any } j \\ \displaystyle\sum_{j=1}^{n} \frac{f_j^{\sup_j^i} - f_j^{\inf_j^i}}{f_j^{\max} - f_j^{\min}}, & \textit{in other case} \end{cases} \tag{11}$$

Where $f_j^{\max} = \max\limits_{i=1\ldots N}\{f_j^i\}$, $f_j^{\min} = \min\limits_{i=1\ldots N}\{f_j^i\}$, $f_j^{\sup_j^i}$ is the value of the *j*th objective for the individual higher adjacent in the *j*th objective to individual *i*, and $f_j^{\inf_j^i}$ is the value of the *j*th objective for the individual lower adjacent in the *j*th objective to individual i.
Ranking of individuals in its slot
Individuals are ordered in $\left(\left\lfloor \sqrt[n-1]{N} \right\rfloor + 1\right)^{n-1}$ slots. An individual *i* belongs to slot s_i such that:

$$s_i = \sum_{j=2}^{n} \left\lceil \frac{f_j^i - f_j^{r\min}}{f_j^{r\max} - f_j^{r\min}} \left\lfloor \sqrt[n-1]{N} \right\rfloor \right\rceil \left(\left\lfloor \sqrt[n-1]{N} \right\rfloor\right)^{n-j} \tag{12}$$

where $f_j^{r\max}$ and $f_j^{r\min}$ are the maximum and minimum values for the *j*th objective if the objective space is bounded; if it is not, then these are bounding reference points so that $f_j^{r\max} \geq f_j^i$ and $f_j^{r\min} \leq f_j^i$ for any individual *i*.
The ranking inside slots is established as an adjustment of the better function (10): given two individuals *k* and *l* belonging to same slot, ranking of individual *k* is lower than ranking of individual *l* in the slot if:

- *k* is feasible and *l* is unfeasible, or
- *k* and *l* are unfeasible and $\max\limits_{j=1\ldots m}\{g_j^k\} \leq \max\limits_{j=1\ldots m}\{g_j^l\}$, or
- *k* and *l* are feasible and *k* dominates *l*, or
- *k* and *l* are feasible and does not dominated each other and $d_k > d_l$.

4. Experiments and results

We consider the second order non-linear plant studied in (Wang & Yen, 1999) and (Yen & Wang, 1998):

$$\begin{aligned}
&y(k) = g(y(k-1), y(k-2)) + u(k) \\
&with \\
&g(y(k-1), y(k-2)) = \frac{y(k-1)y(k-2)(y(k-1)-0.5)}{1 + y^2(k-1) + y^2(k-2)}
\end{aligned}$$

The objective is the approximation of the non-linear component of the plant $g(y(k-1), y(k-2))$ using a fuzzy model. 200 training values and 200 evaluation values are obtained starting at the initial state (0,0) with a random input signal $u(k)$ uniformly distributed in the interval $[-1.5, 1.5]$.
MONEA, ENORA-II and NSGA-II are executed 100 times for 10000 evaluations, with a population of 100 individuals, cross and mutation probabilities of 0.8 and 0.4 respectively. The different variation operators are applied with equal probability. We can compare our results with the results obtained by other approaches proposed in (Wang & Yen, 1999), (Yen & Wang, 1998) and (Roubos & Setnes, 2000) which are shown in Table 3. Table 4 shows the best non-dominated solutions in the last population over 100 runs. Solutions with 4 rules are chosen which are shown in Figure 1 and Table 5.

Reference	M	L	Train *MSE*	Eval *MSE*
Wang & Yen, 1999	40 (initial)	40	3.3 E-4	6.9 E-4
	28 (optimized)	28	3.3 E-4	6.0 E-4
Yen & Wang, 1998	36 (initial)	12	1.9 E-6	2.9 E-3
	24 (optimized)	12	2.0 E-6	6.4 E-4
Roubos & Setnes, 2000	7 (initial)	14	1.8 E-3	1.0 E-3
	5 (optimized)	5	5.0 E-4	4.2 E-4

Table 3. Fuzzy models for the second order non-linear plant reported in literature.

M	L	Train *MSE*	Eval *MSE*	*S*
		MONEA		
1	2	0.041882	0.043821	0.000000
2	3	0.004779	0.005533	0.249887
3	4	0.002262	0.002749	0.232016
4	4	0.000216	0.000248	0.249021
		ENORA-II		
1	2	0.041882	0.043821	0.000000
2	3	0.004951	0.005722	0.242090
3	4	0.001906	0.002411	0.249391
4	4	0.000161	0.000194	0.249746
		NSGA-II		
1	2	0.041882	0.043821	0.000000
2	3	0.004870	0.005639	0.249998
3	4	0.001885	0.002343	0.249999
4	4	0.000249	0.000314	0.250000

Table 4. Non-dominated solutions (best results over 100 runs) obtained in this paper for the second order non-linear plant.

*R*1	If *y(k-1)* is LOW and *y(k-2)* is LOW	then *g* = 0.4327*y(k-1)* + 0.0007*(k-2)* – 0.2008
*R*2	If *y(k-1)* is LOW and *y(k-2)* is HIGH	then *g* = -0.4545*y(k-1)* – 0.0131*(k-2)* + 0.2368
*R*3	If *y(k-1)* is HIGH and *y(k-2)* is LOW	then *g* = -0.3968*y(k-1)* – 0.0044*(k-2)*+ 0.1859
*R*4	If *y(k-1)* is HIGH and *y(k-2)* is HIGH	then *g* = 0.43645*y(k-1)* – 0.0052*(k-2)*– 0.2110
y(k-1)	*LOW* = (-1.5966, 2.0662)	*HIGH* = (1,7679, 2.6992)
y(k-2)	*LOW* = (-1.7940, 3.1816)	*HIGH* = (1.5271, 2.1492)

Table 5. Fuzzy model with 4 rules for the non-linear dynamic plant obtained by ENORA-II.

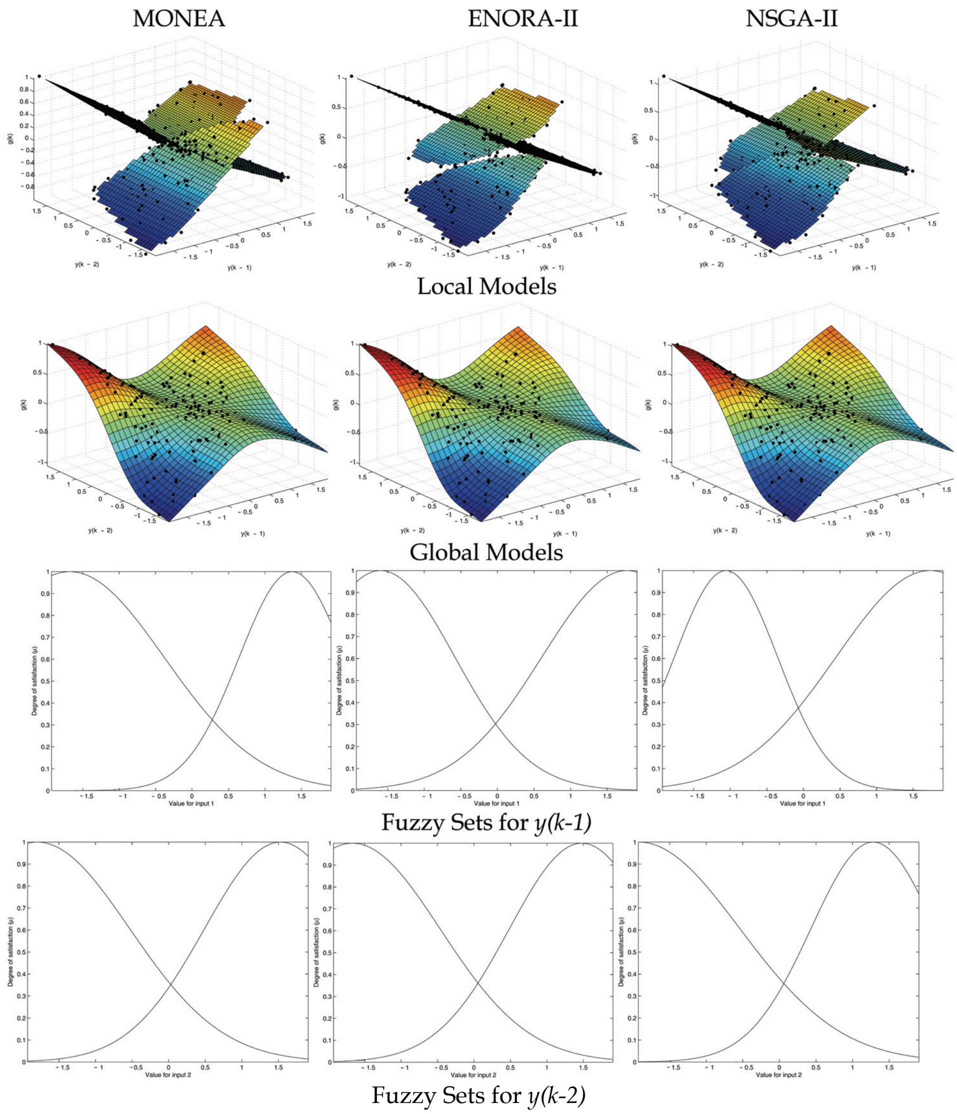


Figure 1. Solutions with 4 rules obtained in this paper for the second order non-linear plant.

To compare the algorithms, we use the hypervolume indicator (ν) which calculates the fraction of the objective space which is non-dominated by any of the solutions obtained by the algorithm in (Deb, 2001), (Laumans et al., 2001) and (Zitzler et al., 2003). The aim is to minimize the value of ν. This indicator estimates both the distance of solutions to the real Pareto front and the spread. Whenever a set of solutions is preferable to other with respect to weak Pareto dominance, the indicator value for the first set of solution will be at least as good as the indicator value for the second; it is, therefore, a Pareto compliant quality indicator. Value ν can be calculated for a population P_0 which is composed by the N_0 non-dominated solutions of P.

Algorithms were executed 100 times, so we have obtained a 100 sample for each algorithm. The statistics showed in Table 6 indicate that MONEA and ENORA-II obtain lower localization values than NSGA-II while NSGA-II obtains the greatest dispersion values. Finally, the 90% confidence intervals for the mean obtained with t-test show that ENORA-II obtains lower values than MONEA and this obtains lower than NSGA-II. That is, the approximation sets obtained by ENORA-II are preferable to those of MONEA and those of NSGA-II under hypervolume indicator ν. t-test is robust with no normal samples which are greater than 30 individuals, so the results are significant and we can conclude that there is statistical difference between the hypervolume values obtained by the algorithms. The Boxplots showed in Figure 2 confirm the above conclusions.

	MONEA	ENORA-II	NSGA-II
Minimum	0.3444	0.3337	0.3318
Maximum	0.4944	0.4591	0.9590
Mean	0.3919	0.3799	0.5333
S.D	0.0378	0.0334	0.1430
C.I. Low	0.3856	0.3743	0.5096
C.I. High	0.3982	0.3854	0.5571

S.D = Standard Deviation of Mean

C.I. = Confidence Interval for the Mean (90%)

Table 6. Statistics for the hypervolume obtained with 100 runs of MONEA, ENORA-II and NSGA-II for the second order non-linear plant.

Taking all the above, we can conclude that the hypervolume values obtained with ENORA-II are significantly better than the values obtained with MONEA and NSGA-II. The statistical analysis shows, therefore, that for the kind of multi-objective problems we are considering, Pareto search based on the space search partition in linear slots is most efficient than general search strategies exclusively based on diversity functions, as in NSGA-II.

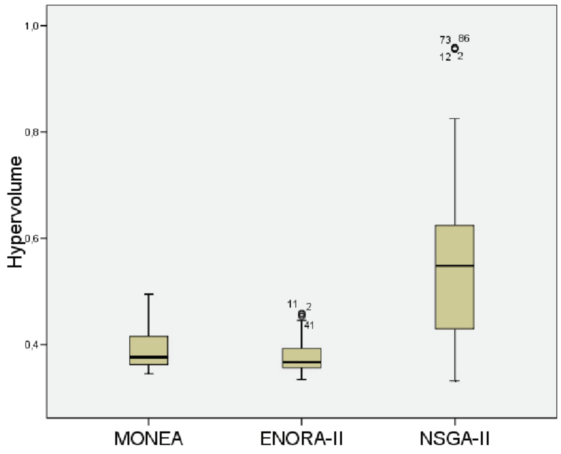


Figure 2. Boxplots for the hypervolume obtained with 100 runs of MONEA, ENORA-II and NSGA-II for the second order non-linear plant.

5. Conclusions

This chapter remarks on some results in the combination of Pareto-based multi-objective evolutionary algorithms, neural networks and fuzzy modeling. A multi-objective constrained optimization model is proposed in which criteria such as accuracy, transparency and compactness have been taken into account. Three multi-objective evolutionary algorithms (MONEA, ENORA-II and NSGA-II) have been implemented in combination with neural network based and rule simplification techniques. The results obtained improve on other more complex techniques reported in literature, with the advantage that the proposed technique identifies a set of alternative solutions. Statistical tests have been performed over the hypervolume quality indicator to compare the algorithms and it has shown that, for the non linear plant problem, ENORA-II obtains better results than MONEA and NSGA-II algorithms.
Future improvements of the algorithms will be the automatic parameter tuning, and a next application of these techniques will be on medicine data.

6. Acknowledgements

Research supported in part by MEC and FEDER under project PET2006 0406 and TIN 2006-15460-C04-01.

7. References

Casillas, J., Cordón, O., Herrera, F., Magdalena, L. (2003) Interpretability improvements to find the balance interpretability-accuracy in fuzzy modeling: an overview, in J. Casillas, O. Cordón, F. Herrera, L. Magdalena (Eds.), *Interpretability Issues in Fuzzy Modeling, Studies in Fuzziness and Soft Computing*, Springer, pp. 3-22.

Coello, C.A., Veldhuizen, D.V., Lamont, G.B. (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic/Plenum publishers, New York.

Deb, K., Agrawal, S., Pratap, A., Meyarivan, T. (2000). *A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II*. In Proceedings of the Parallel Problem Solving from Nature VI (PPSN-VI), pp. 849-858.

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons, LTD.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Gómez-Skarmeta, A.F., Jiménez, F. (1999). Fuzzy modeling with hibrid systems. *Fuzzy Sets and Systems*, 104:199-208.

Gómez-Skarmeta, A.F., Jiménez, F., Sánchez, G. (2007). Improving Interpretability in Approximative Fuzzy Models via Multiobjective Evolutionary Algorithms. *International Journal of Intelligent Systems*, 22:943-969, 2.007

Ishibuchi, H., Murata, T., Trksen, I. (1997). Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. *Fuzzy Sets and Systems*, 89:135-150

Ishibuchi, H., Nakashima, T., Murata, T. (1999). Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. *IEEE Transactions on Systems, Man, and Cubernetics - Part B: Cybernetics*, 29(5):601-618.

Jiménez, F., Gómez-Skarmeta, A.F., Sánchez, G., Deb, K. (2002). An evolutionary algorithm for constrained multi-objective optimization, in: *Proceedings IEEE World Congress on Evolutionary Computation*.

Laumanns, M., Zitzler, E., and Thiele, L. (2001). On the Effects of Archiving, Elitism, and Density Based Selection in Evolutionary Multi-objective Optimization. *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization* (EMO 2001) E. Zitzler et al. (Eds.), 181-196.

Roubos, J.A., Setnes, M. (2000). Compact fuzzy models through complexity reduction and evolutionary optimization. In *Proceedings of FUZZ-IEEE-2000*, 762-767, San Antonio, Texas, USA.

Russo., M. (1998). FuGeNeSys - a fuzzy genetic neural system for fuzzy modeling. *IEEE Transactions on Fuzzy Systems*, 6(3):373-388.

Sánchez, G., Jiménez, J., Vasant, P. (2007). Fuzzy Optimization with Multi-Objective Evolutionary Algorithms: a Case Study. *IEEE Symposium of Computational Intelligence in Multicriteria Decision Making (MCDM)*. Honolulu, Hawaii

Setnes, M. (1995). *Fuzzy Rule Base Simplification Using Similarity Measures*. M.Sc. thesis, Delft University of Technology, Delft, the Netherlands.

Setnes, M., Babuska, R., Verbruggen, H.B. (1998). Rule-based modeling: Precision and transparency. *IEEE Transactions on Systems, Man and Cybernetics, Part C*: Applications & Reviews, 28:165-169.

Takagi, T., Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15:116-132.

Valente de Oliveira, J. (1999). Semantic constraints for membership function optimization. *IEEE Transactions on Fuzzy Systems*, 19(1):128-138.

Wang, L., Yen, J. (1999). Extracting fuzzy rules for system modeling using a hybrid of genetic algorithms and Kalman filter. *Fuzzy Sets and Systems*, 101:353-362.

Yen, J., Wang, L. (1998). Application of statistical information criteria for optimal fuzzy model construction. *IEEE Transactions on Fuzzy Systems*, 6(3):362-371.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V. (2003) Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117-132.

Multi-objective Uniform-diversity Genetic Algorithm (MUGA)

Ali Jamali[1], Nader Nariman-zadeh[1,2] and Kazem Atashkari[1]
[1]*Dept. of Mechanical Engineering, Faculty of Engineering, University of Guilan, Rasht*
[2]*Intelligent-based Experimental Mechanics Center of Excellence, School of Mechanical Engineering, Faculty of Engineering, University of Tehran, Tehran*
Iran

1. Introduction

Optimization in engineering design has always been of great importance and interest particularly in solving complex real-world design problems. Basically, the optimization process is defined as finding a set of values for a vector of design variables so that it leads to an optimum value of an objective or cost function. In such single-objective optimization problems, there may or may not exist some constraint functions on the design variables and they are respectively referred to as constrained or unconstrained optimization problems. There are many calculus-based methods including gradient approaches to search for mostly local optimum solutions and these are well documented in (Arora, 1989; Rao, 1996). However, some basic difficulties in the gradient methods such as their strong dependence on the initial guess can cause them to find a local optimum rather than a global one. This has led to other heuristic optimization methods, particularly Genetic Algorithms (GAs) being used extensively during the last decade. Such nature-inspired evolutionary algorithms (Goldberg, 1989; Back et al., 1997) differ from other traditional calculus based techniques. The main difference is that GAs work with a population of candidate solutions, not a single point in search space. This helps significantly to avoid being trapped in local optima (Renner & Ekart, 2003) as long as the diversity of the population is well preserved.
In multi-objective optimization problems, there are several objective or cost functions (a vector of objectives) to be optimized (minimized or maximized) simultaneously. These objectives often conflict with each other so that as one objective function improves, another deteriorates. Therefore, there is no single optimal solution that is best with respect to all the objective functions. Instead, there is a set of optimal solutions, well known as Pareto optimal solutions (Srinivas & Deb, 1994; Fonseca & Fleming, 1993; Coello Coello & Christiansen, 2000; Coello Coello & Van Veldhuizen, 2002), which distinguishes significantly the inherent natures between single-objective and multi-objective optimization problems. V. Pareto (1848-1923) was the French-Italian economist who first developed the concept of multi-objective optimization in economics (Pareto, 1896). The concept of a Pareto front in the space of objective functions in multi-objective optimization problems (MOPs) stands for a set of solutions that are non-dominated to each other but are superior to the rest of solutions in the search space. Evidently, changing the vector of design variables in such a Pareto optimal

solutions consisting of these non-dominated solutions would not lead to the improvement of all objectives simultaneously. Consequently, such change leads to a deterioration of at least one objective to an inferior one. Thus, each solution of the Pareto set includes at least one objective inferior to that of another solution in that Pareto set, although both are superior to others in the rest of search space.

The inherent parallelism in evolutionary algorithms makes them suitably eligible for solving MOPs. The early use of evolutionary search is first reported in 1960s by Rosenberg (Rosenberg, 1967). Since then, there has been a growing interest in devising different evolutionary algorithms for MOPs. Basically, most of them are Pareto-based approaches and use the well-known non-dominated sorting procedure. In such Pareto-based approaches, the values of objective functions are used to distinguish the non-dominated solutions in the current population. Among these methods, the Vector Evaluated Genetic Algorithm (VEGA) proposed by Schaffer (Schaffer, 1985), Fonseca and Fleming's Genetic Algorithm (MOGA) (Fonseca & Fleming, 1993), Non-dominated Sorting Genetic Algorithm (NSGA) by Srinivas and Deb (Srinivas & Deb, 1994), and Strength Pareto Evolutionary Algorithm (SPEA) by Zitzler and Thiele (Zitzler & Thiele, 1998), and the Pareto Archived Evolution Strategy (PAES) by Knowles and Corne (Knowles & Corne, 1999) are the most important ones. A very good and comprehensive survey of these methods has been presented in (Coello Coello, 1999; Deb, 2001; Khare et al., 2003). Coello (Coello Coello, home page) has also presented an internet based collection of many papers as a very good and easily accessible literature resource. Basically, both NSGA and MOGA as Pareto-based approaches use the revolutionary non-dominated sorting procedure originally proposed by Goldberg (Goldberg, 1989).

There are two important issues that have to be considered in such evolutionary multi-objective optimization methods: driving the search towards the true Pareto-optimal set or front and preventing premature convergence or maintaining the genetic diversity within the population (Toffolo & Benini, 2003). The lack of elitism was also a motivation for modification of that algorithm to NSGA-II (Deb et al., 2002) in which a direct elitist mechanism, instead of a sharing mechanism, has been introduced to enhance the population diversity. This modified algorithm represents the state-of-the-art in evolutionary MOPs (Coello Coello & Becerra, 2003). A comparison study among SPEA and other evolutionary algorithms on several problems and test functions showed that SPEA clearly outperforms the other multi-objective EAs (Zitzler et al., 2000). Some further investigations reported in reference (Toffolo & Benini, 2003) demonstrated, however, that the elitist variant of NSGA (NSGA-II) equals the performance of SPEA. Despite its popularity and effectiveness, NSGA-II is modified in this work to enhance its diversity preserving mechanism especially for problems with more than two objective functions.

In this chapter, a new simple algorithm in conjunction with the original Pareto ranking of non-dominated optimal solutions is proposed and tested for MOPs including some test functions and engineering problems in power and energy conversion. In the Multi-objective Uniform-diversity Genetic Algorithm (MUGA), a ϵ-elimination diversity approach is used such that all the clones and/or ϵ-similar individuals based on normalized Euclidean norm of two vectors are recognized and simply eliminated from the current population. The superiority of MUGA is shown in comparison with NSGA-II in terms of diversity of population and Pareto fronts both for bi-objective and multi-objective optimization problems.

2. Multi-objective optimization

Multi-objective optimization which is also called multicriteria optimization or vector optimization has been defined as finding a vector of decision variables satisfying constraints to give optimal values to all objective functions (Coello Coello & Christiansen, 2000; Homaifar et al., 1994) . In general, it can be mathematically defined as:

find the vector $X^* = \left[x_1^*, x_2^*, \ldots, x_n^*\right]^T$ to optimize:

$$F(X) = \left[f_1(X), f_2(X), \ldots, f_k(X)\right]^T \tag{1}$$

subject to *m* inequality constraints:

$$g_i(X) \le 0 \quad , \quad \mathrm{i=1\ to\ m} \tag{2}$$

and *p* equality constraints:

$$h_j(X) = 0 \quad , \quad \mathrm{j=1\ to\ p} \tag{3}$$

where $X^* \in \Re^n$ is the vector of decision or design variables, and $F(X) \in \Re^k$ is the vector of objective functions. Without loss of generality, it is assumed that all objective functions are to be minimized. Such multi-objective minimization based on the Pareto approach can be conducted using some definitions:

Definition of Pareto dominance

A vector $U = \left[u_1, u_2, \ldots, u_k\right] \in \Re^k$ dominates to vector $V = \left[v_1, v_2, \ldots, v_k\right] \in \Re^k$ (denoted by $U \prec V$) if and only if $\forall i \in \{1,2,\ldots,k\},\ u_i \le v_i \ \wedge\ \exists j \in \{1,2,\ldots,k\} : u_j < v_j$. It means that there is at least one u_j which is smaller than v_j whilst the rest u's are either smaller or equal to corresponding v's.

Definition of Pareto optimality

A point $X^* \in \Omega$ (Ω is a feasible region in $\Re^n$ satisfying equations (2) and (3)) is said to be Pareto optimal (minimal) with respect to all $X \in \Omega$ if and only if $F(X^*) \prec F(X)$. Alternatively, it can be readily restated as $\forall i \in \{1,2,\ldots,k\}$, $\forall X \in \Omega - \{X^*\}$ $f_i(X^*) \le f_i(X)$ $\wedge$ $\exists j \in \{1,2,\ldots,k\} : f_j(X^*) < f_j(X)$. It means that the solution X^* is said to be Pareto optimal (minimal) if no other solution can be found to dominate X^* using the definition of Pareto dominance.

Definition of Pareto Set

For a given MOP, a Pareto set $\mathcal{P}^*$ is a set in the decision variable space consisting of all the Pareto optimal vectors, $\mathcal{P}^* = \{X \in \Omega \mid \nexists\, X' \in \Omega : F(X') \prec F(X)\}$. In other words, there is no other X' in Ω that dominates any $X \in \mathcal{P}^*$.

Definition of Pareto front

For a given MOP, the Pareto front $\mathcal{PF}^*$ is a set of vectors of objective functions which are obtained using the vectors of decision variables in the Pareto set $\mathcal{P}^*$, that is,

$\mathcal{PF}^* = \{F(X) = (f_1(X), f_2(X), \ldots, f_k(X)) : X \in \mathcal{P}^*\}$. Therefore, the Pareto front $\mathcal{PF}^*$ is a set of the vectors of objective functions mapped from $\mathcal{P}^*$.

Evolutionary algorithms have been widely used for multi-objective optimization because of their natural properties suited for these types of problems. This is mostly because of their parallel or population-based search approach. Therefore, most difficulties and deficiencies within the classical methods in solving multi-objective optimization problems are eliminated. For example, there is no need for either several runs to find the Pareto front or quantification of the importance of each objective using numerical weights. It is very important in evolutionary algorithms that the genetic diversity within the population be preserved sufficiently. This main issue in MOPs has been addressed by much related research work (Toffolo & Benini, 2003). Consequently, the premature convergence of MOEAs is prevented and the solutions are directed and distributed along the true Pareto front if such genetic diversity is well provided. The Pareto-based approach of NSGA-II (Deb et al., 2002) has been recently used in a wide range of engineering MOPs because of its simple yet efficient non-dominance ranking procedure in yielding different levels of Pareto frontiers. However, the crowding approach in such a state-of-the-art MOEA (Coello Coello & Becerra, 2003) works efficiently for two-objective optimization problems as a diversity-preserving operator which is not the case for problems with more than two objective functions. The reason is that the sorting procedure of individuals based on each objective in this algorithm will cause different enclosing hyper-boxes. Thus, the overall crowding distance of an individual computed in this way may not exactly reflect the true measure of diversity or crowding property. In order to show this issue more clearly, some basics of NSGA-II are now represented. The entire population R_t is simply the current parent population P_t plus its offspring population Q_t which is created from the parent population P_t by using usual genetic operators. The selection is based on non-dominated sorting procedure which is used to classify the entire population R_t according to increasing order of dominance (Deb et al., 2002). Thereafter, the best Pareto fronts from the top of the sorted list is transferred to create the new parent population P_{t+1} which is half the size of the entire population R_t. Therefore, it should be noted that all the individuals of a certain front cannot be accommodated in the new parent population because of space. In order to choose exact number of individuals of that particular front, a crowded comparison operator is used in NSGA-II to find the best solutions to fill the rest of the new parent population slots. The crowded comparison procedure is based on density estimation of solutions surrounding a particular solution in a population or front. In this way, the solutions of a Pareto front are first sorted in each objective direction in the ascending order of that objective value. The crowding distance is then assigned equal to the half of the perimeter of the enclosing hyper-box (a rectangular in bi-objective optimization problems). The sorting procedure is then repeated for other objectives and the overall crowding distance is calculated as the sum of the crowding distances from all objectives. The less crowded non-dominated individuals of that particular Pareto front are then selected to fill the new parent population. It must be noted that, in a <u>two-objective</u> Pareto optimization, if the solutions of a Pareto front are sorted in a <u>decreasing</u> order of importance to one objective, these solutions are then automatically ordered in an <u>increasing</u> order of importance to the second objective. Thus, the hyper-boxes surrounding an individual solution remain unchanged in the objective-wise sorting procedure of the crowding distance of NSGA-II in the two-objective Pareto optimization problem. However, in multi-objective Pareto optimization problems with more than two objectives, such sorting procedure of individuals based on each objective in this algorithm will cause different enclosing hyper-boxes. Thus, the overall crowding distance of an individual computed in this way may not exactly reflect the true measure of diversity or crowding property for the multi-objective Pareto optimization problems with more than two objectives.

In our work, a new method is presented to modify NSGA-II so that it can be safely used for any number of objective functions (particularly for more than two objectives) in MOPs. Such a modified MOEA is then used for four-objective thermodynamic optimization of subsonic turbojet engines.

3. Multi-objective Uniform-diversity Genetic Algorithm (MUGA)

The multi-objective uniform-diversity genetic algorithm (MUGA) uses non-dominated sorting mechanism together with a ε-elimination diversity preserving algorithm to get Pareto optimal solutions of MOPs more precisely and uniformly.

3.1 The non-dominated sorting method

The basic idea of sorting of non-dominated solutions originally proposed by Goldberg (Goldberg, 1989) used in different evolutionary multi-objective optimization algorithms such as in NSGA-II by Deb (Deb et al., 2002) has been adopted here. The algorithm simply compares each individual in the population with others to determine its non-dominancy. Once the first front has been found, all its non-dominated individuals are removed from the main population and the procedure is repeated for the subsequent fronts until the entire population is sorted and non-dominately divided into different fronts.

A sorting procedure to constitute a front could be simply accomplished by comparing all the individuals of the population and including the non-dominated individuals in the front. Such procedure can be simply represented as following steps:

```
1-Get the population (pop)
2-Include the first individual {ind(1)} in the front P* as P*(1), let P*_size=1;
3-Compare other individuals {ind (j), j=2, Pop_size)} of the pop with { P*(K), K=1,  P*_size} of the P*;
If ind(j)<P*(K) replace the P*(K) with ind(j)
If P*(K)<ind(K), j=j+1, continue comparison;
Else include ind(j) in P*, P*_size= P*_size+1, j=j+1, continue comparison;
4-End of front P*;
```

It can be easily seen that the number of non-dominated solutions in *P** grows until no further one is found. At this stage, all the non-dominated individuals so far found in *P** are removed from the main population and the whole procedure of finding another front may be accomplished again. This procedure is repeated until the whole population is divided into different ranked fronts. It should be noted that the first rank front of the final generation constitute the final Pareto optimal solution of the multi-objective optimization problem.

3.2 The ε-elimination diversity preserving approach

In the ε-elimination diversity approach that is used to replaced the crowding distance assignment approach in NSGA-II (Deb et al., 2002), all the clones and ε-similar individuals are recognized and simply eliminated from the current population. Therefore, based on a value of ε as the elimination threshold, all the individuals in a front within this limit of a particular individual are eliminated. It should be noted that such ε-similarity must exist both in the space of objectives and in the space of the associated design variables. This will ensure that very different individuals in the space of design variables having ε-similarity in the space of objectives will not be eliminated from the population. The pseudo-code of the ε-elimination approach is depicted in Fig. 1. Evidently, the clones and ε-similar individuals are replaced from the population by the same number of new randomly generated individuals. Meanwhile, this will additionally help to explore the search space of the given MOP more effectively. It is clear that such replacement does not appear when a front rather than the entire population is truncated for ε-similar individual.

```
ε-elim= ε-elimination(pop)                    // pop includes design variables and
                                              objective function
i=1; j=1;
get K (K=1 for the first front);
While i,j <pop_size

 e(i,j)= ‖ X(i,:),X(j,:) ‖ / ‖ X(i,:) ‖ ; X(i),X(j) ϵ $P^*_k$ Ụ $PF^*_k$  //finding mean value of ε

                                                         within pop.
end
ε=mean(e);
i=1;
until i+1<pop_size;
j=i+1
        until j<pop_size
        if e(i,j)<ε
        then {pop}={pop}/ {pop(j)}          //remove the ε-similar individual
        j=j+1
        end
i=i+1
end
```

Fig. 1. The ε-elimination diversity preserving pseudo-code

3.3 The main algorithm of MUGA

It is now possible to present the main algorithm of MUGA which uses both non-dominated sorting procedure and ε-elimination diversity preserving approach which is given in Fig.2. It first initiates a population randomly. Using genetic operators, another same size population is then created. Based on the ε-elimination algorithm, the whole population is then reduced by removing ε-similar individuals. At this stage, the population is re-filled by randomly generated individuals which helps to explore the search space more effectively. The whole population is then sorted using non-dominated sorting procedure. The obtained fronts are then used to constitute the main population. It must be noted that the front which must be truncated to match the size of the population is also evaluated by ε-elimination procedure to identify the ε-similar individuals. Such procedure is only performed to match the size of population within ±10 percent deviation to prevent excessive computational effort to population size adjustment. Finally, unless the number of individuals in the first rank front is changing in certain number of generations, randomly created individuals are inserted in the main population occasionally (e.g. every 20 generations of having non-varying first rank front).

```
Get N                          //population size
t=1 ;                          //set generation number
Random_N($P_t$);               //generate the first population ($P_1$) randomly
$Q_t$=Recomb($P_t$)            //generate population $Q_t$ from $P_t$ by genetic operators
$R_t=P_t \cup Q_t$             //union of both parent and offspring population
$R_t'$=ε-elimination ($R_t$)   //remove ε-similar individuals in $R_t$
$R_t''= R_t' \cup$ Random_($R_t$_size-$R'_t$_size) ($P_t'$)    //add random individuals to fill $R_t$ to 2N

Do non-dominate sorting procedure ($R_t''$)   //$R_t''=P^*_1 \cup P^*_2 \cup \ldots \cup P^*_k$  where k is total
                                                                                 number of fronts
i=1
$P_{t+1}=\Theta$
While not $P_{t+1}$_size>N                //includes fronts into new population
                    $P_{t+1}= P_{t+1} \cup P^*_i$
                    i=i+1
end
N'=N- $P_{t+1}$_size
While not (0.9 N'< $P_{t+1}$_size<1.1 N')        //remove the ε-similar individuals within
                                                  the tolerance of ±10 percent
          F'=ε-elimination ($P^*_{i-1}$)
      If  F'_size< N'
          e=1.1*e
          else
          e=0.9 * e            //adjust the value of threshold to get the right population
                               size of the last front
          end
end
t=t+1                               //Start next generation
```

Fig. 2. The pseudo-code of the main algorithm of MUGA

4. Numerical results of MUGA using test functions

In this section four test functions which have been widely used in literature (Deb et al., 2002) are adopted here to test and compare the effectiveness of MUGA with that of NSGA-II. These test functions are all bi-objective and have no constraint. A generation number of 250 with a population size of 100 have been used in all experiments. The probabilities of crossover and mutation have been chosen as 0.9 and 0.1, respectively. Each test function has been run for 5 times to compute the mean and variance of the metric of non-uniformity of the solutions obtained in the final Pareto front.

In order to evaluate the diversity of the obtained Pareto front, a metric, Δ, has been adopted here to measure the spread and uniformity of the achieved non-dominated solutions along a Pareto front (Deb et al., 2002). Such metric basically calculates the relative Euclidean distance of consecutive solutions from their average value. Hence, a lower value of Δ (zero in ideal case) indicates a better uniformly spread non-dominated solutions. It is therefore possible to simply compare the performance of MUGA with that of NSGA-II in term of uniformity using the same metric.

Four different functions which have been used to test and compare the results of MUGA with those of NSGA-II are as follows:

$$1\text{-}\begin{cases} f_1(x)=x^2 \\ f_2(x)=(x-2)^2 \end{cases} \quad x\in[-1000,1000]$$

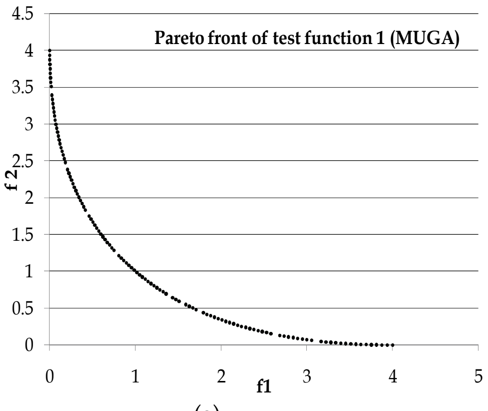
$$2\text{-}\begin{cases} f_1(x)=x_1 \\ f_2(x)=g(x)\left[1-\sqrt{x_i/g(x)}\right] \\ g(x)=1+9\left(\sum_{i=2}^{n} x_i\right)\Big/(n-1) \end{cases} \quad x\in[0,1], n=30$$

$$3\text{-}\begin{cases} f_1(x)=x_1 \\ f_2(x)=g(x)\left[1-\left(x_i/g(x)\right)^2\right] \\ g(x)=1+9\left(\sum_{i=2}^{n} x_i\right)\Big/(n-1) \end{cases} \quad x\in[0,1], n=30$$

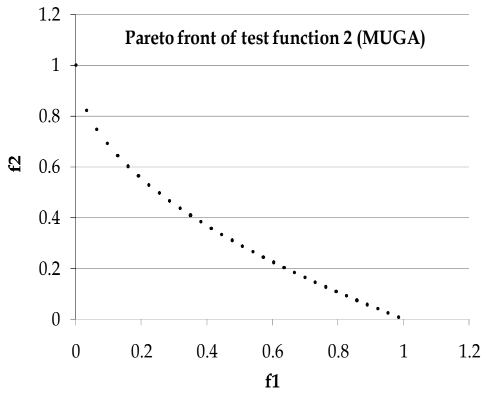
$$4\text{-}\begin{cases} f_1(x)=\sum_{i=1}^{n-1}\left(-10\exp\left(-0.2\sqrt{x_i^2+x_{i+1}^2}\right)\right) \\ f_2(x)=\sum_{i=1}^{n}\left(\left|x_i\right|^{0.8}+5\sin x_i^3\right) \end{cases} \quad x\in[-5,5], n=3$$

Figure 3 depicts the Pareto fronts obtained for test functions 1 and 2 using MUGA. Figure 4 depicts the same for test functions 3 and 4. The uniformity of the well spread-out of the non-dominated solutions is evident from these figures.

In order to compare the uniformity of the results of this work (MUGA) with those of NSGA-II, Table 1 shows the means and variances of metric Δ of both methods for multiple runs (Deb et al., 2002).

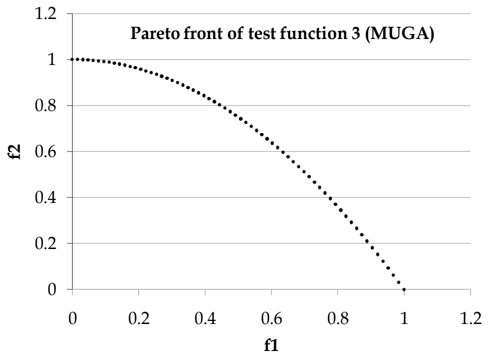


(a)

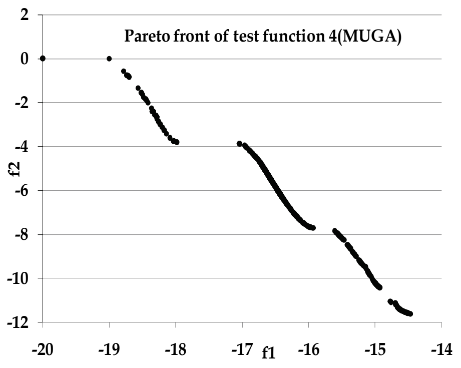


(b)

Fig. 3. Pareto fronts obtained by MUGA: (a) Test function 1 (b):Test function 2



(a)



(b)

Fig. 4. Pareto fronts obtained by MUGA: (a) Test function 3 (b):Test function 4

Methods	Test function 1	Test function 2	Test function 3	Test function 4
NSGA-II	0.449265	0.463292	0.435112	0.442195
	0.002062	0.041622	0.024607	0.001498
SPEA	1.021110	0.784525	0.755148	0.852490
	0.004372	0.004440	0.004521	0.002619
PAES	1.063288	1.229794	1.165942	1.079838
	0.002868	0.004839	0.007682	0.013772
MUGA (this work)	0.162595	0.273347	0.225211	0.402798
	2.9E-06	0.000261	2.1E-07	0.0

Table 1. Comparison of mean and variance of metric Δ of different methods (Deb et al., 2002) with those of MUGA (shaded rows are mean values and un-shaded rows are variances)

It is very evident from Table 1 that the performance of MUGA is better than that of other methods in achieving lower Δ in obtaining more uniform non-dominated solutions for these test functions. Further, the very small value of variances of that metric obtained in multiple runs simply demonstrates the robustness of finding uniform Pareto fronts in MOPs using MUGA.

5. Multi-objective thermodynamic optimization of turbojet engines with two design variables

Turbojet engines use air as the working fluid and produce thrust based on the variation of kinetic energy of burnt gases after combustion. The study of the thermodynamic cycle of a turbojet engine involves different thermo-mechanical aspects such as specific thrust, thermal and propulsive efficiencies, and thrust-specific fuel consumption (Atashkari, et al., 2005). A detailed description of the thermodynamic analysis and equations (Mattingly, 1996) of ideal turbojet engines is given in Appendix A (Atashkari, et al., 2005). This elementary thermodynamic model is sufficient to capture the principles of behaviour and interactions among different input and output parameters in a multi-objective optimal sense. Furthermore, this provides a suitable real-world engineering benchmark for comparing purpose between MOEA using the new diversity preserving mechanism of this work.
The input parameters involved in such thermodynamic analysis in an ideal turbojet engine given in Appendix A are flight Mach number (M_0), input air temperature (T_0, K), specific heat ratio (γ), heating value of fuel (h_{pr}, kJ/kg), exit burner total temperature (T_{t4}, K), and pressure ratio, π_c. The output parameters involved in the thermodynamic analysis in the ideal turbojet engine given in Appendix A are, specific thrust, (*ST*, N/kg/s), fuel-to-air ratio (*f*), thrust-specific fuel consumption (*TSFC*, kg/s/N), thermal efficiency (η_t), and propulsive efficiency (η_p). However, in the multi-objective optimization study, some input parameters are already known or assumed as, T_0 = 216.6 K, γ =1.4, h_{pr} =48000 kJ/kg, and T_{t4} = 1666 K. The input flight Mach number $0 < M_0 \leq 1$ and the compressor pressure ratio $1 \leq \pi_c \leq 40$ are considered as design variables to be optimally found based on multi-objective optimization of 4 output parameters, namely, *ST*, *TSFC*, η_t, and η_p.

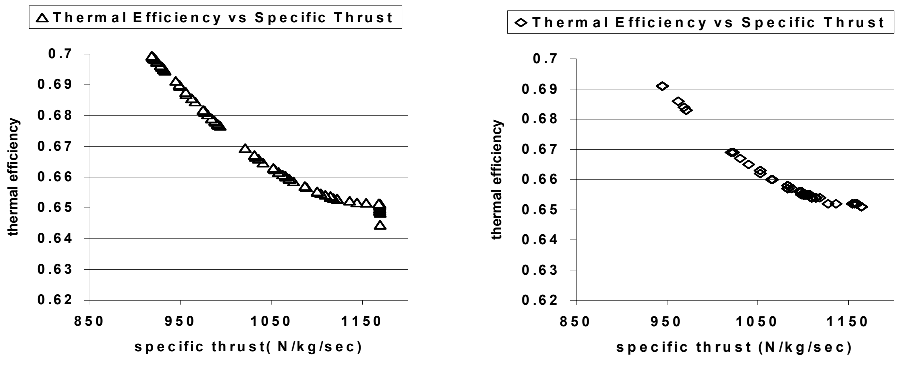
5.2 Two-objective thermodynamic optimization of turbojet engines

In order to investigate the optimal thermodynamic behaviour of subsonic turbojet engines, 5 different sets, each including two objectives of the output parameters, are considered individually. Such pairs of objectives to be optimized separately have been chosen as (η_p, $TSFCx10^5$), (η_p, *ST*), (η_t, $TSFCx10^5$), (η_t, *ST*), and (η_p, η_t). Evidently, it can be observed that η_p, η_t, and *ST* are maximized whilst *TSFC* is minimized in those sets of objective functions.
A population size of 100 has been chosen in all runs with crossover probability P_c and mutation probability P_m as 0.8 and 0.1, respectively.
The results of the two-objective optimizations considering those 5 different combinations of objectives are summarized in Table 2. Some Pareto fronts of each pair of two objectives have been shown through figures (5-6) using both the approach of this work and that of NSGA-II.

	Pairs of objectives in two-objective optimizations						
	(ηp, *TSFC*)		(ηt, *TSFC*)	(ηp, *ST*)		(ηt, *ST*)	(η_p, η_t)
	$0 < \eta p \leq 0.39$	$0.4 < \eta p \leq 0.55$	$.65 \leq \eta t \leq 0.7$	$0.41 < \eta p \leq 0.5$	$0 < \eta p\ 0.39$	$0.64 \leq \eta t\ 0.7$	$0.4 \leq \eta p \leq 0.56$
	$2.1 \leq TSFC \leq 2.43$	$3.16 \leq TSFC \leq 6.8$	$2.1 \leq TSFC \leq 2.43$	$515 \leq ST \leq 817$	$906 \leq ST \leq 1169$	$890 \leq ST \leq 1169$	$0.16 \leq \eta t \leq 0.55$
M_0	$0 < M_o \leq 1$	1	$0 < M_o \leq 1$	$0.85 \leq M_o \leq 1$	$0 < M_o \leq 1$	$0 < M_o \leq 1$	1
π_c	$\pi c = 40$	$1.0 \leq \pi c \leq 8.25$	$\pi c = 40$	$1.2 \leq \pi c \leq 4.28$	$13.5 \leq \pi c \leq 39.3$	$37.3 \leq \pi c \leq 40$	$1 \leq \pi c \leq 8.78$

Table 2. Values of decision variables and objective functions in various two-objective optimizations (Atashkari, et al., 2005)

It is clear from these figures that choosing appropriate values for the decision variables, namely flight Mach number (M_0) and pressure ratio (π_c), to obtain a better value of one objective would normally cause a worse value of another objective. However, if the set of decision variables is selected based on each of a Pareto front, it will lead to the best possible combination of that pair of objectives. In other words, if any other pair of decision variables M_0 and π_c is chosen, the corresponding values of the particular pair of objectives will locate a point inferior to that Pareto front. The inferior area in the space of objective functions (plane in these cases) for figures (5-6) are in fact bottom/left sides. A better diversity of results obtained using the approach of this work than those of NSGA-II can also observed in these figures. Evidently, figures (5-6) reveal some important and interesting optimal relationships among the thermodynamic parameters in the ideal thermodynamic cycle of turbojet engines that may not have been found without a multi-objective optimization approach. Such relationships have been called a worthwhile task for a designer by Deb in (Deb, 2003). These figures and the associated values of the decision variables and the objective functions given in Table 1 simply covers all the 4 objectives studied in the two-objective Pareto optimization.



(a) (b)

Fig. 5. Pareto front of thermal efficiency and specific thrust in 2-objective optimization: (a) MUGA (b) NSGA-II

However, other pairs of objective functions in the two-objective Pareto optimization together with their associated values of the decision variables have been shown in Table 1. A careful investigation of these Pareto optimization results reveals some interesting and informative design aspects. It can be observed that a small value of pressure ratio ($\pi_c < 8.7$) is required in large value of Mach number ($0.85 < M_0 < 1$) when high value of η_p is important to the designer ($0.4 < \eta_p < 0.55$). In this case both *ST* and *TSFC* get their worse values (*ST* becomes smaller and *TSFC* becomes larger), whilst η_t varies between small and medium values ($0.16 < \eta_t < 0.55$) depending on the value of flight Mach number. However, with high value of pressure ratio ($37 < \pi_c < 40$) in a wide range of flight Mach number ($0 < M_0 < 1$), *TSFC*, *ST*, and η_t improve whilst η_p cannot be better than 0.4. The specific values of these objectives depend on the exact value of flight Mach number which have been given in Table 1. However, such important and worthwhile information can be simply discovered using a four-objective Pareto optimization, which will be presented in the next section.

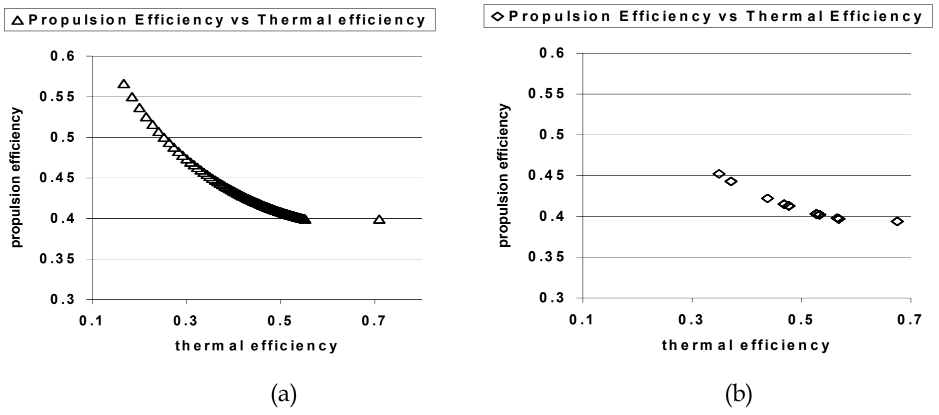


Fig. 6. Pareto front of propulsive efficiency and thermal efficiency in 2-objective optimization: (a) MUGA (b) NSGA-II

Moreover, figures (5-6) also reveal some important and interesting optimal relationships of such objective functions in ideal thermodynamic cycle of turbojet engines that may have not been known without a multi-objective optimization approach. For example, figure (3) demonstrates that the optimal behaviours of η_t with respect to *ST* can be readily represented by

$$\eta_t \propto (ST)^2 \tag{4}$$

Figure (4) represents a non-linear optimal relationship of η_t and η_p in the form of

$$\eta_t \propto (\eta_p)^2 \tag{5}$$

It should be noted that these relationships, which have been obtained from the two-objective Pareto optimization results, are valid when the corresponding two-objective optimization of such functions is of importance to the designer and, in fact, demonstrates the optimal compromise of such pairs of objectives.

5.3 Four-objective thermodynamic optimization of turbojet engines

A multi-objective thermodynamic optimization including all four objectives simultaneously can offer more choices for a designer. Moreover, such 4-objective optimization can subsume all the 2-objective optimization results presented in the previous section. Therefore, in this section, four objectives, namely, *TSFC, ST*, η_p, and η_t, are chosen for multi-objective optimization in which *ST*, η_p, and η_t are maximized whilst *TSFC* is minimized simultaneously. A population size of 200 has been chosen with crossover probability P_c and mutation probability P_m as 0.8 and 0.02, respectively.

Figure (7) depicts the non-dominated individuals in both 4-objective and previously obtained 2-objective optimization in the plane of (η_t-*ST*). Such non-dominated individuals in both 4 and 2-objective optimization have alternatively been shown in the plane of (η_p-η_t) in figure (8). It should be noted that there is a single set of individuals as a result of 4-objective optimization of *TSFC, ST*, η_p, and η_t that are shown in different planes together with the corresponding 2-objective optimization results. Therefore, there are some points in each plane that may dominate others in the same plane in the case of 4-objective optimization. However, these individuals are all non-dominated when considering all four objectives simultaneously. By careful investigation of the results of 4-objective optimization in each plane, the Pareto fronts of the corresponding two-objective optimization can now be observed in these figures. It can be readily observed that the results of such 4-objective optimization include the Pareto fronts of each 2-objective optimization and provide, therefore, more optimal choices for the designer.

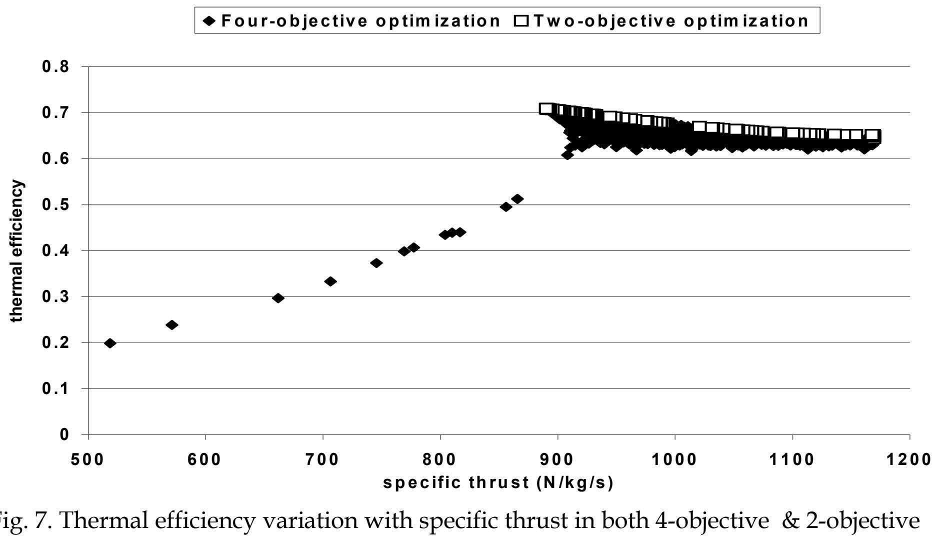


Fig. 7. Thermal efficiency variation with specific thrust in both 4-objective & 2-objective optimisation

The non-dominated individuals obtained in 4-objective optimization demonstrate some interesting behaviours in terms of design variables. Two different parts can be easily observed in figures (7-8). One of these parts which is less populated corresponds to high value of pressure ratio ($0.4<\eta_p<0.55$), unlike the rest of objective functions which all together degrades in their values simultaneously, that is, $3<TSFC\text{x}10^5<6.3$, $515<ST<890$, $0.2<\eta_t<0.52$. The corresponding values of objectives for the second part can be given as, $0<\eta_p<0.4$, $2<TSFC\text{x}10^5<3$, $900<ST<1169$, $0.6<\eta_t<0.71$ which can be appropriately chosen by the designer. Such facts would be very important to the designer to switch from one optimal solution to another for achieving different trade-off requirements of the objectives (Deb, 2003).

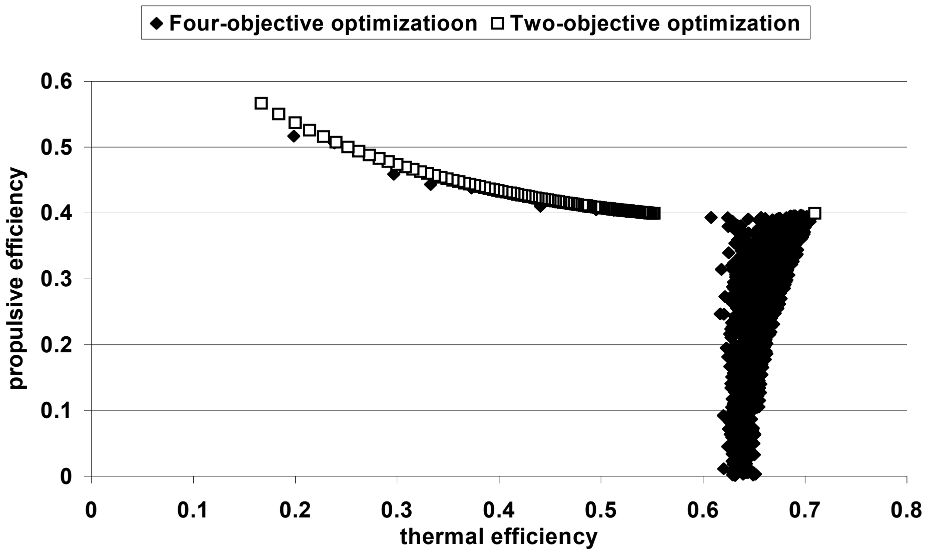


Fig. 8. Propulsive efficiency variation with thermal efficiency in both 4-objective & 2-objective optimization

Additionally, there are some more profound optimal design relationships among the objective functions and the decision variables which have been discovered by the four-objective thermodynamic Pareto optimization of ideal turbojet engines. Such important optimal design facts could not have been found without the multi-objective Pareto optimization. Firstly, figure (9) shows the variation of 4 optimized objective functions *ST*, *TSFC*, η_p, and η_t with the pressure ratio. It can be seen that for pressure ratio less than 14, three objectives *ST*, *TSFC*, and η_t become worse, unlike η_p which gradually starts getting better. The slope of such degradation for *ST*, *TSFC*, and η_t becomes faster especially in *TSFC* and η_t when the pressure ratio becomes smaller than 6. However, for high pressure ratios, the variation of optimal values of *TSFC* and η_t are small whilst there are a wide range of selections for $\eta_p \approx 0.4$. Secondly, figure (10) demonstrates the behaviours of *ST* and η_p with respect to flight Mach number in high pressure ratios. It can be readily seen that the optimal values of *ST* changes linearly with M_0, that is

$$ST = -264.75\, M_0 + 1164.5 \tag{7}$$

with a R-squared value of 0.999. The optimal relationship of η_p with M_0 is non-linear and is represented as

$$\eta_p = -0.0977\,(M_0)^2 + 0.491\, M_0 + 0.0013 \tag{8}$$

with a R-squared value of 0.998.

Therefore, such multi-objective optimization of *ST*, *TSFC*, η_p, and η_t provide optimal choices of design variables based on Pareto non-dominated points.

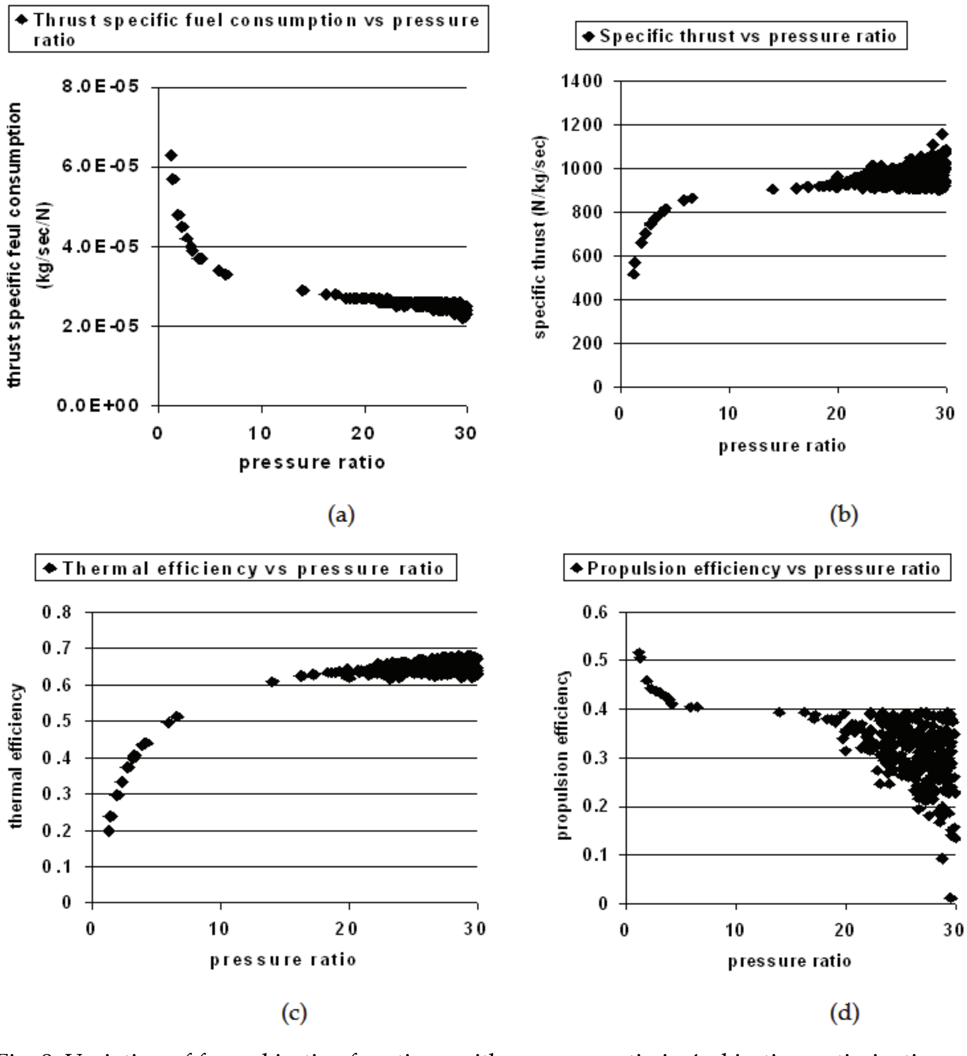


Fig. 9. Variation of four objective functions with pressure ratio in 4-objective optimization

6. Conclusion

A new multi-objective uniform-diversity genetic algorithm (MUGA) has been proposed and successfully used for some test functions and for thermodynamic cycle optimization of ideal turbojet engines. It has been shown that the performance of this algorithm is superior to that

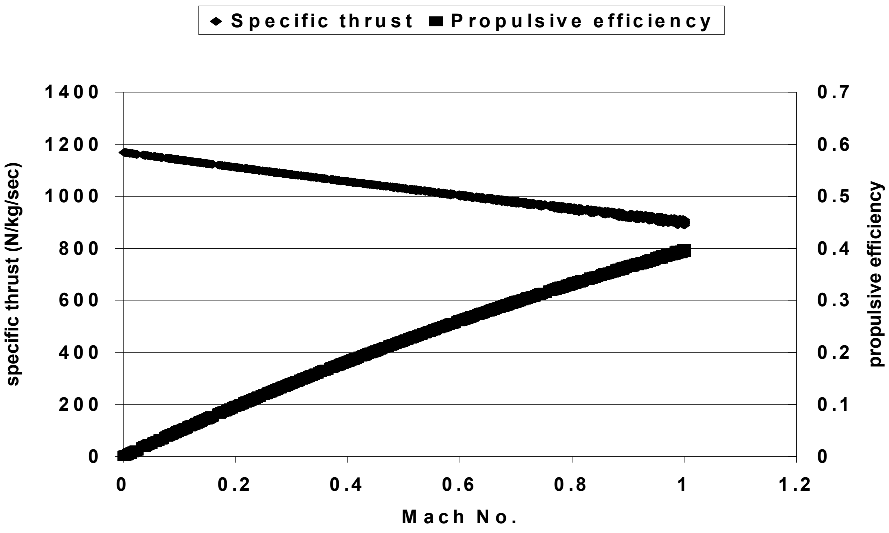


Fig. 10. Relationships of specific thrust & propulsive efficiency with flight Mach No. in 4-objective optimization (Atashkari, et al., 2005)

of NSGA-II in terms of diversity and the uniformity of Pareto front obtained for both 2-objective and 4-objective optimization processes. The robustness of uniform Pareto fronts obtained using MUGA has been shown by the very small values of variance of the metric Δ in multiple runs in comparisons with that of other methods. Further, such multi-objective optimization led to the discovering of important relationships and useful optimal design principles in thermodynamic optimization of ideal turbojet engines both in the space of objective functions and decision variables. The evolutionary multi-objective optimization process has helped to discover important relationships with relatively few efforts of modeling preparation that would otherwise have required at least a very through mathematical analysis. If the underlying objective modeling becomes more complex (like deviating from the ideality of components behaviour) evolutionary multi-objective optimization process may even be expected to become the sole present-time means of attaining respective solutions.

Appendix A

Thermodynamic model of ideal turbojet engine

Assumptions: Inlet diffuser, compressor, turbine and exit nozzle, all operate isentropically. No pressure loss in the burner.

f =(fuel/air)<<1, P_e (turbojet exit pressure)= P_0 (ambient pressure), C_P =1.004 (kJ/kg.K)

$T_0 = 216.6\ \text{K}$, $\gamma = 1.4$, $h_{PR} = 48000\ \frac{\text{kJ}}{\text{kg}}$, $T_{t4} = 1666\ \text{K}$ (in 2 design variables), g_c=1 (kg-m/(N-s^2))

Input parameters:

$$M_0, T_0(K), \gamma, c_P(\frac{kJ}{kg.K}), h_{PR}(\frac{kJ}{kg}), T_{t4}(K), \pi_c$$

Output parameters:

$$ST = \frac{F}{\dot{m}_0}(\frac{N}{kg/s}), SFC(\frac{kg/s}{N}), \eta_t, \eta_P$$

Equations:

$$R = \frac{\gamma-1}{\gamma}.c_P \tag{A-1}$$

$$a_0 = \sqrt{\gamma R g_c T_0} \tag{A-2}$$

$$\tau_r = 1 + \frac{\gamma-1}{2} M_0^2 \tag{A-3}$$

$$\tau_\lambda = \frac{T_{t4}}{T_0} \tag{A-4}$$

$$\tau_c = (\pi_c)^{(\gamma-1)/\gamma} \tag{A-5}$$

$$\tau_t = 1 - \frac{\tau_r}{\tau_\lambda}(\tau_c - 1) \tag{A-6}$$

$$\frac{V_9}{a_0} = \sqrt{\frac{2}{\gamma-1}\frac{\tau_\lambda}{\tau_r\tau_c}(\tau_r\tau_c\tau_t - 1)} \tag{A-7}$$

$$ST = \frac{F}{\dot{m}_0} = \frac{a_0}{g_c}\left(\frac{V_9}{a_0} - M_0\right) \tag{A-8}$$

$$f = \frac{c_P T_0}{h_{PR}}(\tau_\lambda - \tau_r\tau_c) \tag{A-9}$$

$$SFC = \frac{f}{ST} \tag{A-10}$$

$$\eta_t = 1 - \frac{1}{\tau_r\tau_c} \tag{A-11}$$

$$\eta_p = \frac{2M_0}{V_9/a_0 + M_0} \tag{A-12}$$

7. References

Arora, J.S. (1989). Introduction to Optimum Design, McGraw-Hill

Atashkari, K.; Nariman-zadeh, N., Pilechi, A.; Jamali, A.; Yao, X.,(2005), Thermodynamic Pareto Optimization of Turbojet Engines using Multi-objective Genetic Algorithms, *International Journal of Thermal Sciences*, 44, 1061-1071

Back, T.; Fogel, D. B. & Michalewicz, Z. (1997). Handbook of Evolutionary Computation, Institute of Physics Publishing and New York: Oxford University Press

Coello Coello, C.A, http://www.lania.mx/~ccoello/EMOO

Coello Coello, C.A. (1999). A comprehensive survey of evolutionary based multiobjective optimization techniques, *Knowledge and Information Systems: An Int. Journal*, (3), pp 269-308

Coello Coello, C.A. & Becerra, R.L. (2003). Evolutionary Multiobjective Optimization using a Cultural Algorithm, *IEEE Swarm Intelligence Symp.*, pp 6-13, USA

Coello Coello, C.A. & Christiansen, A.D. (2000). Multiobjective optimization of trusses using genetic algorithms, *Computers & Structures*, 75, pp 647-660

Coello Coello, C.A.; Van Veldhuizen, D.A. & Lamont, G.B. (2002). Evolutionary Algorithms for Solving Multi-objective problems, *Kluwer Academic Publishers*, NY

Deb, K. (2003). Unveiling innovative design principles by means of multiple conflicting objectives, *Engineering Optimization*, Vol. 35, No. 5, pp. 445--470, October

Deb, K.; Agrawal, S.; Pratap, A. & Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II, *IEEE Trans. On Evolutionary Computation*, 6(2):182-197

Deb., K. (2001). Multi-objective Optimization using evolutionary algorithms, *John Wiley*, UK

Fonseca, C.M. & Fleming, P.J. (1993). Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization, *In Proc. Of the Fifth Int. Conf. On genetic Algorithms*, Forrest S. (Ed.), San Mateo, CA, Morgan Kaufmann, pp 416-423

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning, *Addison-Wesley*

Homaifar, A.; Lai, H. Y. & McCormick, E. (1994). System optimization of turbofan engines using genetic algorithms, *Applied Mathematical Modelling*, Volume 18, Issue 2, Pages 72-83

Khare, V.; Yao, X. & Deb, K. (2003). Performance Scaling of Multi-objective Evolutionary Algorithms, *Proc. Of Second International Conference on Evolutionary Multi-Criterion Optimization*, (EMO'03), Portugal

Knowles, J. & Corne, D. (1999). The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization, *in Proc. Of the 1999 congress on Evolutionary Computation*, Piscataway, NJ: IEEE Service Center, pp 98-105

Mattingly, J.P. (1996). Elements of Gas Turbine Propulsion, *Mc-Graw Hill*

Pareto, V. (1986). Cours d'economic politique, Lausanne, Switzerland, Rouge

Rao, S.S. (1996). Engineering Optimization: Theory and Practice, *John Wiley & Sons*

Renner, G. & Ekart, A. (2003). Genetic algorithms in computer aided design, *Computer-Aided Design,* Vol. 35, pp 709-726

Rosenberg, R.S. (1967). Simulation of genetic populations with biochemical properties, PhD Thesis, University of Michigan, Ann Harbor, Michigan

Schaffer, J.D. (1985). Multiple objective optimization with vector evaluated genetic algorithms, in Grefenstette, J.J., (ed.) *Proc. of First Int. Conf. On Genetic Algorithms and Their Applications*, pp 93-100, London, Lawrence Erlbaum

Srinivas, N. & Deb, K. (1994). Multiobjective optimization Using Nondominated Sorting in Genetic Algorithms, *Evolutionary Computation*, Vol. 2, No. 3, pp 221-248

Toffolo, A. & Benini, E. (2003). Genetic Diversity as an Objective in Multi-objective evolutionary Algorithms, *Evolutionary Computation* 11(2):151-167, MIT Press

Zitzler, E. & Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: The strength Pareto approach, *Tech. Report 43, Computer engineering and communication network Lab*, Swiss federal ins. of Tech., Zurich

Zitzler, E.; Deb, K. & Thiele, L. (2000). Comparison of multi objective evolutionary algorithms: empirical results, *Evolutionary Computation* 8(2), pp 173-175

EA-based Problem Solving Environment over the GRID

Mohamed Wahib[1], Asim Munawar[1], Masaharu Munetomo[2] and Kiyoshi Akama[2]
[1] *Graduate School Of Information Science and Technology*
[2] *Division of Large Scale Computing Systems, Information Initiative Center*
Hokkaido University
Sapporo, Japan

1. Introduction

EA-based problem solving environments have progressively evolved in the last two decades from explicit one-problem serial solvers to multi-solvers platforms running on vast distributed heterogeneous resources. Significant efforts in the literature were devoted towards designing EA-based problem solving environments. Those research efforts were mainly directed to innovating new EAs with a parallel implementation (Cantu-Paz, 2000), and the counterpart for those research efforts were directed towards designing and constructing parallel computing environments (Weise, 2007) that could host parallel and distributed implementations of EAs. Still for the evolution of the problem solving paradigms, problem solving environments have not fully shifted to parallel and distributed models, and even up till today practices of serially implementing EAs problems of medium complexity are still noticeable. These practices prevailed in part due to the continuous increase in clock speeds, multicore processors, and problem nature.
Yet, in the past few years, the significant increase in distributed resources, high bandwidth/low latency networks and cheap data storage along with the wide expansion in problem scope and addressing new problem types that were not attainable before, all combined together strongly motivated to rethinking the strategy of designing EA-based problem solving environments. Various distributed computing paradigms were used as platforms for EA-based problem solving environments, (Munawar et al., 2008) gives a brief illustration of those paradigms. In this chapter we concentrate on a modern distributed computing paradigm, namely grid computing (Foster & Kesselman, 1999). In the recent years, grid computing acquired widespread attention from both research and industrial institutions, as it provides contextual establishment of open standard platforms for distributed computing (more details in section 2.1)
Constructing an EA-based problem solving environment requires two main streams of working, one is the algorithm design and the other is the challenges associated with constructing a Grid based platform. The algorithm design is significantly affected when using distributed technologies, therefore many points should be taken into account when designing algorithms for distributed environments: fault tolerance, support of

interconnection for loosely coupled resources, support of late binding and dynamic migration. The other main stream which is the challenges accompanied with the grid computing environments both that are general (i.e. grid computing traditional problems) and specific (i.e. challenges related to EA-based solvers deployment).

In this chapter we present MHGrid (Meta Heuristics Grid), a service-oriented grid application that provides easy to use robust environment for meta heuristics optimization solvers, including EAs, over a grid. The objective of MHGrid is to offer a framework, using which a user can solve complex global optimization problems using EAs over a grid with minimal effort. MHGrid is designed in a service-oriented fashion and offers the following services to the user:

1. Allow the user to use any of the solvers registered with MHGrid to solve a problem with minimal input and in a black box manner.
2. Allow solver developers to write and register a new EA-based solver with MHGrid.
3. Allow solver developers to write and register a new objective function with MHGrid.
4. Ability to control the parallelization model of the solver and objective function for high complexity problems.
5. Provide all the preceding services at both the application layer and middleware layer.

This chapter is intended for a reader interested in the implementation of grid based problem solving environments of EAs. The reader is expected to have the basic background about EAs so the chapter scope will be focused on the grid computing problem solving environment and the effect of using the grid on the algorithms (i.e. parallelization and solver–to-objective function relation). We have tried not to overload the chapter with details by providing a very brief summary of the most notable and significant related work. So the chapter is focusing the discussion on the MHGrid platform and not devoted to being a comprehensive overview or survey of the previous work done in the area.

The chapter is organized as follows, section 2 discusses grid-based EAs problem solving environments, it briefly investigates the related work of grid-based EAs. Section 3 discusses the design, architecture and implementation of MHGrid as a problem solving environment. Section 4 presents a close-up, from the service orientation perspective, to the SOA (Service Oriented Architecture) that MHGrid encompasses and also the modelling of MHGrid solvers as services. Section 5 illustrates a full test case starting from a user registering his service to using the registered service. Finally, section 6 concludes the whole chapter and gives an insight for the future work.

2. Grid-based problem solving environment in EAs

This section will give a very brief introduction of grid computing and why use grid computing with EA followed by showing the impact of the grid on algorithm design. Also a revision of the related work is discussed.

2.1 Grid computing

The most commonly used definition to abstractly define a grid is: "Coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organization" (Foster & Kesselman, 1999). The most common among the categories of grid are:

- *Computational Grids:* Grids that basically aggregate computational resource to offer transparent computational power to the applications that use them.

- *Data Grids:* Used to manage and control access to huge distributed data stored on heterogeneous storage devices.
- *Utility Grid:* A market-oriented Grid that applies utility computing concepts in designing the grid.

EAs problem solving environments when associated with grid fall under the category of computational grids. Yet in some problem solving environments that require extensive data handling, techniques that are basic components in data grids such as data replication and staging are introduced in the computational grid. Now almost every production level computational grid has support to what is known as workflow (transfer of data and files across the grid). Nonetheless, grid computing when addressed in EAs conventionally means computational grids.

The grid architecture as shown in figure 2 is a revised version of the traditional grid architecture. The traditional grid architecture is composed of three layers only, the resources, the middleware layer and the application layer. The middleware is a software layer that resides between an application and the underlying platform, in grids the middleware hides the underlying low-level details and complexities from the application layer. Yet, practically in grids, a big semantic gap lies between the middleware and the application layer, so (Abramson, 2006) revised the traditional architecture and modified it by splitting the middleware layer into two layers, the upper middleware layer and the lower middleware layer. This architecture was adopted in MHGrid due to its enhanced subjective representation and ease of modelling.

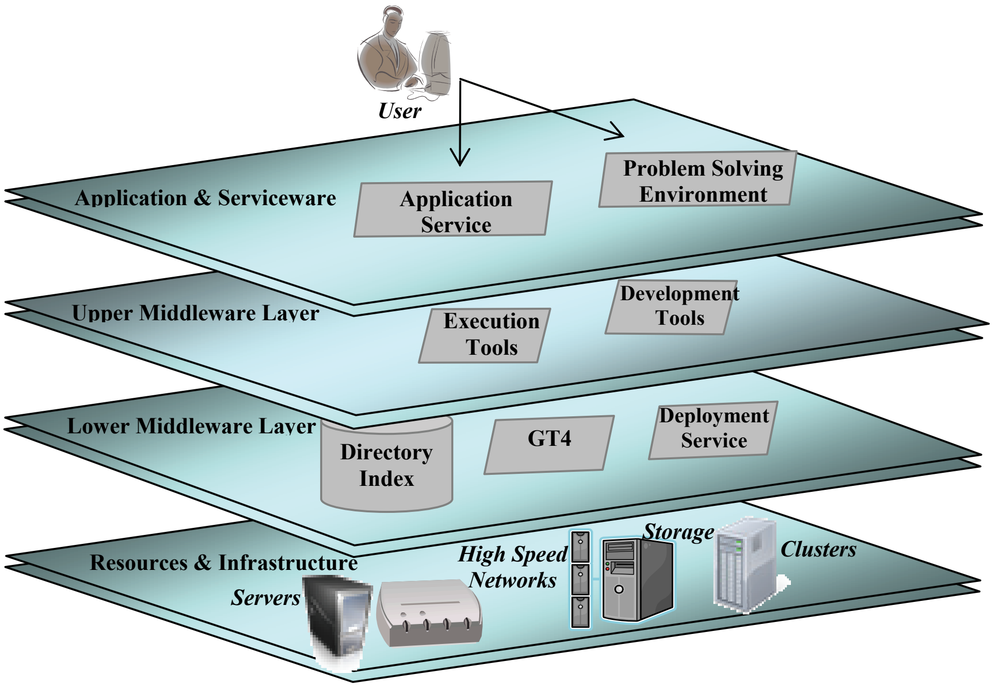


Fig.2. General revised architecture of MHGrid as a computational grid.

2.2 Why grid computing for EAs

An often repeatedly aroused question is why use grid computing for EAs, as it naturally adds a significant overhead to the performance compared to other technologies such as cluster computing. Also designing and implementing a problem solving environment over the grid involves much more complexity than compared to other techniques. The answer to that question lies in a three point checklist by Ian Foster (Foster, 2002), that is when satisfied, classifies the distributed computing framework as a grid. The checklist is:

- Resources are not administered centrally.
- Open standard, general-purpose interfaces and protocols are used.
- Non-trivial quality of service is achieved.

From the checklist above, considering the non-trivial quality of service, grid will be a good choice as a distributed computing paradigm. The major non trivial quality of service is the grid application hosting environment. As the grid application can be available over the Internet and accessed through a Web portal (this is the case in MHGrid), so the hosting environment in this case is the Internet, and the user could be any person accessing the portal and having a valid grid certificate. Other parallel computing paradigms on the other hand (e.g. cluster computing and supercomputing) are available locally in the scale of a LAN, and thus the users in this case, are users having direct access to the resources. This feature of grid computing (i.e. availability over Internet and Intranets) is a basic advantage that attracts developers in the case of applications that are intended to be accessed widely with remote resources.

Other non trivial qualities of service include availability, latency and throughput. A more detailed study on quality of service metrics and aspects in grid is at (Daniel & Emiliano, 2004). The handling and presentation of those metrics could be through defining utility functions (Chunlin & Layuan, 2007) or by defining the provided functionalities as services and thus have a SLA (Service Level Agreement) for each service. One more case that will be most suitable to adopt grid technology with EAs and that is the case of using grid to aggregate resources to provide a huge underlying computational power that enables addressing new complex and relatively expensive problems that were not addressed before due to resource limitation. One fine example to this case is (Chrabakh & Wolski, 2006) in which the authors were able to solve problems that were not solved before due to resource limitation. (Chrabakh & Wolski, 2006) is mainly designated for SAT problems but it still gives a clear evidence of how the grid can be used to address problems of higher complexity compared to other distributed computing paradigms.

Summarizing the need of using grid for EAs; the ability to use non-trivial quality of service metrics rather than speedup, and the ability to use the application over the Internet rather than direct local access is particularly the most important non trivial quality of service. Another reason will be the ability to address new problem of high order complexity and cost depending on the grid ability to aggregate heterogeneous geographically dispersed resources.

2.3 Impact of grid on algorithms

A common practice of running EAs over grid is to use legacy EAs that were written to run on another parallel computing paradigm and running it intactly on the grid. This practice for some algorithms will not be suitable and will be error-prone (i.e. an algorithm that is tightly coupled with out being able to tolerate communication delays will have very significant performance degradation.). From the other side, if the algorithm design did not take into account the nature of the grid it will not benefit most from using a grid and will at best expectations run without any degradation in performance. Therefore the following points should be taken into account when designing EAs for a grid:

- The algorithm should be designed and implemented in a manner that supports interconnection of loosely coupled entities.
- The algorithm should be able to tolerate communication delays for up to 100's milliseconds without significant performance degradation.
- The algorithm should have interfaces allowing for late binding to allow a space for dynamic scheduling and workflows.
- The algorithm should be able to rely on remote data sources as copying the data locally before executing might not be feasible.
- The Algorithm should be fault tolerant.

2.4 Related work

Projects using EAs over grids or EAs problem solving environmets over grid are numerous. Table 1 summarizes some of the notable efforts in this direction and also projects trageted to optimization problem solving environments in general. The table has a comparison of MHGrid with different projects, of different scopes and using different technologies, it gives a close-up to the relation of optimization problems with grids.

NEOS (Czyzck, 1998)*:* The only non-grid based project among the other projects in the table, yet it later motivated using grids for the similar functionality. NEOS is simply a client-server system that is dedicated to solving optimization problems by allowing the user to submit his optimization problems as well as allowing the user to add a solver of his own through NEOS management. The user has no control over the solver parallelization.

Folding@Home (Larson, 2003): This project is categorized as what is called desktop grids, utilizing processor cycles of distributed non-dedicated normal PCs, it was designed to perform computationally intensive simulations of protein folding and other molecular dynamics, it involves GROMACS optimization, and it does not allow user interaction with the job running, the user just installs the client and offer his resources for usage. Folding@Home has not provided optimization solving problem solving environment, yet it is a well known example of how aggregated resources when combined can address new problem scope.

Nimrod/O (Abramson et al., 2000): A very significant project as the authors not only designed the problem solving environment but they also added and modified the grid middleware to adapt with the grid application. Nimrod/O offers namely 4 optimization solving packages solving non-linear optimization problems, but it doesn't allow the user to add his own solver and limits him/her to the provided solvers. Further to mention, Nimrod/O uses an ontology based module to guide the user to the best solver considering his/her problem.

GEODISE (Cox et al., 2002): Specific to optimization problems in computational fluid dynamics, it uses Application Service Provision (ASP) and offers the services through a custom Matlab toolbox, it was designed for production and like Nimrod/O and had a commercial version.

Project	Scope	Black-Box	Arch-itecture	Middleware	Info. Exchange	Adding New Solver By User	Security	Solver Parall-elization	User Inte-rface	User Guid-ence to Solvers
NEOS	Any Optim-ization	Not Supp-orted	Trivial Web Applic-ation	Non-Standard	Plain Sockets	Allowed(Thr-ough Managem-ent)	No	Predefined	Web-based Job Submission	Text Describing Each Solver
Folding @Home	GROMACS Optim-ization	Not Supp-orted	Distri-buted	SMP Client	Client-Server Sockets	Not Allowed	Yes (2048 bit Digital Signat-ure)	Independent Work Units	Folding@Home Client	N/A
Nimrod/O	Non-linear Optimization	Not Supp-orted	Event Driven	OGSA Compliant	Active Sheets	Not Allowed	Yes (GSI)	Predefined	Web Portal	Ontology
GEODISE	Fluid Dyn-amics	Not Supp-orted	ASP	OGSA Compliant	Environment API	Not Allowed	Yes (GSI)	Predefined	Matlab Toolbox/ Portal	Ontology
OSP	Decision Supp-ort System	Not Supp-orted	ASP	Aggregated Components	AMPL/M PL	Not Allowed	Yes	Predefined	Web Portal	Designat-ed Tools
GE-HPGA	Island-model Genetic Algorithm	Supp-orted	Trivial Distribut-ed Appli-cation	OGSA Compliant	Environment API	Not Allowed	Yes (GSI)	Fixed Island Model	None	None
MW	Combinator-ial optimization	Not Supp-orted	ASP	OGSA Compliant	Java Interfaces	Alowed(pa rtially)	Yes (GSI)	Structured Definition	Java Classes Integration	None
MHGrid	Global Optimization with metaheuristics (GA,SA,EDA ,...)	Supp-orted	Service orient-ed grid applic-ation	OGSA Compliant	MHML (XML-based)	Allowed (Through Portal)	Yes (GSI)	Solver Writer defines his own Parallel/ Distributed Model	Web Portal OR Web Services	SLD + SLA

Table 1. Comparison of Different projects providing optimization solving environments.

OSP (Optimization Service Provider, www.osp.org): A recent EU funded project using ASP for solving decision support systems optimization problems. It was later extended to another project (WEBOPT, www.webopt.org) that uses the E-service model instead. Both of them intend to offer a web-based DSS optimization solving environment.

GE-HPGA (Lim et al., 2007): It is similar to MHGrid in offering black-box optimization, the framework is limited to only one solver and the main target was to offer speedup compared to other distributed models. To achieve it's target, GE-HPGA used the island model GA that splits the population into sub-populations to minimize the program inter-communication as much as possible and thus minimize the grid overhead as much as possible.

MW (Glankwamdee & Linderoth, 2006): A framework that is targeting to offer combinatorial optimization solvers over the grid, MW has a very interesting feature for solver and task definition where through MW API (Java interfaces), the user can implement the interfaces to define his task, and also his solver. This technique solves the problem of solver deployment but on the other hand enforces the user to use Java language which is relatively slow, yet it eases the usage of MW by defining flexible interfaces.

MHGrid is a service oriented grid-based framework compliant with OGSA, Open Grid Services Architecture (Foster et al., 2005). It offers various solvers to global optimization problems. All solvers belong to the meta heuristics family of solvers (meta heuristics is a wide category containing EAs and other solver types like search heuristics). Solvers that are meta heuristics based support black box optimization in which the user provides the input and receives the output without knowledge of the underlying computation, black box optimization is a highly desirable feature in optimization solvers to relief the user from involvement in too much details. As for the user interface, the user could use MHGrid's web portal or directly use the Web services of MHGrid. Information interchange between the user and the system is maintained through MHML (Meta Heuristics Mark up Language), Details for MHML are in section 3.4.

3. MHGrid: A grid-based global optimization problem solving environment

MHGrid is a framework dedicated for solving optimization problems over Grid. The main target of the framework is global optimization problems (global optimization is a branch of applied mathematics and numerical analysis that deals with the optimization of a function or a set of functions to some criteria). The framework is intended for the solvers based on heuristic or meta heuristic searching methods.

MHGrid targets general purpose global optimization problems, a major challenge is that according to the No Free Lunch theorem, NFL, (Wolpert & Macready, 1995), no single optimization algorithm will give good results will all problems. The strategy that MHGrid uses to overcome this part is by offering diverse techniques for global optimization covering a wide range of problem type, and also offering mediation between the problem-solver pairs to assure that the solver used is the most adequate to the problem in hand. The strategies enforced by MHGrid to overcome the NFL problem are discussed later in sections 4.1, 4.2.

MHGrid provides the following functions to the user:

- Allows the usage of a solver registered with MHGrid to solve a problem in hand, this is done with a minimal input.
- Enables solver developers to write a new solver that is integrated with MHGrid sing MHAPI, and register it.
- Enables solver developers write a new objective function and register it.

- Do all the previous either through MHGrid's web portal or by directly consuming MHGrid's Web services.

The key contribution is combining the computational power offered by grid technology along with the optimization efficiency of meta heuristics algorithms to give an easy to use general purpose Problem Solving Environment (PSE) for global optimization problems. All MHGrid Web services are WSRF complaint web service to enable the user to use the services directly or through the portal. We have used a unique hybrid parallelization technique that employs GridRPC (Symour et al., 2002) + GridMPI (Ishikawa et al., 2005) approach to dynamically adapt to the grain size of the solver. We have also developed an XML based mark up language, MHML, which acts as an interface between the user and MHGrid Web services.

3.1 MHGrid architecture

Figure 3 gives an overview of MHGrid's architecture, it shows the services that are directly or indirectly used by MHGrid. As the figure shows, the base layer is a high performance grid network, on the top of that runs our Web services in a globus GT4 container (Foster, 2006). All other technologies and services are either build on top of globus or they use globus in one way or the other. Globus Toolkit Monitoring and Discovery Service (MDS) are used by the Condor-G scheduler (Frey et al., 2001) to collect information about the current state of the dynamically changing Grid environment. This information is used by the Condor-G based scheduler to negotiate SLA (Service Level Agreement) with the web service and also to manage and schedule the jobs in a better way.

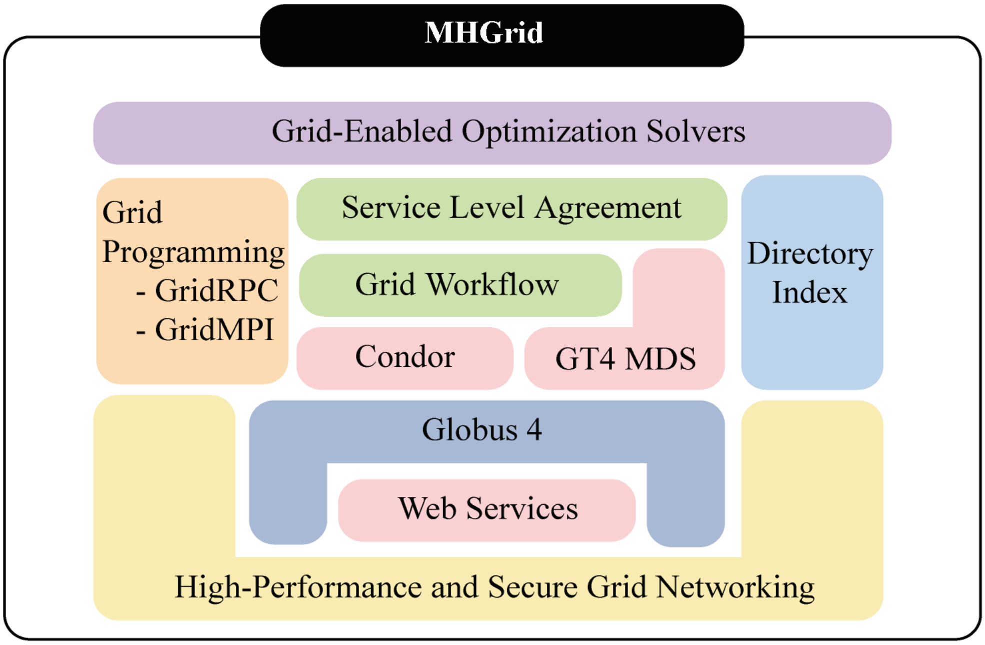

MHGrid
Grid-Enabled Optimization Solvers
Grid Programming - GridRPC - GridMPI
Service Level Agreement
Grid Workflow
Condor
GT4 MDS
Directory Index
Globus 4
Web Services
High-Performance and Secure Grid Networking

Fig.3. MHGrid architecture at an abstract level.

GridRPC (Symour et al., 2002) - MHGrid uses Ninf-G (Tanaka et al., 2003) implementation of GridRPC- and GridMPI (Ishikawa et al., 2005) are also built on top of the Grid technologies, they are Grid variants of the famous Remote Procedure Call (RPC) and Message Passing Interface (MPI) technologies respectively and their use is almost similar to that of their non-Grid counterparts. Next is the Directory index, which is responsible for storing the logs and maintaining the indexes for the solvers and objective functions. A Workflow management module is needed for managing data staging in case of solvers requiring remote datasets. Service Level Agreement (SLA) layer is used for controlling the negotiations between the resource broker (i.e. Condor-G Central Manager) and the users submitting jobs. On top of all these layers come the solvers that run on the Grid to solve global optimization problems.

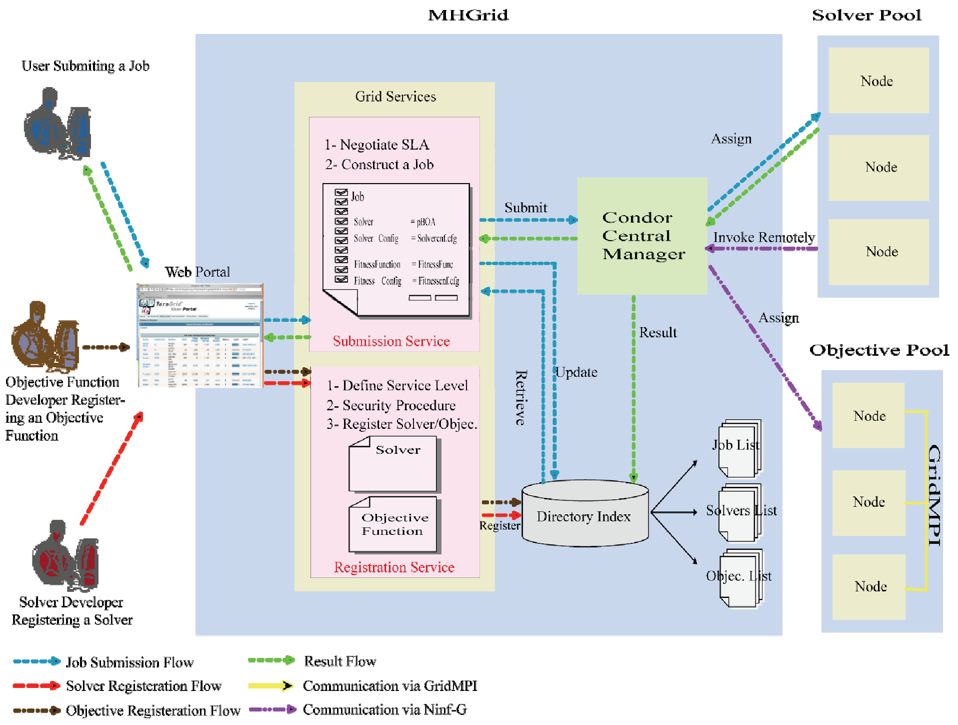


Fig.4. A close-up to MHGrid internals.

Figure 4 gives an insight to the internals of MHGrid and the flow of information inside MHGrid. The arrows with short dashed show the information flow for a user submitting a job, while the dashed-single dotted show an objective function developer registering an objective function and the long dashed are of a solver developer registering a solver. Different modules and functionalities provided by the framework are visible from the figure. The modules of the framework are as follows:

Web Portal: A 2nd generation portal using Gridshpere (Novotny, 2004) as a portlet container. Custom JSR compliant portlets are added to enable the user to use MHGrid with minimal effort. The portal is simply a client application consuming MHGrid's Web services on behalf of the user.

MHGrid's Web services: Runs in a globus container and are the core of MHGrid connecting all components together. Three main services exist, one for retrieving the list of solvers and objective functions registered, one for adding a new solver or objective function to MHGrid and the last is for job submission.
Directory Index: A database that consists of all the objective functions and solvers registered with the framework. It maintains a list of all the jobs and is also responsible for keeping a log of all the previous runs along with the obtained results.
Condor-G based Scheduler: A simple scheduler that is responsible for scheduling jobs to appropriate resources in the grid.

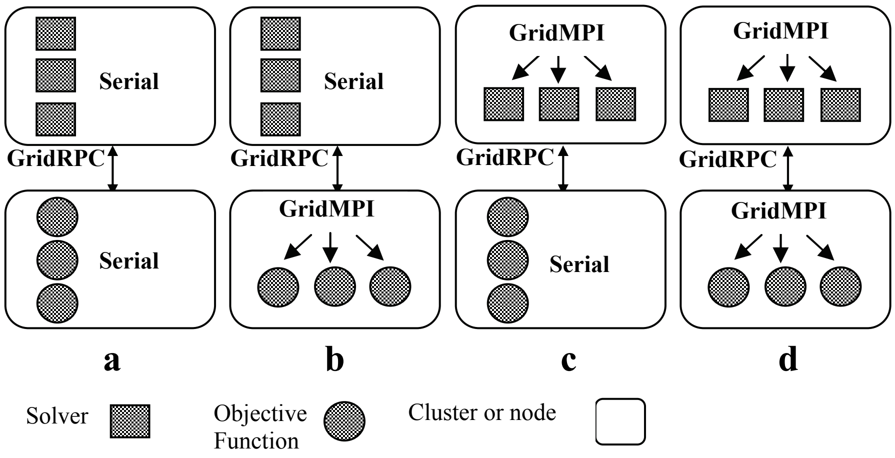


Fig. 5.Different grain size depending on parallelization combination. a) A solver running in serial fashion and objective function computing also running in serial, simplest scenario with no parallelization. b) The solver running serial but the objective computing is running parallel in another cluster, Master-slave GAs are an example that will use this scenario. The Master here is the solver process running in serial and the GridMPI objective function processes are the slaves. c) The solver running in parallel while objective function calculation is serial, a solver like parallel BOA will use this scenario where the objective function calculation is not heavy while the solver involves heavy computation (candidate selection). In this scenario one of the GridMPI solver processes is a controlling node that will call upon objective function calculation. d) Both solver and objective function are running in parallel on different clusters. This scenario will have one of the GridMPI solver processes acting as a controlling node that will be acting as a master for the GridMPI objective function processes.

3.2 Dynamic grain size in MHGrid

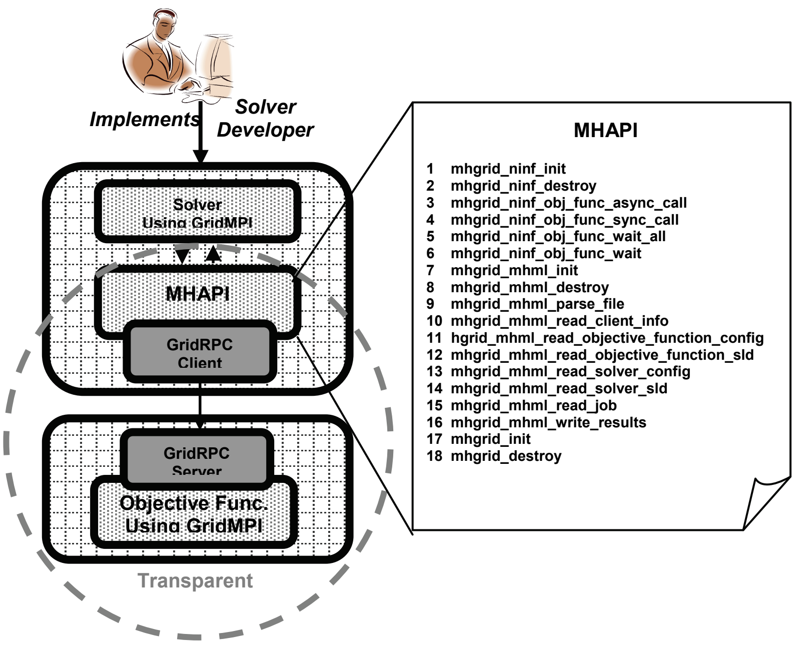
Parallelization in meta heuristics in general differs depending on the algorithm communication/computation ratio. To offer an environment that will host a variety of solvers, there is a necessity of having a mechanism that allows the usage of different parallelization technologies to be used within the solvers and objective functions. MHGrid uses a hybrid of two technologies, GridMPI and GridRPC (MHGrid uses Ninf-G, a wrapper for GridRPC). Figure 5 shows how the mixed use of GridMPI and GridRPC can offer

different parallelization models providing the solver developer with flexibility in designing his solver. This unique parallelization technique employing GridRPC and GridMPI was first used in (Takemiya et al, 2006) for a specific problem. MHGrid deploys this technique as a general model for dynamic grain size definition.
The deployment of solvers and objective functions in such a way to provide those parallelization models is a complicated process that uses both Ninf-G and Condor-G deployment techniques. Detailed method of objective function deployment is discussed in (Munawar et al., 2008).

3.3 Solver developing and integration to MHGrid

When a user requires adding a solver to MHGrid, he is required to provide two things, the first is the solver source files and the other is an MHML file including the SLD of the solver to be added (the SLD part of MHML usage will be explained later in section 4.2). On the other hand for the user to be able to integrate his solver with MHGrid, he/she needs to use MHAPI. MHAPI is an API provided by MHGrid that includes a set of functions that allow the user to run and deploy his solver on MHGrid. As shown in figure 6 the solver developer writes the solver and uses the APIs in MHAPI for the following:

- Reading the input and configuration data from the job's MHML file.
- Calling the objective function calculation whenever needed.
- Initialize the deployment of the objective function. Then MHGrid will transparently deploy the objective function on behalf of the user.



```
1  mhgrid_ninf_init
2  mhgrid_ninf_destroy
3  mhgrid_ninf_obj_func_async_call
4  mhgrid_ninf_obj_func_sync_call
5  mhgrid_ninf_obj_func_wait_all
6  mhgrid_ninf_obj_func_wait
7  mhgrid_mhml_init
8  mhgrid_mhml_destroy
9  mhgrid_mhml_parse_file
10 mhgrid_mhml_read_client_info
11 hgrid_mhml_read_objective_function_config
12 mhgrid_mhml_read_objective_function_sld
13 mhgrid_mhml_read_solver_config
14 mhgrid_mhml_read_solver_sld
15 mhgrid_mhml_read_job
16 mhgrid_mhml_write_results
17 mhgrid_init
18 mhgrid_destroy
```

Fig.6.Main functionalities provided by MHAPI. Note that every thing is kept transparent from the solver developer.

Two points to note here about objective function calling and objective function deployment. For objective function calling, the writer of the objective function is usually different from the writer of the solver, so for an objective function to be used by solvers in MHGrid, it must comply with a predefined Ninf IDL. This IDL defines the interfacing between the solver and any objective function that will be used with it with eyes on the different problem encodings that can be used (e.g. binary, real, combinatorial … etc). Figure 7 shows how a simple Ninf-IDL file looks like.

```
// Sample IDL file

Module obj;
Define obj-func(IN int in_length_of_chromosome, IN float in_chromosome[length], OUT float *out_fitness)
“sga on rpc”
Required “obj_func.o”
{
            Extern float obj_func(float length, float *x);
            *out_fitness = obj_func(int in_length_of_chromosome, in_chromosome);
}
```

Fig. 7.Simple sample of a Ninf-IDL file.

3.4 MHML

MHML is an XML-based language providing all the functionalities required from a language to describe meta heuristics information interchange. Full details about MHML is beyond the scope of this chapter, MHML language is fully demonstrated in (Munawar et al., 2007), we will only summarize why the need to use MHML and the basic features of MHML.

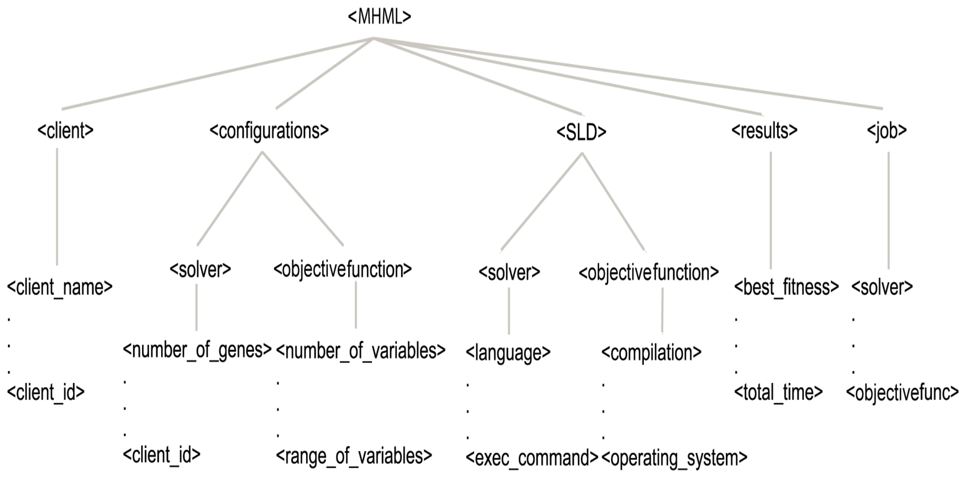


Fig.8.Top level hierarchy of MHML.

The rationale behind MHML was the need for standardizing the communication interface. Standardizing the communication interface not only enables a flexible design, but also eases the process of extendibility and interoperability. XML was chosen as it appears the most promising information interchange language, and its wide dominance in the area of web-based information interchange.
MHML basically is an extension/modification to an earlier attempt by (Alba et al., 2003). (Alba et al., 2003) proposed a language to configure optimization algorithms as XML DTD. Yet, it failed to address important issues considering the configuration of optimization algorithms. MHML offers many advantages compared to (Alba et al., 2003), from the top-level hierarchy of MHML shown in figure 8, it is clear that MHML has the capability to represent: *Job configuration, Solver description and configuration, Objective function description and configuration, submitting client information* and *job results*

4. Service orientation aspect in MHGrid

Creating a general framework for global optimization problem solving is challenged with two major problems that will compromise the generality-to-performance trade-off; the first problem is that if the set of available solvers is fixed then the overall scope of the framework will be limited to the solvers in hand. The second problem is the reduced efficiency due to week or non existing relation between the solver and the problem using the solver. Added to the complexity of the second problem is that the nature and availability of the underlying resources is dynamically changing in Grid-based systems. Another complexity added to the second problem is the compound nature of meta heuristics based solvers, as Meta heuristic based solvers constitute of the main solver code and the objective function which is a computationally independent, cost expensive and repeatedly called function. Thus, the need to formulate the interaction between solver and objective function counterparts.
MHGrid tackles these two problems by adopting service oriented architecture (SOA), this SOA is attained in MHGrid by applying a set of strategies in both the vertical and horizontal direction. And by applying these set of strategies that melt down MHGrid in a SOA frame, the performance of MHGrid as a framework is leveraged to the desired level of being a general framework (i.e. addressing problems of different scope.) while still offering a reasonable performance to the problems submitted. Figure 9 shows three different models with different problem type to performance relations. The Narrow scope-High Quality model is the typical case of optimization problem solvers according to NFL (Wolpert & Macready., 1995). The Wide scope low quality model is a model having a set of robust solvers. This model targets average performance for wide scope of problem types. The last model, MHGrid, targets a wide problem scope with performance that is high above the average by modelling MHGrid in a SOA through applying strategies to expand in the horizontal and vertical directions.
This section will give a close-up to the SOA of MHGrid by discussing the strategies used to model MHGrid into a SOA. An important point to note here is that MHGrid doesn't embrace SOA by just using OGSA and Web services in the middleware layer, as normally in SOA context, modelling a framework to fit into a SOA implies using Web services. This is not the case in MHGrid, as Web services – though used in all modules of MHGrid – are just tools in the middleware layer. The SOA referred to here is effective at the application layer (i.e. solvers as services), section 4.2 discusses this point in details. The next sections will discuss the horizontal expansion strategies, vertical expansion strategies and finally the impact of those expansions on the adaptation of MHGrid into a SOA.

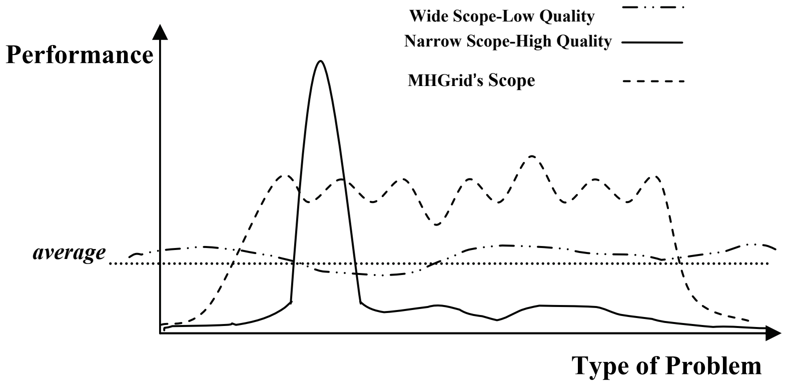


Fig.9. MHGrid scope according to problem-type space.

4.1 Horizontal expansion strategies

Expanding MHGrid in the horizontal direction is mainly directed to widen the solvers base. The strategies that MHGrid use to expand horizontally can be summarized in two points:

- Offer a variety of state-of-art robust solvers that make the framework suitable for different problem types.
- Allow the user to add his own solver(s) and objective function(s).

For the first point, a set of robust solvers developed by the information systems design laboratory at the information initiative center, Hokkaido university are to be used in MHGrid platform. These solvers are the fuel of MHGrid that provide the ability to address a wide variety of problems. The second point is as mentioned before in section 3.3, providing a mechanism to allow the solver developers to add their solvers and objective functions.

4.2 Vertical expansion strategies

The vertical expansion strategies are much more complicated as they are mainly concerned with increasing the semantics of the solver to problem relation. The following are the strategies:

- Solvers and objective functions are represented as services in MHGrid, thus binding a Service Level Description (SLD) with each solver/objective function to describe the service level offered by the solver/objective function. MHML has two main sections one for solvers and the other for objective functions. The SLD part should be submitted with newly added solvers/objective functions. The SLD section contains information like what problem type is the solver targeting, problem encoding and what model of parallelization is used (e.g. Island model GA will use any parallel model while master/slave pBOA requires the solver to run in parallel on the same cluster). The SLD information is later used to guide the user for which solver to select to the problem in hand and to check if the grid resources will support the parallelization model required.
- Having an M-N relation between the solvers and objective functions registered with MHGrid, where the user can run the same solver against many objective functions and vice versa. This strategy is handled through the Ninf-IDL interface described in section 3.3.

- Allowing the solver developer to control the parallelization model in the solver /objective function he writes. The solver developer can choose the parallelization model and thus the grain size as mentioned in section 3.2.
- Offering two SAPs (Service Access Points) for the user of MHGrid, one of them is the web portal and the other is by consuming the MHGrid's Web services directly. Accessing MHGrid services through the portal will be shown in the test case of section 5, also there is another SAP that can be used in case the user wants to avoid the overhead in using the web portal and also to use MHGrid's services automatically in case he needs that.
- Having a Service Level Agreement (SLA) for each job submitted to MHGrid. Initially upon job submission and after the user chooses the solver/objective function pair, the scheduler checks the state of the available resources, then the SLA manager using the state of resources along with the solver/objective function SLD informs the user with the expected scenario that rises from running the selected solver/objective function running on the current available resources. The SLA in the case of MHGrid is at the application layer and not the middleware layer, and therefore refraining from the expected SLA procedure at middleware (i.e. SLA based scheduling). SLA at the application layer guarantee to the user that his problem is well matched to a solver, while if at middleware layer will be targeting QoS metrics such as time, cost and resources availability. The current SLA implementation is rather trivial, but different options are now being investigated and it is anticipated that SLA mechanism will later use e-contracts at the application level.

4.3 MHGrid as a grid application benefiting from SOA

Grid applications are combined with SOA and service fundamentals in many projects, and often the grid application that are modelled after SOA are referred to as *service-oriented grid applications.* The case of MHGrid despite being a service-oriented application, yet it used a different approach to combine SOA with grid technology. MHGrid as a framework is designed to be a *general* framework for global optimization, yet this goal was challenged with the NFL theorem, and so the expansion in both directions was thought of in order to enable more generality for MHGrid. This expansion design for MHGrid was clearly consistent with SOA fundamentals and concepts, for example the following are the SOA projections mapped to the vertical expansion strategies:

- Solvers as services with SLDs. Mapping: A well known practice of SOA, where every service in a SOA model should have a description of what it is doing in order to be used later for QoS process. Analogous to WSDL associated with Web services.
- M-N solver to objective function relation (one solver can be associated to many objective functions and vice versa). *Mapping: Service interoperability is a main concept in SOA.*
- Solver developer control over the parallelization model. Mapping: From SOA perspective, this is providing strong semantics for inter-services relations.
- Offering two SAPs. *Mapping: The two service access points for the solvers in MHGrid comes in favour of ease of use, this polymorphic interfacing to the services is indeed a merit from SOA perspective.*
- Having an SLA for each job submitted. *Mapping: A straightforward SOA pillar.*

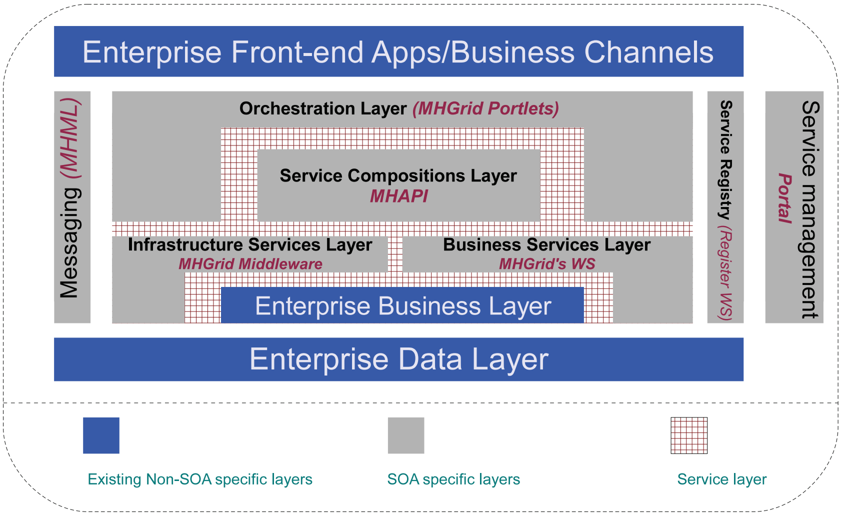

Enterprise Front-end Apps/Business Channels
Messaging (MHML)
Orchestration Layer (MHGrid Portlets)
Service Compositions Layer
MHAPI
Infrastructure Services Layer
MHGrid Middleware
Business Services Layer
MHGrid's WS
Enterprise Business Layer
Service Registry (Register WS)
Service management
Portal
Enterprise Data Layer
Existing Non-SOA specific layers
SOA specific layers
Service layer

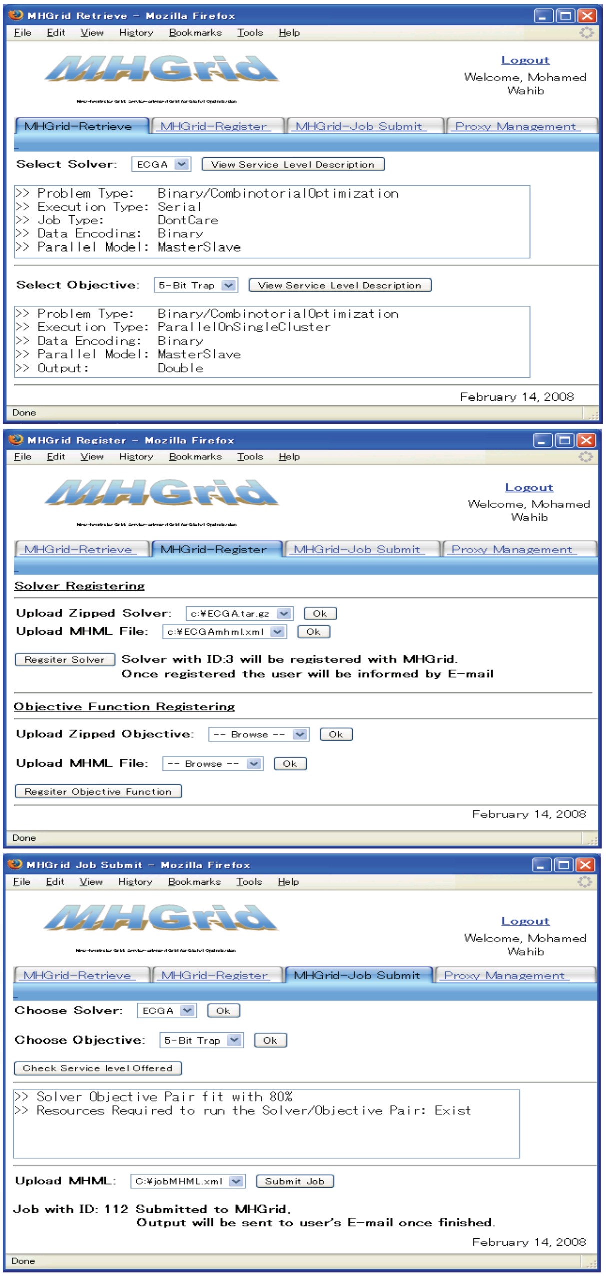
Fig.10. MHGrid modules mapped to a typical SOA layout.

The point of concern that can be concluded from merging MHGrid as a grid application with SOA, is that MHGrid was not designed as a SOA compliant model in order to benefit from the typical advantages of SOA such as ease of extensibility, but MHGrid was framed into a SOA model to achieve the basis of having a general problem solving framework in terms of wide problem type support. Figure 10 shows a mapping of MHGrid modules to a typical SOA layout.

5. Test case of MHGrid from user perspective

This section illustrates a test case example for MHGrid from the user perspective. The illustration will start by a solver developer registering a solver he wrote for MHGrid, then as a user retrieving the list of solvers and objective functions and finally submitting a job to MHGrid.

- Solver registration: The solver developer will initially write his solver that uses MHAPI, and then write the MHML file with the SLD section of the solver. Then the solver developer logins to the portal opens the solver registration portlet and uploads both the solver tar ball and the MHML file. The solver developer will be notified though his e-mail registered with the portal. Figure 11 middle snapshot shows the solver registration portlet while registering a solver.
- Job submission: The job submission is done in two steps, first the user uses the retrieve portlet to get a list of all the registered solvers and the registered objective functions. For each solver and objective function displayed, the information for the corresponding SLD is displayed to give guidance to the user. Figure 11 top snapshot shows the retrieve portlet where the user can view the solvers and objective functions before deciding which one to use. The next step where the user actually submits the job, the user will switch to the job submission portlet and choose a solver/objective function pair, and then the portlet will give the user an indication of how the current available resources are coherent with the SLDs of the solver and objective function. After the user decides which solver/objective function pair to use, he has to supply the MHML job file, and he'll later get the MHML result file on his e-mail. Figure 11 bottom snapshot shows the job submission portlet while in submission process.


MHGrid Retrieve – Mozilla Firefox
File Edit View History Bookmarks Tools Help
MHGrid
Logout
MHGrid-Retrieve
MHGrid-Register
MHGrid-Job Submit
Proxy Management
View Service Level Description
Done
MHGrid Register – Mozilla Firefox
Regsiter Solver
Regsiter Objective Function
MHGrid Job Submit – Mozilla Firefox
Check Service level Offered
Submit Job
Ok

Welcome, Mohamed Wahib

Select Solver: ECGA

```
>> Problem Type:    Binary/CombinotorialOptimization
>> Execution Type:  Serial
>> Job Type:        DontCare
>> Data Encoding:   Binary
>> Parallel Model:  MasterSlave
```

Select Objective: 5-Bit Trap

```
>> Problem Type:    Binary/CombinotorialOptimization
>> Execution Type:  ParallelOnSingleCluster
>> Data Encoding:   Binary
>> Parallel Model:  MasterSlave
>> Output:          Double
```

February 14, 2008

Welcome, Mohamed Wahib

Solver Registering

Upload Zipped Solver: c:¥ECGA.tar.gz

Upload MHML File: c:¥ECGAmhml.xml

Solver with ID:3 will be registered with MHGrid.
Once registered the user will be informed by E-mail

Objective Function Registering

Upload Zipped Objective: -- Browse --

Upload MHML File: -- Browse --

February 14, 2008

Welcome, Mohamed Wahib

Choose Solver: ECGA

Choose Objective: 5-Bit Trap

```
>> Solver Objective Pair fit with 80%
>> Resources Required to run the Solver/Objective Pair: Exist
```

Upload MHML: C:¥jobMHML.xml

Job with ID: 112 Submitted to MHGrid.
Output will be sent to user's E-mail once finished.

February 14, 2008

Fig.11. Top Snapshot: A user registering a solver with MHGrid through the web portal. Middle snapshot: A user retrieving information about the solvers and objective functions registered with MHGrid through the web portal. Bottom snapshot: A user submitting a job to MHGrid through the web portal

This was a simple example case just to acknowledge the reader with how MHGrid is viewed from the user perspective, nevertheless, MHGrid can still be accessed directly from the Web services, but illustration for that was skipped to refrain the user from details outside the scope of the book.

6. Conclusions and future work

This chapter presented a grid based problem solving environment that uses EAs and other algorithms all falling under the meta heuristics category to offer black box global optimization for the user. The chapter first highlighted the grid computing technology and then discussed with reasons behind using the grid for MHGrid, Meta Heuristics Grid, and the benefits of the grid technology compared to other distributed paradigms.

Then a comparison of MHGrid with related work was discussed, to imply the concepts behind the design of optimization solving grid applications. The design and implementation of MHGrid was explained, including the layered architecture, the workflow inside the framework and explanation of MHAPI, a library that allows the solver developers to integrate their solvers with MHGrid.

MHGrid as a model was expanded in both the vertical and horizontal directions in order to widen the base of MHGrid to be a general framework rather than being tailored to one problem type. The expansion strategies reformed the architecture of MHGrid into a SOA, the main impact for MHGrid adopting SOA was the representation of solvers and objective functions as services and thus having the service oriented grid application mostly affecting the application layer whilst using OGSA and Web services at the middleware layer. A sample example case was demonstrated to acknowledge the reader with the user perspective of MHGrid.

For the future work, modifications and extensions will cover different aspects. Major points will include adopting a more sophisticated SLA mechanism, defining new interfaces that allow one solver to use another solver, for example pBOA algorithm can internally use Tabu search for candidate offspring selection, and one more important point is to conduct more study on the dynamic grain size in EAs to reach the best formulation of parallelization models adopted. Other minor points will include enchantments on the portlets to auto generate the MHML files on behalf of the users.

7. References

Abramson, D. ; Lewis, A. & Peachy, T. (2000). Nimrod/O: A Tool for Automatic Design Optimization, *Proceedings of 4th International Conference on Algorithms & Architectures for Parallel Processing; ICA3PP 2000*, pp. 90-98, ISBN, Hong Kong, December 2000, World Scientific Publishing

Abramson, D. (2006). Applications Development for the Computational Grid, *Proceedings of APWEB 2006*, pp. 1-12, China, January 2006, Springer, Harbin

Alba, E.; Garc-Nieto, J. & Nebro, A. (2003). *On the Configuration of Optimization Algorithms by Using XML Files,* Technical report, http://neo.lcc.uma.es/publications/Publi2003

Cantu-Paz, E. (2000). *Efficient and Accurate Parallel Genetic Algorithms,* Springer, 0792372212, Chicago IL.

Chrabakh, W. & Wolski, R. (2006). GridSAT: Design and Implementation of a

Computational Grid Application. *Journal of Grid Computing,* Vol. 4, No. 2, (June 2006) pp. 177-193. 1570-7873

Chunlin, L. & Layuan, L. (2007). Utility Based Multiple QoS Guaranteed Resource Scheduling Optimization in Grid Computing. *Proceeding of the International Conference on Computing: Theory and Applications, 2007. ICCTA apos;07,* pp 165-169, India, October 2007, Kolkata.

Cox, S.; Chen, L.; Campobasso, S.; Duta, M.; Eres, M.; Giles, M.; Goble, C.; Jiao, Z.; Keane, A.; Pound, G.; Roberts, A.; Shadbolt, N.; Tao, F.; Wason, J. & Xu, F. (2002). *Grid Enabled Optimization and Design Search (GEODISE),* Technical report, www.geodise.org

Czyzck, J.; Mesnier, M. & More, J. (1998). The NEOS Server. *IEEE Journal on Computational Science and Engineering,* Vol. 5, No. 3, (May 1998) pp. 68-75.

Daniel, M. & Emiliano, C. (2004). Quality of Service Aspects and Metrics in Grid Computing. *Proceeding of Computer Measurement Group Conference,* pp 102-111, USA, December 2004, Las Vegas, NV.

Foster, I. & Kesselman, C. (1999). *The Grid : Blueprint for a New Computing Infrastructure,* Morgan Kaufmann Publishers, 1558604758, Chicago IL.

Foster, I. (2002). What is the Grid? A three point checklist. *GRIDtoday,* Vol. 1, No. 6, (February 2002)

Foster, I.; Kishimoto, H..; Sava, A.; Berry, D.; Djaoui, A.; Grimshaw, A.; Horn, B.; Maciel, F.; Siebenlist, F.; Subramaniam, R.; Treadwell, J. & Reich, J. (2005). *The Open Grid Services Architecutre Version 1.0,* GGF informational document Global Grid forum, www.globalgridforum.org

Foster, I. (2006). Globus toolkit version 4: Software for service oriented systems, *Proceedings of IFIP International Conference on Network and Parallel Computing,*, pp. 2-13, Japan, October 2006, Sprinder-Verlag LINCS 3779, Tokyo

Frey, J.; Tannenbaum, T.; Foster, I.; Livny, M. & Tuecke, S. (2001). Condor-G: A Computation Management Agent for Multi-institutional Grids, *Proceedings of 10th IEEE symposium on High Performance Distributed Computing*, pp. 7-19, USA, August 2001, Morgan Kaufmann Publishers, San Francisco, California

Glankwamdee, V. & Linderoth, J. (2006). MW: A Software Framework for Combinatorial Optimization on Computational Grids, In: *Parallel Combinatorial Optimization,* Talbi, E., pp. 239-256, Wiley-Interscience, 0471721018

Ishikawa, Y.; Kaneo, Y.; Edamoto, M.; Okazaki, F.; Koie, H.; Takano, R.; Kudoh, T. & Kodama. Y., (2005). Overview of GridMPI Version 1.0, *Proceedings of SWoPP'05*, pp. 116-127, Japan, October, Tokyo

Larson, S.; Snow, C. & Pande, V. (2003). *Folding@Home and Genome@Home: Using Distributed Computing to Tackle Previously Intractable Problems in Computational Biology*, R. Grant, Horizon Press

Lim, D.; Ong, Y.; Jin, Y.; Sendhoff, B. & Lee, S. (2007). Efficient Hierarchical Parallel Genetic Algorithms using Grid Computing. *Future Generation Computational systems,* Vol. 4, No. 23, (May 2007) pp. 658-670.

Munawar, A.; Wahib, M.; Munetomo, M. & Akama, K. (2007). Optimization Problem Solving Framework Employing GAs with Linkage Identification over a Grid Environment, *Proceedings of CEC2007: IEEE Congress on Evolutionary Computation*, pp. 3659-3661, Singapore, September 2007

Munawar, A.; Wahib, M.; Munetomo, M. & Akama, K. (To Appear). Parallel GEAs with Linkage Analysis over Grid, In: *Linkage in Genetic and Evolutionary Algorithms*, Springer.

Novotny, J.; Russell, M. & Wehrens, O. (2004). Gridsphere: A Portal Framework for Building Collaborations, *Concurrent Computing: Practices and Exercises,* Vol. 5, No. 16, （June 2004) pp. 503-513

Symour, K.; Nakada, H.; Matsuoka, S.; Dongarra, J.; Lee, C. & Casanova, H. (2002). Overview of GridRPC: A Remote Procedure Call API for Grid Computing, *Proceedings of 3rd International Workshop of Grid Computing*, pp. 274-278, USA, November 2002, Morgan Kaufmann Publishers, Baltimore, Maryland

Takemiya, H.; Tanaka, Y.; Sekiguchi, S.; Ogata, S.; Kalia, R.; Nakano, A. & Vashishta, P. (2006). Sustainable Adaptive Grid Supercomputing: Multiscale Simulation of Semiconductor Processing Across the Pacific, *Proceedings of the 2006 ACM/IEEE conference on Super Computing; SC'06*, pp. 106-118, USA, November 2006, Morgan Kaufmann Publishers, New York, NY

Tanaka, Y.; Nakada, H.; Sekiguchi, S.; Suzumura, T. & Matsuoka, S. (2003). Ninf-G: A Reference Implementation of RPC based Programming Middleware for Grid Computing, *Journal of Grid Computing,* Vol. 3, No. 7, (June 2003) pp. 41-51

Wahib, M.; Munawar, A.; Munetomo, M. & Akama, K. (2007). MHGrid: Towards an Ideal Optimization Environment for Global Optimization Problems using Grid Computing, *Proceedings of Parallel and Distributed Computing, Applications and Technologies; PDCAT2007*, pp. 217-220, Australia, December 2007, Morgan Kaufmann Publishers, Adelaide

Weise, T. (2007). *Global Optimization Techniques and Genetic Programming Applied to Distributed Computing,* Thomas Weise, Online as e-book.

Wolpert, H. & Macready, G. (1995). *No Free Lunch Theorems for Search,* Technical report, SFI-TR-95-02-010 Santa Fe, NM

Part IV:

Applications

Evolutionary Methods for Learning Bayesian Network Structures

Thierry Brouard, Alain Delaplace and Hubert Cardot
Université Francois-Rabelais de Tours - Laboratoire Informatique
France

1. Introduction

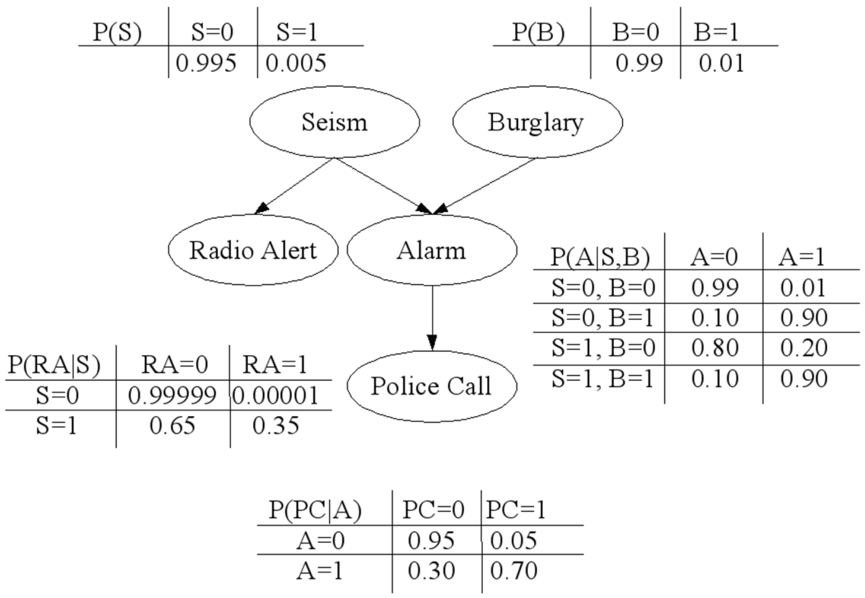
Bayesian networks (BN) are a family of probabilistic graphical models representing a joint distribution for a set of random variables. Conditional dependencies between these variables are symbolized by a Directed Acyclic Graph (DAG). Two classical approaches are often encountered when automaticaly determining an appropriate graphical structure from a database of cases,. The first one consists in the detection of (in)dependencies between the variables (Spirtes et al., 2001; Cheng et al., 2002). The second one uses a scoring metric (Chickering, 2002a). But neither the first nor the second are really satisfactory. The first one uses statistical tests which are not reliable enough when in presence of small datasets. If numerous variables are required, it is the computing time that highly increases. Even if score-based methods require relatively less computation, their disadvantage lies in that the searcher is often confronted with the presence of many local optima within the search space of candidate DAGs. Finally, in the case of the automatic determination of the appropriate graphical structure of a BN, it was shown that the search space is huge (Robinson, 1976) and that is a NP-hard problem (Chickering et al., 1994) for a scoring approach.
In this field of research, evolutionary methods such as Genetic Algorithms – GAs (De Jong, 2006) have already been used in various forms (Larrañaga et al., 1996; Muruzábal & Cotta, 2004; Wong et al., 1999; Wong et al., 2002; Van Dijk et al., 2003b; Acid & De Campos, 2003). Among these works, two lines of research are interesting. The first idea is to effectively reduce the search space using the notion of equivalence class (Pearl, 1988). In (Van Dijk et al., 2003b) for example the authors have tried to implement a genetic algorithm over the partial directed acyclic graph space in hope to benefit from the resulting non-redundancy, without noticeable effect. Our idea is to take advantage both from the (relative) simplicity of the DAG space in terms of manipulation and fitness calculation and the unicity of the equivalence classes' representations.
One major difficulty when tackling the problem of structure learning with scoring methods — evolutionary methods included — is to avoid the premature convergence of the population to a local optimum. When using a genetic algorithm, local optima avoidance is often ensured by preserving some genetic diversity. However, the latter often leads to slow convergence and difficulties in tuning the GA's parameters.
To overcome these problems, we designed a general genetic algorithm based upon dedicated operators: mutation, crossover but also a mutual information-driven repair

operator which ensures the closeness of the previous. Various strategies were then tested in order to find a balance between speed of convergence and avoidance of local optima. We focus particularly onto two of these: a new adaptive scheme to the mutation rate on one hand and sequential niching techniques on the other.
The remaining of the chapter is structured as follows: In the second section we will define the problem, ended by a brief state of the art. In the third section, we will show how an evolutionary approach is well suited to this kind of problem. After briefly recalling the theory of genetic algorithms, we will describe the representation of a Bayesian network adapted to genetic algorithms and all the needed operators necessary to take in account the inherent constraints to Bayesian networks. In the fourth section the various strategies will then be developed: Adaptive scheme to the mutation rate on one hand and niching techniques on the other hand. The fifth section will describe the test protocol and the results obtained compared to other classical algorithms. A study of the behaviour of the used strategies will also be given. And finally, the sixth section will present future search in this domain.

2. Problem settings and related work

2.1 Settings

A probabilistic graphical model can represent a whole of conditional relations within a field $X = \{X_1, X_2, \ldots, X_n\}$ of random variables having each one their own field of definition. Bayesian networks belong to a specific branch of the family of the probabilistic graphical models and appear as a directed acryclic graph (DAG) symbolizing the various dependences existing between the variables represented. An example of such a model is given Fig. 1.



P(S)	S=0	S=1
	0.995	0.005

P(B)	B=0	B=1
	0.99	0.01

P(A|S,B)	A=0	A=1
S=0, B=0	0.99	0.01
S=0, B=1	0.10	0.90
S=1, B=0	0.80	0.20
S=1, B=1	0.10	0.90

P(RA|S)	RA=0	RA=1
S=0	0.99999	0.00001
S=1	0.65	0.35

P(PC|A)	PC=0	PC=1
A=0	0.95	0.05
A=1	0.30	0.70

Fig. 1. Example of a Bayesian network.

A Bayesian network is denoted $B = \{G, \theta\}$. Here, $G = \{X, E\}$ is a directed acyclic graph whose set of vertices X represents a set of random variables and its set of arcs E represents the dependencies between these variables. The set of parameters θ holds the conditional probabilities for each vertice, depending on the values taken by its parents in G. The probability $\theta_i = \{P(X_i \mid Pa(X_i))\}$, where $Pa(X_i)$ are the parents of variable X_i in G. If X_i has no parents, then $Pa(X_i) = \emptyset$.
The main convenience of Bayesian networks is that, given the representation of conditional independences by its structure and the set θ of local conditional distributions, we can write the global joint probability distribution as:

$$P(X_1,\ldots,X_n) = \prod_{k=1}^{n} P(X_k \mid Pa(X_k)) \tag{1}$$

2.2. Field of applications of Bayesian networks

Bayesian networks are encountered in various applications like filtering junk e-mail (Sahami et al., 1998), assistance for blind people (Lacey & MacNamara, 2000), meteorology (Cano et al., 2004), traffic accident reconstruction (Davis, 2003), image analysis for tactical computer-aided decision (Fennell & Wishner, 1998), market research (Jaronski et al., 2001), user assistance in sofware use (Horvitz et al. 1998), fraud detection (Ezawa & Schuermann, 1995), human-machine interaction enhancement (Allanach et al., 2004).
The growing interest, since the mid-nineties, that has been shown by the industry for Bayesian models is growing particularly through the widespread process of interaction between man and machine to accelerate decisions. Moreover, it should be emphasized their ability, in combination with Bayesian statistical methods (i.e. taking into account prior probability distribution model) to combine the knowledge derived from the observed domain with a prior knowledge of that domain. This knowledge, subjective, is frequently the product of the advice of a human expert on the subject. This property is valuable when it is known that in the practical application, data acquisition is not only costly in resources and in time, but, unfortunately, often leads to a small knowledge database.

2.3 Training the structure of a Bayesian network

Learning Bayesian network can be broken up into two phases. As a first step, the network structure is determined, either by an expert, either automatically from observations made over the studied domain (most often). Finally, the set of parameters θ is defined here too by an expert or by means of an algorithm.
The problem of learning structure can be compared to the exploration of the data, i.e. the extraction of knowledge (in our case, network topology) from a database (Krause, 1999). It is not always possible for experts to determine the structure of a Bayesian network. In some cases, the determination of the model can therefore be a problem to solve. Thus, in (Yu et al., 2002) learning the structure of a Bayesian network can be used to identify the most obvious relationships between different genetic regulators in order to guide subsequent experiments. The structure is then only a part of the solution to the problem but itself a solution.
Learning the structure of a Bayesian network may need to take into account the nature of the data provided for learning (or just the nature of the modelled domain): continuous variables— variables can take their values in a continuous space (Lauritzen & Wermuth,

1989; Lerner et al. 2001, Cobb & Shenoy, 2006) —, incomplete databases (Lauritzen, 1995; Heckerman, 1995). We assume in this work that the variables modelled take their values in a discrete set, they are fully observed, there is no latent variable i.e. there is no model in the field of non-observable variable that is the parent of two or more observed variables.
The methods used for learning the structure of a Bayesian network can be divided into two main groups:

1. Discovery of independence relationships: these methods consist in the testing procedures on allowing conditional independence to find a structure;
2. Exploration and evaluation: these methods use a score to evaluate the ability of the graph to recreate conditional independence within the model. A search algorithm will build a solution based on the value of the score and will make it evolve iteratively.

Without being exhaustive, belonging to the statistical test-based methods it should be noted first the algorithm PC, changing the algorithm SGS (Spirtes et al. 2001). In this approach, considering a graph $G\ (X, E, \theta)$, two vertices X_i and X_j from X and a subset of vertices $S_{Xi,Xj} \in X / \{X_i, X_j\}$, the vertices X_i and X_j are connected by an arc in G if there is no $S_{Xi,Xj}$ such as $(X_i \perp X_j \mid S_{Xi,Xj})$ where $\perp$ denotes the relation of conditional independence. Based on an undirected and fully connected graph, the detection of independence allows us to remove the corresponding arcs until the obtention the skeleton of the expected DAG. Then followed two distinct phases: i) detection and determination of the V-structures[1] of the graph and ii) orientation of the remaining arcs. The algorithm returns a directed graph belonging to the Markov's equivalence class of the sought model. The orientation of the arcs, except those of V-structures detected, does not necessarily correspond to the real causality of this model. In parallel to the algorithm PC, another algorithm, called IC (Inductive Causation) has been developed by the team of Judea Pearl (Pearl & Verma, 1991). This algorithm is similar to the algorithm PC, but starts with an empty structure and links couples of variables as soon as a conditional dependency is detected (in the sense that there is no identified subset conditioning $S_{Xi,Xj}$ such as $(X_i \perp X_j \mid S_{Xi,Xj})$. The common disadvantage to the two algorithms is the numerous tests required to detect conditional independences. Finally, the algorithm BNPC — Bayes Net Power Constructor — (Cheng et al., 2002) uses a quantitative analysis of mutual information between the variables in the studied field to build a structure G. Tests of conditional independence are equivalent to determine a threshold for mutual information (conditional or not) between couples of involved variables. In the latter case, a work (Chickering & Meek, 2003) comes to question the reliability of BNPC.
Many algorithms, by conducting casual research, are quite similar. These algorithms propose a gradual construction of the structure returned. However, we noticed some remaining shortcomings. In the presence of an insufficient number of cases describing the observed domain, the statistical tests of independence are not reliable enough. The number of tests to be independently carried out to cover all the variables is huge. An alternative is the use of a measure for evaluating the quality of a structure knowing the training database in combination with a heuristic exploring a space of options.
Scoring methods use a score to evaluate the consistency of the current structure with the probability distribution that generated the data. Thus, in (Cooper & Herskovits, 1992) a formulation was proposed, under certain conditions, to compute the Bayesian score,

[1] We call V-structure, or convergence, a triplet (x, y, z) such as y depends on x and z ($x \rightarrow y \leftarrow z$).

(denoted BD and corresponds in fact to the marginal likelihood we are trying to maximize through the determination of a structure G). In (Heckerman et al. 1995a) a variant of Bayesian score based on an assumption of equivalency of likelihood is presented. BDe, the resulting score, has the advantage of preventing a particular configuration of a variable X_i and of its parents $Pa(X_i)$ from being regarded as impossible. A variant, BDeu, initializes the prior probability distributions of parameters according to a uniform law. In (Kayaalp & Cooper, 2002) authors have shown that under certain conditions, this algorithm was able to detect arcs corresponding to low-weighted conditional dependencies. AIC, the Akaike Information Criterion (Akaike, 1970) tries to avoid the learning problems related to likelihood alone. When penalizing the complexity of the structures evaluated, the AIC criterion focuses the simplest model being the most expressive of extracted knowledge from the base D. AIC is not consistent with the dimension of the model, with the result that other alternatives have emerged, for example CAIC - Consistent AIC - (Bozdogan, 1987). If the size of the database is very small, it is generally preferable to use AICC - Akaike Information Corrected Criterion - (Hurvich & Tsai, 1989). The MDL criterion (Rissanen, 1978; Suzuki, 1996) incorporates a penalizing scheme for the structures which are too complex. It takes into account the complexity of the model and the complexity of encoding data related to this model. Finally, the BIC criterion (Bayesian Information Criterion), proposed in (Schwartz, 1978), is similar to the AIC criterion. Properties such as equivalence, breakdown-ability of the score and consistency are introduced. Due to its tendency to return the simplest models (Bouckaert, 1994), BIC is a metric evaluation as widely used as the BDeu score.

To efficiently go through the huge space of solutions, algorithms use heuristics. We can found in the literature deterministic ones like K2 (Cooper & Herskovits, 1992), GES (Chickering, 2002b), KES (Nielsen et al., 2003) or stochastic ones like an application of Monte Carlo Markov Chains methods (Madigan & York, 1995) for example. We particularly notice evolutionary methods applied to the training of a Bayesian network structure. Initial work is presented in (Larrañaga et al., 1996; Etxeberria et al., 1997). In this work, the structure is build using a genetic algorithm and with or without the knowledge of a topologically correct order on the variables of the network. In (Larrañaga et al., 1996) an evolutionary algorithm is used to conduct research over all topologic orders and then the K2 algorithm is used to train the model. Cotta and Muruzábal (Cotta & Muruzábal, 2002) emphasize the use of phenotypic operators instead of genotypic ones. The first one takes into account the expression of the individual's allele while the latter uses a purely random selection. In (Wong et al., 1999), structures are learned using the MDL criterion. Their algorithm, named MDLEP, does not require a crossover operator but is based on a succession of mutation operators. An advanced version of MDLEP named HEP (Hybrid Evolutionary Programming) was proposed (Wong et al., 2002). Based on a hybrid technique, it limits the search space by determining in advance a network skeleton by conducting a series of low-order tests of independence: if X and Y are independent variables, the arcs $X \rightarrow Y$ and $X \leftarrow Y$ can not be added by the mutation operator. The algorithm forbids the creation of a cycle during and after the mutation. In (Van Dijk et al., 2003a, Van Dijk et al., 2003b, Van Dijk & Thierens, 2004) a similar method was proposed. The chromosome contains all the arcs of the network, and three alleles are defined: *none*, $X \rightarrow Y$ and $X \leftarrow Y$. The algorithm acts as Wong's one (Wong et al., 2002) but only recombination and repair are used to make the individuals evolve. The results presented in (Van Dijk & Thierens, 2004) are slightly better than these obtained by HEP. A search, directly done in the equivalence graph space, is presented in (Muruzábal & Cotta, 2004, Muruzábal & Cotta, 2007). Another approach, where the algorithm works in the limited partially directed acyclic graph is reported in (Acid & De Campos, 2003). These are a special form of PDAG where many of these could fit the same equivalence class. Finally, approaches such as Estimation of Distribution Algorithms (EDA) are applied in (Mühlenbein & Paab, 1996). In (Blanco et al., 2003), the authors have implemented two approaches (UMDA and PBIL) to search structures over the PDAG space. These algorithms were applied to the distribution of arcs in the adjacency matrix of the expected structure. The results appear to support the approach PBIL. In (Romero et al., 2004), two approaches (UMDA and MIMIC) have been applied to the topological orders space. Individuals (i.e. topological orders candidates) are themselves evaluated with the Bayesian scoring.

2.5 Our contribution

For the training of the structure of a Bayesian network with a score function and without prior knowledge like the topology of the structure sought, one often use a greedy search algorithm over the space of structures or in the equivalence classes. But these methods have the disadvantage of being frequently trapped into a solution corresponding to a local optimum of the evaluation function. This is due to the presence of many local optima in space solutions. The smaller the training base is the numerous the optima are. The main reason for a premature convergence is that a greedy algorithm considers, at each moment, only one solution. The search stops if there is no better evaluated solution around a given point. The most widespread technique to avoid this is to use multiple initialization of the greedy algorithm, from very different initial structures and keep the best solution obtained. This technique has the disadvantage to dramatically increase the computing time but also offer no guarantee of obtaining x distinct solutions for x different initialization of the algorithm.

Evolutionary algorithms have two major advantages when processing a problem with many local optima. On the one hand, they allow us to maintain a population of solutions, i.e. several points in the space of solutions. With the maintenance and development of alternatives it becomes possible to reduce the chances of being trapped in a single locally optimum. On the other hand, stochastic behaviour of these methods through the mutation operator can amplify the robustness to local optima attraction (conditionally on the use of parameters and adapted operators) by allowing an exploration of the solutions area which is no longer limited to the immediate neighbourhood of individuals in the population.

3. Genetic algorithm design

Genetic algorithms are a family of computational models inspired by Darwin's theory of Evolution. Genetic algorithms encode potential solutions to a problem in a chromosome-like data structure, exploring and exploiting the search space using dedicated operators. Their actual form is mainly issued from the work of J.Holland (Holland, 1992) in which we can find the general scheme of a genetic algorithm (see Fig. 2) called *canonical GA*. Throughout the years, different strategies and operators have been developed in order to perform an efficient search over the considered space of individuals: selection, mutation and crossing operators, etc.

```
/* Initialization*/
t ← 0;
Randomly and uniformly generate an initial population $P_0$ of $\lambda$ individuals and
evaluate them using a fitness function $f$
/* Evolution */
Select $P_t$ individuals for the reproduction
Build new individuals by application of the crossing operator on the
beforehand selected individuals
Apply a mutation operator to the new individuals: individuals obtained are
affected to the new population $P_{t+1}$
/* Evaluation */
Evaluate the individuals of $P_{t+1}$ using $f$
t ← t + 1
/* Stop */
If a definite criterion is met then stop else start again the evolution phase
```

Fig. 2. Holland's canonical genetic algorithm (Holland, 1992)

Applied to the search for Bayesian networks structures, genetic algorithm pose two problems:

- The constraint on the absence of circuits in the structures creates a strong link between the different genes — and alleles — of a person, regardless of the chosen representation. Ideally, operators should reflect this property;
- Often, a heuristic searching over the space of solutions (genetic algorithm, greedy algorithm and so on.) finds itself trapped in a local optimum. This makes it difficult to find a balance between a technique able to avoid this problem, with the risk of overlooking many quality solutions, and a more careful exploration with a good chance to compute only a locally-optimal solution.

If the first item involves essentially the design of a thoughtful and evolutionary approach to the problem, the second point characterizes an issue relating to the multimodal optimization. For this kind of problem, there is a particular methodology: the niching.

We now proceed to a description of a genetic algorithm adapted to find a good structure for a Bayesian network.

3.1 Representation

As our search is performed over the space of directed acyclic graphs, each invidual is represented by an adjacency matrix. Denoting with N the number of variables in the domain, an individual is thus described by an $N \times N$ binary matrix Adj_{ij} where one of its coefficients a_{ij} is equal to 1 if an oriented arc going from X_i to X_j in G exists.

Whereas the traditional genetic algorithm considers chromosomes defined by a binary alphabet, we chose to model the Bayesian network structure by a chain of N genes (where N is the number of variables in the network). Each gene represents one row of the adjacency matrix, that's to say each gene corresponds to the set of parents of one variable. Although this non-binary encoding is unusual in the domain of structure learning, it is not an uncommon practice among genetic algorithms. In fact, this approach turns out to be especially practical for the manipulation and evaluation of candidate solutions.

3.2 Fitness function

We chose to use the Bayesian Information Criterion (BIC) score as the fitness function for our algorithm:

$$S_{BIC}(B,D)=\log\left(L(D|B,\theta^{MAP})\right)-\frac{1}{2}Dim(B)\times\log(N) \tag{2}$$

where D represents the training data, θ^{MAP} the MAP-estimated parameters, and $Dim()$ is the dimension function defined by Eq. 3:

$$Dim(B)=\sum_{i=1}^{n}(r_i-1)\times\prod_{X_k\in Pa(X_i)}r_k \tag{3}$$

where r_i is the number of possible values for X_i. The fitness function $f(individual)$ can be written as in Eq. 4:

$$f(individual)=\sum_{k=1}^{n}f_k(X_k,Pa(X_k)) \tag{4}$$

where f_k is the local BIC score computed over the family of variable X_k.
The genetic algorithm takes advantage of the breakdown of the evaluation function and evaluates new individuals from their inception, through crossing, mutation or repair. The impact of any change on local an individual's genome shall be immediately passed on to the phenotype of it through the computing of the local score. The direct consequence is that the evaluation phase of the generated population took actually place for each individual, depending on the changes made, as a result of changes endured by him.

3.3 Seting up the population

We choose to initialize the population of structures by the various trees (depending on the chosen root vertice) returned by the MWST algorithm. Although these n trees are Markov-equivalent, the initialization can generate individuals with relevant characteristics. Moreover, since early generations, the combined action of the crossover and the mutation operators provides various and good quality individuals in order to significantly improve the convergence time. We use the undirected tree returned by the algorithm: each individual of the population is initialized by a tree directed from a randomly-chosen root. This mechanism introduces some diversity in the population.

3.4 Selection of the individuals

We use a rank selection where each one of the λ individuals in the population is selected with a probability equal to:

$$P_{select}(individual)=2\times\frac{\lambda+1-rank(individual)}{\lambda\times(\lambda+1)} \tag{5}$$

This strategy allows promote individuals which best suit the problem while leaving the weakest one the opportunity to participate to the evolution process. If the major drawback of this method is to require a systematic classification of individuals in advance, the cost is negligible. Other common strategies have been evaluated without success: the roulette wheel (prematured convergence), the tournament (the selection pressure remained too strong) and the fitness scaling (Forrest, 1985; Kreinovich et al., 1993). The latter aims to allow in the first instance to prevent the phenomenon of predominance of "super individuals" in the early generations while ensuring when the population converges, that the mid-quality individuals did not hamper the reproduction of the best ones.

3.5 Repair operator

In order to preserve the closeness of our operators over the space of directed acyclic graphs, we need to design a repair operator to convert those invalid graphs (typically, cyclic directed graphs) into valid directed acyclic graphs. When one cycle is detected within a graph, the operator suppresses the one arc in the cycle bearing the weakest mutual information. The mutual information between two variables is defined as in (Chow & Liu, 1968):

$$W(X_A, X_B) = \sum_{x_a, x_b} \frac{N_{ab}}{N} \log\left(\frac{N_{ab}N}{N_a N_b}\right) \tag{6}$$

Where the mutual information $W(X_A,X_B)$ between two variables X_A and X_B is calculated according to the number of times N_{ab} that $X_A=a$ and $X_B=b$, N_a the number of times $X_A=a$ and so on. The mutual information is computed once for a given database. It may happen that an individual has several circuits, as a result of a mutation that generated and/or inverted several arcs. In this case, the repair is iteratively performed, starting with deleting the shortest circuit until the entire circuit has been deleted.

3.6 Crossover operator

A first attempt was to create a one-point crossover operator. At least, the operator used has been developed from the model of (Vekaria & Clack, 1998). This operator is used to generate two individuals with the particularity of defining the crossing point as a function of the quality of the individual. The form taken by the criterion (BIC and, in general, by any decomposable score) makes it possible to assign a local score to the set $\{X_i, Pa(X_i)\}$. Using these different local scores we can therefore choose to generate an individual which received the best elements of his ancestors. This operation is shown Fig. 3.
This generation can be performed only if a DAG is produced (the operator is closed). In our experiments, P_{cross}, the probability that an individual is crossed with another is set to 0.8.

3.7 Mutation operator

Each node of one individual has a P_{mute} probability to either lose or gain one parent or to see one of its incoming arcs reverted (i.e. reversing the relationship with one parent).

3.8 Other parameters

The five best individuals from the previous population are automatically transferred to the next one. The rest of the population at $t+1$ is composed of the $S-5$ best children where S is the size of the population.
Now, after describing our basic GA, we will present how it can be improved by i) a specific adaptive mutation scheme and ii) an exploration strategy: the niching.

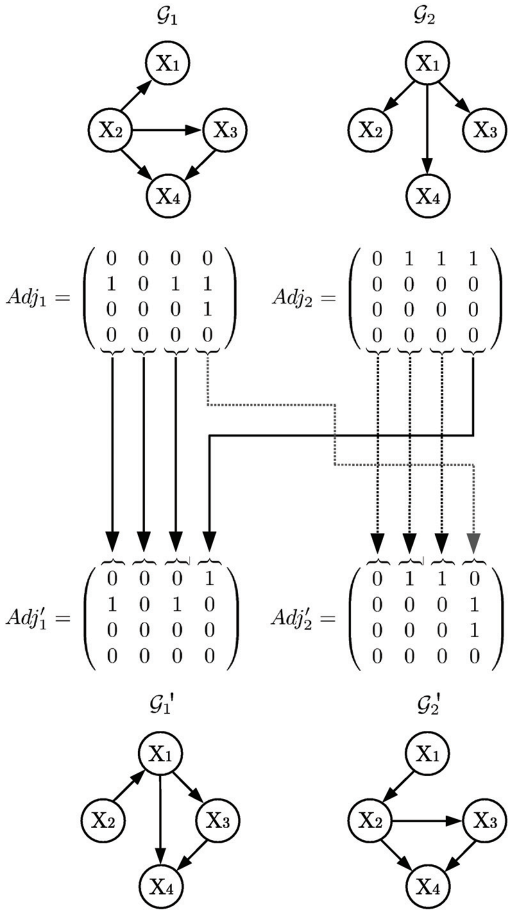

$\mathcal{G}_1$
$\mathcal{G}_2$
X1
X2
X3
X4
$Adj_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$
$Adj_2 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$
$Adj'_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$
$Adj'_2 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$
$\mathcal{G}_1'$
$\mathcal{G}_2'$

Fig. 3. The crossover operator and the transformation it performs over two DAGs.

4. Strategies

The many parameters of a GA are usually fixed by the user and, unfortunately, usually lead to sub-optimal choices. As the amount of tests required to evaluate all the conceivable sets of parameters will be eventually exponential, a natural approach consists in letting the different parameters evolve along with the algorithm. (Eiben et al., 1999) defines a terminology for self-adaptiveness which can be resumed as follows:

- Deterministic Parameter Control: the parameters are modified by a deterministic rule;
- Adaptive Parameter Control: consists in modifying the parameters using feedback from the search;
- Self-adaptive Parameter Control: parameters are encoded in the individuals and evolve along.

We now present three techniques. The first one, an adaptive parameter control, aims at managing the mutation rate. The second one, an evolutionary method tries to avoid local optima using a penalizing scheme. Finaly, the third one, another evolutionary method, makes many populations evolve granting sometimes a few individuals to go from one population to another.

4.1 Self-adaptive scheme of the mutation rate

As for the mutation rate, the usual approach consists in starting with a high mutation rate and reducing it as the population converges. Indeed, as the population clusters near one optimum, high mutation rates tend to be degrading. In this case, a self-adaptive strategy would naturally decrease the mutation rate of individuals so that they would be more likely to undergo the minor changes required to reach the optimum.

However, applying this kind of policy can do more harm than good. When there are many local optima, as in our case, we can be confronted with the *bowl effect* described in (Glickman & Sycara, 2000). That is: when the population is clustered around a local optimum and the mutation rate is too low to allow at least one individual to escape this local optimum, a strictly decrementing adaptive policy will only trap the population around this optimum.

Other strategies have been proposed which allow the individual mutation rates to either increase or decrease, such as in (Thierens, 2002). There, the mutation step of one individual induces three differently rated mutations: greater, equal and smaller than the individual's actual rate. The issued individual and its mutation rate are chosen accordingly to the qualitative results of the three mutations. Unfortunately, as the mutation process is the most costly operation in our algorithm, we obviously cannot choose such a strategy. Therefore, we designed two adaptive policies.

The first one is given Fig. 4:

```
At each mutation process, given one individual $I$, its fitness value $f(I)$ and its
mutation rate $P_{m,\omega} < 1$, $\gamma > 1$:
    1. Mutate individual $I$ according to its mutation rate $P_m$: $(I,P_m) \rightarrow (I')$
    2. If $f(I') > f(I)$: allocate mutation rate $\omega \times P_m$ to individual $I'$ and $\gamma \times P_m$
       to individual $I$,
    3. If $f(I') \leq f(I)$: allocate mutation rate $\gamma \times P_m$ to individual $I'$ and $\omega \times P_m$ to
       individual $I$
```

Fig. 4. Basic adaptive mutation rate scheme.

This principle is based on the fact that, during an evolution-based process, the less fit individuals have the best chances to produce new, fitter individuals. Our scheme is based on the idea of maximizing the mutation rate of less fit individuals while reducing the mutation rate of the fitter. However, in order to control the computational complexity of the algorithm as well as to leave to the best individuals the possibility to explore their neighbourhood, we define a maximum threshold $Mute_{max}$ and a minimum threshold $Mute_{min}$ for the mutation rate of all individuals. Since we also apply an elitist strategy, we added a deterministic rule in order to control the mutation rate of the best individuals: At the end of each iteration multipliy the mutation rates of the best D individuals by ω where D is the degree of our elitist policy.

An improvement of this approach is now proposed. Indeed, the computed probability concerns all the possible mutation operations. But, perharps some could be benefits, others none. So we propose to conduct the search over the space of solutions by taking into account information on the quality of later searchs. Our goal is to define a probability distribution which drives the choice of the mutation operation. This distribution should reflect the performance of the mutation operations being applied over the individuals during the previous iterations of the search.

Let us define $P(i,j,op_{mute})$ the probability that the coefficient a_{ij} of the adjacency matrix is modified by the mutation operation opmute. The mutation decays according to the choice of *i, j* and *opmute*. We can simplify the density of probability by conditionning a subset of $\{i,j,op_{mute}\}$ by its complementary; this latter being activated according to a static distribution of probability. After studying all the possible combination, we have chosen to design a process to control $P(i\,|\,op_{mute},j)$. This one influences the choice of the source vertex knowing the destination vertex and for a given mutation operation. So the mutation operator can be rewritten such as shown by Fig. 5.

```
for j = 1 to $n$ do
    if $Pa(X_j)$ mute with a probability $P_{mute}$ then
        choose a mutation operation among these allowed on $Pa(X_j)$
        apply $op_{mute}(i, j)$ with the probability $P(i\,|\,op_{mute}, j)$
    end if
end for
```

Fig. 5. The mutation operator scheme

Assuming that the selection probability of $Pa(X_j)$ is uniformly distributed and equals a given P_{mute}, Eq. 7 must be verified:

$$\begin{cases} \sum_{op_{mute}} \delta^{(i,j)}_{op_{mute}} P(i|op_{mute}, j) = 1 \\ \delta^{(i,j)}_{op_{mute}} = \begin{cases} 1 \text{ if } op_{mute}(i,j) \text{ is allowed} \\ O \text{ else} \end{cases} \end{cases} \tag{7}$$

The diversity of the individuals lay down to compute $P(i\,|\,op_{mute},j)$ for each allowed op_{mute} and for each individual X_j. We introduce a set of coefficients denoted $\zeta(i,j,op_{mute}(i,j))$ where $1\leq i,j\leq n$ and $i\neq j$ to control $P(i\,|\,op_{mute},j)$. So we define:

$$P(i|op_{mute}, j) = \frac{\zeta(i,j,op_{mute}(i,j))}{\sum \delta^{(i,j)}_{op_{mute}} \zeta(i,j,op_{mute}(i,j))} \tag{8}$$

During the initialisation and without any prior knowledge, $\zeta(i,j,op_{mute}(i,j))$ follows an uniform distribution:

$$\zeta(i,j,op_{mute}(i,j)) = \frac{1}{n-1} \quad \begin{cases} \forall 1 \leq i,j \leq n \\ \forall\, op_{mute} \end{cases} \tag{9}$$

Finally, to avoid the predominance of a given op_{mute} (probability set to 1) and a total lack of a given op_{mute} (probability set to 0) we add a constraint given by Eq.10:

$$0.01 \leq \zeta(i,j,op_{mute}(i,j)) \leq 0.9 \quad \begin{cases} \forall 1 \leq i,j \leq n \\ \forall\, op_{mute} \end{cases} \tag{10}$$

Now, to modify $\zeta(i,j,op_{mute}(i,j))$ we must take in account the quality of the mutations and either their frequencies. After each evolution phase, the $\zeta(i,j,op_{mute}(i,j))$ associated to the op_{mute}

applied at least one time are reestimated. This compute is made according to a parameter γ which quantifies the modification range of $\zeta(i,j,op_{mute}(i,j))$ and depends on ω which is computed as the number of successful applications of op_{mute} minus the number of detrimental ones in the current population. Eq.11 gives the computation. In this relation, if we set γ=0 the algorithm acts as the basic genetic algorithm previoulsy defined.

$$\zeta(i,j,op_{mute}(i,j)) \leftarrow \begin{cases} \min(\zeta(i,j,op_{mute}(i,j)) \times (1-\gamma)^{\omega}, 0.9) \text{ if } \omega > 0 \\ \max(\zeta(i,j,op_{mute}(i,j)) \times (1-\gamma)^{\omega}, 0.01) \text{ else} \end{cases} \quad (11)$$

The regular update $\zeta(i,j,op_{mute}(i,j))$ leads to standardize the $P(i \mid op_{mute}, j)$ values and avoids a prematured convergence of the algorithm as seen in (Glickman & Sycara, 2000) in which the mutation probability is strictly decreasing. Our approach is different from an EDA one: we drive the evolution by influencing the mutation operator when an EDA makes the best individuals features probability distribution evolve until then generated.

4.2 Niching

Niching methods appear to be a valuable choice for learning the structure of a Bayesian network because they are well-adapted to multi-modal optimization problem. Two kind of niching techniques could be encountered: spatial ones and temporal ones. They all have in common the definition of a distance which is used to define the niches. In (Mahfoud, 1995), it seemed to be expressed a global consensus about performance: spatial approch gives better results than temporal one. But the latter is easier to implement because it consists in the addition of a penalizing scheme to a given evolutionnary method.

4.2.1 Sequential niching

So we propose two algorithms. The first one is apparented to a sequential niching. It makes a similar trend to that of a classic genetic algorithm (iterated cycles evaluation, selection, crossover, mutation and replacement of individuals) except for the fact that a list of optima is maintained. Individuals matching these optima see their fitness deteriorated to discourage any inspection and maintenance of these individuals in the future.

The local optima, in the context of our method, correspond to the equivalence classes in the meaning of Markov. When at least one equivalence class has been labelled as corresponding to an optimum value of the fitness, the various individuals in the population belonging to this optimum saw the value of their fitness deteriorated to discourage any further use of these parts of the space of solutions. The determination of whether or not an individual belongs to a class of equivalence of the list occurs during the evaluation phase, after generation by crossover and mutation of the new population. The graph equivalent of each new individual is then calculated and compared with those contained in the list of optima. If a match is determined, then the individual sees his fitness penalized and set to at an arbitrary value (very low, lower than the score of the empty structure).

The equivalence classes identified by the list are determined during the course of the algorithm: if, after a predetermined number of iterations Ite_{opt}, there is no improvement of the fitness of the best individual, the algorithm retrieves the graph equivalent of the equivalence class of it and adds it to the list.

It is important to note here that the local optima are not formally banned in the population. The registered optima may well reappear in our population due to a crossover. The evaluation of these equivalence classes began, in fact until the end of a period of change; an optimum previously memorized may well reappear at the end of the crossover operation and the individual concerned undergo mutation allowing to explore the neighbourhood of the optimum.

The authors of (Beasley et al., 1993) carry out an evolutionary process reset after each determination of an optimum. Our algorithm continues the evolution considering the updated list of these optima. However, by allowing the people to move in the neighbourhood of the detected optima, we seek to preserve the various building blocks hitherto found, as well as reducing the number of evaluations required by multiple launches of the algorithm.

At the meeting of a stopping criterion, the genetic algorithm completes its execution thus returning the list of previously determined optima. The stopping criterion of the algorithm can also be viewed in different ways, for example:

- After a fixed number of local optima detected;
- After a fixed number of iterations (generations).

We opt for the second option. Choosing a fixed number of local optima may, in fact, appear to be a much more arbitrary choice as the number of iterations. Depending on the problem under consideration and/or data learning, the number of local optima in which the evolutionary process may vary. The algorithm returns a directed acyclic graph corresponding to the instantiation of the graph equivalent attached to the highest score in the list of optima.

An important parameter of the algorithm is, at first glance, the threshold beyond which an individual is identified as qu'optimum of the evaluation function. It is necessary to define a value of this parameter, which we call Ite_{opt} that is:

- Neither too small: take it too hasty a class of equity as a local optimum hamper space exploration research of the genetic algorithm, and it amalga over too many optima;
- Nor too high: loss of the benefit of the method staying too long in the same point in space research: the local optima actually impede the progress of the research.

Experience has taught us that Ite_{opt} value of between 15 and 25 iterations can get good results. The value of the required parameter Ite_{opt} seems to be fairly stable as it allows both to stay a short time around the same optimum while allowing solutions to converge around it. The value of the penalty imposed on equivalence classes is arbitrary. The only constraint is that the value is lowered when assessing the optimum detected is lower than the worst possible structure, for example: -10^{15}.

4.2.2 Sequential and spatial niching combined

The second algorithm uses the same approach as for the sequential niching combined with a technique used in parallels GAs to split the population. We use an island model approach for our distributed algorithm. This model is inspired from a model used in genetic of populations (Wright, 1964). In this model, the population is distributed to k islands. Each island can exchange individuals with others avoiding the uniformization of the genome of the individuals. The goals of all of this is to preserve (or to introduce) genetic diversity.

Some additional parameters are required to control this second algorithm. First, we denote I_{mig} the migration interval, i.e. the number if iteration of the GA between two migration phases. Then, we use R_{mig} the migration rate: the rate of individuals selected for a migration.

N_{isl} is the number of islands and finaly I_{size} represents the number of individuals in each island.
In order to remember the local optima encountered by the populations, we follow the next process:

- The population of each island evolves during I_{mig} iterations and then transfert $R_{mig} \times I_{size}$ individuals
- Local optima detected in a given island are registered in a shared list. Then they can be known by all the islands.

5. Evaluation and discussion

From an experimental point of view, the training of the structure of a Bayesian network consists in:

- to have an input database containing examples of instantiation of the variables
- to determine the conditional relationship between the variables of the model
    - Either from statistical tests performed on several subsets of variables;
    - Either from measurements of a match between a given solution and the training database
- to compare the learned structures to determine the respective qualities of the different algorithms used

5.1 Tested methods

So that we can compare with existing methods, we used some of the most-used learning methods: the K2 algorithm, the greedy algorithm applied to the structures space, denoted GS; the greedy algorithm applied to the graph equivalent space, noted GES; the MWST algorithm, the PC algorithm. These methods are compared to our four evolutionary algorithms learning: the simple genetic algorithm (GA); genetic algorithm combined with a strategy of sequential niching (GA-SN); the hybrid sequential-spatial genetic approach (GA-HN); the genetic algorithm with the dynamic adaptive mutation scheme GA-AM.

5.2 The Bayesian networks used

We apply the various algorithms in search of some common structures like: Insurance (Binder et al., 1997) consisting of 27 variables and 52 arcs; ALARM (Beinlich et al. 1989) consisting of 37 variables and 46 arcs. We use each of these networks to summarize:

- Four training data sets for each network, each one containing a number of databases of the same size (250, 500, 1000 & 2000 samples);
- A single and large database (20000 or 30000 samples) for each network. This one is supposed to be sufficiently representative of the conditional dependencies of the network it comes from.

All these data sets is obtained by logic probabilistic sampling (Henrion, 1988): the value of vertices with no predecessors is randomly set, according to the probability distributions of the guenine network, and then the remaining variables are sampled following the same principle, taking into account the values of the parent vertices. We use several training databases for a given network and for a given number of cases, in order to reduce any bias due to sampling error. Indeed, in the case of small databases, it is possible (and it is common) that the extracted statistics are not exactly the conditional dependencies in the guenine network. After training with small databases, the BIC score of the returned structures by the different methods are computed from the large database mentioned earlier, in order to assess qualitative measures.

5.3 Experiments

GAs: The parameters of the evolutionary algorithms are given in Table 1.

Parameter	**Value**	**Remarks**
Population size	150	
Mutation probability	$1/n$	
Crossover probability	0.8	
Recombination scheme	elitist	The best solution is never lost
Stop criterion	1000 iter.	
Initialisation		See footnote[2]
Ite_{opt}	20	For GA-SN only
γ	0.5	For D1-GA & GA-AM
I_{mig}	20	For GA-HN only
R_{mig}	0.1	For GA-HN only
N_{isl}	30	For GA-HN only
I_{size}	30	For GA-HN only

Table 1. Parameters used for the evolutionary algorithms.

GS: This algorihtm is initialized with a tree returned by the MWST method, where the root vertice is randomly chosen.
GES: This algorithm is initialized with the empty structure.
MWST: it is initialized with a root node randomly selected (it had no effect on the score of the structure obtained).
K2: This algorithm requires a topological order on the vertices of the graph. We used for this purpose two types of initialization:

- The topological order of a tree returned by the MWST algorithm (method K2-T);
- A topological order random (method K2-R).

For each instance of K2-R — i.e. for each training database considered — we are proceeding with $5 \times n$ random initialization for choosing only those returning the best BIC score.
Some of these values (crossover, mutation probability) are coming from some habits of the domain (Bäck, 1993) but especially from experiments too. The choice of the iteration number is therefore sufficient to monitor and interpret the performance of the method considered while avoiding a number of assessments distorting the comparison of results with greedy methods.
We evaluate the quality of the solutions with two criteria: the BIC score from one hand, and a graphic distance measuring the number of differences between two graphs on the other

[2] The populations of the evolutionary methods are all initialized like GS. We make sure, however, that each vertice will be selected at least once as root.

hand. The latter is defined from 4 terms: (D) the total number of different arcs between two graphs G_1 and G_2, (+) the number of arcs existing in G_1 but not in G_2, (-) the number of arcs existing in G_2 but not in G_1 and (inv) the number of arcs inverted in G_1 comparing to G_2. These terms are important because, when considering two graphs of the same equivalence class, some arcs could be inverted. This implies that the corresponding arcs are not oriented in the corresponding PDAG. The consequence is that G_1 and G_2 have the same BIC score but not the same graphic distance. To compare the results with we also give the score of the empty structure G_0 and the score of the reference network G_R.

5.4 Results for the INSURANCE network

Results are given Table 2 & Table 3. The evaluation is averaged over 30 databases. Table 2 shows the means and the standard deviations of the BIC scores. For a better seeing, values are all divided by 10. Values labelled by † are significantly different from the best mean score (Mann-Whitney's test).

The results in Table 2 give an advantage to evolutionary methods. While it is impossible to distinguish clearly the performance of the different evolutionary methods, it is interesting to note that these latter generally outperform algorithms like GES and GS. Only the algorithm GS has such good results as the evolutionary methods on small databases (250 and 500). We can notice too, according to a Mann-Whitney's test that, for large datasets, GA-SN & GA-AM returns a structure close to the original one. Standard deviations are not very large for the GAs, showing a relative stability of the algorithms and so, a good avoidance of local optima.

	250	500	1000	2000
GA	−32135 ± 290	−31200 ± 333	**−29584** ± 359	−28841 ± 89†
GA-SN	−31917 ± 286	−31099 ± 282	−29766 ± 492	**−28681**±156
GA-HN	−31958±246	**−31075** ± 255	**−29428** ± 290	−28715 ± 164
GA-AM	**−31826**±270	−31076 ± 151	−29635 ± 261	−28688 ± 165
GS	−32227 ± 397	−31217 ± 314	−29789 ± 225†	−28865 ± 151†
GES	−33572 ± 247†	−31952 ± 273†	−30448 ± 836†	−29255 ± 634†
K2-T	−32334 ± 489†	−31772 ± 339†	−30322 ± 337†	−29248 ± 163†
K2-R	−33002 ± 489†	−31858 ± 395†	−29866 ± 281†	−29320 ± 245†
MWST	−34045 ± 141†	−33791 ± 519†	−33744 ± 296†	−33717 ± 254†
G_R	−28353			
G_0	−45614			

Table 2. Means and standard deviations of the BIC scores (INSURANCE).

Table 3 shows the mean structural differences between the original network and these delivered by some learning algorithms. There, we can see that evolutionary methods, particularly GA-SN, return the structures which are the closest to the original one. This network was chosen because it contains numerous low-valued conditional probabilities. These are difficult to find using small databases. So even if the BIC score is rather close to the original one, graphical distances reveals some differences. First, we can see that *D* is rather high (the original network G_R is made with only 52 arcs, compared to D which minimum is 24.4) even if the BIC score is very close (resp. -28353 compared to -28681). Second, as expected, *D* decreases when the size of the learning database grows, mainly because of the (-) term. Third, GAs obtains the closest models to the original in 11 cases over 16; the 5 others are provided by GES.

	250				500			
	D	+	inv	-	D	+	inv	-
GA	39.6	4.4	7.2	28	34	3.1	7.6	23.3
GA-SN	**37**	3.5	7.1	26.4	35.1	3.7	7.4	24
GA-HN	38.1	3.5	7.5	27.1	**33.3**	3	7.3	23
GA-AM	37.5	4.3	6.6	26.6	33.9	3.2	7.7	23
GS	42.1	4.6	9.4	28.1	37.7	4.5	9.4	23.8
GES	39.5	3.7	7.1	28.7	35.1	3	7.1	25
K2-T	42.7	5.1	8.4	29.2	40.8	5.4	8.8	26.6
K2-R	42.4	4.8	7.2	30.4	41.8	6.5	8.8	26.6
MWST	41.7	4	7.7	30	41.3	3.5	8.3	29.5
	1000				**2000**			
	D	+	inv	-	D	+	inv	-
GA	39.6	4.4	7.2	28	27.8	4.7	8	15.1
GA-SN	30.8	3.8	7.4	19.6	**24,4**	3.4	6.7	14.3
GA-HN	**29.3**	3.6	6.5	19.2	26.6	3.6	8.6	14.4
GA-AM	31.4	4	8	19.4	27	4.3	8.4	14.3
GS	35.9	5.1	10	20.8	31.9	5.2	11.4	15.3
GES	32.4	4.1	8.1	20.2	27.5	4	8.4	15.1
K2-T	38.7	5.9	11	21.8	34.6	7.3	10.9	16.4
K2-R	39.6	8.3	8.3	23	36.1	8.5	8.5	9.1
MWST	37.7	1.7	8.3	27.7	36.3	1.2	7.9	27.2

Table 3. Mean structural differences between the original INSURANCE network and the best solutions founded by some algorithms

5.5 Results for the ALARM network

The results are shown Table 4 & Table 5. This network contains more vertices than the INSURANCE one, but less low-valued arcs. The evaluation is averaged over 30 databases. Table 4 shows that evolutionary algorithms obtain the best scores. But while GES provides less qualitative solutions accordingly to the BIC score, these solutions are closest to the original one if we consider the graphical distance. Here, a strategy consisting in gradually building a solution seems to produce better structures than an evolutionary search. In this case, a GA has a huge space (3×10^{237} when applying the Robinson's formula) into which one it enumerates solutions. If we increases the size of the population the results are better than these provided by GES.

	250	500	1000	2000
GA	−36239 ± 335	−34815 ± 317	−33839 ± 159	−33722 ± 204†
GA-SN	**−36094**±297	−34863 ± 346	−33865 ± 203	−33640 ± 196†
GA-HN	−36144 ± 326	−34864 ± 337	**−33723 ± 251**	**−33496 ± 170**
GA-AM	−36104 ± 316	**−34791**±340	−33942 ± 198†	−33722 ± 204†
GS	−36301 ± 309†	−35049 ± 380†	−33839 ± 109†	−33638 ± 964†
GES	−36124 ± 315	−34834 ± 288	−33801 ± 562†	−33593 ± 692†
K2-T	−36615 ± 308†	−35637 ± 328†	−34427 ± 200†	−34045 ± 818†
K2-R	−37173 ± 435†	−35756 ± 264†	−34579 ± 305†	−34128 ± 173†
MWST	−37531 ± 185†	−37294 ± 737†	−37218 ± 425†	−37207 ± 366†
G_R	−33097			
G_0	−63113			

Table 4. Means and standard deviations of the BIC scores (ALARM).

	250				500			
	D	+	inv	-	D	+	inv	-
GA	34.2	4.8	13.9	15.5	25.7	4.5	10.2	11
GA-SN	33.1	4.6	13.5	15	25.6	4.2	10.6	10.8
GA-HN	33.6	4.6	13.8	15.2	25.1	3.7	10.7	10.7
GA-AM	33	4.6	13.4	15	26.2	4	11.5	10.7
GS	33.7	5	12.6	16.1	30.2	5	13.5	11.7
GES	**32.5**	4.5	12.7	15.3	**23.3**	3.8	8	11.5
K2-T	34.5	5.1	13.1	16.3	35.1	7.2	15.2	12.7
K2-R	36.5	6.6	10.2	19.6	35	8.7	11.3	11.5
MWST	38.5	6.9	14.7	16.9	36.5	4.7	17.1	14.7
	1000				2000			
	D	+	inv	-	D	+	inv	-
GA	19.7	3.7	9	6.9	23	5.3	11.8	5.9
GA-SN	22	4.5	10.4	7.1	20.1	4.1	10.2	5.8
GA-HN	**18.3**	3.3	10.1	4.9	18.9	3.6	9	6.3
GA-AM	27	6.4	13.1	7.4	29	7.4	16	6.3
GS	27.8	6.2	14.5	7.1	25.4	6.2	13.6	5.6
GES	20.2	4.3	8.5	7.3	**17.3**	3.5	8.2	5.6
K2-T	35.4	10.4	15.7	9.3	36.9	12.3	17.4	7.2
K2-R	37.1	11.4	15.1	10.6	40.2	14.6	16.1	9.5
MWST	35.1	4.4	16.3	14.4	34.1	14	16.1	14

Table 5. Mean structural differences between the original ALARM network and the best solutions founded by some algorithms

5.5 Behaviour of the GAs

Now look at some measures in order to evaluate the behaviour of our genetic algorithms.
A repair operator was designed to avoid individuals having a cycle. Statistics computed during the tests show that the rate of individuals repaired does not seem to depend neither on the algorithm used nor and on the size of the training set. It seems to be directly related to the complexity of the network. Thus, this rate is about 15% for the INSURANCE network and about 7% for the ALARM network.
The mean number of iterations before the GA found the best solution returned for the INSURANCE network is given Table 6. The data obtained for the ALARM network are the same order of magnitude. We note here that GA-HN quickly gets the best solution. This makes it competitive in terms of computing time if we could detect this event.

	250	500	1000	2000
GA	364	454	425	555
GA-AM	704	605	694	723
GA-SN	398	414	526	501
GA-HN	82	106	166	116

Table 6. Mean of the necessary number of iterations to find the best structure (INSURANCE).

The averaged computing time of each algorithm is given Table 7 (for the ALARM network). We note here that GA-HN is only three times slower than GES. We note too that these computing times are rather stable when the size of the database increases.

	250	500	1000	2000
GA	3593 ± 47	3659 ± 41	3871 ± 53	4088 ± 180
GA-AM	3843 ± 58	3877 ± 44	4051 ± 59	4332 ± 78
GA-SN	3875 ± 32	4005 ± 43	4481 ± 46	4834 ± 52
GA-HN	9118 ± 269	9179 ± 285	9026 ± 236	9214 ± 244
GS	9040 ± 1866	9503 ± 1555	12283 ± 1403	16216 ± 2192
GES	3112 ± 321	2762 ± 166	4055 ± 3.4	5759 ± 420
K2-T	733 ± 9	855 ± 25	1011 ± 14	1184 ± 8
K2-R	3734 ± 61	4368 ± 152	5019 ± 67	5982 ± 43
MWST	10 ± 1	10 ± 2	11 ± 1	12 ± 1

Table 7. Averaged computing times (in seconds) and standard deviations (ALARM).

An example of the evolution of the fitness of the population is given Fig. 6. The curves for GA, GA-SN and GA-AM are very similar. The curve associated with GA-HN increases through levels, a consequence of spatial niching policy who promptly exchange some individuals between islands. Although the average quality is progressing more slowly, it is revealed fairly quickly, however, better than in other genetic algorithms. Although the curve corresponding to the algorithm GA seems well placed, Tables 4 and 5 learn us a bit more. First, the score is not considered equivalent: the algorithm GA-HN have the best one. Second, the graphical distance of GA-HN is the lowest. Although GA-SN seems more remote, the results presented in Tables 4 and 5 show that the BIC score obtained by GA-SN is closer to the optimal, and the editing distance of GA-SN is better than the GA one.

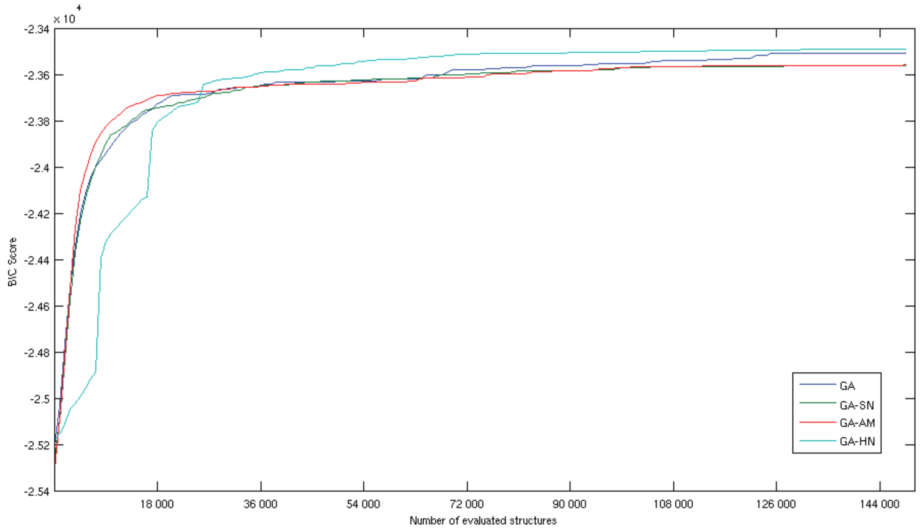


Fig. 6. Evolution of the individuals' fitness (ALARM, 2000 training samples).

6. Future search

We will continue the development of the hybrid niching technique. The first step is the distribution over a cluster of computers. Then we plan to develop new strategies implying a global behaviour like in GA-HN and a dynamic mutation scheme like this one used in GA-AM. The next goal will be the definition of a stopping criterion based on population's statistics to make our algorithm competitive in term of computing time.

7. Conclusion

We have presented three methods for learning the structure of a Bayesian network. The first one consists in the control of the probability distribution of mutation in the genetic algorithm. The second one is to incorporate a scheme penalty in the genetic algorithm so that it avoids certain areas of space research. The third method is to search through several competing populations and to allow timely exchange among these populations. We have shown experimentally that different algorithms behaved satisfactorily, in particular that they were proving to be successful on large databases. We also examined the behaviour of proposed algorithms. Niching strategies are interesting, especially using the spatial one, which focuses quickly on the best solutions.

8. Acknowledgements

This work was realized using Matlab and two toolboxes dedicated to Bayesian networks manipulation: Bayesian Net Toolbox from K. P. Murphy (Murphy, 2001) and Structure Learning Package (SLP) from P. Leray & O. François (Francois & Leray, 2004).

9. References

Acid, S. & De Campos, L.M. (2003). Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, Vol. 18, 05/03, 445-490, 11076-9757

Akaike, H. (1970). Statistical predictor identification. *Annals of the Institute of Statistical Mathematics*, Vol. 22, No. 1, 12/70, 203–217, 0020-3157

Allanach, J.; Tu, H.; Singh, S.; Pattipati, K. & Willett, P. (2004). Detecting, tracking and counteracting terrorist networks via hidden markov models, *Proceedings of IEEE Aerospace Conference*, pp. 3257, 0-7803-8155-6, 03/2004

Bäck, T. (1993). Optimal mutation rates in genetic search, *Proceedings of International Conference on Genetic Algorithms*, pp. 2-8, 1-55860-299-2, San Mateo (CA), Morgan Kaufmann

Beasley, D.; Bull, D.R.; & Martin, R.R. (1993). A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, Vol. 1, No. 2, 101–125, 1063-6560

Beinlich, I.A.; Suermondt, H.J.; Chavez, R.M. & Cooper, G.F. (1989). The alarm monitoring system : A case study with two probabilistic inference techniques for belief networks, *Proceedings of European Conference on Artificial Intelligence in Medicine*, pp. 247–256, London, Springer-Verlag, Berlin

Binder, J.; Koller, D.; Russell, S.J. & Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, Vol. 29, No. 2-3, 11/97, 213–244, 0885-6125

Blanco, R. ; Inza, I. ; & Larrañaga, P. (2003). Learning bayesian networks in the space of structures by estimation of distribution algorithms. *International Journal of Intelligent Systems*, Vol. 18, No. 2, 205–220, 0884-8173

Bouckaert, R. (1994). Properties of bayesian belief network learning algorithms, *Proceedings of Uncertainty in Artificial Intelligence*, pp. 102–10, Morgan Kaufmann, San Francisco (CA)

Bozdogan, H. (1987). Model selection and Akaike's information criteria (AIC): The general theory and its analytical extentions. *Psychometrika*, Vol. 52, 354–370, 0033-3123

Cano, R.; Sordo, C.; & Gutiérrez, J. (2004). Applications of Bayesian Networks in Meteorology, In *Advances in Bayesian Networks*, Gámez J.A, Moral S. & Salmerón A. (Eds.), 309-327, Springer, 3540208763

Cheng, J.; Bell, D.A. & Liu, W. (2002). Learning belief networks from data: An information theory based approach. *Artificial Intelligence*, Vol. 137, No. 1-2, 43–90

Chickering, D.M. (2002b). Learning equivalence classes of Bayesian network structures. *Journal of Machine Learning Research*, Vol. 2, 03/02, 445–498, 1532-4435

Chickering, D.M. & Meek, C. (2003). Monotone DAG faithfulness : A bad assumption. Technical Report MSR-TR-2003-16, Microsoft Research

Chickering, D.M. (2002a). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, Vol. 3, 03/03, 507–554, 1532-4435

Chickering, D.M.; Geiger, D. & Heckerman, D. (1994). Learning Bayesian networks is np-hard. Technical Report MSR-TR-94-17, Microsoft Research

Chow, C. & Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, Vol. 14, No. 3, 05/68, 462-467, 0018-9448

Cobb, B.R. & Shenoy, P.P. (2006). Inference in hybrid bayesian networks with mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, Vol. 41, No. 3, 04/06, 257–286, 0888-613X

Cooper, G. & Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, Vol. 9, No. 4, 10/92, 309–347, 0885-6125

Cotta, C. & Muruzábal, J. (2002). Towards a more efficient evolutionary induction of bayesian networks, *Proceedings of Parallel Problem Solving from Nature*, pp. 730-739, Granada, 09/2002

Davis, G.A. (2003) Bayesian reconstruction of traffic accidents, *Law, Probability and Risk*, Vol. 2, No. 2, 69-89, 1470-8396

De Jong, K. (2006). *Evolutionary Computation: A Unified Approach*, MIT Press, 0262041944

Eiben, A.E.; Hinterding, R. & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 2, 124-141, 1089-778X

Etxeberria, R.; Larrañaga, P. & Picaza, J.M. (1997). Analysis of the behaviour of genetic algorithms when learning bayesian network structure from data. *Pattern Recognition Letters*, Vol. 18, No. 11-13, 11/97, 1269–1273, 0167-8655

Ezawa, K.J. & Schuermann, T. (1995) Fraud/Uncollectible Debt Detection Using a Bayesian Network Based Learning System: A Rare Binary Outcome with Mixed Data Structures, *Proceedings of Uncertainty in Artificial Intelligence*, pp.157-16, Morgan Kaufmann, San Francisco (CA)

Fennell, M.T. & Wishner, R.P. (1998). Battlefield awareness via synergistic SAR and MTI exploitation. *IEEE Aerospace and Electronic Systems Magazine*, Vol. 13, No. 2, 39-43, 0885-8985

Forrest, S. (1985). Documentation for prisoner's dilemma and norms programs that use the genetic algorithm, Technical Report, Univ. of Michigan.

Francois, O. & Leray, P. (2004). BNT structure learning package: documentation and experiments, Technical Report, Univ. of Rouen (France).

Glickman, M. & Sycara, K. (2000). Reasons for premature convergence of self-adapting mutation rates, *Proceedings of Evolutionary Computation*, pp. 62 – 69, 07/2000

Heckerman, D. (1995a). A tutorial on learning bayesian networks, Technical Report MSR-TR-95-06, Microsoft Research

Heckerman, D.; Mamdani, A. & Wellman, M.P. (1995b). Real world applications of bayesian networks. *Communications of the ACM*, Vol. 38, No. 3, 03/95, 24-30, 0001-0782

Henrion, M. (1988). Propagation of uncertainty by probabilistic logic sampling in bayes networks. *Proceedings of Uncertainty in Artificial Intelligence*, pp. 149–164, Morgan Kaufmann, San Francisco (CA)

Holland, J.H. (1992). *Adaptation in natural and artificial systems*, The MIT Press, 0262581116

Horvitz, E.; Breese, J.; Heckerman, D.; Hovel, D. & Rommelse, K. (1998) The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users, *Proceedings of Uncertainty in Artificial Intelligence*, 07/1998, Morgan Kaufmann, San Francisco (CA).

Hurvich, C.M. & Tsai, C.-L. (1989). Regression and time series model selection in small samples. *Biometrika*, Vol. 76, No. 2, 297-307, 0006-3444

Jaronski, W.; Bloemer, J.; Vanhoof, K. & Wets, G. (2001). Use of bayesian belief networks to help understand online audience, Proceedings of ECML/PKDD, 09/2001, Freiburg, Germany.

Kayaalp, M. & Cooper, G.F. (2002). A bayesian network scoring metric that is based on globally uniform parameter priors, *Proceedings of Uncertainty in Artificial Intelligence*, pp. 251-258, Morgan Kaufmann, San Francisco (CA)

Krause, P.J. (1999). Learning probabilistic networks. *The Knowledge Engineering Review*, Vol. 13, No. 4, 321–351, 0269-8889

Kreinovich, V., Quintana, C. & Fuentes, O. (1993). Genetic algorithms: What fitness scaling is optimal ? *Cybernetics and Systems*, Vol. 24, No. 1, 9–26, 0196-9722

Lacey, G. & MacNamara, S. (2000). Context-aware shared control of a robot mobility aid for the elderly blind. *International Journal of Robotic Research*, Vol. 19, No. 11, 1054-1065, 0278-3649

Larranãga, P.; Poza, M.; Yurramendi, Y.; Murga, R. & Kuijpers. C. (1996). Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions PAMI*, Vol. 18, No. 9, 912–926, 0162-8828

Lauritzen, S.L. (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics & Data Analysis*, Vol. 19, No. 2, 191 201, 0167-9473

Lauritzen, S.L. & Wermuth, N. (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, Vol. 17, No. 1, 31-57, 0090-5364

Lerner, U. ; Segal, E. & Koller, D. (2001). Exact inference in networks with discrete children of continuous parents, *Proceedings of Uncertainty in Artificial Intelligence*, pp. 319-32, Morgan Kaufmann, San Francisco (CA)

Madigan, D. & York, J. (1995). Bayesian graphical models for discrete data. *International Statistical Review*, Vol. 63, No. 2, 215–232, 03067734

Mahfoud, S.W. (1995). Niching methods for genetic algorithms. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA. IlliGAL Report 95001

Mühlenbein, H. & Paab, G. (1996). From recombination of genes to the estimation of distributions, Proceedings of Parallel Solving from Nature, pp. 178–187.

Murphy, K. (2001). The Bayes net toolbox for matlab. *Computing Science and Statistics*, Vol. 33, 331-350

Muruzábal, J. & Cotta, C. (2007). A study on the evolution of bayesian network graph structures. *Studies in Fuzziness and Soft Computing*, Vol. 213, 193-214, 1434-9922.

Muruzábal, J. & Cotta, C. (2004). A primer on the evolution of equivalence classes of Bayesian network structures, *Proceedings of Parallel Problem Solving from Nature*, pp. 612–621, Birmingham, 09/2004

Nielsen, J.D.; Kocka, T. & Peña, J.M. (2003). On local optima in learning bayesian networks, *Proceedings of Uncertainty in Artificial Intelligence*, pp. 435–442, Acapulco, Morgan Kaufmann, San Francisco (CA)

Pearl, J. & Verma, T.S. (1991). A theory of inferred causation, *In: Principles of Knowledge Representation and Reasoning*, Allen J.F., Fikes R.& Sandewall, E. (Eds.), pp. 441-452, Morgan Kaufmann, San Francisco (CA)

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 0934613737, San Francisco (CA)

Rissanen, J. (1978). Modelling by shortest data description. *Automatica*, Vol. 14, 465–471, 0005-1098

Robinson, R. (1976). Counting unlabeled acyclic digraphs, *Proceedings of Combinatorial Mathematics*, pp. 28–43

Romero, T. ; Larrañaga, P. & Sierra, B. (2004). Learning bayesian networks in the space of orderings with estimation of distribution algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 18, No. 4, 607–625, 0218-0014

Sahami, M.; Dumais, S.; Heckerman, D. & Horvitz, E. (1998). A bayesian approach to filtering junk e-mail, *Proceedings of the AAAI-98 Workshop on Text Categorization*, pp.55-62, Madison (WI), 07/1998, AAAI Press.

Schwartz, G. (1978). Estimating the dimensions of a model. *The Annals of Statistics*, Vol. 6, No. 2, 461–464, 0090-5364

Spirtes, P.; Glymour, C. & Scheines, R. (2001). Causation, Prediction and Search, The MIT Press, 0262194406,

Suzuki, J. (1996). Learning bayesian belief networks based on the minimum description length principle: An efficient algorithm using the B&B technique, *Proceedings of International Conference on Machine Learning*, pp. 462-470

Thierens, D. (2002). Adaptive mutation rate control schemes in genetic algorithms, Technical Report UU-CS-2002-056, Utrecht University

Van Dijk, S. & Thierens, D. (2004). On the use of a non-redundant encoding for learning bayesian networks from data with a GA, *Proceedings of Parallel Problem Solving from Nature*, pp. 141–150, Birmingham, 09/2004

Van Dijk, S., Thierens, D., & Van Der Gaag, L.C. (2003a). Building a GA from design principles for learning bayesian networks, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 886–897.

Van Dijk, S., Van Der Gaag, L.C. & Thierens, D. (2003b). A skeleton-based approach to learning bayesian networks from data, *Proceedings of Principles and Practice of Knowledge Discovery in Databases*, pp. 132–143, Cavtat-Dubrovnik, 09/2003

Vekaria, K. & Clack, C. (1998). Selective crossover in genetic algorithms: An empirical study, *Proceedings of Parallel Problem Solving from Nature*, pp. 438-447, Amsterdam, 09/1998

Wong, M., Lee, S.Y. & Leung, K.S. (2002). A hybrid data-mining approach to discover bayesian networks using evolutionary programming, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 214-222.

Wong, M.L., Lam,W., & Leung, K.S. (1999). Using evolutionary programming and minimum description length principle for data mining of bayesian networks. *IEEE Transactions PAMI*, Vol. 21, No. 2, 174-178, 0162-8828

Wright, S. (1964). Stochastic processes in evolution, In *Stochastic models in medecine and biology*, Gurland, J. (Ed.), 199-241, Univ. of Wisconsin Press, Madison

Yu, J.; Smith, V.A.; Wang, P.P.; Hartemink, A.J. & Jarvis., E.D. (2002). Using bayesian network inference algorithms to recover molecular genetic regulatory networks, *Proceedings of International Conference on Systems Biology*

18

Design of Phased Antenna Arrays using Evolutionary Optimization Techniques

Marco A. Panduro[1], David H. Covarrubias[2] and Aldo L. Mendez[1]
[1]Reynosa-Rodhe Multidisciplinary Academic Center,
Universidad Autónoma de Tamaulipas
[2]CICESE Research Center, Tijuana
México

1. Introduction

Mobile and wireless communication systems have now arrived at the point where substancial advances in antenna technology have become a critical issue. The majority of these systems consist of an antenna array combined with an appropriate signal processing (Soni et al., 2002; Godara, 2002), i.e., the antenna elements are allowed to work in concert by means of array element phasing, which is accomplished with hardware or is performed digitally.
In these systems, the antenna array performance over a certain steering range is of primary concern. In this case, the antenna array design problem consists of finding weights that make the radiation pattern satisfy the desired characteristics (a maximum directivity, a minimum side lobe level, etc), so the direction of the main beam can be steered at will.
Generally, the design problem is formulated as an optimization problem. The design of antenna arrays has a nonlinear and non-convex dependence of elements parameters [Kurup et al. 2003], because of that, the interest has been focused on stochastic search techniques, such as simulated annealing (Murino et al., 1996), and mainly, genetic algorithms (GA's) (Ares-Pena et al., 1999; Haupt, 1994; Haupt, 1995; Panduro et al., 2005; Rahmat-Samii et al, 1999; Weile et al., 1997; Yan et al., 1997), widely used in electromagnetic problems, including the synthesis of phased antenna arrays (Mailloux, 2005; Hansen, 1998).
The antenna arrays optimization for improving performance represents an open line of research in the antenna design field. In the application of evolutionary optimization techniques for designing antenna arrays, it has been considered the design of different phased array structures, such as the linear arrays (Bray et al., 2002; Panduro, 2006) and the circular arrays (Panduro et al., 2006), among others. The design of planar arrays is dealt with in (Bae et al., 2005). In many design cases, it has been considered the optimization in the design of scannable arrays with non-uniform separation (Bray et al., 2002; Bae et al., 2004; Junker et al., 1998; Tian et al., 2005; Lommi et al., 2002).
In this chapter it is considered the case of designing scannable arrays with the optimization of the amplitude and phase excitations for maximum side lobe level reduction in a wide scanning range.
The purpose of this chapter is to investigate the behavior of the radiation pattern for the design of different phased array structures (linear and circular arrays) considering the

optimization of the amplitude and phase excitation across the antenna elements, by using the well-known method of Genetic Algorithms. Due to the great variety of parameters involved, optimization techniques such as Genetic Algorithms are very appropriate tools to search for the best antenna array models.

The primary focus of this chapter is to present a study of the application of GA techniques to the design of scannable linear and circular arrays in a uniform geometry considering the optimization of the amplitude and phase excitation across the antenna elements. This study considers the design of scannable linear and circular arrays to be a problem optimizing a simple objective function. This objective function considers the synthesis of the radiation diagram with desired characteristics of the side lobe level and the directivity in a wide steering range. The contribution of this work is to present a model for the design of scannable linear and circular arrays that includes the synthesis of the radiation diagram using the method of genetic algorithms.

The remainder of this chapter is organized as follows. Section 2 states the design of phased linear arrays. A description of the objective function used by the genetic algorithm and the obtained results for this design problem are presented in this section. Following the same design philosophy, the design of phased circular arrays is presented in the section 3. Discussions and open problems are presented in the section 4. Finally, the summary and conclusions of this work are presented in the section 5.

2. Design of phased linear arrays

The design of scannable linear arrays has been dealt with in many papers. In these documents, the study has been focused mainly to design scannable linear arrays with non-uniform separation (Bray et al., 2002; Bae et al., 2004; Junker et al., 1998; Tian et al., 2005; Lommi et al., 2002; Panduro et al., 2005)., i.e, the performance of the array is improved, in the sense of the side lobe level, optimizing the spacing between antenna elements. In this case, it is presented the design of scannable linear arrays optimizing the amplitude and phase excitations across the antenna elements. It is believed by the authors that the performance of the array could be improved substantially, with respect to the linear array with the conventional progressive phase excitation, if the amplitude and phase excitations are set or optimized in an adequate way. Next, it is presented the theoretical model for the design of scannable linear arrays.

2.1 Theoretical model

Consider a scannable linear array with N antenna elements uniformly spaced, as shown in figure 1. If the elements in the linear array are taken to be isotropic sources, the radiation pattern of this array can be described by its array factor (Stutzman, 1998). The array factor for a conventional linear array in the *x-y* plane is given by (Balanis, 2005)

$$AF(\theta,\mathbf{I})=\sum_{n=1}^{N} I_n \exp\left\{j\left[kd_n\left(\cos\theta-\cos\theta_0\right)\right]\right\} \tag{1}$$

In this case, the array factor for a linear array with phase excitation is created by adding in the appropriate element phase perturbations, $\mathbf{P}$ = [$\delta\beta_1$, $\delta\beta_2$, ..., $\delta\beta_N$], $\delta\beta_i$ represents the phase perturbation of the *i*th element of the array, such that

$$AF(\theta,\mathbf{I},\mathbf{P})=\sum_{n=1}^{N} I_n \exp\left\{j\left[\psi_n+\delta\beta_n\right]\right\}. \quad (2)$$

In these equations, $\mathbf{I} = [I_1, I_2, ..., I_N]$, I_i represents the amplitude excitation of the *i*th element of the array, $\psi_n=kd_n\cos\theta_0$, θ_0 is the direction of maximum radiation, $k = 2\pi/\lambda$ is the phase constant and θ is the angle of incidence of a plane wave, λ is the signal wavelength.

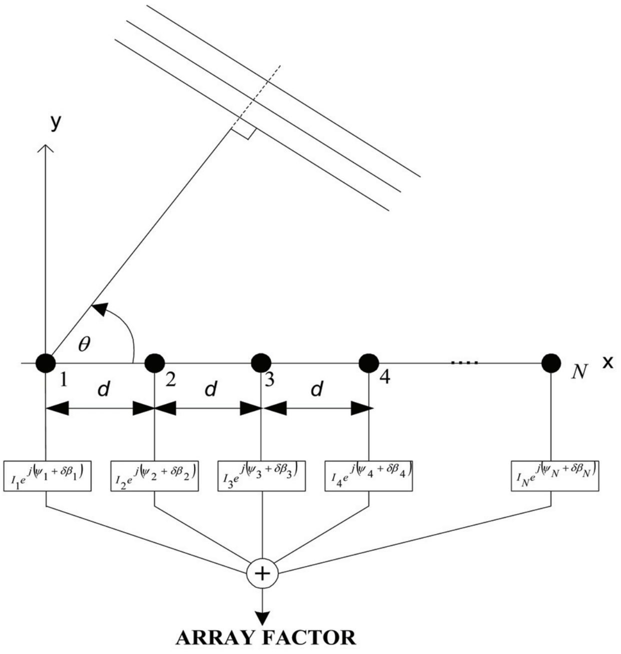

y
x
θ
1
2
3
4
N
d
d
d
....
$I_1e^{j(\psi_1+\delta\beta_1)}$
$I_2e^{j(\psi_2+\delta\beta_2)}$
$I_3e^{j(\psi_3+\delta\beta_3)}$
$I_4e^{j(\psi_4+\delta\beta_4)}$
$I_Ne^{j(\psi_N+\delta\beta_N)}$
+
ARRAY FACTOR

Figure 1. Steerable linear array with antenna elements uniformly spaced.

The idea of adding perturbations in the conventional array factor is that the optimization algorithm searches possible optimal phase excitations in angles near the direction of desired maximum gain. The optimization process developed in this paper for generating arrays that have radiation patterns with low side lobe level will be based on (2).
We now need to formulate the objective function we want to optimize.

2.2 Objective function used to optimize the design of linear arrays.

The objective function is the driving force behind the GA (Goldberg, 1989). It is called from the GA to determine the fitness of each solution string generated during the search. In this case, each solution string represents possible amplitude excitations and phase perturbations of antenna elements. As already being pointed out, the objective of the present study is to evaluate the radiation pattern of scannable linear arrays in a uniform geometry considering the optimization of the amplitude and phase excitation across the antenna elements. In this case, it is studied the behavior of the array factor for the scanning range of $50°\leq\theta_0\leq130°$ with an angular step of 10°. In order to calculate the objective function of an individual, the procedure described below is followed.

1. A set of 1000 points is used to specify a desired radiation pattern with direction of maximum gain in each angle of the scanning range. Each point represents the *i*th desired normalized radiation pattern value.
2. An individual is generated by the GA (amplitude excitations and phase perturbations of antenna elements). Each individual is in general represented by a vector of real numbers, i.e., $\mathbf{I} = [I_1, I_2, ..., I_N]$, and a vector of real numbers restrained on the range $(0, 2\pi)$, i.e., $\mathbf{P} = [\delta\beta_1, \delta\beta_2, ..., \delta\beta_N]$.
3. The value of the objective function is calculated as

$$of = (|\mathrm{AF}(\theta_{MSL}, \mathbf{I}, \mathbf{P})| \,/\, max|\mathrm{AF}(\theta, \mathbf{I}, \mathbf{P})|) + (1/DIR(\theta, \mathbf{I}, \mathbf{P})) \tag{3}$$

where θ_{MSL} is the angle where the maximum side lobe is attained, and DIR the directivity for the radiation pattern. In this case, the design problem is formulated as minimize the objective function *of*.
4. A random population of individuals is generated and the genetic mechanisms of crossover, survival and mutation are used to obtain better and better individuals, until the GA converges to the best solution or the desired goals are achieved.

The results of using a GA for the design of scannable linear arrays are described in the next section.

2.3 Results obtained for the design of phased linear arrays

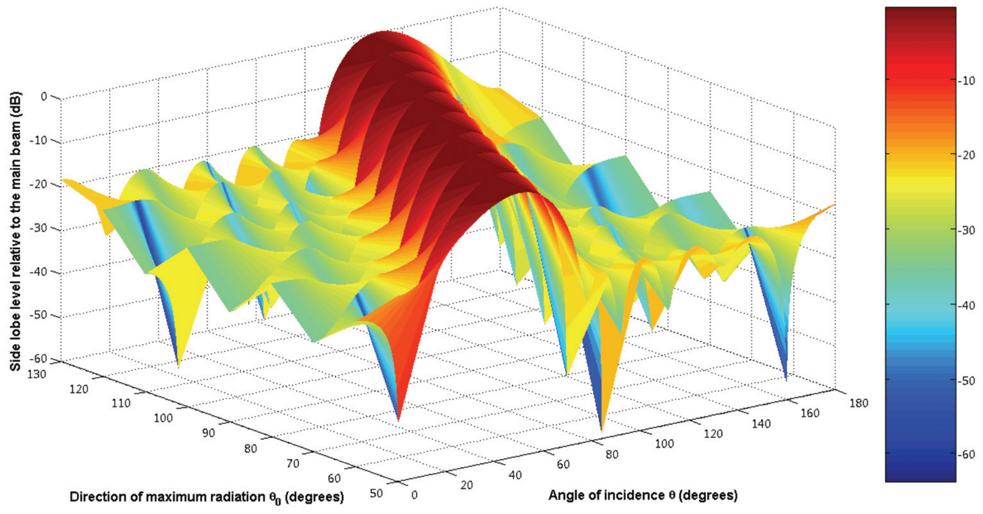
The method of Genetic Algorithms was implemented to study the behavior of the radiation pattern for scannable linear arrays. In this case, it is studied the behavior of the array factor for the scanning range of $50° \leq \theta_0 \leq 130°$. Several experiments were carried out with different number of antenna elements. In the experiments the algorithm parameters, after a trial and error procedure, were set as follows: maximum number of generations *rmax* = 500, population size *gsize* = 200, crossover probability *pc* = 1.0 and mutation probability *pm* = 0.1. A selection scheme combining Fitness Ranking and Elitist Selection (Goldberg, 1989) was implemented instead of a common weighted roulette wheel selection. The used genetic operators are standard: the well known two point crossover (Goldberg, 1989) along with a single mutation where a locus is randomly selected and the allele is replaced by a random number uniformly distributed in the feasible region. The obtained results are explained below.

Figure 2 shows the behavior of the radiation pattern for a scannable linear array with the amplitude and phase excitation optimized by the GA. The separation between antenna elements is set as $d=0.5\lambda$. In this case, we illustrate the examples for a) $N=6$, b) $N=8$, c) $N=12$.

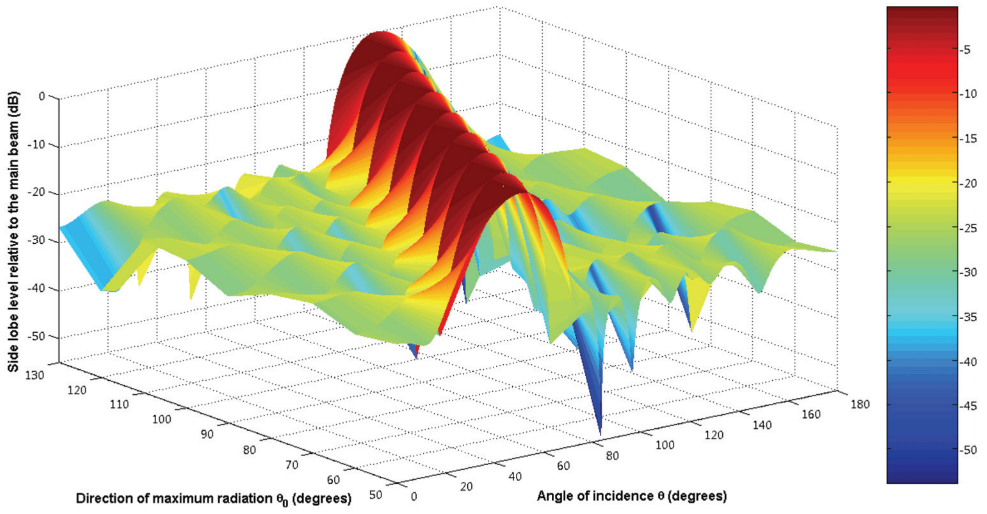
As shown in the examples of the Figure 2, the Genetic Algorithm generates a set of amplitude and phase excitations in each angle of the scanning range to provide a normalized array factor with a side lobe level < -20 dB in the steering range. The optimization of the array can maintain a low side lobe level without pattern distortion during beam steering.

Numerical values of the side lobe level, directivity, amplitude and phase perturbation distributions for the array factor illustrated in Figure 2 are presented in the Table 1.

Table 1 illustrates that the design case with the amplitude and phase optimized by the GA could provide a better performance in the side lobe level with respect to the conventional case. These low values of the side lobe level for the optimized design case could be achieved with very similar values of directivity and the same aperture in both design cases.

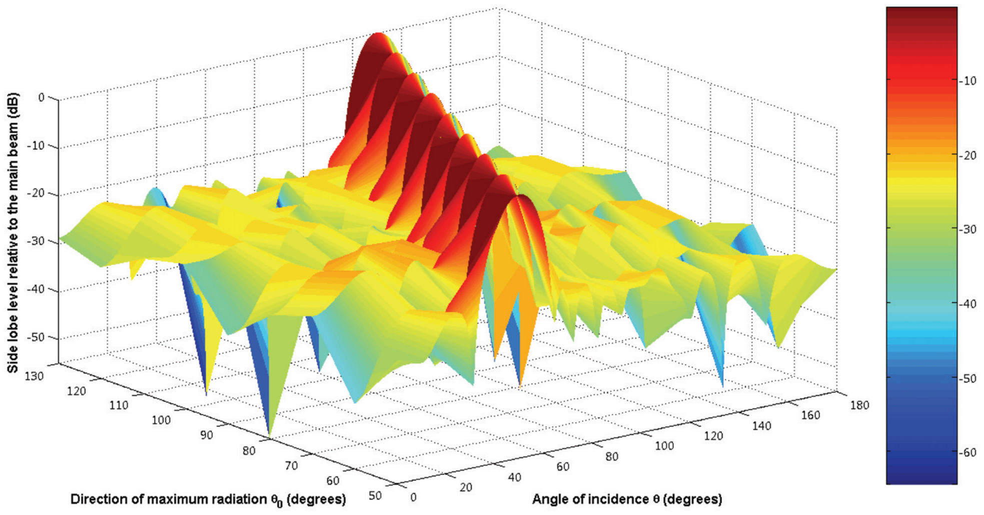


(a)



(b)

Figure 2. Behavior of the radiation pattern for a scannable linear array in a steering range of $50° \leq \theta_0 \leq 130°$ with the amplitude and phase excitation optimized by the GA, a) N=6, b) N=8, c) N=12.



(c)

Figure 2. (continued).

Design case with the amplitude and phase excitation optimized by the GA (N=6)					Conventional case	
θ_0	*SLL* (dB)	*DIR* (dB)	Normalised amplitude distribution	Phase perturbation distribution (deg)	*SLL* (dB)	*DIR* (dB)
50°	-20.57	5.95	5.2812 8.3264 11.1717 10.071 7.8738 5.0515	93.46, 103.56, 100.2, 102.06, 103.95, 94.22	-12.42	6.41
60°	-22.97	6.54	4.6352 8.5603 11.8683 11.6959 9.0046 5.0082	87.23, 92.48, 89.33, 90.30, 89.66, 89.65	-12.42	6.73
70°	-21.95	7.01	4.861 8.0166 11.481 11.6839 9.3365 5.5832	52.80, 58.69, 60.12, 60.05, 56.70, 53.96	-12.42	7.32
80°	-23.01	7.09	4.2881 8.753 11.427 11.4502 8.1468 4.6648	81.17, 73.84, 70.18, 71.08, 69.03, 83.96	-12.42	7.39
90°	-24.08	7.15	4.4101 8.6824 11.8247 11.9992 9.0169 4.3549	108.1, 114.48, 115.7, 114.46, 117.9, 107.2	-12.42	7.60
100°	-23.13	7.16	4.2918 7.6439 10.8717 11.7129 9.058 5.2613	90.28, 91.94, 95.46, 95.45, 93.59, 94.37	-12.42	7.39
110°	-23.49	6.86	4.1223 7.0177 10.4515 11.5757 9.0148 4.8233	75.35, 64.63, 60.83, 61.18, 64.72, 74.03	-12.42	7.32
120°	-21.13	6.60	4.5337 7.1531 9.9966 10.2654 7.6541 4.8863	83.29, 87.44, 94.70, 94.38, 87.42, 83.89	-12.42	6.73
130°	-20.14	6.03	5.3514 8.16 10.6545 10.117 7.8955 5.3136	89.32, 84.89, 89.40, 87.97, 84.96, 87.70	-12.42	6.41

(a)

Table 1. Numerical values of the side lobe level (*SLL*), directivity (*DIR*), amplitude and phase perturbation distribution for the array factor illustrated in Fig. 2, a) N=6, b) N=8, c) N=12.

Design case with the amplitude and phase excitation optimized by the GA (*N*=8)					**Conventional case**	
θ_0	***SLL* (dB)**	***DIR* (dB)**	**Normalised Amplitude distribution**	**Phase perturbation distribution (deg)**	***SLL* (dB)**	***DIR* (dB)**
50°	-23.37	7.22	4.087 4.9365 8.5532 10.2015 10.799 9.7949 7.5327 4.0191	100.41, 97.92, 103.90, 98.9, 101.2, 105.25, 101.56, 105.16	-12.79	7.48
60°	-21.34	7.90	4.2851 8.0845 9.094 11.5372 11.8051 10.633 7.121 5.8245	107.84, 95.16, 102.25, 105.2, 100.63, 105.05, 92.13, 105.7	-12.79	8.25
70°	-23.12	8.22	4.5748 6.4426 9.221 11.8394 11.792 10.3245 7.1393 4.827	66.60, 60.46, 59.03, 62.87, 63.14, 62.06, 62.34, 65.21	-12.79	8.51
80°	-22.20	8.43	4.5666 7.6476 9.81 11.5279 10.9044 9.513 6.7872 4.23	102.55, 104.3, 115.14, 103.7, 106.7, 109.5, 104, 94.6	-12.79	8.76
90°	-22.36	8.54	4.4301 7.6973 9.0958 10.86 10.8595 9.2315 5.927 4.6922	114.10, 102.9, 104.58, 99.58, 98.06, 104.4, 99, 111.62	-12.79	8.89
100°	-21.71	8.41	4.6048 6.5117 9.263 11.1713 11.982 9.9245 8.7118 4.8284	52.6, 76.44, 68.51, 77.12, 72.34, 73.01, 73.56, 57.24	-12.79	8.76
110°	-22.32	8.18	4.6991 7.3316 10.1467 11.8 11.00 9.1493 5.9517 4.2841	56.36, 53.19, 59.30, 47.77, 43.77, 54.19, 46.66, 53.58	-12.79	8.51
120°	-20.79	7.89	4.57 6.0843 9.1019 10.2557 9.7068 7.6138 6.6535 4.0621	105.23, 92.6, 104.36, 90.73, 95.37, 103.52, 92.73, 109.056	-12.79	8.25
130°	-23.37	7.25	4.141 6.4183 8.7271 11.4705 11.2 9.9689 7.0337 4.3883	101.21, 99.28, 102.08, 97.16, 97.26, 97.17, 98.01, 98.06	-12.79	7.48

(b)

Design case with the amplitude and phase excitation optimized by the GA (*N*=12)					**Conventional case**	
θ_0	***SLL* (dB)**	***DIR* (dB)**	**Normalised Amplitude distribution**	**Phase perturbation distribution (deg)**	***SLL* (dB)**	***DIR* (dB)**
50°	-21.43	9.20	5.834 5.819 7.759 8.9656 11.427 11.96 11.109 11.0136 10.232 7.295 6.952 4.6	97.6, 105.9, 101.5, 110, 90.5, 102.12, 95.96, 100.8, 102.54, 102, 103.1, 101.34	-13.05	9.48
60°	-23.56	9.58	4.001 6.1168 8.7636 10.192 11.69 11.563 11.6994 11.0176 8.62 6.31 4.908 4.13	88.59, 84.56, 86.26, 96.48, 87.56, 90.73, 86.58, 85.11, 102.20, 83.35, 94.8, 83.9	-13.05	10.05
70°	-19.08	10.01	5.971 7.8776 7.052 11.7027 10.658 10.886 11.539 10.817 10.136 8.486 6.636 6.371	67.94, 88.3, 85.7, 105.95, 98.62, 87.08, 92.3, 103.06, 99.9, 88.22, 96.42, 64.1	-13.05	10.41
80°	-19.11	10.23	7.3289 6.4067 9.1657 9.121 9.5598 11.419 11.164 9.1777 8.9068 6.812 6.353 4.022	17.87, 23.62, 11.35, 38.56, 19.02, 10.57, 17.68, 33.39, 19.07, 21.61, 32.24, 359.9	-13.05	10.62
90°	-23.77	10.18	4.3008 4.705 7.9 9.1098 11.2055 11.325 10.077 9.723 7.3412 5.32 4.784 4.0317	86.99, 94.23, 92.56, 93.22, 98.9, 101.5, 106.6, 103.37, 96.13, 92.3, 89.9, 86.84	-13.05	10.69
100°	-22.33	10.13	4.683 5.0336 8.207 10.2406 11.407 11.664 11.9685 10.84 8.61 6.992 5.307 5.076	95.2, 99.54, 104.57, 92.94, 104.3, 89.63, 87.54, 99.45, 89.9, 103.15, 97.5, 100.9	-13.05	10.62
110°	-22.38	10.01	4.748 6.2578 7.7439 8.9026 10.5646 11.617 11.78 10.328 10.562 6.653 6.363 4.356	84.30, 94.71, 79.67, 86.31, 73.17, 92.29, 82.51, 78.79, 84.01, 81.04, 87.27, 90.22	-13.05	10.41
120°	-21.25	9.67	4.5847 6.208 7.4537 9.5612 11.112 11.413 10.195 10.736 9.0616 7.4537 4.866 4.806	86.35, 70.13, 70.10, 71.82, 74.06, 81.96, 68.87, 82.64, 72.91, 66.77, 70.65, 88.11	-13.05	10.05
130°	-22.59	9.15	4.9722 5.9917 6.805 9.83 11.2548 11.3448 11.99 10.71 9.3777 7.7632 5.5745 4.92	91.46, 106.02, 94.3, 100.5, 101.1, 103.9, 100.8, 104.9, 99.45, 95.6, 99.57, 95.76	-13.05	9.48

(c)

Table 1. (continued)

From the results shown previously, it is illustrated a perspective of designing scannable linear arrays in a uniform structure with amplitude and phase optimization using genetic algorithms. The genetic algorithm efficiently computes a set of antenna element amplitude and phase excitations in each angle of the steering range in order to provide a radiation pattern with maximum side lobe level reduction in all scanning range. The optimized design case provides a considerable side lobe level reduction with respect to the conventional phased array, with very similar values of directivity and maintaining the same aperture.
The design case for phased circular arrays is presented in the next section.

3. Design of phased circular arrays

Among antenna array configurations, the phased linear array is the most common form employed in cellular and personal communication systems (PCS) (Song et al., 2001). However, 360° scanning of the radiation beam can be obtained by combining a few linear arrays whose sector scans add to give the desired 360° scan. This could result in objectionably high costs, i.e., the array cost, the control complexity, and the data processing load are increased. Furthermore, the radiation pattern varies with the scan angle, i.e., the gain of a linear array degrades in its end-fire directions giving way to interference coming from other directions (Durrani et al., 2002). Unlike the linear array, the performance of the circular arrays (Du, 2004; Goto et al., 1977; Tsai et al., 2001; Tsai et al., 2004; Vescovo, 1995; Watanabe, 1980) has not been extensively studied. Therefore, in this section it is presented the design of scannable circular arrays optimizing the amplitude and phase excitations across the antenna elements. It is believed by the authors that an evaluation of the array factor for scannable circular arrays optimized by GA's considering a scanning range in all azimuth plane (360°) has not been presented previously. Depending on the performance improvement that we could get (in terms of the side lobe level and the directivity) with respect to the circular array with the conventional progressive phase excitation, this information could be interesting for antenna designers. Next, it is presented the theoretical model for this design case.

3.1 Theoretical model

Consider a circular antenna array of N antenna elements uniformly spaced on a circle of radius a in the x-y plane. The array factor for the circular array shown in Figure 1, considering the center of the circle as the phase reference, is given by

$$AF(\phi,\mathbf{I}) = \sum_{n=1}^{N} I_n \exp\left[jka\left(\cos(\phi-\Delta\phi_n)-\cos\left(\phi_0-\Delta\phi_n\right)\right)\right] \tag{4}$$

where $\Delta\phi_n=2\pi(n-1)/N$ for $n=1,2, \ldots, N$ is the angular position of the nth element on the x-y plane, $ka=Nd$, i.e., $a=Nd\lambda/2\pi$, $\mathbf{I} = [I_1, I_2, \ldots, I_N]$, I_n represents the amplitude excitation of the nth element of the array, ϕ_0 is the direction of maximum radiation and ϕ is the angle of incidence of the plane wave.
As it was established for the linear array case, the array factor with phase excitation is created by adding in the appropriate element phase perturbations, $\mathbf{P} = [\delta\beta_1, \delta\beta_2, \ldots, \delta\beta_N]$, $\delta\beta_i$ represents the phase perturbation of the ith element of the array, such that

$$\varphi_n = ka[\cos(\phi - \Delta\phi_n) - \cos(\phi_0 - \Delta\phi_n)] \tag{5}$$

where $\varphi_n = ka[\cos(\phi - \Delta\phi_n) - \cos(\phi_0 - \Delta\phi_n)]$.
It is important to mention that as the center of the circle is taken as the phase reference in the array factor, it is considered a symmetrical excitation for the optimization process, i.e, the phase perturbation would be given in the next way $I_1\exp(j\delta\beta_1)$, …, $I_{N/2}\exp(j\delta\beta_{N/2})$, $I_{N/2+1}\exp(j\delta\beta_{N/2+1}) = I_1\exp(-j\delta\beta_1)$, …, $I_N\delta\beta_N = I_{N/2}\exp(-j\delta\beta_{N/2})$. Note that we will have $N/2$ amplitude and phase excitations in the optimization process.

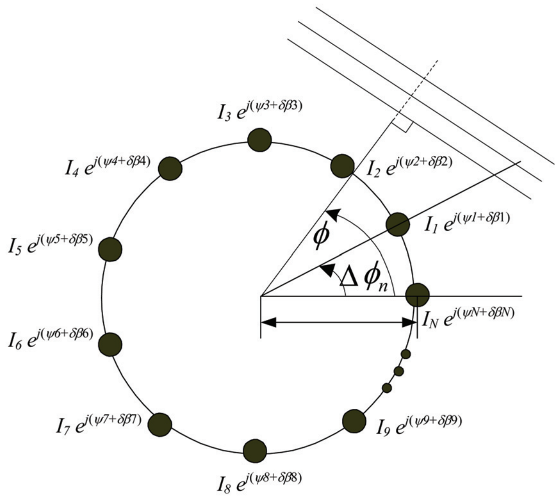

$I_3\, e^{j(\psi 3+\delta\beta 3)}$
$I_4\, e^{j(\psi 4+\delta\beta 4)}$
$I_2\, e^{j(\psi 2+\delta\beta 2)}$
$I_1\, e^{j(\psi 1+\delta\beta 1)}$
$I_5\, e^{j(\psi 5+\delta\beta 5)}$
ϕ
$\Delta\phi_n$
$I_N\, e^{j(\psi N+\delta\beta N)}$
$I_6\, e^{j(\psi 6+\delta\beta 6)}$
$I_7\, e^{j(\psi 7+\delta\beta 7)}$
$I_9\, e^{j(\psi 9+\delta\beta 9)}$
$I_8\, e^{j(\psi 8+\delta\beta 8)}$

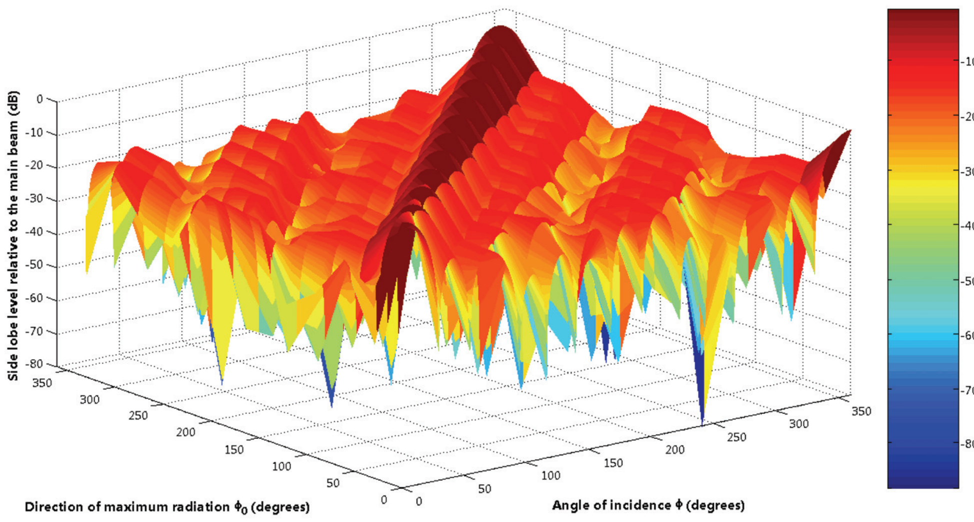
Figure 3. Array geometry for an N element uniform circular array with inter-element spacing d.

As already being pointed out, the objective of this section is to present an evaluation of the array factor for scannable circular arrays in a uniform geometry considering the optimization of the amplitude and phase excitation across the antenna elements. In this case, it is studied the behavior of the array factor for the scanning range of $0° \leq \phi_0 \leq 360°$ with an angular step of 30°. In this case, the objective function and the optimization process are set as they were presented for the linear array case, with the considerations of the scanning range and the symmetrical excitation aforementioned.
The results of using the GA for the design of scannable circular arrays are described in the next section.

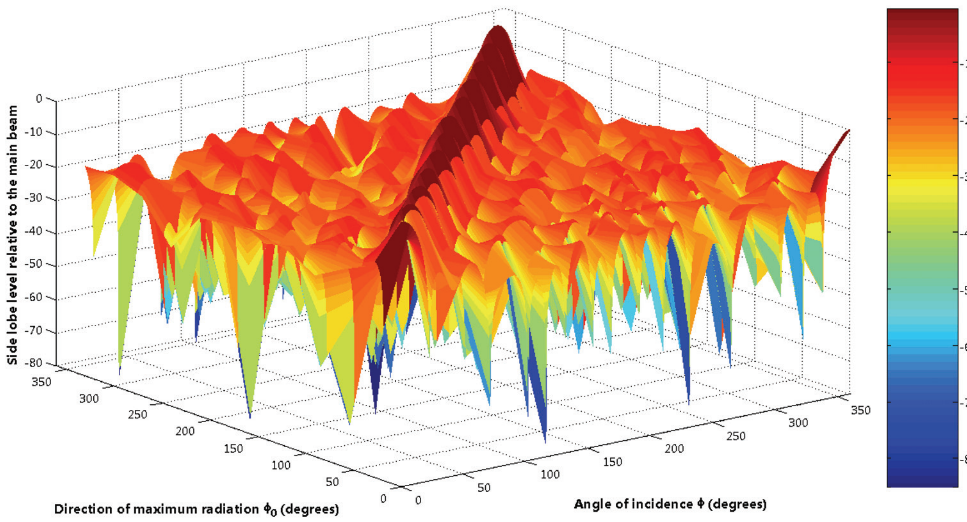
3.3 Results obtained for the design of phased circular arrays

The application of a phased circular array has sense when it is used to have a scanning range in all azimuth plane (360°). Therefore, the method of GA's was implemented to evaluate the behavior of the array factor for the scanning range of $0° \leq \phi_0 \leq 360°$ with an angular step of 30°. Next, some examples of the obtained results for the design of scannable circular arrays are explained.
Figure 4 shows the behavior of the array factor for a scannable circular array with the amplitude and phase excitation optimized by the GA. In this case, the separation between

antenna elements is set as d=0.5λ, and it is illustrated the examples for a) N=12 and b) N=18. The numerical values of the side lobe level, directivity, amplitude and phase perturbation distributions for the array factor shown in Figure 4 are presented in the Table 2.



(a)



(b)

Figure 4. Behavior of the radiation pattern for a scannable circular array in a steering range of $0° \leq \phi_0 \leq 360°$ with the amplitude and phase excitation optimized by the GA, a) N=12, b) N=18.

As illustrated in the Figure 4 and the Table 2, the results of the side lobe level and the directivity for the optimized design are surprising. Observing the results, the conventional case of progressive phase excitation provides a *SLL*= -7.16 dB, and *DIR*=10.6 dB for a) *N*=12, and a *SLL*= -7.9 dB, *DIR*=12 dB for b) *N*=18. For the case of the optimized design, it is obtained a SLL_{min}= -12.17 dB, SLL_{max}= -13.68 dB and DIR_{min}=11.35 dB, DIR_{max}=11.56 dB for a) *N*=12, and a SLL_{min}= -13.50 dB, SLL_{max}= -16.74 dB and DIR_{min}=12.96 dB, DIR_{max}=13.23 dB for b) *N*=18.

These values mean a substantial improvement in the performance of the array for the design optimized by the GA with respect to the conventional case, i.,e, it is obtained a substantial improvement in the sense of the side lobe level and an improvement of about 1 dB in the directivity, maintaining the same scanning range and the same aperture.

Design case with the amplitude and phase excitation optimized by the GA (*N*=12)					**Conventional case**	
θ_0	***SLL* (dB)**	***DIR* (dB)**	**Normalised Amplitude distribution**	**Phase perturbation distribution (deg)**	***SLL* (dB)**	***DIR* (dB)**
0°	-13.26	11.50	8.9861, 7.0281, 6.0653, 7.1273, 8.6156, 13.6634	165.62, -109.48, 179.99, 108.64, -161.28, 149.09	-7.167	10.62
30°	-12.17	11.35	13.9516, 12.0757, 6.3265, 6.0282, 6.4862, 10.6097	18.52, -6.39, 39.77, -10.41, -26.71, 9.63	-7.165	10.65
60°	-12.78	11.53	10.444, 13.9698, 10.5989, 6.9121, 6.5014, 6.8659	-17.52, 25.97, -13.76, 57.97, -2.92, -59.70	-7.164	10.66
90°	-13.01	11.54	6.5076, 10.5052, 13.9753, 10.7816, 6.2717, 6.1824	65.56, -20.46, 26.50, -15.54, 57.09, 1.70	-7.165	10.66
120°	-13.42	11.50	6.0569, 6.8042, 10.9555, 13.8561, 8.9444, 6.1517	-17.08, 76.07, -11.95, 26.74, -30.83, 66.82	-7.167	10.66
150°	-13.18	11.56	6.2821, 6.0246, 7.0327, 9.669, 13.7562, 8.3516	114.21, 169.51, -114.93, 168.12, -151.91, 160.20	-7.165	10.65
180°	-12.48	11.50	9.1939, 6.015, 6.3482, 6.2204, 11.0839, 13.9433	14.83, -52.36, -5.56, 59.88, -11.88, 24.69	-7.164	10.65
210°	-13.24	11.47	13.7931, 9.391, 6.2871, 6.0677, 6.8493, 10.4153	-24.87, 19.04, -51.57, -19.56, 70.88, -12.32	-7.165	10.65
240°	-13.36	11.51	9.0782, 13.9387, 9.9896, 6.0713, 6.0022, 6.6342	-157.31, 155.52, -168.26 114.07, -176.6, -107.67	-7.167	10.66
270°	-12.74	11.50	7.3426, 11.2421, 13.9757, 11.0413, 7.2521, 6.0368	-52.84, 14.82, -26.20, 14.92, -51.18, -1.47	-7.165	10.66
300°	-12.51	11.48	6.0446, 7.609, 11.5202, 13.7836, 11.3168, 7.6133	-176.94, 125.17, -161.71 153.62, -165.07, 130.3	-7.164	10.66
330°	-13.68	11.54	6.0884, 6.0135, 6.2152, 9.2554, 13.9733, 10.1944	-113.97, -178.77, 110.13 -160.49, 152.53, -165.7	-7.165	10.66

(a)

Table 2. Numerical values of the side lobe level (*SLL*), directivity (*DIR*), amplitude and phase perturbation distribution for the array factor illustrated in Fig. 4, a) *N*=12, b) *N*=18.

Design case with the amplitude and phase excitation optimized by the GA (*N*=18)					**Conventional case**	
θ_0	***SLL* (dB)**	***DIR* (dB)**	**Normalised Amplitude distribution**	**Phase perturbation distribution (deg)**	***SLL* (dB)**	***DIR* (dB)**
0°	-16.74	12.98	9.606, 10.0796, 8.7306, 7.4423, 6.0729, 13.1574, 10.1184, 12.7586, 13.948	151, 137.36, -175.89, -3.65, 18.19, -175.31, -119.92, 176.34, -147.15	-7.9	11.91
30°	-14.59	13.15	13.8924, 13.4948, 12.384, 6.1859, 9.089, 7.2624, 8.7293, 6.1218, 10.978	-19.84, -9.665, -2.166, 27.55, -85.76, -19.785, 75.877, 25.36, 1.656	-7.9	12.0
60°	-14.85	13.05	7.7885, 10.2233, 13.7296, 10.4833, 8.1398, 10.8737, 6.5203, 6.1301, 9.315	140.02, 172.19, 156.37, 172.09, 139.6, -178.963, -40.93, 70.3, -170.776	-7.9	11.96
90°	-14.20	13.23	9.5658, 6.1554, 13.1213, 13.7249, 13.4988, 11.421, 6.0703, 8.8405, 7.243	101.49, -172.8, 175.48 166.3, 159.385, 177.2, 172.646, 105.8, -170.73	-7.9	12.0
120°	-14.08	12.96	6.8649, 6.1881, 10.5908, 7.0292, 11.4631, 13.7734, 9.398, 8.1573,10.8491	-137.484, 99.2, -9.642, - 43.582, 1.26, -17.302, -10.043, -23.553, -9.312	-7.9	11.96
150°	-14.16	13.12	6.2198, 8.1045, 7.2332, 8.8087, 6.0134, 11.5031, 13.8558, 13.7171, 11.9379	-3.078, 76.838, 0.51, -77.027, 14.863, 2.441, -12.856, -10.865, 2.579	-7.9	12.0
180°	-16.21	13.17	11.2931, 6.9325, 9.8873, 6.0995, 7.026, 6.1086, 11.0604, 12.6843, 13.1782	-7.805, 28.515, 9.038, -102.785, -178.52, 8.36, - 62.315, -27.75, -29.808	-7.9	11.95
210°	-14.85	13.18	13.4792, 13.806, 13.7676, 6.2475, 10.5248, 6.23, 8.2037, 6.1015, 9.0618	-144.58, -178.87, -165.5, 110.127, -112.7, -137.6, 111.07, 144.61, 176.76	-7.9	12.0
240°	-14.74	13.00	8.1184, 9.5482, 13.75, 11.8184, 7.7963, 12.1036, 6.0628, 6.5981, 8.9808	-149, -171.964, -159.465 176.53, -141.74, 178.43, 42.03, -50.722, 171.964	-7.9	11.96
270°	-14.07	13.15	8.649, 6.4937, 12.5178, 13.9531, 13.1291, 9.6387, 6.0478, 6.2315, 7.3053	76.656, -29.518, 4.1517, 6.344, 23.728, 7.36, 27.052, 71.543, -18.029	-7.9	12.0
300°	-13.50	13.00	6.6366, 6.4168, 10.2879, 7.7545, 9.5766, 13.9128, 11.4554, 7.9931, 11.7014	-68.05, 65.273, -169.852 -144.74, -179.74, -161.4, -176.4, -155.88, -165.66	-7.9	11.96
330°	-14.80	13.17	6.2548, 9.9819, 7.4138, 8.7991, 6.002, 10.5322, 13.7914, 13.7537, 13.7233	-116.06, 98.633, 136.89, -120.18, -126.33, -166.8 -152.76, -167, -165.53	-7.9	12.0

(b)

Table 2. (continued).

Now, if the results of the side lobe level and the directivity for the scannable circular array optimized by the GA (for *N*=12, shown in the Table 2a) are compared with the linear array case with conventional phase excitation (for *N*=12, shown in the Table 1c), we observe that the values of the *SLL* and *DIR* are a little better for the circular array case with the great advantage of having a scanning range several times bigger than the linear array case.

4. Discussions and open problems

The main objective of this chapter is to illustrate the application of an evolutionary optimization technique in the problem of designing scannable antenna arrays with geometry lineal and circular. A genetic algorithm is applied to evaluate the performance of scannable linear and circular arrays optimizing the amplitude and phase excitations across the antenna elements. The results obtained for the design of scannable linear and circular arrays reveal that the performance of the phased array could be improved substantially, with respect to the conventional case of progressive phase excitation, if the amplitude and phase excitations are optimized in an adequate way by an evolutionary algorithm.

There are many remaining open problems. In this case, we propose the following questions:

- Which is the best evolutionary algorithm for the problem in terms of solution quality and in terms of computation time?
- Given the algorithm, what is the best representation and the best genetic operators to use?
- Is there a better way to model or represent the problem in such a way to avoid the evaluation of the *SLL* and the *DIR* for each angle in the scanning range?
- What are the limits of performance for non-uniformly spaced phased arrays? How do these limits compare with the ones obtained by uniformly spaced phased arrays?

5. Conclusions

This chapter illustrates how to model the design of phased linear and circular arrays with the optimization of the amplitude and phase excitations for improving the performance of the array in the sense of the side lobe level and the directivity.

In the case of the scannable linear arrays, the experimental results illustrated that the design of scannable linear arrays with the amplitude and phase optimized with the use of genetic algorithms could provide a lower side lobe level (<-20 dB), with respect to a conventional phased linear array. In this case, these values of the side lobe level for the optimized design case are achieved with very similar values of directivity and the same aperture in both design cases.

For the case of the scannable circular arrays, the obtained results illustrated that the optimization of the array could provide a substantial improvement in the side lobe level and an improvement of about 1 dB in the directivity, with respect to the conventional case of progressive phase excitation. These improvements in the performance of the array are achieved maintaining the same scanning range, i.e., in all azimuth plane (360°), and the same aperture.

Future research will be aimed at considering the application and performance evaluation of new evolutionary algorithms in the design of different array geometries to understand which algorithm fits best a given problem. Also, the answer for the proposed set of questions will be investigated. Furthermore, it will be investigated the application of evolutionary techniques in the optimization of different phased arrays considering the feeding network in order to simplify the beam-forming network.

6. Acknowledgements

This work was supported by the Mexican National Science and Technology Council, CONACyT, under grant J50839-Y, and the Science and Technology Council of Tamaulipas Mexico (COTACYT) under grant 2007-C13-73901.

7. References

Ares-Pena, F. J., Rodriguez-Gonzalez, J. A., Villanueva-Lopez, E., & Rengarajan, S. R. (1999). Genetic algorithms in the design and optimization of antenna array patterns. *IEEE Transactions on Antennas and Propagation,* 47, 506-510.

Bae, J., Kim, K., & Pyo, C. (2005). Design of steerable linear and planar array geometry with non-uniform spacing for side-lobe reduction. *IEICE Transactions on Communications,* E88-B (1), 345-357.

Bae, J., Kim, K., Pyo, C., & Chae, J. S. (2004). Design of scannable non-uniform planar array structure for maximum side-lobe reduction. *ETRI Journal*, 26 (1), 53-56.

Balanis, C. (2005). *Antenna Theory-Analysis and Design*. Third Edition, New York: Wiley.

Bray, M. G., Werner, D. H., Boeringer, D. W., & Machuga, D. W. (2002). Optimization of thinned aperiodic linear phased arrays using genetic algorithms to reduce grating lobes during scanning. *IEEE Transactions on Antennas and Propagation,* 50, 1732–1742.

Du, K. L. (2004). Pattern Analysis of Uniform Circular Array. *IEEE Transactions on Antennas and Propagation*, 52 (4), 1125-1129.

Durrani, S., & Bialkowski, M. E. (2002). An investigation into the interference rejection capability of a linear array in a wireless communications system. *Microwave and Optical Technology Letters*, 35, 445-449.

Godara, L. C. (2002). *Handbook of Antennas in Wireless Communications*. CRC Press.

Golberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Massachusetts.

Goto, N., & Tsunoda, Y. (1977). Sidelobe reduction of circular arrays with a constant excitation amplitude. *IEEE Transactions on Antennas and Propagation*, 25 (6), 896-898.

Hansen, R. C. (1998). *Phased Array Antennas*. New York: Wiley.

Haupt, R. (1994). Thinned arrays using genetic algorithms. *IEEE Transactions on Antennas and Propagation*, 42, 993-999.

Haupt, R. L. (1995). An introduction to genetic algorithms for electromagnetics. *IEEE Antennas and Propagation Magazine,* 37, 7-15.

Junker, G. P., Kuo, S. S., & Chen, C. H. (1998). Genetic algorithm optimization of antenna arrays with variable interlement spacings. *Proceedings of IEEE Antennas and Propagation Society International Symposium: Vol. 1* (pp. 50-53), Atlanta GA.

Kurup, D., Himdi, M., & Rydberg, A. (2003). Synthesis of uniform amplitude unequally spaced antenna arrays using the differential algorithm. *IEEE Transactions on Antennas and Propagation*, 51, 2210-2217.

Lommi, A., Massa, A., Storti, E., & Trucco, A. (2002). Sidelobe reduction in sparse linear arrays by genetic algorithms. *Microwave and Optical Technology Letters,* 32, 194-196.

Mailloux, R. J. (2005). *Phased array antenna handbook.* Artech House, Boston, Second edition.

Murino, V., Trucco, A., & Regazzoni, C. S. (1996). Synthesis of unequally spaced arrays by simulated annealing. *IEEE Transactions on Signal Processing*, 44, 119–123.

Panduro, M. A., Covarrubias, D. H., Brizuela, C. A., & Marante, F. R. (2005). A multi-objective approach in the linear antenna array design. *AEU International Journal of Electronics and Communications,* 59 (4), 205-212.

Panduro, M. A. (2006). Optimization of non-uniform linear phased array using genetic algorithms to provide maximum interference reduction in a wireless communication system. *Journal of the Chinese Institute of Engineers JCIE*, 29 (7), Special Issue: Communications, 1195-1201.

Panduro, M. A., Mendez, A. L., Dominguez, R., & Romero, G. (2006). Design of non-uniform circular antenna arrays for side lobe reduction using the method of genetic algorithms. *AEU International Journal of Electronics and Communications,* 60 (10), 713-717.

Rahmat-Samii, Y., & Michielssen, E. (1999). *Electromagnetic Optimisation by Genetic Algorithms*. New York: Wiley-Interscience.

Soni, RA, Buehrer, R. M., & Benning, R. D. (2002). Intelligent antenna system for cdma2000. *IEEE Signal Processing Magazine*, 19, 54-67.

Song, Y. S., Kwon, H. M., & Min, B. J. (2001). Computationally efficient smart antennas for CDMA wireless communications. *IEEE Transactions on Vehicular Technology,* 50, 1613-1628.

Stutzman, W. L., & Thiele, G. A. (1998). *Antenna Theory and Design*. Wiley, second edition.

Tian, Y. B., & Qian, J. (2005). Improve the performance of a linear array by changing the spaces among array elements in terms of genetic algorithm. *IEEE Transactions on Antennas and Propagation*, 53 (7), 2226–2230.

Tsai, J. A., & Woerner, B. D. (2001). Adaptive beamforming of uniform circular arrays (UCA) for wireless CDMA system. *35th Asimolar Conference*, Pacific Grove, CA.

Tsai, J. A., Buehrer, R. M., & Woerner, B. D. (2004). BER performance of a uniform circular array versus a uniform linear array in a mobile radio environment. *IEEE Transactions on Wireless Communications*, 3, 695-700.

Vescovo, R. (1995). Constrained and Unconstrained Synthesis of Array Factor for Circular Arrays. *IEEE Transactions on Antennas and Propagation*, 43 (12), 1405-1410.

Watanabe, F., Goto, N., Nagayama, A., & Yoshida, G. (1980). A Pattern Synthesis of Circular Arrays by Phase Adjustment. *IEEE Transactions on Antennas and Propagation*, 28 (6), 857-863.

Weile, D. S., & Michielsen, E. (1997). Genetic algorithm optimization applied to electromagnetics: A review. *IEEE Antennas and Propagation Magazine*, 45, 343-353.

Yan, K., & Lu, Y. (1997). Sidelobe reduction in array-pattern synthesis using genetic algorithm. *IEEE Transactions on Antennas and Propagation*, 45, 1117–1122.

Design of an Efficient Genetic Algorithm to Solve the Industrial Car Sequencing Problem

A. Zinflou[1], C. Gagné[2] and M. Gravel[2]
[1] *Département des sciences appliquées, Université du Québec à Chicoutimi,*
[2] *Département d'informatique et de mathématique, Université du Québec à Chicoutimi,*
[1,2]*Canada*

1. Introduction

In many industrial sectors, decision makers are faced with large and complex problems that are often multi-objective. Many of these problems may be expressed as a combinatorial optimization problem in which we define one or more objective functions that we are trying to optimize. Thus, the car sequencing problem in an assembly line is a well known combinatorial optimization problem that cars manufacturers face. This problem involves scheduling cars along an assembly line composed of three consecutive shops: body welding and construction, painting and assembly. In the literature, this problem is most often treated as a single objective problem and only the capacity constraints of the assembly shop are considered (Dincbas *et al.*, 1988). In this workshop, each car is characterized by a set of different options and the workstations where each option is installed are designed to handle a certain percentage of cars requiring the same options. To smooth the workload at the critical assembly workstations, cars requiring high work content must be dispersed throughout the production sequence. Industrial car sequencing formulation subdivides the capacity constraints into two categories, that are the capacity constraints linked to the high-priority options and the capacity constraints linked to the low-priority options.
However, the reality of industrial production does not only take into account the assembly shop requirements. The industrial formulation proposed by French automobile manufacturer Renault, in the context of the ROADEF 2005 Challenge, also takes into account the paint shop requirements. In this workshop, the minimization of the amount of solvent used to purge the painting nozzles for colour changeovers, or when a known maximum number of vehicle bodies of the same colour have been painted, is an important objective to consider. Indeed, long sequences of cars of the same colour tend to render visual quality controls inaccurate. To ensure this quality control, the number of cars of the same colour must not exceed an upper limit.
The industrial car sequencing problem (ICSP) is thus a multi-objective problem in nature, with three conflicting objectives to minimize. In the assembly shop, one tries to minimize the number of violations of capacity constraints related to high-priority options (HPO) and to low-priority options (LPO). In the paint shop, one tries to minimize the number of colour changes (COLOUR). In the 2005 ROADEF Challenge, the Renault automobile manufacturer proposes to tackle the problem by treating the three objectives lexicographically.

Among the resolution methods proposed by the participants of the challenge, one finds essentially neighbourhood search methods as simulated annealing, iterative tabu search, iterative local search and variable neighbourhood search (Briant *et al.*, 2007; Cordeau *et al.*, 2007; Estellon *et al.*, 2007; Ribiero *et al.*, 2007a; Gavranović, 2007; Benoist, 2007), an ant colony optimization algorithm (ACO) (Gagné *et al.*, 2006) and a genetic algorithm (GA) (Jaszkiewicz *et al.*, 2004). Since the work of all the participating teams was not published, the previous enumeration is not exhaustive. After the challenge, other authors proposed to solve the problem using an integer linear programming model (Estellon *et al.*, 2005; Gagné *et al.*, 2006; Prandtstetter and Raidl, 2007), an algorithm hybridizing variable neighbourhood search and integer linear programming (Prandtstetter and Raidl, 2007) or an iterative local search approach (Ribeiro *et al.*, 2007b).
One may note that few authors proposed GAs to solve this multi-objective problem, except for Jaszkiewicz *et al.* (Jaszkiewicz *et al.*, 2004). Moreover, this team was not amongst the twelve finalists of the 2005 ROADEF Challenge that included 55 teams from 15 countries at the beginning. As for the ICSP, one may only find the GAs proposed by Warwick and Tsang (1995), Terada *et al.* (2006) and Zinflou *et al.* (2007) in the literature for the standard version of the car sequencing problem. Among them, only Zinflou *et al.* (2007) succeeded in proposing an efficient GA, suggesting that this metaheuristic is not well suited to deal with the specificities of this problem.
The main purpose of this chapter is to show that GAs can be efficient approaches for solving the ICSP when the different mechanisms of the algorithm are specially design to deal with the specificities of the problem. To achieve this, we present the different choices made during the design of the genetic operators. In particular, we propose two new crossover operators dedicated to the multi-objective characteristic of the problem. The performance of the proposed approaches is assessed experimentally using the different instances of the 2005 ROADEF Challenge and compared with the best results obtained during the challenge.
This chapter is organized as follows: Section 2 briefly defines the industrial car sequencing problem and Section 3 describes the new crossover operators proposed for this multi-objective problem. The basic features of the proposed GA are presented in Section 4. Section 5 is dedicated to computational experiments and comparisons with previous results from literature. Finally, the conclusion of this research work is given in Section 6.

2. The industrial car sequencing problem

This section provides the main elements to describe the ICSP. The reader may consult Nguyen & Cung (2005) and Solnon *et al.* (2007) for a complete description of the problem. On each production day, customer orders are sent in real time to the assembly plant. The daily task of the planners is then: (1) to assign a production day to each ordered vehicle, according to production line capacities and delivery dates that were promised to customers; and (2) to schedule the cars within each production day while satisfying as many of the requirements as possible of the three manufacturing workshops, as illustrated in Figure 1. The sequence thus found is then applied to the whole assembly line.
In the definition of ICSP proposed during the 2005 ROADEF Challenge, the Renault car manufacturer stated that technologies used in the plants are such that the body shop does not set requirements for the daily schedule. The ICPS then consists in scheduling a set of cars (*Nb_cars*) for a production day taking into consideration the paint shop and assembly shop requirements.

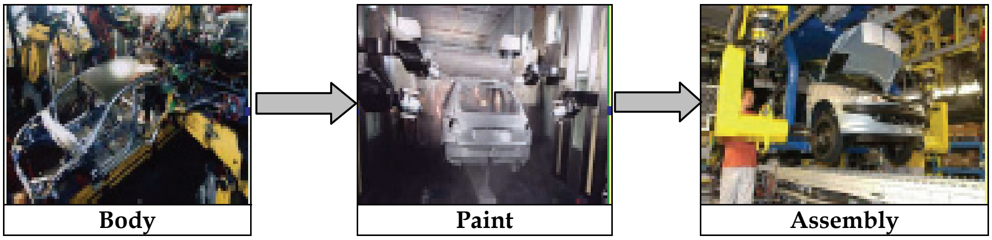


Fig. 1. The three workshops of the assembly line (Nguyen & Cung, 2005)

In the paint shop, production scheduler tries to group cars by paint colour to minimize the number of colour changes. Painting nozzles must be purged with solvent when changing car colour, or after a maximum number of cars (rl_{max}) painted the same colour, to ensure quality. Each purge requires a colour change. Then, each solution with more consecutive cars than rl_{max} to be painted the same colour must be considered unfeasible.

In the assembly shop, many elements are added to the painted body to complete the car assembly. Each car is characterized by a set of different options O for which the workstations, where these options are installed, are designed to handle up to a certain percentage of the cars requiring the same options. These capacity constraints may be expressed by a ratio r_0/s_0, that means that any consecutive subsequence of s cars must include at most r cars with option o. Cars requiring the same configuration of options must be dispersed throughout the production sequence to smooth out the workload at various critical workstations. If, for a subsequence of length s, it is impossible to satisfy the capacity constraint for option o, the number of cars that exceeds r defines what is called *conflicts* or *violations*. As mentioned previously, the ICSP subdivides the capacity constraints of the assembly shop into two groups; the constraints related to the high-priority options and those related to the low-priority options. In this shop, production scheduler tries to optimize two different objectives: the number of capacity constraint violations related to the high-priority options (HPO) and the number of capacity constraints violations related to the low-priority options (LPO).

We choose to cluster the cars requiring the same configuration of high-priority and low-priority options into V car classes, for which we know the exact number to produce (c_v). These quantities represent the production constraints of the problem. Table 1(a) shows an example of the industrial problem for producing 25 cars (*Nb_cars*) having 5 options (O) with 6 car classes (V) and a possibility of 4 different colours across each class. One defines a production sequence Y by two vectors representing respectively the car classes (*Classes*) and the car colour codes (*Colours*) as shown in Figure 1(b). A production sequence will be designated by Y = {*Classes*/*Colours*} in the remainder of the chapter and the element at position i of the sequence will be defined by $Y(i)$ = *Classes*(i)/*Colours*(i).

Another interesting feature of the ICSP is that it links the different production days. Thus, the evaluation of a solution must take into account the end of the previous production day and must extrapolate the minimum number of conflicts generated with the next production day. Similarly, a colour change will be added if the colour of the first car of the current day is different from the colour of the last car of the previous day.

To evaluate the number of conflicts for each option, we first construct binary matrix S of size O * *Nb_cars* using solution vector Y. We have $S_{oi} = 1$ if the class of car assigned to position i of the solution vector requires option o, otherwise it is equal to 0. The decomposition of the

Classes vector of solution Y from Table 1 into its different options to obtain *S* is given in Table 2. In Table 2(a), we also report the end of the previous production day sequence to allow to evaluate the number of conflicts related to the link of these two production days. In Table 2(b), we also evaluate the solution based on the next day, assuming cars without any option.

			Class #					
o	*r*	*s*	1	2	3	4	5	6
1	1	2	0	1	1	0	0	0
2	2	5	1	0	1	0	1	1
3	1	3	0	1	0	0	0	0
4	3	5	0	0	0	1	0	1
5	2	3	0	1	1	0	1	0
c_v			5	5	4	4	3	4
Colour #		1	2	1	1	2	1	1
		2	1	1	0	2	1	1
		3	1	3	2	0	0	2
		4	1	0	1	0	1	0

(a)

Y	1	2	3	4	5	6	…..	21	22	23	24	25
Classes	3	5	5	4	6	4		3	1	4	5	1
Colours	4	4	2	2	2	2		3	3	1	1	1

(b)

Table 1. Example and solution of an ICSP

		Previous day (*D-1*)					**Current day (*D*)**						
Positions		-5	-4	-3	-2	-1	1	2	3	4	5	6	………
Classes		4	1	4	4	2	3	5	5	4	6	4	
OPTION	1/2	0	0	0	0	**1**	**1**	0	0	0	0	0	
	2/5	0	1	0	0	0	**1**	**1**	**1**	**0**	**1**	**0**	
	1/3	0	0	0	0	1	0	0	0	0	0	0	
	3/5	1	0	1	1	0	0	0	0	1	1	1	
	2/3	0	0	0	0	**1**	**1**	**1**	**1**	0	0	0	

(a)

		Current day (*D*)						**Next day (*D+1*)**				
Positions		….	21	22	23	24	25	26	27	28	29	30
Classes			3	1	4	5	1					
OPTION	1/2		1	0	0	0	0	0	0	0	0	0
	2/5		1	**1**	**0**	**1**	**1**	**0**	0	0	0	0
	1/3		0	0	0	0	0	0	0	0	0	0
	3/5		0	0	1	0	0	0	0	0	0	0
	2/3		1	0	0	1	0	0	0	0	0	0

(b)

Table 2. Evaluation of the solution shown in Table 1

For the current production day *D*, options 1, 3 and 4 do not cause any violation in this part of the solution. Indeed, for each of these three options, we never have a subsequence of size *s*, with more than *r* cars with the option. However, for option 2, there are two conflicts located between positions *1* to *5* since we have 4 cars having the option while the capacity constraint limits the maximum to 2. In addition, there is one conflict located between positions *2* to *6,* another conflict between positions 20 to 24 and two other conflicts between positions *21* to *25,* since capacity constraint 2/5 is not satisfied. For option 5, we also have one conflict as capacity constraint 2/3 is not satisfied between positions *1* to *3*.
For the link with previous production day *D-1*, we have one conflict located between positions *-1* to *1* for option 1 , two conflicts located between positions -2 to 3 and positions -1 to 4 for option 2, and another conflict between positions -1 to 2 for option 5. For the link with next production day *D+1*, we only have one conflict located between positions 22 to 26 for option 2.
Considering that the first three options are high-priority and that the other two are low-priority, we therefore have 10 conflicts for the HPO objective and 2 conflicts for the LPO objective for this solution *Y*. Then, we only have to count the number of colour changes (COLOUR) to complete the evaluation of solution *Y*.
The 2005 ROADEF Challenge proposed to tackle the problem using a weighted sum method that assigns different weights w_1, w_2 and w_3 to each objective according to their priority level, in order to evaluate a solution *Y*. The quality of solution *Y* is then given by:

$$F(Y)=w_1*obj_1+w_2*obj_2+w_3*obj_3 \tag{1}$$

where obj_1, obj_2 and obj_3 correspond respectively to the values obtained for a solution *Y* on each objective according to the priority level assigned. The weights w_1, w_2 and w_3 are respectively set at 1000000, 1000 and 1 (Nguyen & Cung, 2005). According to the different configurations of the Renault plants, the three following objective hierarchies are possible: HPO-COLOUR-LPO, HPO-LPO-COLOUR and COLOUR-HPO-LPO.

3. Introducing problem knowledge in crossover design for the industrial car sequencing problem

Traditional crossover operators are not well suited to deal with the specificities of the car sequencing problem. Indeed, Warwick and Tsang (1995), and Terada *et al.* (2006) used such operators to solve the single objective car sequencing problem found in the literature and their results were not competitive. However, Zinflou *et al.* (2007) obtained very competitive results using two highly-specialized crossover operators for the same problem.
For the multi-objective ICSP, Jaszkiewicz *et al.* (2004) proposed to use a common sequence preserving crossover. Basically, the purpose of this operator is to create an offspring using the common maximum subsequence of the indices of the groups in two given solutions (parents). However, even if the results of this approach are promising, they did not allow the authors to be part of the twelve finalists during the 2005 ROADEF Challenge.
The crossover operators proposed by Zinflou *et al.* (2007) for the single objective car sequencing problem, called *non-conflict position crossover* (*NCPX*) and *interest based crossover* (*IBX*), use problem-knowledge to perform recombination. The concept used by *NCPX* and *IBX* crossovers to use problem-knowledge is called *interest*. The idea behind this concept is to penalize the conflicting car classes, by counting the number of new conflicts caused by the addition of these classes as a cost. Conversely, if the addition of a car class does not cause

new conflicts, then this is counted as a profit equal to the difficulty of class D_v as proposed by Gottlieb *et al.* (2003). Basically, *NCPX* crossover tries to minimize the number of relocated cars by emphasizing non conflict position information from both parents. The *IBX* crossover, in turn, rather tries to keep the cars in the same area of the chromosome as it occupied with one of the two parents. For more details about these two crossover operators, the reader may consult Zinflou *et al.* (2007).
The following sections will show how to adapt the two *NCPX* and *IBX* crossover operators to the multi-objective ICSP.

3.1 Adaptation of the interest calculation for the industrial car sequencing problem

To present the different adaptation of the crossover operators, we must redefine the interest concept to be able to take into account the multi-objective nature of the ICSP. We define the *total weighted interest* (TWI) to establish if it is interesting to add a car of class *v*, of colour *colour* at a position *i* in the sequence. The total weighted interest is expressed by:

$$TWI_{v,colour,i} = I_{v,i,HPO} * w_{HPO} + I_{v,i,COLOUR} * w_{COLOUR} + I_{v,i,LPO} * w_{LPO} \tag{2}$$

where w_{HPO}, w_{COLOUR} and w_{LPO} correspond respectively to the weight of each objective (1000000, 1000 or 1 according to their priority levels) and $I_{v,i,HPO}$, $I_{v,i,COLOUR}$ and $I_{v,i,LPO}$ correspond to the interest in inserting a car of class *v* at the position *i* for each objective. The interest concept may be defined according to each objective.
According to Equation 3, the interest $I_{v,i,COLOUR}$ to insert a car of class *v* at position *i* to minimize objective COLOUR is set at 1 if it is possible to complete the current colour subsequence with a car of class *v*. If it isn't possible, the interest is set to -1.

$$I_{v,i,COLOUR} = \begin{cases} 1 & \text{if } nb(v_{colour(i-1)}) > 0 \;\&\; run\_length < rl_{\max} \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

nb($v_{colour(i-1)}$) indicates the number of cars of class *v* painted the same colour as the car in position *i-1*, *run_length* indicates the size of the consecutive subsequence of cars of the same colour as the car in position *i-1* and rl_{max} indicates the maximum length of a subsequence of the same colour. This notion serves to favour the classes of cars that have the same colour as the car located in the previous position, to lengthen the colour subsequence to the maximum size. Conversely, we penalize the car classes for which the addition implies a colour change.
$I_{v,i,HPO}$ and $I_{v,i,LPO}$ indicate the interest to insert a car of class *v* at position *i* in the sequence to minimize objectives HPO and LPO respectively. According to Equation 4, the interest corresponds to the difficulty for class *v* if the addition of this class does not cause new conflicts respectively on high-priority options (*k* = HPO) and on low-priority options (*k* = LPO). In the opposite case, we will define the cost that corresponds to the number of new conflicts produced on the high-priority or low-priority options, to discourage the insertion of this class at position *i*.

$$I_{v,i,k} = \begin{cases} D_{v,k} & \text{if } NbNewConflicts_{v,i,k} = 0 \\ -NbNewConflicts_{v,i,k} & \text{otherwise} \end{cases} \tag{4}$$

$NbNewConflicts_{v,i,k}$ corresponds to the number of new conflicts for the high-priority options (k = HPO) or for the low-priority options (k = LPO) caused by the addition of a car of class v at position i. $D_{v,k}$ indicates the difficulty of class v for high-priority options (k = HPO) and for low-priority options (k = LPO). The idea behind this concept is simply to penalize the classes of cars for which the addition leads to additional conflicts for the high-priority or low-priority options, on considering this number of new conflicts as a cost. Conversely, if the addition of a class does not cause new conflicts on the options, we then evaluate the benefit of placing this class according to its difficulty. Gottlieb *et al.* (2003) established that the difficulty of a class of cars v for high-priority or low-priority options ($D_{v,k}$) is the sum of the utilization rates of the high-priority options (k = HPO) or low-priority options (k = LPO) that compose that class. The utilization rate of an option may be expressed as the ratio between the number of cars requiring this option and the maximum number of cars that may have this option such that the r_o/s_o constraint is satisfied.

3.2 The multi-objective *NCPX* crossover operator (*NCPX*^{MO})

The *NCPX*^{MO} procedure for the ICSP is inspired by the *NCPX* crossover proposed for the single objective car sequencing problem (Zinflou *et al.*, 2007) and is carried out in two main steps. Step 1 consists of selecting a parent P_1 and establishing in this chromosome the number of positions that are not part of a conflict for objectives HPO ($nbpos_{HPO}$) and LPO ($nbpos_{LPO}$) and the number of positions where there is no colour change ($nbpos_{COLOUR}$). Then, we randomly select a number nbg_k between 0 and $nbpos_k$ for each objective k (k = *HPO, LPO, COLOUR*). These three numbers are used to determine, for each objective k, the number of "good" genes that will maintain in offspring *E1* the same position they had in P1. To take into account the priority of the objectives, we must make sure that the number of "good" genes kept for the main objective is greater or equal to the number of "good" genes selected for the secondary objective, and so forth. Once we establish these numbers, starting position (*sPos*) that is between 1 and *Nb_cars*, is randomly selected in the offspring to be created. The process of copying the good genes of P_1 to the offspring being created starts from *sPos* by first considering the main objective. If we reach the end of the chromosome and the number of genes copied for objective k is less than its corresponding nbg_k, the copy process restarts this time from the beginning of the offspring up to *sPos-1*. The same process is repeated for the other objectives, taking into account the already copied genes. Thereafter, the remainder of the genes from P_1 are used to constitute a non-orderly list L for the cars that must still be placed. We then randomly determine a position (*Pos*) from which the remaining positions of chromosome E_1 will be completed.

In Step 2, the cars in L are sorted according to their TWI. In case of a tie in TWI, if one of the cars is in *P2* at the position to be completed, this car is then selected. In the opposite case, we randomly select a car amongst those of equal ranking.

The operation of this cross operator is illustrated in Figure 2 for two parents P_1 = {21352446/62224622} and P_2 = {32621454/26242622} with the following objective hierarchy HPO-LPO-COLOUR. Let us assume that the evaluation of P_1 gives 5 positions without conflicts for objective HPO and for objective LPO (expressed by 0 in vectors "conflicts on HPO and LPO" below chromosome P_1), 4 positions where there is no colour change (expressed by 0 in vector the "colour changes" below chromosome P_1) and the values for numbers nbg_{HPO} = 4, nbg_{LPO} =2, nbg_{COLOUR} =1 and *sPos* = 3 by random setting. Starting with *sPos* and considering objective HPO, we may copy genes 5/2, 4/6, 4/2 and 2/6 in the

offspring. Repeating the same procedure with LPO, one notes that the three good genes 5/2, 4/2 and 2/6 are already transferred to the offspring, that corresponds to the number of good genes to transfer for this objective. Also, the two good genes 5/2 and 2/6 are already present in the offspring for the COLOUR objective, that corresponds to the number of good genes to transfer for this objective. Genes 1/2, 3/2, 2/4 and 6/2 of P_1 are then used to constitute non-orderly list L. In Step 2, assuming that Pos = 7 and that the TWI calculation places the genes in the order 3/2, 2/4, 6/2, 1/2 with equal TWI value on genes 2/4 and 6/2. We then place 3/2 gene in position 8 and favour placing gene 6/2 in position 3 since it occupies this position in P_2 and genes 2/4 and 1/2 are placed in positions 2 and 5 respectively. In this example, genes 1/2 and 6/2 are directly inherited from P_2 since they have the same position in the second parent. The offspring produced from P_1 and P_2 is then E_1= {22651443/64222622}.
A second offspring is created similarly, this time starting with parent P_2.

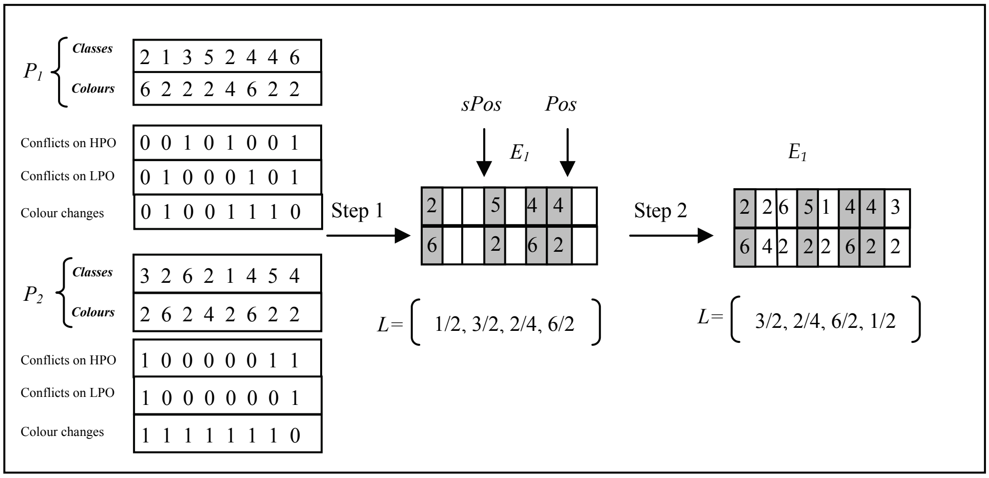


Fig. 2. Schematic of the $NCPX^{MO}$ crossover

3.3 The multi-objective *IBX* crossover operator (*IBX*^{MO})

The IBX^{MO} crossover procedure for the ICSP is inspired by the functioning of the *IBX* for the single objective car sequencing problem (Zinflou *et al.*, 2007) and proceed in three main steps. Step 1 consists in randomly determining two cut-off points for both parents P_1 and P_2. Once these temporary cut-off points are determined, the colours of the preceding cars at the 1st cut-off point and the colour of the cars immediately after the 2nd cut-off point in P_1 are verified so as not to interrupt an ongoing colour subsequence. As long as the colour of the cars located before the 1st cut-off point is the same as the colour of the car located at the cut-off point, we move the cut-off point to the left. Inversely, as long as the colour of the car at the 2nd cut-off point is identical to the colour of the car after that cut-of point, we move the 2nd cut-off point to the right.
In Figure 3, once the cut-off points are set for both parents P_1 = {22351446/46222622} and P_2 = {32421465/24662222}, the genes subsequence {351/222} included between the two cut-off points of the first parent ($a_1 \in P_1$) is directly recopied in the offspring. Thereafter, two non-

orderly lists (L_1 and L_2) are created from subsequence b_3 = {32/24} and b_4 = {465/222} of P_2 and will be used to complete the beginning and the end of offspring E_1. However, during this operation, part of the information may be lost by the addition of duplicates. One effect of this process is that the production requirements will not always be satisfied. In the example in Figure 3, we may thus notice that the production constraints for the 2, 3, 4 and 5 car classes are no longer met. To restore all the genes and to produce exactly c_v cars of the v class, replacement of genes 3/2 and 5/2 (obtained from a_1-a_2) whose number exceeds the production constraints are replaced by genes 4/6 and 2/6 (obtained from a_2-a_1) whose number is now lower than the production constraints. This replacement is done randomly in the second step to adjust the L_1 and L_2 lists.

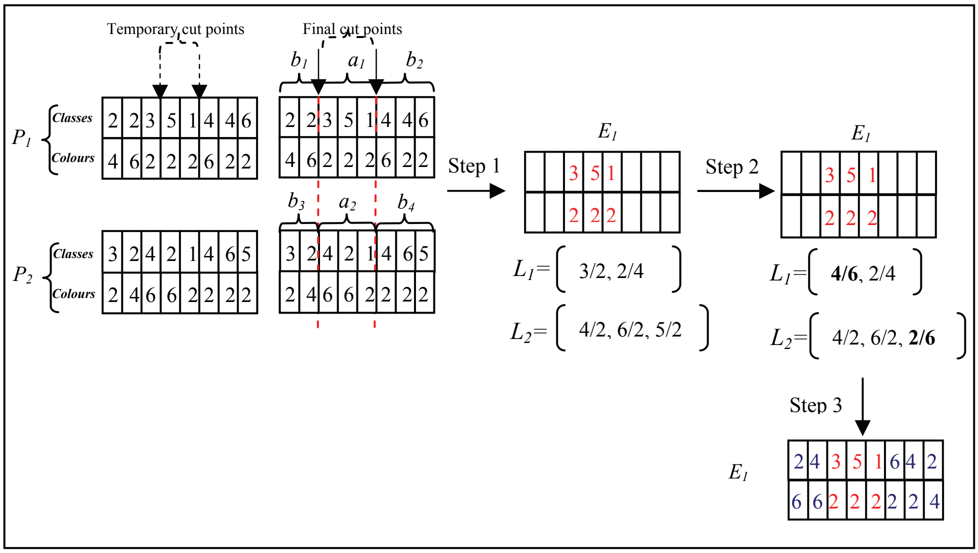


Fig. 3. Schematic of the IBX^{MO} crossover

Finally, the last step consists in rebuilding the beginning and the end of the offspring using the two corrected lists L_1 and L_2 by using TWI as defined in Equation 2. In both cases, the reconstruction starts from the cut-off point towards the beginning or the end of the offspring, depending on the situation. For example, we calculate the TWI for each car $\in L_1$ to reconstruct the beginning of the offspring. The car class v to place is then chosen deterministically in 95% of the cases and in the remaining 5% of the cases the car class v to be placed is chosen probabilistically using the roulette wheel principle (Goldberg, 1989). The second vector of the solution for this position is then completed by the colour associated to this class. We then remove this class from list L_1 and restart the calculations for the next position. The same process is repeated to reconstruct the end of the offspring from list L_2.
A second offspring is created by using the same process, but this time starting from parent P_2.

4. Genetic algorithm for the industrial car sequencing problem

In this section, we present the complete description of the genetic algorithm (GA) used to solve the multi-objective ICSP.

4.1 Representation of the chromosome

As shown previously in Table 1(b), instead of choosing classical bit-string encoding, that seems ill-suited for this type of problem, a chromosome is represented using two vectors of size *Nb_cars* corresponding respectively to the class and the colour of the car.

4.2 Creating the initial population

In the proposed implementation, the individuals of the initial population are generated in two ways: 70 % randomly and 30 % using a greedy heuristic based on the concept of interest. Two greedy heuristics are used according the main objective. If the main objective is to minimize the number of colour changes (COLOUR), the greedy heuristic used is *greedy_colour*. If the main objective is to minimize the number of conflicts on high-priority options (HPO), the greedy heuristic used is *greedy_ratio*. Figure 4 resumes the operation of these two heuristics. Notice that in both cases, one ensures that the individuals produced are feasible solutions.

***greedy _colour* heuristic**

```
1: Start with an individual Y consisting of the D-1
   production day cars
2 : i=1 ; run_length =1
3 : previous_colour = Colours(-1)
4: While there are cars to place
5:     If run_length < $rl_{max}$ and there remain cars with
       previous_colour then
6:         colour = previous_colour
7:         run_lenght ++
8:     Else
9:         Choose randomly previous_colour ≠ colour
10:        run_length = 1
11:    End If
12:    Restricted the choice to the m car classes having the
       selected colour
13:    For each of these m car classes
14:        Evaluate the interest $I_{v,i,COLOUR}$ of adding a car
           class v at position i
15:    End For
16:    Choose randomly a number rnd between 0 and 1
17:    If rnd < 0.95 then
18:        Choose car class v according to Arg Max
           {$I_{v,i,COLOUR}$}
19:        In case of a tie, choose car class v randomly
20:    Else
21:        Choose v using the roulette wheel principle
22:    End If
23:    Y(i) = v / colour
24 :   i=i+1
25: End While
```

***greedy_ratio* heuristic**

```
1: Start with an individual Y consisting of the D-1
   production day cars
2 : i=1; run_length =1
3 : previous_colour = Colours(-1)
4: While there are cars to place
5:     If run_length = $rl_{max}$ then
6:         Exclude the cars for which colour= previous_colour
           from the candidates cars list
7:     End If
8:     For each candidate car class v
9:         Evaluate the interest $I_{v,i,HPO}$ of adding a car class v
           at position i
10:    End For
11:    Choose randomly a number rnd between 0 and 1
12:    If rnd < 0.95 Then
13:        Choose class v according to Arg Max {$I_{v,i,HPO}$}
14:        In case of a tie, break the tie lexicographically by
           using the interest of the second objective and then
           the third objective ($I_{v,i,LPO}$ or $I_{v,i,COLOUR}$). In case of
           ties for the 3 objectives, choose a class randomly
15:    Else
16:        Choose car class v using the roulette wheel
           principle
17:    End If
18:    For the selected car class v, choose colour with Arg
       Max {$I_{v,i,COLOUR}$}. In case of a tie, choose colour
       randomly
19:    Y(i) = v / colour
20:    If run_length = $rl_{max}$ OR colour ≠ previous_colour then
21:        run_length= 1
22 :   Else
23 :        run_length= run_length +1
24 :   End If
25 :   previous_colour=colour
26 :   i=i+1
27: End While
```

Fig. 4. Greedy construction of an individual us the *greedy_colour* or *greedy_ratio* heuristic

Greedy_colour begins with an initial solution composed of the cars planned the previous production day. In fact, to link with the previous production day, we only need to know the maximum value of s_o for all the options and this value determines the length of the sequence required at the end of the previous day to evaluate the current solution. Then, we initialize the counter for positions i at 1, the length of the current colour subsequence (*run_length*) that is also at 1 and the colour of the last car produced the previous day (*previous_colour*) (lines 2-3). The selection iteration process for the next car to place in the building sequence (lines 4-25) begins by selecting a current colour (*colour)* according to rl_{max} and *previous_colour* (lines 5-11). Once the colour of the next car to place is determined, we limit the selection process to the m car classes having that colour. At this step, for each of the m classes, we evaluate the interest $I_{v,i,COLOUR}$ to place a car of class v at the current position i. In 95 % of the cases, the selected class is the one with the largest $I_{v,i,COLOUR}$ (*Arg Max { $I_{v,i,COLOUR}$ }*). For the remaining 5 % of the cases, the car class to place is selected using the roulette wheel principle. Once the colour and the car class are selected, we add the selected car class v and the selected *colour* at position i of sequence Y being built (line 23). This process is thus repeated until an entire sequence of cars is built. The main purpose of this *greedy_colour* heuristic is thus to minimize, in a greedy way, the number of colour changes.
The second proposed construction heuristic, called *greedy_ratio*, also uses a greedy approach to build an individual Y. However, for this heuristic, the main greedy criterion used to select the car to add in the next position of sequence Y being built is the interest $I_{v,i,HPO}$. Just as for the *greedy_colour* heuristic, the *greedy_ratio* procedure starts with an initial solution consisting of cars already sequenced the previous production day. We then initialize the various counters and the colour of the previous car produced on day D-1 the same way as for the *greedy_colour* heuristic. The main loop of the algorithm (lines 4-27) first checks if the maximum length for a subsequence of identical colour, rl_{max}, has not been reached. If rl_{max} is reached, we withdraw all the cars of colour *previous_colour* from the list of classes that may be added at current position i (list of candidate car classes). This step ensures that the generated solution is feasible. Then, for each candidate car class v, we calculate the interest $I_{v,i,HPO}$ to place a car of class v at the current position i according to the HPO objective. Then, the selection of the next car class to place in the sequence is made in 95 % of the cases by selecting the class with the largest $I_{v,i,HPO}$. Note that in case of a tie for the $I_{v,i,HPO}$, the tie is broken using the highest interest for the second objective and then the third objective, respectively. In 5 % of the cases, the car class to place is selected using the roulette wheel principle. Once the car class is selected, we choose the colour of the car to add from the colours available for this class according to $I_{v,i,COLOUR}$. If all the colours for this class of cars are of the same interest, we choose a colour randomly. Thereafter, we add the selected car class and colour at position i in sequence Y being built. Finally, we update the various counters (*run_length* and i) and *previous_colour*. This process is repeated until a complete sequence of cars is done.

4.3 Selection

Several selection strategies could have been considered in the GA based algorithm to solve the multi-objective ICSP. However, since it is easy to implement and that it is efficient for the standard car sequencing problem (Zinflou *et al.*, 2007), the selection procedure chosen to solve the multi-objective ICSP is a binary tournament selection.

4.4 Mutation operator

According to the objective hierarchy, four mutation operators are used here: *reflection*, *random_swap*, *group_exchange* and *block_reflection.* Note that these four operators have often been used in the literature for the ICSP to explore the neighbourhood within a local search method (Solnon *et al.*, 2007). For problems with HPO-COLOUR-LPO and HPO-LPO-COLOUR objective hierarchies, the mutation operators used are reflection and random_swap. A reflection consists in randomly selecting two positions and reversing the subsequence included between these two positions. A random_swap simply consists in randomly exchanging the positions of two cars belonging to different classes. For problems with COLOUR-HPO-LPO objective hierarchy, the mutation operators used are the group_exchange and the block_reflection. The group_exchange mutation consists in randomly exchanging the position of two subsequences of consecutive cars painted the same colour. The block_reflection consists in selecting a subsequence of consecutive cars painted the same colour and in inverting the position of the cars included in this subsequence.

4.5 Replacement strategy

The proposed GA is an elitist approach in that it has explicit mechanisms that keep the best solution found during the search process. To ensure that elitism, the replacement strategy used is a (λ+μ) type of deterministic replacement. In this replacement strategy, the parent and offspring populations are combined and sorted and only the λ best individuals are kept to form the next generation.

```
1: Generate randomly or using the two greedy heuristics of the initial population $POP_0$
2: Evaluate each individual Y $\in$ $POP_0$ and sort $POP_0$
3: While no stop criterion is reached
4:     While | $Q_t$ | < $N$
5:         Choose randomly a number $rnd$ between 0 and 1
6:         If $rnd < p_c$ then
7:             Select parents $P_1$ and $P_2$
8:             Create two offspring $E_1$ and $E_2$ using $NCPX^{MO}$ or $IBX^{MO}$ crossover
9:             Evaluate the generated offspring
10:        else
11:            Generate random migrant using the greedy heuristic
12:        End If
13:        Choose randomly a number $rnd$ between 0 and 1
14:        If $rnd < p_m$ then
15:            Mutate and evaluate the offspring or the migrant
16:        End If
17:        Add $E_1$ and $E_2$ or the migrant to $Q_t$
18:    End While
19:    Sort $Q_t \cup POP_t$
20:    Choose the first $N$ individuals of $Q_t \cup POP_t$ to the next generation $POP_{t+1}$
21:    $t = t + 1$
22: End while
23: Return the best individual found so far
```

Fig. 5. The proposed GA procedure for ICSP

Figure 5 describes the general procedure of our GA for the ICSP. The GA starts building an initial population POP_0 in which each individual $Y \in POP_0$ is evaluated. Then it performs a series of iterations called generations. At each generation *t*, a limited number of individuals are selected to perform recombination according to a crossover probability (p_c). Notice that, occasionally, a new individual is introduced in the offspring population to maintain diversity and avoid stagnation. This individual called *random migrant* is created using the greedy heuristic used to creation the initial population according to the objective hierarchy of the problem to solve. After the crossover, the generated offspring or the migrant is mutated according to mutation probability (p_m). Finally, the current population is updated by selecting the best individuals from the pool of parents (POP_t) and offspring (Q_t). This process is repeated until a stop criterion is reached.

5. Computational experiments

The GA proposed in this chapter was implemented in C++ and compiled with Visual Studio .Net 2005. The computational experiments were run on a Dell Pentium with a Xeon 3.6 GHz processor and 1 Gb of RAM, with Windows XP. For all the experiments performed, the parameters N, p_c, p_m, T_{max} that represent respectively the population size, crossover probability, mutation probability and time limit allowed for the GA are set at the following values: 5, 0.8, 0.35 and 350 seconds. The small population size and the mutation and crossover probabilities were determined using the theoretical results of Goldberg (1989) and the work of Coello Coello and Pulido (2001). According to these authors, a very small population size is sufficient to obtain convergence, regardless of the chromosome length. Thus, the use of a small population with a high crossover probability allows, on one hand, to increase the efficiency of the GA for the ICSP by limiting the computation time required to evaluate the fitness of each individual. In fact, the evaluation of the fitness of a solution for the ICSP requires considerable computation time. On the other hand, a high crossover probability usually allows better exploration of the search space (Grefenstette, 1986). In addition to the difficulties related to the multi-objective nature of the ICSP, a 600 second time limit was set for a Pentium 4/1.6 GHz/Win2000/1 Go RAM computer for the 2005 ROADEF Challenge. To meet this time limit, we set the running time of our GA at 350 seconds, that corresponds roughly to the time limit defined in the Challenge, considering the differences in hardware.

Three versions of our GA will be used for the numerical experiments. The first version integrates the $NCPX^{MO}$ crossover operator ($AG\text{-}NCPX^{MO}$), the second uses the IBX^{MO} crossover operator ($AG\text{-}IBX^{MO}$) and the third version integrates the $NCPX^{MO}$ crossover operator with a local search procedure ($AG\text{-}NCPX^{MO}+LS$).

5.1 Benchmark problems

The performance of the proposed multi-objective GAs is evaluated using three test suites provided by the Renault car manufacturer and that are available from the Challenge website at : http://www.prism.uvsq.fr/~vdc/ROADEF/CHALLENGES/2005/. The first set (SET A) includes 16 sets of data to sequence 334 to 1314 cars that have from 6 to 22 options that create from 36 to 287 cars classes with 11 to 24 different colours. This set allowed to evaluate the teams during the qualification phase and thus to determine the 18 teams who qualified for the next phase of the Challenge. The second set (SET B) consists of a wide range of 45 instances each consisting of 65 to 1270 cars having from 4 to 25 options, with 11 to 339 car classes and 4 to 20 different colours. This set was used by the qualified teams to improve and tune their algorithms. Finally, the last set (SET X) consists of 19 instances having from

65 to 1319 cars to sequence, with 5 to 26 options, 10 to 328 car classes and 5 to 20 different colours. This set remained unknown to the teams until the last phase of the Challenge and was used by the jury to establish the final ranking.
In comparison with the standard car sequencing problem whose largest instances included 400 cars, 5 options and from 18 to 24 car classes, the resolution of the multi-objective ICSP thus represents a large challenge.

5.2 Experimental comparison

To evaluate the performance of the algorithms proposed in this chapter, we compare our results with the best results obtained during the 2005 ROADEF Challenge for the 61 instances of SET A and SET B. All the results of the 2005 ROADEF Challenge are available online from the Challenge website. Thus, Tables 3 to 5 report the comparative results of *GA-NCPX^{MO}*, *GA-IBX^{MO}* and *GA-NCPX^{MO}+LS* with those of the *Challenge Winning Team* and those of the *GLS* (Jaszkiewicz *et al.*, 2004) which is the best evolutionary algorithm proposed during the Challenge. The rank of the solution found by each algorithm for the same instance is listed in Tables 3 to 5 and is based on the results of the 18 qualified teams and the results of the three GAs proposed here .
In these tables, we group instances in three categories:

- those for which the main objective is the minimization of the number of conflicts on high-priority options (HPO) and where the requirements for these high-priority options are considered "easy" according to Renault (Table 3) ;
- those for which the main objective is the minimization of the number of conflicts on high-priority options (HPO) and where the requirements for these high-priority options are considered "difficult" according to Renault (Table 4) ; and
- those for which the main objective is the minimization of the number of colour changes (COLOUR) (Table 5).

Each row of Tables 3 to 5 indicates the name of the instance, the value and the rank of the solution found respectively by the *Winning Team*, the *GLS* (Jaszkiewicz *et al.*, 2004), the *GA-IBX^{MO}*, the *GA-NCPX^{MO}* and the *GA-NCPX^{MO}+LS*. The best results obtained for each instance are highlighted in bold in the different tables. It is important to note that as for the Challenge results, the GAs proposed were run once only and what we report is the solution value obtained for this execution. The results reported in the different tables indicate the objectives weighted sum value ($F(X)$) of the solution as calculated in Equation 1.
Table 3 reports the results for instances with "easy" high-priority options according to Renault. These instances have two possible hierarchies that are HPO-LPO-COLOUR or HPO-COLOUR-LPO. By examining the results of Table 3, one may note that *GA-NCPX^{MO}* outperforms *GA-IBX^{MO}* for all the instances of SET A and SET B, except for instance 028_ch2_S23_J3 with HPO_COLOUR_LPO objective hierarchy where the two algorithms obtain equal results. These results seem to highlight the superiority of the *NCPX^{MO}* crossover operator over the *IBX^{MO}* crossover operator for the ICSP. The best performance of the *NCPX^{MO}* crossover operator may probably be explained by its ability to use information about non-conflict positions. Thus, this crossover is able to do a better search intensification during the allowed time.
Except for instance 028_ch2_S23_J3 with HPO_LPO_COLOUR objective hierarchy, that is trivially solved by all algorithms, *GA-IBX^{MO}* ranks between 11th and 19th while *GA-NCPX^{MO}* ranks between 1st and 17th according to the instances. It should be noted that, contrary to

most algorithms of the Challenge, *GA-IBX^{MO}* and *GA-NCPX^{MO}* do not use a local search procedure in their algorithm.

By comparing the results of *GA-NCPX^{MO}* and *GA-IBX^{MO}* to those of *GLS*, one may note for SET A that *GLS* globally outperforms *GA-IBX^{MO}* but *GA-NCPX^{MO}* clearly outperforms *GLS*. Indeed, *GLS* outperforms *GA-IBX^{MO}* for 3 instances of Set A, is worse for one instance while it obtains identical results for the remaining instance. By contrast, *GLS* is worse than *GA-NCPX^{MO}* for 4 of the 5 instances of SET A shown in Table 3. These results are confirmed with a few slight differences for the instances of SET *B*. Thus, *GLS* outperforms *GA-IBX^{MO}* for 10 instances, is worse for 7 instances while obtaining identical results for the remaining instance. Compared to *GA-NCPX^{MO}*, *GLS* achieves better results for 6 instances, is worse for 8 instances while obtaining identical results for the 4 remaining instances. We may therefore notice a slight advantage for *GA-NCPX^{MO}* for the instances of SET B with easy high-priority options. These results are very promising considering that *GLS* is a memetic algorithm, that is, an approach hybridizing GA with local search method.

When we now compare the results of *GA-NCPX^{MO}* and *GA-IBX^{MO}* to those of the *Winning Team* for the 2005 ROADEF Challenge, one may notice that the results of the two proposed GAs are clearly lower than the results of the *Winning Team* in terms of solution quality. We believe that this gap may be explained by the lack of intensification of the search for this type of approach. By combining *GA-NCPX^{MO}* with a local search procedure inspired from the one proposed by Estellon *et al.* (2007) and using the mutation operators presented in Section 4.4 to explore the neighbourhood, we obtain the results shown in the last column of Table 3. We mention here that *GA-NCPX^{MO}+LS* was executed with the same time limit as the other algorithms presented in this chapter. We observe that adding the local search procedure clearly improves the performance of the algorithm. Indeed, *GA-NCPX^{MO}+LS* clearly outperforms *GA-NCPX^{MO}* and achieves competitive results compared to those of the *Challenge Winning Team* for all instances of SET A with easy high-priority options. In fact, *GA-NCPX^{MO}+LS* ranks first for all these instances and even finds new minimums for instance 022_3_4 with HPO_COLOUR_LPO objective hierarchy and for instance 25_38_1 with HPO_LPO_COLOUR objective hierarchy. For the instances of SET B, *GA-NCPX^{MO}+LS* obtains similar results as those of the *Challenge Winning Team* for 10 of the 16 instances. For the remaining instances, we observe a small gap that comes from the results of the second or the third objective. Indeed, *GA-NCPX^{MO}+LS* is always ranked between 1st and 3rd, except for instance 064_ch1_S22_J3 with HPO_COLOUR_LPO objective hierarchy where it ranks 7th.

Table 4 reports the results obtained by the different algorithms for the instances of SET A and SET B considered by Renault as "difficult" high-priority options. The two possible objective hierarchies for these instances are HPO-LPO-COLOUR and HPO-COLOUR-LPO. We may notice again that *GA-NCPX^{MO}* clearly outperforms *GA-IBX^{MO}*. Therefore, for the instances of SET A, *GA-NCPX^{MO}* obtains better results than *GA-IBX^{MO}* for 6 of the 7 instances while *GA-IBX^{MO}* is better for the only remaining instance. The results are quite the same for the instances of SET B where, this time, *GA-NCPX^{MO}* always outperforms *GA-IBX^{MO}*. *GA-IBX^{MO}* ranks between 12th and 20th while *GA-NCPX^{MO}* ranks between 1st and 19th depending on the instances. Despite the fact these two algorithms do not use a local search procedure, they are quite competitive with the global results of the teams that qualified for the Challenge. However, for the instances with easy high-priority options, we notice that the results of the two proposed algorithms are not competitive with those of the *Challenge Winning Team*.

	Winning Team	*GLS (rank)*	*AG-IBX^{MO} (rank)*	*AG-NCPX^{MO} (rank)*	*AG-NCPX^{MO} +LS (rank)*
SET A					
HPO_COLOUR_LPO					
022_3_4	**31001 (1)**	37000 (14)	32022 (11)	32001 (8)	**31001 (1)**
025_38_1	231452 (4)	262460 (15)	262460 (15)	231772 (6)	**229295 (1)**
064_38_2_ch1	**112759 (1)**	139757 (15)	184775 (17)	164760 (16)	**112759 (1)**
064_38_2_ch2	**34051 (1)**	36056 (15)	37156 (16)	34052 (8)	**34051 (1)**
HPO_ LPO_COLOUR					
025_38_1	99720 (2)	200711 (10)	270686 (14)	150767 (6)	**97076 (1)**
SET B					
HPO_COLOUR_LPO					
022_S22-J1	**19144 (1)**	23144 (13)	21174 (12)	20176 (9)	**19144 (1)**
025_S22-J3	**172180 (1)**	281877 (20)	264156 (19)	222711 (13)	179378 (3)
028_ch_S22_J2	**54049124 (1)**	54059164 (13)	54072436 (19)	54063113 (14)	**54049124 (1)**
028_ch2_S23_J3	**4071 (1)**	**4071 (1)**	5078 (17)	**4071 (1)**	**4071 (1)**
039_ch1_S22_J4	**78089 (1)**	92308 (17)	82731 (14)	79128 (7)	78220 (3)
039_ch3_S22_J4	189146 (4)	199223 (17)	195718 (15)	192160 (12)	189122 (2)
048_ch1_S22_J3	**161378 (1)**	186438 (18)	180440 (16)	170377 (12)	161401 (2)
064_ch1_S22_J3	**130187 (1)**	158222 (14)	183149 (19)	177376 (17)	134368 (7)
064_ch2_S22_J4	**130069 (1)**	**130069 (1)**	130088 (14)	**130069 (1)**	**130069 (1)**
HPO_LPO_COLOUR					
022_S22_J1	**3109 (1)**	3138 (12)	3191 (14)	3186 (13)	**3109 (1)**
025_S22_J3	**3912479 (1)**	3926649 (11)	4041787 (19)	3928619 (12)	**3912479 (1)**
028_ch1_S22_J2	54003079 (4)	54021114 (12)	54065112 (14)	54017116 (9)	54003077 (2)
028_ch2_S23_J3	**70006 (1)**	**70006 (1)**	**70006 (1)**	**70006 (1)**	**70006 (1)**
039_ch1_ S22_J4	**29117 (1)**	29385 (16)	29375 (15)	29272 (11)	**29117 (1)**
039_ch3_ S22_J4	**197 (1)**	276 (12)	315 (17)	290 (13)	201 (2)
048_ch1_S22_J3	**200 (1)**	298 (15)	367 (19)	298 (15)	203 (3)
064_ch1_S22_J3	**182 (1)**	1359 (18)	421 (15)	240 (10)	**182 (1)**
064_ch2_S22_J4	**69130 (1)**	69131 (9)	69161 (19)	69132 (11)	**69130 (1)**

Table 3. Results of the *Winning Team*, *GLS*, *GA-IBX^{MO}*, *GA-NCPX^{MO}* and *GA-NCPX^{MO}+LS* for "easy" high-priority options instances with HPO as the main objective

If we now compare the performance of *GA-IBX^{MO}* and *GA-NCPX^{MO}* with those of *GLS*, we notice that *GLS* clearly outperforms *GA-IBX^{MO}*, both for the instances of SET A and SET B. Thus, *GLS* obtains better results than *GA-IBX^{MO}* for 6 of the 7 instances of SET A and for 11 of 12 instances of SET B. We believe that the poor performance of *GA-IBX^{MO}* may be explained by the difficulty of these instances which, combined with the time limit, more highlight the lack in terms of intensification of the search process of the crossover operator. However, when we compare the results of *GLS* with those of *GA-NCPX^{MO}*, we observe essentially the same results as those obtained in Table 3 for the instances of SET A. Indeed, *GA-NCPX^{MO}* outperforms *GLS* for 6 of the 7 instances of SET A. But, for the SET B instances, the results slightly favour *GLS*. Thus, *GA-NCPX^{MO}* is better than *GLS* for 4 instances, is worse for 5 instances while obtaining identical results for the 3 remaining instances.
These results confirm the previous observations made and once again highlight the need to incorporate more explicit intensification mechanisms in our GA. By analyzing the results of adding a local search procedure to *GA-NCPX^{MO}* (last column of Table 4), we notice a clear

improvement of the performance for all the instances. In fact, the results of *GA-NCPX^{MO}+LS* are competitive with those of the *Challenge Winning Team* by obtaining equal or better results for 9 of the 19 instances of the two sets, while obtaining significantly closer results for the remaining instances. *GA-NCPX^{MO}+LS* always ranks between 1st and 6th except for instance 024_38_5 with HPO_COLOUR_LPO hierarchy where it ranks 12th. Compared to *GLS*, *GA-NCPX^{MO}+LS* always obtains better result except for two instances for which the two algorithms obtain identical results.

	Winning Team	*GLS (rank)*	*AG-IBX^{MO} (rank)*	*AG-NCPX^{MO} (rank)*	*AG-NCPX^{MO} +LS (rank)*
SET A					
HPO_COLOUR_LPO					
024_38_3	**4249083 (1)**	4327229 (12)	4471615 (15)	4304266 (9)	4256186 (3)
024_38_5	**4280079 (1)**	7347154 (17)	26015122 (18)	6501289 (15)	4392151 (12)
039_38_4_ch1	**13129000 (1)**	15179000 (11)	17201000 (14)	14122000 (8)	**13129000 (1)**
048_39_1	175615 (4)	202740 (13)	3286796 (18)	191750 (11)	174690 (2)
HPO_LPO_COLOUR					
024_38_3	**4000306 (1)**	4041506 (8)	5035482 (13)	6015504 (14)	4033403 (6)
024_38_5	**4034309 (1)**	6080457 (18)	58072610 (17)	5068407 (15)	4045349 (6)
048_39_1	**61290 (1)**	83403 (11)	246439 (17)	81406 (10)	63323 (4)
SET B					
HPO_COLOUR_LPO					
023_S23_J3	**48310008 (1)**	48349006 (10)	48465018 (18)	48429000 (13)	48313000 (4)
024_V2_S22_J1	**1074299068 (1)**	1100352464 (8)	1124857475 (16)	1106420563 (10)	1078310188 (4)
029_HPO_ S21_J6	**35167170 (1)**	35192150 (14)	35187151 (12)	35173150 (6)	35168171 (3)
035_ch1_ S22_J3	67036064 (7)	67037063 (12)	67044083 (20)	67037063 (12)	**67036061 (1)**
035_ch2_ S22_J3	**385187351 (1)**	**385187351 (1)**	385187353 (17)	**385187351 (1)**	**385187351 (1)**
048_ch2_ S22_J3	**3094029 (1)**	3126017 (15)	3131944 (17)	3124086 (12)	3094030 (2)
HPO_LPO_COLOUR					
023_S23_J3	48000317 (3)	48000406 (10)	65000453 (19)	48000496 (15)	**48000316 (1)**
024_V2_S22_J1	**1074850430 (1)**	1097921524 (9)	1179022413 (19)	1113997557 (13)	1075884555 (4)
029_HPO _S21_J6	**37150167 (1)**	37150194 (12)	37150402 (18)	37150182 (9)	**37150167 (1)**
035_ch1 _S22_J3	**67052049 (1)**	67052052 (9)	67059057 (19)	67052052 (9)	**67052049 (1)**
035_ch2 _S22_J3	**385341205 (1)**	**385341205 (1)**	388350188 (20)	385353192 (19)	**385341205 (1)**
048_ch2_S22_J3	**3000337 (1)**	3000375 (14)	3000405 (17)	3000356 (7)	**3000337 (1)**

Table 4. Results of the *Winning Team*, *GLS*, *GA-IBX^{MO}*, *GA-NCPX^{MO}* and *GA-NCPX^{MO}+LS* for "difficult" high-priority options instances with HPO as the main objective

Table 5 lists the results of the different algorithms for the instances of SET A and SET B with COLOUR-HPO-LPO objective hierarchy. By comparing first *GA-IBX^{MO}* and *GA-NCPX^{MO}*, we observe once again that *GA-NCPX^{MO}* globally outperforms *GA-IBX^{MO}*. *GA-NCPX^{MO}* obtains better results for 18 instances out of 19 and identical results for the remaining instance. However, contrary to the previous observation, the gap between the two algorithms is smaller for this group of instances. Except for three instances, the two algorithms give the same value for the main objective. For these instances, the gap between the two algorithms is observed for the second and third objective. However, we notice again that the results of the two algorithms are not competitive with those of the *Challenge*

Winning Team, except for instance 35_ch2_S22_J4 with COLOUR_HPO_LPO objective hierarchy for which all algorithms obtain the same result. *GA-IBX*^{MO} ranks between 12th and 20th while *GA-NCPX*^{MO} ranks between 1st and 17th. We also notice that, except for one instance for *GA-NCPX*^{MO} and three instances for *GA-IBX*^{MO}, the two algorithms obtain the same value for the main objective as the *Challenge Winning Team* did. We can make this conclusion considering that the weight of the main objective is set at 1000000 and that the gap between the algorithms is less than this value.

	Winning Team	***GLS (rank)***	***AG-IBX***^{MO} ***(rank)***	***AG-NCPX***^{MO} ***(rank)***	***AG-NCPX***^{MO} ***+LS (rank)***
SET A					
COLOUR_HPO_LPO					
022_3_4	**11039001 (1)**	11041001 (15)	11039131 (12)	11039098 (11)	**11039001 (1)**
039_38_4 _ch1	68161000 (3)	68265000 (15)	68265000 (15)	68249000 (12)	**68155000 (1)**
064_38_2 _ch1	**63423782 (1)**	63435799 (15)	63443831 (17)	**63423782 (1)**	**63423782 (1)**
064_38_2_ch2	**27367052 (1)**	**27367052 (1)**	27367067 (15)	**27367052 (1)**	**27367052 (1)**
SET B					
COLOUR_HPO_LPO					
022_S22_J1	**13022148 (1)**	13022154 (11)	13022189 (19)	13022178 (17)	**13022148 (1)**
023_S23_J3	**51327031 (1)**	54349063 (21)	51735264 (20)	51393130 (17)	51343070 (9)
024_V2_S22_J1	**134023158 (1)**	135226676 (20)	134902740 (19)	134230457 (14)	134057341 (4)
025_S22_J3	**126127589 (1)**	133129840 (21)	126300350 (18)	126136839 (12)	**126127589 (1)**
028_ch1_S22_J2	38098201 (4)	38098251 (9)	38099330 (16)	38098334 (12)	**38098188 (1)**
028_ch2_S23_J3	**4000071 (1)**	**4000071 (1)**	5000078 (18)	**4000071 (1)**	**4000071 (1)**
029_ S21_J6	**52711171 (1)**	52755179 (14)	52905570 (20)	52763341 (15)	52717428 (8)
035_ch1_S22_J3	**6156090 (1)**	6156092 (10)	6156109 (18)	6156092 (10)	**6156090 (1)**
035_ch2_S22_J3	**7651671 (1)**	**7651671 (1)**	**7651671 (1)**	**7651671 (1)**	**7651671 (1)**
039_ch1_S22_J4	**55045096 (1)**	55045235 (9)	55046737 (18)	55045235 (9)	**55045096 (1)**
039_ch3_ S22_J4	**59214671 (1)**	59214698 (12)	59214783 (15)	59214681 (9)	**59214671 (1)**
048_ch1_ S22_J3	**64115670 (1)**	64135847 (14)	64153806 (15)	64124687 (12)	**64115670(1)**
048_ch2_ S22_J3	**58283180 (1)**	58288194 (12)	58312194 (19)	58290183 (13)	**58283180 (1)**
064_ch1_ S22_J3	**62095288 (1)**	62108458 (10)	63116379 (19)	62113381 (12)	62097307 (3)
064_ch2_ S22_J4	**31052178 (1)**	31052184 (9)	32052158 (16)	31053188 (13)	**31052178 (1)**

Table 5. Results of the *Winning Team*, *GLS*, *GA-IBX*^{MO}, *GA-NCPX*^{MO} and *GA-NCPX*^{MO}*+LS* for instances with COLOUR as the main objective

By comparing the results of our algorithms with those of *GLS*, we again notice that *GLS* outperforms *GA-IBX*^{MO} for 2 of 4 instances of SET A, is worse for only one instance while obtaining an identical result for the remaining instance. However, for the SET B instances, *GLS* clearly outperforms *GA-IBX*^{MO} by obtaining better results for 11 instances, worse results for 3 instances and identical results for the remaining instance. By comparing the results of *GA-NCPX*^{MO} with those of *GLS*, one notes that *GA-NCPX*^{MO} obtains better results for all instances of SET A except one where the two algorithms achieve identical results. For the SET B instances, *GA-NCPX*^{MO} obtains better results than *GLS* for 5 instances, is worse for 6 instances while obtaining identical results for the 4 remaining instances. Again, we observe very close performance between the two algorithms.

By now comparing the results of the two GAs to those of the *Challenge Winning Team*, we notice on one hand that *GA-NCPX*^{MO} always reaches the same value for the main objective.

On the other hand, *GLS* doesn't always reach these values. *GLS* even obtains the worst solution for instances 023_S23_J3 and 025_S22_J3 with COLOUR_HPO_LPO objective hierarchy.

By analysing the results of *GA-NCPX^{MO}+LS*, we observe a clear performance improvement for all the instances. Thus, for SET A instances, *GA-NCPX^{MO}+LS* always obtains identical or better results than those of the *Challenge Winning Team*. For SET B instances, *GA-NCPX^{MO}+LS* obtains identical or better results than those of the *Challenge Winning Team* for 11 of the 15 instances. *GA-NCPX^{MO}+LS* always ranks between 1st and 4th except for instances 023_S23_J3 and 029_S21_J6 with COLOUR_HPO_LPO objective hierarchy, where it ranks 9th and 8th respectively. Compared to *GLS*, *GA-NCPX^{MO}+LS* gets better results for 16 of the 19 instances while obtaining identical results for the 3 remaining ones.

Finally, Table 6 gives the results of the different algorithms for the 19 instances of SET X that was used in the 2005 ROADEF Challenge to determine the final ranking. Here, instead of executing the algorithms once as we did in the previous results, we executed the algorithms 5 times as was done for the qualified teams in this phase of the Challenge. The values reported in this table are thus the average results of 5 runs.

	Winning Team	*GLS (rank)*	*AG-IBX^{MO} (rank)*	*AG-NCPX^{MO} (rank)*	*AG-NCPX^{MO} +LS (rank)*
SET X					
HPO_COLOUR_LPO					
023_S49_J2	**192466 (1)**	246268.20 (17)	246268.40 (18)	211879 (12)	193077 (3)
024_S49_J2	**337006 (1)**	421425 (8)	27046420.20 (18)	506015 (11)	346202.20 (2)
029_S49_J5	**110442.60 (2)**	120855 (11)	150969.20 (17)	123029.20 (12)	111093.20 (3)
034_VP_S51_J1_J2_J3	**56386.80 (1)**	76217.60 (17)	74354.20 (15)	66750 (12)	57577.40 (5)
034_VU_S51_J1_J2_J3	8087037 (4)	8091450.20 (10)	8112049 (16)	8103064 (15)	**8087035.80 (1)**
039_CH1_S49_J1	**69239 (1)**	69455.60 (6)	69705 (9)	69479.60 (7)	69355.20 (2)
039_CH3_S49_J1	**231030.20 (2)**	239593.20 (16)	250670 (17)	235475.40 (13)	231030.40 (3)
048_CH1_S50_J4	**197044.80 (3)**	206509.60 (16)	207634 (17)	204182 (14)	197045.40 (4)
048_CH2_S49_J5	**31077916.20 (1)**	31104598.80 (12)	31128931 (18)	31106266.2 (13)	31078317.20 (2)
064_CH1_S49_J1	**61187229.80 (1)**	61229518.80 (12)	61309246.20 (20)	61223429 (10)	61190429 (2)
064_CH2_S49_J4	**37000 (1)**	40400 (14)	42000 (15)	39000 (12)	**37000 (1)**
655_CH1_S51_J2_J3_J4	**30000 (1)**	**30000 (1)**	**30000 (1)**	**30000 (1)**	**30000 (1)**
655_CH2_S52_J1_J2_S01_J1	**153034000 (1)**	153035200 (8)	153047000 (12)	153041000 (11)	**153034000 (1)**
COLOUR_HPO_LPO					
022_S49_J2	**12002003 (1)**	**12002003 (1)**	12002008 (16)	**12002003 (1)**	**12002003 (1)**
035_CH1_S50_J4	**5010000 (1)**	-	**5010000 (1)**	**5010000 (1)**	**5010000 (1)**
035_CH2_S50_J4	**6056000 (1)**	**6056000 (1)**	**6056000 (1)**	**6056000 (1)**	**6056000 (1)**
LPO_COLOUR_HPO					
025_ S49_J1	160407.60 (2)	189390.20 (15)	188118.20 (13)	176454.60 (10)	**160407.20 (1)**
028_CH1_S50_J4	36370094 (4)	36377907.20 (5)	49863125.80 (20)	39634315.20 (12)	**36360092.40 (2)**
028_CH2_S51_J1	**3 (1)**	**3 (1)**	**3 (1)**	**3 (1)**	**3 (1)**

Table 6. Results of the *Winning Team*, *GLS*, *GA-IBX^{MO}*, *GA-NCPX^{MO}* and *GA-NCPX^{MO}+LS* for SET X instances

When we compare the average results of *GA-IBX^{MO}* and *GA-NCPX^{MO}*, we again notice for this set that *GA-NCPX^{MO}* clearly outperforms *GA-IBX^{MO}* by obtaining better results except for 4 instances for which the two algorithms obtain the same average results. We also notice for these 4 instances that the two algorithms always find the same solution for each run. Moreover, the results obtained by the two GAs are the same as those of the *Winning Team*.

By looking more closely at the characteristics of these 4 instances, we notice that they are small instances where the number of cars to schedule is between 65 and 376. These small sizes probably explain why the two algorithms solve these 4 instances trivially. As shown in the previous results, the gap between the two algorithms seems to be related to the size of the instances. Indeed, *GA-IBX^{MO}* seems to have more difficulty to converge towards a good solution for large instances. This situation is again confirmed using instance 024_S49_J2 with HPO_COLOUR_LPO objective hierarchy and 1319 cars to schedule. For this instance, the gap between the average results of the two algorithms for the main objective is over 26 conflicts. Except for the 4 small size instances solved trivially, *GA-IBX^{MO}* ranks between 9th and 20th while *GA-NCPX^{MO}* ranks between 7th and 15th.
If we now compare the results of our two algorithms to those of *GLS*, we observe similar results to those obtained for SET A et SET B. *GA-IBX^{MO}* is worse than *GLS* for 13 instances, better for 3 instances while identical for the 3 other instances. We notice that among the 3 instances for which *GA-IBX^{MO}* achieves better average results than *GLS*, there is one instance (035_CH1_S50_J4 with COLOUR_HPO_LPO hierarchy) for which *GLS* did not provide a feasible solution during this phase of the Challenge. When we now compare *GLS* to *GA-NCPX^{MO}*, we notice that *GA-NCPX^{MO}* outperforms *GLS* for 8 instances, is worse for 7 instances while identical for the 4 remaining instances.
We also notice that the results of *GA-IBX^{MO}* and *GA-NCPX^{MO}* are not competitive with the average results of the *Winning Team*. However, by adding a local search procedure to *GA-NCPX^{MO}*, we considerably improve the performance of the algorithm by obtaining the best average results for 10 instances while obtaining very close average results for the other instances. *GA-NCPX^{MO}+LS* ranks between 1st and 5th for all the instances of SET X.
Now, to compare the performance of the proposed approaches with the results of the teams that qualified for the Challenge, we used the ranking procedure described in the Challenge description, that consists in calculating a *mark* for each instance of SET X according to Equation 5. The *mark* of each algorithm is calculated according to the best and the worst solution found by the 18 teams that qualified for the Challenge and the 3 proposed algorithms. The score is a normalized measure of solution quality that necessarily lies between 0 and 1.

$$mark(A\lg o) = \frac{result_{A\lg o} - Best\_result}{Best\_result - worst\_result} \tag{5}$$

In Equation 5, *Best_result* and *Worst_result* indicate respectively the best and the worst average result found for an instance while *result$_{Algo}$* indicates the average result found by the algorithm for which we compute the mark for the same instance. Then, each row of Table 7 lists the mark of the *Winning Team*, the *GA-IBX^{MO}*, the *GA-NCPX^{MO}* and *GA-NCPX^{MO} +LS* for each instance of SET X. The last row of this table lists the total mark of each algorithm for the whole set. On analysing the results of Table 7, we notice that they confirm the results of Tables 3 to 6, namely that *GA-NCPX^{MO}* is a better performer than *GA-IBX^{MO}* and that *GA-NCPX^{MO}+LS* is the best performer compared to the two other algorithms. It is important to mention that, according to the final rank of the Challenge that is published by the organizers and that is available online from the Challenge website, *GLS* ranks 13th with a mark of 16.8937 while the *Winning Team* has a mark of 18.9935. Based on these results, we may conclude that the difference between the results of our best genetic approach and those of the *Winning Team* is rather small (0.0345). We also notice that both *GA-NCPX^{MO}* obtain a better mark than *GLS*, with and without local search procedure. We may then conclude that the methods proposed in this chapter achieve competitive results for the multi-objective ICSP. Thus, we demonstrate that GAs are well suited to address this category of problem if they incorporate specific knowledge of the problem to design dedicated genetic operators.

	Marks			
SET X	***Winning Team***	***AG-IBX^{MO}***	***AG-NCX^{MO}***	***AG-NCX^{MO}+LS***
HPO_COLOUR_LPO				
023_S49_J2	1	0.5575	0.8403	0.9950
024_S49_J2	1	0.4605	0.9966	0.9998
029_S49_J5	0.9980	0.4249	0.8200	0.9888
034_VP_S51_J1_J2_J3	0.9956	0.7949	0.8799	0.9823
034_VU_S51_J1_J2_J3	1	0.9998	0.9999	1
039_CH1_S49_J1	1	0.9755	0.9873	0.9939
039_CH3_S49_J1	0.9999	0.6368	0.9178	1
048_CH1_S50_J4	0.9999	0.9952	0.9968	1
048_CH2_S49_J5	1	0.9868	0.9927	0.9999
064_CH1_S49_J1	1	0.9799	0.9940	0.9995
064_CH2_S49_J4	1	0.8588	0.9435	1
655_CH1_S51_J2_J3_J4	1	1	1	1
655_CH2_S52_J1_J2_S01_J1	1	0.9999	0.9999	1
COLOUR_ HPO_LPO				
022_S49_J2	1	0.9999	1	1
035_CH1_S50_J4	1	1	1	1
035_CH2_S50_J4	1	1	1	1
HPO_LPO_COLOUR				
025_S49_J1	1	0.9983	0.9990	1
028_CH1_S50_J4	0.9999	0.9553	0.9891	0.9999
028_CH2_S51_J1	1	1	1	1
Total	**18.9935**	**16.6241**	**18.3569**	**18.9590**

Table 7. Marks of the *Winning Team, GA-IBX^{MO}*, *GA-NCPX^{MO}* and *GA-NCPX^{MO}+LS* for SET X instances.

6. Conclusion

In this chapter, we have introduced a GA based on two specialized crossover operators dedicated to the multi-objective nature of the ICSP proposed by French automobile manufacturer Renault for the ROADEF 2005 Challenge. If GAs are known to be well suited for multi-objective optimization (Barichard, 2003; Basseur, 2004; Zinflou *et al.*, 2006), few researchers and industrials decided to use this category of algorithms to solve the ICSP. Among the 18 teams that qualified for the second phase of the Challenge, only one proposed a genetic algorithm based approach. This situation may be explained by the difficulty in defining specific and efficient genetic operators that take into account the specificities of the problem. The approach proposed in this chapter is essentially based on adapting highly specialized genetic crossover operators to the specificities of the industrial version of the single objective car sequencing problem, for which we have three conflicting objectives to optimize. The numerical experiments allowed us to demonstrate the efficiency of the proposed approach for this industrial problem. A natural conclusion of these experimental results is that GAs may be robust and efficient alternative to solve the multi-objective ICSP. These results also again highlight the importance of incorporating specific problem knowledge into genetic operators, even if classical genetic operators could be used. We are also aware of the fact that having known the solutions found by the algorithms of the different qualified teams has facilitated improving and tuning our algorithms. However, the main purpose of this study was to demonstrate that GAs can be an efficient alternative to solve this kind of industrial problem.

The lexicographical treatment of the objectives proposed by Renault is such that it can eliminate several "interesting" solutions for the manufacturer. Indeed, the relaxation of the importance granted to the main objective can highlight other attractive solutions for the company. For example, if an additional violation on the HPO objective allows to avoid 5 colour changes, the production scheduler could then be interested to a such solution to make his final schedule. We therefore believe that the industrial problem introduced by Renault would benefit to be treated to obtain so-called "compromise solutions". In this context, the GAs proposed in this chapter represent very interesting alternatives to find these compromise solutions. In fact, GAs are well suited for multi-objective optimization in the Pareto sense and these approaches have proven their ability to generate compromise solutions in a single optimization step. Since the mid-nineties, an increasing number of approaches exploit the principle of dominance (Zitzler and Thiele, 1998; Deb, 2000; Knowles and Corne, 2000a; Knowles and Corne, 2000b; Coello Coello and Pulido, 2001) in the Pareto sense as defined by Goldberg (1989). These evolutionary multi-objective algorithms use the concepts of dominance, niches and elitism (Deb, 2000; Knowles and Corne, 2000b; Deb and Goel, 2001; Zitzler *et al.*, 2001). The NSGAII algorithm (Deb, 2000), the SPEA2 algorithm (Zitler *et al.*, 2001) and the PMS^{MO} algorithm (Zinflou *et al.*, 2007) are recognized as amongst the best performing of the elitist multi-objective evolutionary algorithms. These algorithms are said to be elitist because they include one or several mechanisms allowing the memorization of the best solutions found during the execution of the GA.

For future work, we will use this type of approaches to consider the objectives simultaneously, without assigning priority or weight. A set of compromise solutions may then be found for comparison to the solution by considering the objectives in lexicographical order. It will thus be possible to highlight different solutions that are much more financially interesting for a manufacturer and that are better suited to industrial reality.

7. References

Barichard, V. (2003). *Approches hybrides pour les problèmes multiobjectifs*, Ph.D. Thesis, Université d'Angers, France.

Basseur, M. (2004). *Conception d'algorithmes coopératifs pour l'optimisation multi-objectifs : Application aux problèmes d'ordonnancement de type flow-shop*, Ph.D. Thesis, Université des Sciences et Technologies de Lille, France.

Benoit, T. (2007). Soft car sequencing with colors: Lower bounds and optimality proofs, *European Journal of Operational Research*: doi:10.1016/j.ejor.2007.04.035.

Briant, O.; Naddef, D. & Mounié, G. (2007). Greedy approach and multi-criteria simulated annealing for the car sequencing problem, *European Journal of Operational Research*: doi:10.1016/j.ejor.2007.04.052.

Coello Coello, A. C. & Pulido, G. T. (2001). Multiobjective optimization using a micro-genetic Algorithm, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001),* 274-282, San Francisco, California.

Cordeau, J.-F.; Laporte, G. & Pasin, F. (2007). An iterated local search heuristic for the car sequencing problem, *European Journal of Operational Research*: doi:10.1016/j.ejor.2007.04.048.

Deb, K. (2000). A fast elitist non-dominated sorting genetic algorithm for multiobjective optimization : NSGA II, *Proceedings of Parallel problem Solving form Nature – PPSN VI,* Lecture Notes in Computer Science, M. Schoenauer *et al.* (Eds), Springer, 849-858.

Deb, K. & Goel, T. (2001). Controlled elitist non-dominated sorting genetic algorithms for better convergence, *Proceedings of Evolutionary Multi-Criterion Optimization*, Lecture Notes in Computer Science 1993, E. Zitler *et al.* (Eds), Springer-Verlag.

Dincbas, M.; Simonis, H. & van Hentenryck, P. (1988). Solving the car sequencing problem in constraint logic programming, *Proceedings of the European Conference on Artificial Intelligence (ECAI-88)*, Munich, Germany, Pitmann Publishing, London, 290-295.

Estellon, B. ; Gardi, F. & Nouioua, K. (2005). Ordonnancement de véhicules: une approche par recherche locale à grand voisinage, *Proceedings of Journées Francophones de Programmation par Contraintes*, 21-28, Lens, France.

Estellon, B.; Gardi, F. & Nouioua, K. (2007). Two local search approaches for solving real-life car sequencing problem, *European Journal of Operational Research,* doi:10.1016/j.ejor.2007.04.043.

Gagné, C.; Gravel, M. & Price, W. L. (2006). Solving real car sequencing problems with ant colony optimization, *European Journal of Operational Research*, 174(3), 1427-1448.

Gavranović, H. (2007). Local search and suffix tree for car-sequencing problem with colors, *European Journal of Operational Research*, doi:10.1016/j.ejor.2007.04.051.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Massachusetts, Addison-Wesley, Reading.

Gottlieb, J.; Puchta, M. & Solnon, C. (2003). A study of greedy, local search and ant colony optimization approaches for car sequencing problems, *Computers Science*, 246-257.

Grefenstette, J. J. (1986). Optimization of Control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1), 122-128.

Jaszkiewicz, A.; Kubiak, M. & Kominek, P. (2004). The application of the genetic local search algorithm to Car Sequencing Problem, *Proceedings of the 7th National Conference on Evolutionary Algorithms and Global Optimization*, Kazimierz Dolny, Poland.

Knowles, J. D. & Corne, D. W. (2000a). M-PAES : A Memetic Algorithm for Multiobjective Optimization, *Proceedings of the 2000 Congress on Evolutionary Computation*, 325-332.

Knowles, J. D. & Corne, D. W. (2000b). The pareto-envelope based selection algorithm for multiobjective optimization, *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI)*, 839-848, Berlin.

Nguyen, A. & Cung, V.-D. (2005). Le problème du Car Sequencing Renault et le challenge ROADEF' 2005, *Proceedings of Journées Francophones de Programmation par Contraintes*, 3-10, 2005.

Prandtstetter, M. & Raidl, G. R. (2007). An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem, *European Journal of Operational Research*, doi:10.1016/j.ejor.2007.04.044.

Ribeiro, C. C.; Aloise, D. Noronha, T. F. Rocha, C. & Urrutia, S. (2007a). An efficient implementation of a VNS/ILS heuristic for a real-life car sequencing problem, *European Journal of Operational Research*, doi:10.1016/j.ejor.2007.02.003.

Ribiero, C. C.; Aloise, D. Noronha, T. F. Rocha, C. & Urrutia, S. (2007b). A hybrid heuristic for a multi-objective real-life car sequencing, *European Journal of Operational Research*, doi:10.1016/j.ejor.2007.04.034.

Solnon, C.; Cung, V.-D. & Artigues, C. (2007). The car sequencing problem: overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem, *European Journal of Operational Research*, doi:10.1016/j.ejor.2007.04.033.

Terada, J.; Vo, H. & Joslin, D. (2006). Combining genetic algorithms with squeaky-wheel optimization, *Proceedings of Genetic and Evolutionary Computation COnference (GECCO) 2006*, Seattle.

Warwick, T. & Tsang, E. (1995). Tackling car sequencing problem using a generic genetic algorithm, *Evolutionary Computation*, 3(3), 267-298.

Zinflou, A., Gagné, C. & Gravel, M. (2007). Crossover operators for the car-sequencing problem, *Proceedings of the Seventh European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP 2007)*, LNCS 4446, C. Cotta and J. van Hemert (Eds.), Springer-Verlag Berlin Heidelberg, 229-239.

Zinflou, A. ; Gagné, C. Gravel, M. & Price, W. L. (2006). Pareto memetic algorithm for multiple objectives optimization with an industrial application, *Journal of Heuristics*, doi: 10.1007/s10732-007-9042-2.

Zitzler, E.; Laumanns, M. & Thiele, L. (2001). SPEA2: Improving the strength pareto evolutionary algorithm, *Technical Report 103*, Computer Engineering and Communication Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Switzerland.

Zitzler, E. & Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: the strength pareto approach, *Technical Report 43*, Computer Engineering and Communication Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Switzerland.

Symbiotic Evolution Genetic Algorithms for Reinforcement Fuzzy Systems Design

Chia-Feng Juang* and I-Fang Chung**
Department of Electrical Engineering, National Chung-Hsing University
***Institute of Biomedical Informatics, National Yang-Ming University*
Taiwan, R.O.C.

1. Introduction

The advent of fuzzy logic controllers has inspired the allocation of new resources for the possible realization of more efficient methods of control. In comparison with traditional controller design methods requiring mathematical models of the plants, one key advantage of fuzzy controller design lies in its model-free approach. Conventionally, the selection of fuzzy if-then rules often relies heavily upon the substantial amounts of heuristic observation to express the strategy's proper knowledge. It is very difficult for human experts to examine all the input-output data from a complex system, and then to design a number of proper rules for the fuzzy logic controllers. Many design approaches for automatic fuzzy rules generation have been developed in an effort to tackle this problem (Lin & Lee, 1996). The neural learning method is one of them. In (Miller et al., 1990), several neural learning methods including supervised and reinforcement based control configurations are studied. For many control problems, the training data are usually difficult and expensive, if not impossible, to obtain. Besides, many control problems require selecting control actions whose consequences emerge over uncertain periods for which training data are not readily available. In reinforcement learning, agents learn from signals that provide some measure of performance which may be delivered after a sequence of decisions being made. Hence, when the above mentioned control problems occur, reinforcement learning is more appropriate than supervised learning.

Genetic algorithms (GAs) are stochastic search algorithms based on the mechanics of natural selection and natural genetics (Goldberg, 1989). Since GAs do not require or use derivative information, one appropriate application for their use is the circumstance where gradient information is unavailable or costly to obtain. Reinforcement learning is an example of such domain. The link of GAs and reinforcement learning may be called genetic reinforcement learning (Whitley et al., 1993). In genetic reinforcement learning, the only feedback used by the algorithm is the information about the relative performance of different individuals and may be applied to reinforcement problems where the evaluative signals contain relative performance information. Besides GAs, another general approach for realizing reinforcement learning is the temporal difference (TD) based method (Sutton & Barto, 1998). One generally used TD-based reinforcement learning method is Adaptive Heuristic Critic (AHC) learning algorithm. AHC learning algorithm relies upon both the learned evaluation network and the learned action network. Learning of these two networks is based on gradient-descent learning algorithms with errors derived from internal and external reinforcement signals. In comparison with the GAs, one disadvantage of AHC learning algorithms is that they usually suffer the local minimum problem in network learning due to the use of the gradient descent method. Overall performance comparisons between TD-based reinforcement learning methods, including AHC and Q-learning, and GAs are made in (Whitley et al., 1993; Moriarty & Miikkulainen, 1996). The results show that GAs achieve better performance both in CPU time and number of control trials. In the past, some studies on the combination of GAs with TD-based reinforcement learning methods were proposed (Lin & Jou, 1999; Juang, 2005a). These studies show that the combination approach achieves better performance than using only GAs or the TD-based method.

Many approaches to fuzzy system design using GAs have been proposed (Cordón et al., 2004). If we distinguish them by individual representation in GAs, the major ones include Pittsburgh, Michigan, and the iterative rule learning (IRL) approach (Cordón et al., 2001). In the Pittsburgh approach, each individual represents an entire fuzzy rule set. A population of candidate rule sets is maintained by performing genetic operators to produce new generations of rule sets. Most GA-based fuzzy controller design methods belong to this approach (Karr, 1991; Homaifar & McCormick, 1995; Shi et al., 1999; Belarbi & Titel, 2000; Chung et al., 2000; Juang, 2004; Chou, 2006). In (Karr, 1991), Karr applied GAs to the design of the membership functions of a fuzzy controller, with the fuzzy rule set assigned in advance. Since the membership functions and rule sets are co-dependent, simultaneous design of these two approaches would be a more appropriate methodology. Based upon this concept, many researchers have applied GAs to optimize both the parameters of the membership functions and the rule sets. Differences between the approaches depend mainly on the type of coding and the way in which the membership functions are optimized. The disadvantage of this approach is the computational cost, since a population of rule set has to be evaluated in each generation. Also, the dimension of search space increases significantly, making it substantially difficult to find good solutions. In the Michigan approach, each individual of the population represents a single rule and a rule set is represented by the entire population. All researches in (Valenzuela-Rendon, 1991; Bonarini, 1993; Furuhashi et al., 1995) belong to this approach. As the evolutionary process is applied to the individual rule base, this approach invariably leads to consideration of both cooperation and competition. Obviously, it is difficult to obtain a good cooperation among the fuzzy rules that compete with each other. To solve this cooperation versus competition problem, a complex credit assignment policy is required, which is a disadvantage of this approach. This credit assignment task becomes more difficult especially for controller design based upon reinforcement learning problems, where the reinforcement signal is available after a long sequence of control actions. Besides, if the rule number in the designed fuzzy system is small, the small rule set in the population may easily converge to a local optimum and degrade the search speed. Like the Michigan approach, in IRL, each individual represents a single rule. However, in contrast to the former, only the best rule is adopted and added to the rule set in every GA run. The process is run several times to obtain the complete rule set. The IRL approach is considered to design genetic processes for off-line inductive learning problems and is not suitable to the controller design problem considered here.

Recently, the adoption of coevolutionary GAs for fuzzy system design has also been proposed. In GAs, coevolution refers to the simultaneous evolution of two or more species with coupled fitness (Zhao, 1998; Paredis, 1995). Coevolution may be performed at the population level (Potter et al., 1995; Peña -Reyes & Sipper, 2001) or at the individual level (Juang et al., 2000). The idea of coevolutionary GA is similar to the Michigan approach. The cooperative coevolutionary GAs (Potter & DeJong, 2000; Peña -Reyes & Sipper, 2001) and Symbiotic GAs (Moriarty & Miikkulainen, 1996; Moriarty & Miikkulainen, 1998; Juang et al., 2000; Juang, 2005b; Lin & Xu, 2006) are of this type. In (Moriarty & Miikkulainen, 1996; Moriarty & Miikkulainen, 1998), Symbiotic Adaptive Neuro-Evolution (SANE) and hierarchical SANE were proposed for neural networks design. In (Juang et al., 2000), a Symbiotic-Evolution-based Fuzzy Controller (SEFC) was proposed and the performance of SEFC is shown to be better than SANE. In (Juang, 2005b), a coevolutionary GA with divide-and-conquer (CGA-DC) technique was proposed. The CGA-DC not only performs a GA search on separate fuzzy rules, but also on a global fuzzy network simultaneously. Therefore, the performance of CGA-DC is better than SEFC. This chapter extends the idea of CGA-DC to both feedforward and recurrent fuzzy systems design, and the design method is called hierarchical SEFC (HSEFC).

Besides GAs, another factor that may influence fuzzy controller performance is its structure. Depending on the property of a controlled plant, different types of fuzzy controller structures are used in this chapter. A feedforward fuzzy controller is designed for a static plant. For a dynamic plant, whose output depends upon either previous states or control inputs or both, a recurrent controller should be a better choice. To apply a feedforward controller to this type of plant, we need to know the exact order of the plant in advance, and the inclusion of the past values to the controller input increases the controller size. Several recurrent fuzzy systems have been proposed (Zhang & Morris, 1999; Juang & Lin, 1999; Lee &. Teng, 2000; Juang, 2002). The performance of these systems has been demonstrated to be superior to that of recurrent neural networks. Based on this observation, a recurrent fuzzy controller should be a better choice compared to a recurrent neural controller under genetic reinforcement learning.

This Chapter introduces feedforward and recurrent fuzzy controllers design using HSEFC. For a static plant control problem under reinforcement learning environment, HSEFC for feedforward fuzzy controller design (HSEFC-F) is introduced, while for a dynamic plant, HSEFC for recurrent fuzzy controller (HSEFC-R) is proposed. In HSEFC-F, two populations are created. One of the populations is for searching the well-performed local rules, and each individual in the population represents only a fuzzy rule. Within the other population, each individual represents a whole fuzzy controller. The objective of the population is to search the best fuzzy system participating rules selected from the rule population, and the relationship between each rule is cooperative. Concurrent evolution of the local-mapping and global-mapping stages increases the design efficiency. With the above techniques, HSEFC-F performs an efficient fuzzy controller design task with a small population size. HSEFC-R is applied to the design of a recurrent fuzzy controller obtained by adding feedback structures into the feedforward fuzzy systems. In the local-mapping stage, each recurrent fuzzy rule is divided into two sub-rules, one representing a spatial mapping and the other doing a temporal mapping. These two sub-rules are considered as two distant species, and two populations are created for each sub-rule search, which is a technique based on the divide-and-conquer concept. In the global-mapping search stage, the third population is created to seek the best combination of spatial sub-rules, temporal sub-rules or both.

This chapter is organized as follows. Section 2 describes the types and functions of the fuzzy controller to be designed, including feedforward and recurrent fuzzy controllers. Section 3 describes the concepts of symbiotic evolution for fuzzy systems. Section 4 introduces HSEFC for fuzzy controller design, including HSEFC-F and HSEFC-R. Section 5 presents simulation results, where HSEFC-F is applied to control a cart-pole balancing system and HSEFC-R is applied to control a dynamic system with delays. Comparisons with SEFC for the same task are also made in this section. The conclusions are summarized in the last section.

2. Fuzzy controller

Control Systems represent an important application for reinforcement learning algorithms. From the perspective of controller learning, since GAs only require the appropriate evaluation of the controller performance to yield the fitness values for evolution, they are suitable for fuzzy controller design under reinforcement learning problems.

2.1 Feedforward fuzzy controller

Several types of fuzzy systems have been proposed depending on the types of fuzzy if-then rules and fuzzy reasoning. In this chapter, each rule in the feedforward fuzzy controller is presented in the following form:

$$\text{Rule } i : \text{ IF } x_1(t) \text{ is } A_{i1} \text{ And} \ldots \text{And } x_n(t) \text{ is } A_{in} \text{ Then } u(t+1) \text{ is } b_i \tag{1}$$

where x_j is the input variable, u is the control output variable, A_{ij} is a fuzzy set, and b_i is a fuzzy singleton. For a fuzzy set A_{ij}, a Gaussian membership function with

$$M_{ij}(x_j) = \exp\{-(\frac{x_j - m_{ij}}{\sigma_{ij}})^2\} \tag{2}$$

is used, where m_{ij} and σ_{ij} denote the mean and width of a fuzzy set A_{ij}, respectively. In the fuzzification process, crisp input x_j is converted into a fuzzy singleton and is mapped to the fuzzy set A_{ij} with degree $M_{ij}(x_j)$. In the inference engine, the fuzzy AND operation is implemented by the algebraic product in fuzzy theory. Given an input data set $\boldsymbol{x} = (x_1, ..., x_n)$, the firing strength $\mu_i(\boldsymbol{x})$ of rule i is calculated by

$$\mu_i(\boldsymbol{x}) = \prod_{j=1}^{n} M_{ij} = \exp\{-\sum_{j=1}^{n} (\frac{x_j - m_{ij}}{\sigma_{ij}})^2\} \tag{3}$$

The output from each rule is a crisp value. The fuzzy logic control action is the combination of the output of each rule using the weighted average defuzzification method. Suppose that a fuzzy controller consists of r rules, and then the output of the controller is

$$u = \frac{\sum_{i=1}^{r} \mu_i(\boldsymbol{x}) b_i}{\sum_{i=1}^{r} \mu_i(\boldsymbol{x})} \tag{4}$$

In applying HSEFC to the feedforward fuzzy controller design, only the number of rules should be assigned in advance. Instead of grid type partition, the rules are flexibly partitioned in the input space. If the total number of rules is set to r, then the number of fuzzy sets in each input variable as well as the number of fuzzy singletons in the consequent part are also equal to r.

2.2 Recurrent fuzzy controller

For a dynamic plant control, a recurrent controller appears to be a better choice than a feedforward controller. In the previous studies (Juang & Lin, 1999; Juang, 2002), performance recorded by applying recurrent fuzzy systems to dynamic problems solving has been shown to be superior to recurrent neural networks. The recurrent fuzzy controller designed in this chapter is a slight modification of that used in TSK-type recurrent fuzzy network (TRFN) (Juang, 2002) in that the consequent part is of zero-order instead of first-order TSK type. Figure 1 shows structure of the recurrent fuzzy system. Suppose the system consists of two rules. Each recurrent fuzzy if-then rule is in the following form

Rule i: IF $x_1(t)$ is A_{i1} AND … AND $x_n(t)$ is A_{in} AND $h_i(t)$ is G

$$\text{THEN } u(t+1) \text{ is } b_i \text{ AND } h_1(t+1) \text{ is } w_{1i} \text{ AND } h_2(t+1) \text{ is } w_{2i},\ i = 1, 2 \tag{5}$$

where A_{ij} and G are fuzzy sets, u is the output variable, h_i is the internal variable, w_{ij} and b_i are the consequent parameters for inference outputs h_i and u, respectively. The recurrent reasoning implies that the inference output $u(t+1)$ is affected by the internal variable $h_i(t)$, and the current internal output $h_i(t+1)$ is a function of previous output value $h_i(t)$, i.e., the internal variable $h_i(t)$ itself forms a recurrent reasoning. As in a feedforward fuzzy controller, Gaussian membership function is used for the fuzzy set A_{ij}. For the fuzzy set G, a global membership function $G(x) = 1/(1+e^{-x})$ is used. Given an input set $\boldsymbol{x} = (x_1, ..., x_n)$, the inference and internal outputs of the recurrent fuzzy controller are calculated, respectively, by

$$u(t+1) = \frac{\sum_{i=1}^{r} \mu_i(\boldsymbol{x}) b_i}{\sum_{i=1}^{r} \mu_i(\boldsymbol{x})} \tag{6}$$

and

$$h_i(t+1) = \sum_{k=1}^{r} \mu_k(\boldsymbol{x}) w_{ik} \tag{7}$$

where

$$\mu_i(\boldsymbol{x}) = G(h_i(t)) \cdot \prod_{j=1}^{n} M_{ij}(x_j) \tag{8}$$

In applying HSEFC to the recurrent fuzzy controller design, only the number of recurrent fuzzy rules should be assigned in advance. Suppose there are r rules in total, then the numbers of fuzzy sets A's on each external input variable x_i and the internal variable h_i, are all equal to r.

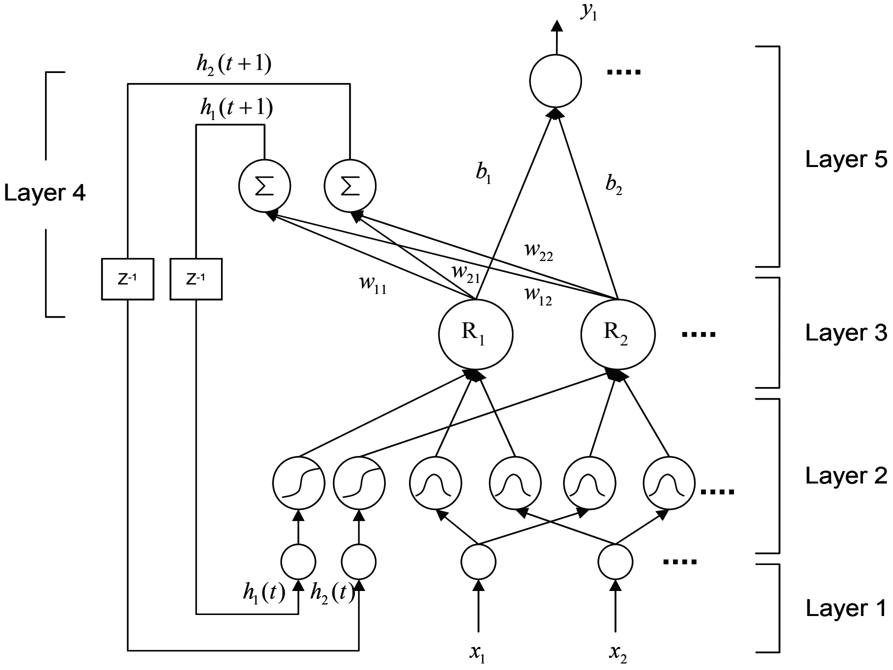


Fig. 1. Recurrent fuzzy system structure.

3. Symbiotic Evolution-based Fuzzy Controller (SEFC)

The symbiotic evolution-based fuzzy controller (SEFC) was proposed in (Juang et al., 2000), and the idea was used in many later studies (Mahfouf et al., 2001, Jamei et al., 2004, Kuo et al., 2004, Juang, 2005b, Lin & Xu, 2006). Unlike general GAs' evolution algorithms which operate on a population of full solutions to a problem (the Pittsburgh approach), symbiotic evolution assumes that each individual in the population represents only a partial solution; complete solutions are formed by combining several individuals. Figure 2(a) and (b) show codings of a fuzzy system using the general and symbiotic evolutions, respectively. Each individual in Fig. 2(a) represents a whole fuzzy system. On the contrary, each individual in Fig. 2(b) represents a single fuzzy rule. In general GAs, a single individual is responsible for the overall performance, with the fitness value assigned to itself according to its performance. In symbiotic evolution, the fitness of an individual (a partial solution) depends on others. Partial solutions can be characterized as *specializations*. The specialization property tries to keep search diversity which prevents convergence of the population. The symbiotic evolution appears to be a faster and more efficient search scheme than the general evolution approaches for reinforcement learning problems (Moriarty & Miikkulainen, 1996; Juang et al., 2000; Lin & Xu, 2006).

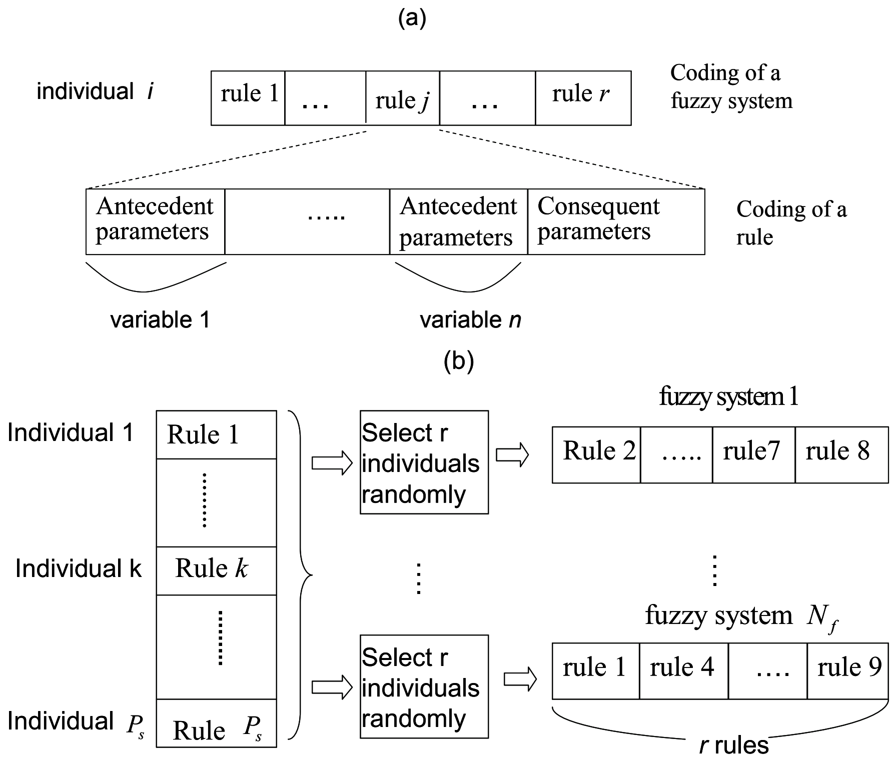


Fig. 2. Coding of a fuzzy system using (a) the general and (b) symbiotic evolutions.

The basic idea of SEFC is on the representation of a single fuzzy rule by an individual. A whole fuzzy system is formed by combining r randomly selected rules from a population. With the fitness assignment performed by symbiotic evolution and the local property of a fuzzy rule, symbiotic evolution and the fuzzy system design can complement each other. If a normal GA evolution scheme is adopted for fuzzy system design, only the overall performance of a fuzzy system is known, not the performance of each fuzzy rule. The method to replace the unsuitable fuzzy rules that degrade the overall performance of a fuzzy system is through random crossover operations, followed by observing the performance of the offspring. Only when the overall performance of the fuzzy system is good do we know that the unsuitable rules have been replaced. In SEFC, the performance of each fuzzy rule may be implicitly evaluated. Such implicit evaluation is especially suitable for reinforcement learnimg problems which require only evaluation instead of instructive feedback information. With the local property of a fuzzy rule, the fitness assignment performed by the SEFC is quite representative. In this way, symbiotic evolution and fuzzy system design can complement each other, which result in a fast, efficient genetic search for reinforcement learning problems.

4. Hierarchical SEFC (HSEFC)

In SEFC, a GA search is performed only on the separate rules. The information about the participating rules in a well-performed fuzzy network is not propagated from generation to generation. If, besides the local rule search, we can propagate the information to the next generation and perform a global fuzzy network search simultaneously, then a better design performance could be achieved. This is the motivation of Hierarchical SEFC (HSEFC). This section introduces the detailed HSEFC implementation algorithm. Subsection 3.1 presents the HSEFC implementation algorithm for feedforward fuzzy controller design (HSEFC-F). In subsection 3.2, the HSEFC for recurrent fuzzy controller design (HSEFC-R) is presented.

4.1 HSEFC for feedforward fuzzy controller design (HSEFC-F)

Figure 3 shows the structure of the HSEFC-F design system. It consists of two populations. In population 1, each individual represents only a single fuzzy rule in the form described in (1). A whole fuzzy system constituted by r fuzzy rules is built by randomly selecting r individuals from population 1. The selected rules are recorded in an individual of population 2. Therefore, each individual in population 2 indicates a whole fuzzy system, with each gene representing a rule selected from population 1. Each constituted fuzzy controller in population 2 is applied to the plant to be controlled with a controller performance evaluation returned and used as the fitness value. This fitness value is assigned not only to the action system in population 2, but also distributed to the rules participating in the system. The concurrent evolution of populations 1 and 2 leads to an efficient algorithm. Detailed processes of these two stages are described as follows.

4.1.1 Local mapping stage

This stage performs evolution of population 1. Like general GAs, the evolution consists of three major operations: reproduction, crossover, and mutation. Initially, this stage randomly generates a population of individuals, each of which represents a set of parameters for the fuzzy rule in (1). The population size is denoted as P_{S1}, which indicates that P_{S1} fuzzy rules are generated. Each gene is represented by a floating number and the encoded form of each rule (individual) is as follows,

$$| m_{i1} | \sigma_{i1} | m_{i2} | \sigma_{i2} | ... | m_{in} | \sigma_{in} | b_i |$$

After creating a whole population with real-valued individuals, an interpreter takes one from the population and uses it to set part of the parameters in a fuzzy system. Suppose there are r rules in a fuzzy system, then a whole fuzzy system is constructed by selecting r

individuals from population 1. The fuzzy system runs in a feedforward fashion to control the plant until a failure or a success occurs. Then, in the reinforcement control problem, we should assign a credit to the fuzzy system. From the view point of temporal credit, if the fuzzy system can control the plant for a longer time, then the degree of success is higher and a higher credit should be assigned. Based on this concept, for each individual in population 2, the fitness value is assigned at the moment of failure.

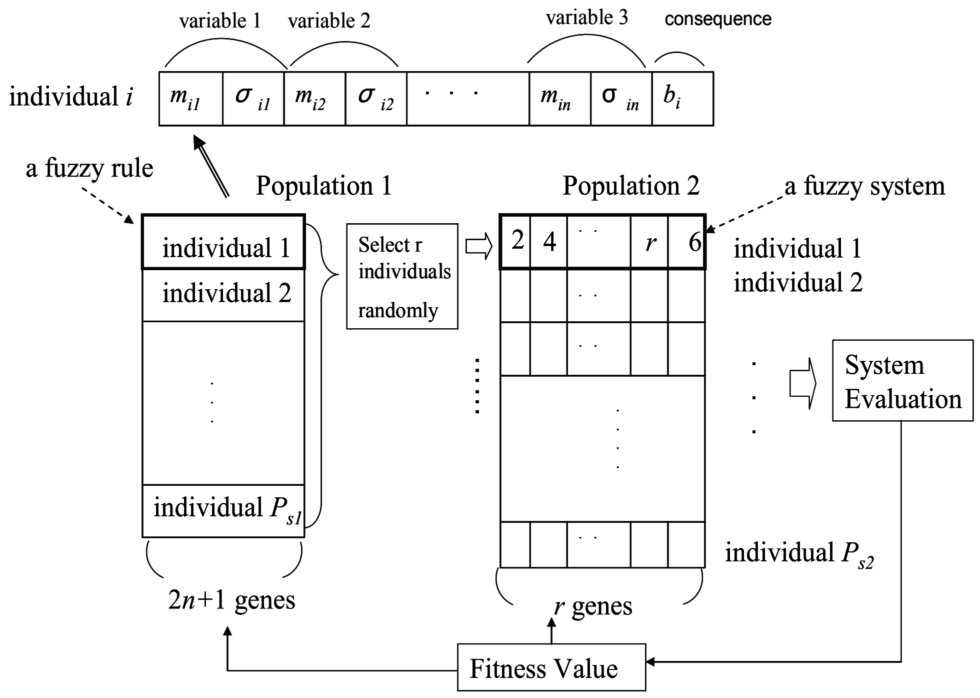


Fig. 3. Structure of the HSEFC for feedforward fuzzy system design (HSEFC-F).

To evaluate the performance of each individual in population 1, the credit assignment is not as direct as that used in population 2. We should apportion the credit to the participating rules in a fuzzy system. This credit assignment problem also occurs in the Michigan approach. In the Michigan approach, many different credit assignment schemes have been proposed (Cordón et al., 2001). The two most important ones are the bucket bridge algorithm (BBA) (Booker et al., 1989) and the profit sharing plan (PSP) (Grefenstette, 1988). The BBA adjusts the strength of an individual rule classifier by distributing the obtained reward across the sequence of active rule classifiers that are directly or indirectly contributed to the past actions by the fuzzy classifier system. It uses only the local interactions between rules to distribute credit. In contrast to the BBA, the PSP is a global learning scheme and typically achieves a better performance than the BBA. In (Ishibuchi et al., 1999), a simpler credit assignment algorithm is proposed. In this algorithm, there is always a unique winner rule to be rewarded or penalized depending on whether it correctly predicts the class of the training example. The fitness value of each rule is determined by the total reward assigned to the rule. Basically, the aforementioned schemes are based on an economic analogy and consist of a bid competition between fuzzy rules. They measure only the goodness of an individual rule and do not consider the ability to cooperate with the remaining ones in the population. In fact, in the Michigan approach, since a population represents a whole system, each individual cooperates with the same ones in each generation. The quality of cooperation is difficult to obtain among these competing individuals.

In HSANE-F, many fuzzy systems are formed in each generation, and each individual may combine with different ones in each fuzzy system construction. By taking the average system fitness value in which an individual participates, we can approximately measure the individual cooperation ability. The measure is based on the fact that the fitness of each individual depends on the quality of the whole system it participates in, thus measuring how well it cooperates to solve the problem. As to the goodness of each individual, owing to the local mapping property, a well-performed rule will also have certain contribution to the system performance. On the contrary, a wrongly-mapped rule will degrade the system performance. The contribution of each rule to the system depends on its firing strength. However, the fitness value is available only when the control fails, during which the firing strength of each rule varies with time. It would be complex to distribute the fitness value among the participating rules based on the firing strength. A simple way is to equally distribute the system fitness value among the participating rules to measure its goodness. Therefore, by taking the average system fitness values in which an individual participates, we can approximately measure both the cooperation and goodness of the individual. Effectiveness of this fitness value distribution approach will be verified in simulations. Detailed steps of the approach are as follows.

Step 1. Divide the fitness value by r and accumulate the divided value to the fitness record of the r selected rules with their fitness set to zero initially.

Step 2. The above rule selection, plant control, and fitness division process are repeated P_{S2} (the size of population 2) times. The process ends when each rule has been selected for a sufficient number of times. Record the number of times each rule has participated in a fuzzy system.

Step 3. Divide the accumulated fitness value of each rule by the number of times it has been selected. The average fitness value represents the performance of an individual.

When the average fitness of each individual in population 1 is obtained, the HSEFC then looks for a better set of individuals to form a new population in the next generation by using genetic operators, including reproduction, crossover, and mutation. The detailed description of the three operations is as follows.

In reproduction operation, the elite strategy and tournament selection techniques are used. The top-half of best-performing individuals in the population are sorted according to their fitness value. Based on the elite strategy, these elites are advanced directly to the next generation. Also, to keep a non-decreasing best-so-far fitness value, the rules participating in the best-performing system in each generation are directly reproduced in the next generation. The remaining individuals are generated by performing tournament selection and crossover operations on the elites.

In crossover operation, the tournament selection scheme is used to select parents. Two individuals in the top-half of best-performing individuals are selected at random in the tournament selection, and their fitness values are compared. The individual with higher fitness value is selected as one parent. The other is also selected in the same way. Two offspring are created by performing crossover on the selected parents. Here, one point crossover operation is performed. The top-half of worst-performing individuals are replaced by the newly produced offspring. The adopted crossover may be regarded as a kind of elite crossover.

In mutation operation, the gene in an individual is altered randomly by mutation. Uniform mutation is used, where a mutated gene is drawn randomly and uniformly from its search domain. In the following simulations, mutation probability p_m is set to 0.1.

The elite strategy above can improve the searching speed, but the population diversity may also be lost quickly at the same time. To overcome this disadvantage, a population renewal technique is used. In each generation, the relationship between each individual is monitored. Since half of the next generation population is generated by performing crossover on the top-half of best-performing individuals, it is only necessary to check the similarities between the top-half of best-performers. The cross correlation value between two neighbouring individuals in a performance ranked list is calculated and averaged. The mathematical form for this measure is as follows:

$$S = \frac{2}{P_{S1}} \sum_{i=1}^{P_{S1}/2-1} \frac{(D_i^T D_{i+1})^2}{D_i^T D_i D_{i+1}^T D_{i+1}} \tag{9}$$

where D_i is the i th best-performing individual in the rank list. The dimension of D_i is $\ell \times 1$, where ℓ is the number of genes in the individual. With this measure, if all of the individuals are exactly the same, then S is equal to 1. This similarity measure is performed for each generation. When the measurement similarity is higher than a pre-specified threshold T_{hr}, it reflects that the elites have moved to a degree of convergence. If this phenomenon occurs, then most parts of the individuals are renewed. In the renewal process, only a portion of the top best-performing individuals are reproduced to the next generation, and the remaining parts are replaced by newly generated individuals. After the renewal process, the similarity value is again calculated on each subsequent generation, and the renewal process is performed when the value is higher than the threshold T_{hr}. The renewal process can always keep the diversity of the population and thus helps to find the best fuzzy rules.

4.1.2 Global mapping stage

This stage performs evolution of population 2. The function of population 2 consists of both exploitation and exploration of the rule-combination in a fuzzy system. In exploitation, the information about which rules are combined together in a well-performed fuzzy system is propagated from generation to generation. On the other hand, evolutionary procedure is performed on population 2 to search the best-combination in exploration. Without population 2, in each generation, the rules participating in a fuzzy system should be randomly selected from population 1. Population 2 helps to concentrate the search on the best rule combination. Since populations 1 and 2 are evolved concurrently, if individuals in the former are updated frequently, the search in the latter might be meaningless. To avoid this phenomenon, as stated in the local-mapping-search-stage, the top-half of best-performing individuals in population 1 are reproduced directly to the next generation. Only the remaining poorly-performed individuals are updated. Owing to the local mapping property, the update of these rules has only local influence on its participating fuzzy system. In general, the newly generated rules outperform the original poorly-performing ones. So the evolution of population 1 is also helpful to that of population 2, that is, both are cooperative. This property will be verified in the simulations.

In this stage, an integer-value encoding scheme is used, and the alleles have values in the set {1, 2, …, P_{S1}}. There are r genes in each individual, and it has the following form,

$$| 2 | 7 | 5 | 9 | \dots | P_{S1} | \dots | 8 | 1 |$$

The population size of population 2 is set to be P_{S2}, indicating that P_{S2} fuzzy controllers are applied to plant control in each generation. Due to the following two reasons, the genetic operation used in this stage is different from that used in the local-mapping search stage. First, since a flexible partition of precondition part is adopted and reinforcement learning is performed, the rule number r in a fuzzy system is usually small. Population 2 converges quickly due to the small individual length. Second, the character of population 2 in the whole searching task is auxiliary and should always maintain diversity to coordinate the evolution of population 1 from generation to generation. If population 2 converges faster than population 1, then the evolution of population 1 is useless. In order to maintain population diversity, the following genetic operation is used. The top-half of best-performing individuals in population 2 are sorted according to their fitness values. To select the parents for crossover operation, the tournament select scheme is adopted and performed on the top-half of best-performing individuals. By performing the one-point crossover operation on the selected parents, offspring can be created and half of the new population is produced in this way. As stated in the local-rule searching stage, it is desired to maintain a non-decreasing best-so-far fitness value, consequently the best-performing individual is directly reproduced in the next generation. As to the remaining half of the population, they are created by randomly generated individuals. For the mutation operation, a mutated gene is selected randomly and uniformly from the integer set {1, 2, …, P_{S1}}. The mutation probability is set to 0.1.

After the above crossover and mutation operations, overlapping of rules might occur. If this occurs, then the total number of rules in a fuzzy system is less than r. For this situation, the overlapping of each rule is regarded as a weighting factor F of its firing strength. If the overlapping number of rule i is n_i, then $F_i = n_i$. With the weighting factor, the output equation of the fuzzy system in (4) can be rewritten as

$$u = \frac{\sum_{i=1}^{r'} \mu_i(\boldsymbol{x}) F_i b_i}{\sum_{i=1}^{r'} \mu_i(\boldsymbol{x}) F_i} \tag{10}$$

where $r' \le r$ is the total number of rules and $\sum_{i=1}^{r'} F_i = r$.

4.2 HSEFC for recurrent fuzzy controller design (HSEFC-R)

This subsection introduces the application of HSEFC to recurrent fuzzy controller design. The recurrent fuzzy rule to be designed is previously described in (5). By regarding each recurrent rule as an individual in population 1 of HSEFC-F, the recurrent fuzzy controller can be designed. However, this approach does not use the recurrent fuzzy system structure to its full advantage. To speed up the design process, the HSEFC-R designed specifically for the recurrent fuzzy rule is proposed. The divide-and-conquer concept is incorporated in HSEFC-R (Juang, 2005b). Based on this concept, the recurrent fuzzy rule in (5) is decomposed into two sub-rules, the spatial sub-rule and the temporal sub-rule. The antecedent parts of both sub-rules are the same as that in (5). The consequent part in each spatial sub-rule considers only the output variable u, while the consequent part in each temporal sub-rule considers only the variables h_1, …, h_r. In HSEFC-R, the spatial and temporal sub-rules are evolved simultaneously. Figure 4 shows the HSEFC-R structure,

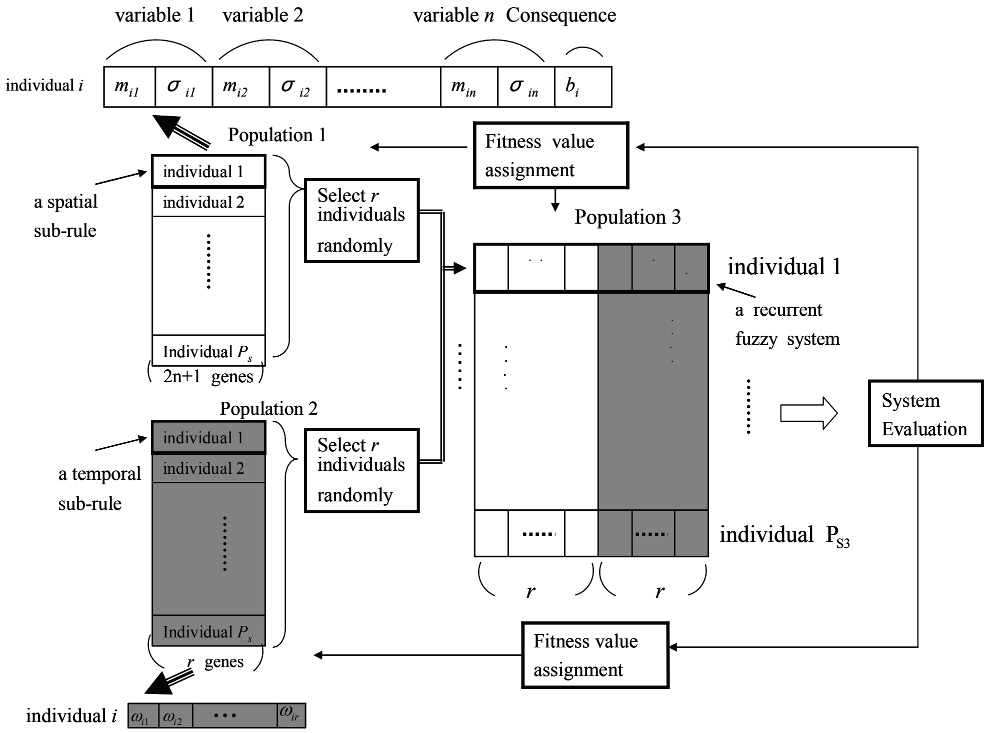


Fig. 4. Structure of the HSEFC for recurrent fuzzy system design (HSEFRC-R).

where there are three populations. Populations 1 and 2 are responsible for spatial and temporal sub-rules searches, respectively. Population 3 is responsible for the whole recurrent fuzzy system search. Each individual in population 1 represents a spatial sub-rule, whereas each individual in population 2 represents a temporal sub-rule. Since the spatial and temporal sub-rules share the same antecedent part, the antecedent parameters are encoded in population 1 only. A recurrent fuzzy system consisting of r rules is constructed by randomly selecting r individuals from both populations 1 and 2. The selected individuals from both populations are recorded in population 3. Each individual in population 3 represents a whole fuzzy system. The task of creating population 3 is not only to search the best combinations of the r spatial sub-rules or temporal sub-rules selected from each population, but also to search for the best match of both types of sub-rules. Each recurrent fuzzy system encoded in population 3 is applied to a dynamic plant control with the return of a performed evaluation. The evaluation is used as the fitness value of the controller. As in HSEFC-F, the fitness value for each individual in population 3 is set to the time steps until failure for each control trial. This fitness value is then assigned to the participating sub-rules selected from populations 1 and 2. With the distributed fitness value, evolution of populations 1 and 2 is performed in the local-mapping search stage, while evolution of population 3 is performed in the global-mapping search stage. These two stages are executed concurrently until a successful control is achieved. Detailed operations of these two stages are described as follows.

4.2.1 Local mapping stage

The objective of this stage is to explore the well-performing spatial and temporal sub-rules in each local input region. First, populations 1 and 2 are created by randomly generated individuals. The sizes of both the populations are equal to and are denoted as P_S. The real-value encoding scheme is used in both populations. Each individual in population 1 encodes a spatial sub-rule and has the following form:

$$| m_{i1} | \sigma_{i1} | m_{i2} | \sigma_{i2} | \ldots | m_{in} | \sigma_{in} | b_i |$$

Each individual in population 2 encodes only the consequent part of a temporal sub-rule because the spatial and temporal sub-rules share the same antecedent part. Each individual has the following form:

$$| w_{i1} | w_{i2} | \ldots | w_{ir} |$$

The relationship between the individuals in populations 1 and 2 is cooperative. The genetic operation of each population is executed independently and concurrently. The fitness value decision method of an individual is similar to that used in HSEFC-F. If the fitness value of the recurrent fuzzy system consisting of r recurrent rules is *Fit*, then the distributed fitness value of each participating individuals from populations 1 and 2 is set to *Fit/r*. When the fitness value of each individual in both populations is given, new populations are generated by using genetic operations. Like HSEFC-F, the elite strategy and tournament selection techniques are used here. The reproduction, crossover, and mutation operations are the same as those used in the local-mapping search stage of HSEFC-F. The mutation probability is set at 0.1. To keep population diversity, the population renewal technique is applied to both the populations. A threshold value, T_{hr}, is set for both populations. If the similarity value of the top-half of best-performing individuals is higher than T_{hr} in each individual, then the renewing technique is applied to that population.

4.2.2 Global mapping stage

This stage performs evolution of population 3. An integer-value encoding scheme is used. Each individual contains $2r$ genes. The first r genes represent the r spatial sub-rules selected from population 1, while the remaining r genes represent the r temporal sub-rules selected from population 2. Each gene representing the selected sub-rule has value in the integer set {1, 2, …, P_S}. Each individual has the following form

$$|7|2|4|\ldots|\ P_S\ |\ \ldots|11|4|\ |2|13|7|\ldots|\ P_S\ |\ldots|15|4|$$

The temporal sub-rule recorded in position $r+k$ shares the same antecedent part with the spatial sub-rule in position k. The population size is set to P_{S3}, indicating that P_{S3} recurrent fuzzy controllers are built and applied to a dynamic plant control in each generation. The fitness value of each individual is assigned according to the controller performance evaluation. For the genetic operation, in addition to the crossover operation, the other operations used are the same as those used in the global-mapping search stage of HSEFC-F. In the crossover operation, to exchange the spatial and temporal sub-rule combination information of each population, a two-point crossover operation is performed. One crossover site is located at the first r genes, indicating the exchange of spatial sub-rule combination information; the other is located at the last r genes, indicating the exchange of temporal sub-rule combination information.

5. Simulations

This section presents simulation results of HSEFC for feedforward and recurrent fuzzy controller design under genetic reinforcement learning environments. All simulations in the following examples are written in C++ program, and run on a Pentium-1G personal computer. For the fuzzy rule number selection, it is somewhat heuristic and depends on the complexity of the plant to be controlled. In the following examples, the number of rules in each fuzzy system is set to five, i.e., r = 5.

5.1 Feedforward fuzzy controller design

Example 1. Cart-Pole Balancing System. In this example, HSEFC-F is applied to a classic control problem referred to as the cart-pole balancing problem. This problem is often used as an example of inherently unstable and dynamic systems to demonstrate both modern and classic control techniques, and is now used as a control benchmark (Andersonm 1989). The cart-pole balancing problem is the problem of learning how to balance an upright pole. There are four state variables in the system: θ, the angle of the pole from an upright position (in degrees); $\dot{\theta}$, the angular velocity of the pole (in degrees/second); x, the horizontal position of the center of the cart (in meters); and $\dot{x}$, the velocity of the cart (in m/s). The only control action is u, which is the amount of force (Newton) applied to the cart to move it toward its left or right. The system fails when the pole falls past a certain angle (12 degrees is used here) or the cart runs into the bounds of its track (the distance is 2.4m from the center to both bounds of the track). Details of the control system description can be found in (Juang et al. 2000). A control strategy is deemed successful if it can balance a pole for 120000 time steps. In designing the fuzzy controller, the four states are fed as the controller inputs, and the controller output is u. In HSEFC-F, the number of individuals (rules) in population 1 is set to 50 (i.e., P_{S1}=50). The size of population 2 is set to 50 (i.e. P_{S2}=50), indicating that fifty fuzzy controllers are built and evaluated per generation. The evaluation of a fuzzy controller consists of a single trial to the cart-pole system. The similarity measure threshold T_{hr} in the renewing process is set at 0.5. The fitness value is equal to the number of time steps in which the pole remains balanced. For each control trial, the initial values of $(x, \dot{x}, \theta, \dot{\theta})$ are random values in region [-2, 2]x[-1.5, 1.5]x[-5, 5]x[-40, 40]. In this example, 100 runs are simulated, and a run ends when a successful controller is found or a failure run occurs. The definition of a failure run is if no successful fuzzy controller is found after 25,000 trials. The number of pole-balance trials and the CPU time (the time from the first trial to the end of a successful trial) are measured. The average CPU time and trial number of the HSEFC-F are 4.0 (sec) and 179, respectively. Figure 5 (a) and (b) show the control results of position and angle in the first 1000 time steps of three different runs with different initial states. For SEFC, the average results are 5.1 (sec) and 256 trials. The results show that the performance of HSEFC-F is better than SEFC. Since the performance of SEFC has been shown to be better than other compared reinforcement learning methods in (Juang et al., 2000), only SEFC is compared this example.

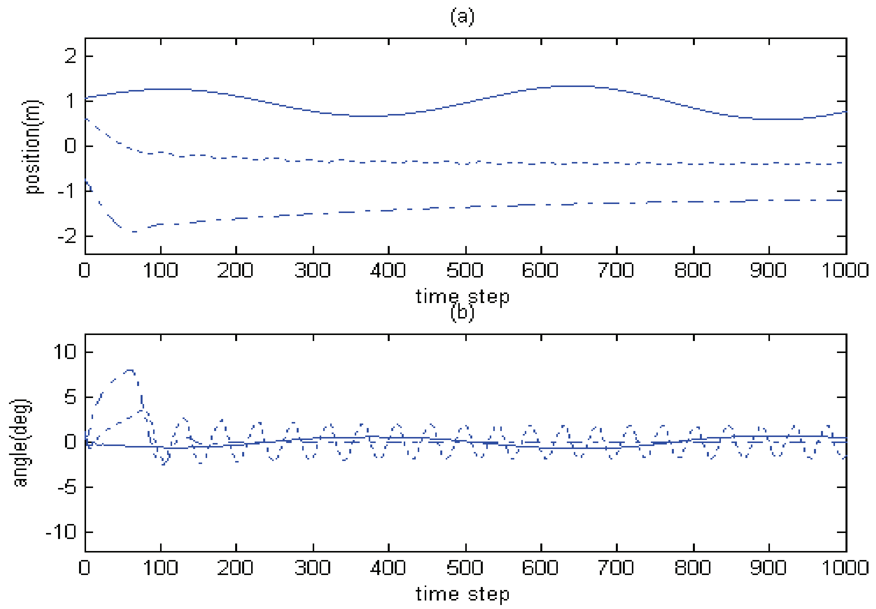


Fig. 5. Control results of (a) position (b) angle in the first 1000 time steps of three different runs with different initial states in Example 1.

5.2 Recurrent fuzzy controller design

Example 2. Dynamic Plant Control. The dynamic plant to be controlled is described by the following equation

$$y_p(k+1) = 0.6y_p(k) + 0.03y_p(k-1)u(k) + 0.01u^2(k-2) + 0.2u(k-3) \qquad (11)$$

The current output of the plant depends on two previous outputs and four previous inputs. In (Kim et al., 1998), it is shown that for this type of plant, a poor performance is achieved by a linear predictive control. The controller input u is in the range [-20, 20]. The initial states are $y_p(0) = y_p(1) = 3$. The regulation point y_{ref} is set to 10. The performance of a controller is measured by the number of time steps in which the controlled state y_p satisfies the following constraint. The constraint set starts from the initial state and after 10 times of control, the state of y_p should be within the region [y_{ref} -0.2, y_{ref} +0.2], otherwise a failure occurs. A recurrent fuzzy controller designed by HSEFC-R is applied to the plant. In HSEFC-R, the sizes of populations 1, 2, and 3 are all set to 100. The similarity measure threshold T_{hr} in the renewal process is set to 0.35. Since a recurrent fuzzy controller is used, only the current state $y_p(k)$ and reference state y_{ref} are fed as the controller inputs. Since a recurrent fuzzy controller consists of five recurrent fuzzy rules, the number of genes in each individual of populations 1 or 2 is equal to 5. One hundred runs are simulated, and a run ends when a successful controller is found. A failure run is said to occur if no successful fuzzy controller is found after 100,000 trials. The average CUP time and trial number of HSANE-R are 0.65 (sec) and 1853, respectively. For SEFC, the results are 1.37 (sec) and 3960 trials. The performance of HSANE-R is much better than SEFC. Detailed comparisons of different design methods can be found in (Juang, 2005b).

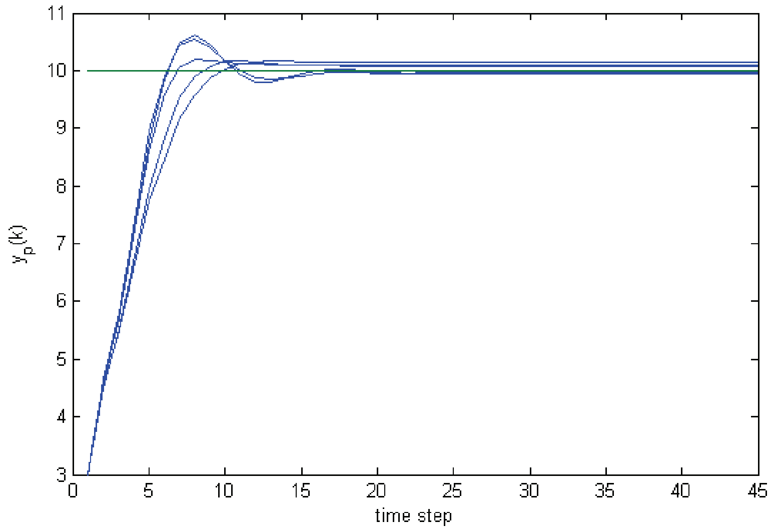


Fig. 6. Dynamic plant control results of five different runs using HSEFC-R in Example 2.

6. Conclusion

This chapter introduces a unified symbiotic evolution framework (the HSEFC) for feedforward and recurrent fuzzy controller design in reinforced learning environments. The design of a fuzzy controller is divided by the HSEFC into two iterative search stages: the local-mapping search stage and the global-mapping search stage. In this way, the population in each stage is evolved independently and concurrently. Furthermore, to avoid the premature population phenomenon, modifications of general genetic operations are also incorporated in the design process. In the interests of utility and economy, the HSEFC operates under two formats; HSEFC-F and HSEFC-R. For feedforward fuzzy controller design, HSEFC-F is presented, while for recurrent fuzzy controller design, HSEFC-R, which uses the divide-and-conquer technique on spatial and temporal sub-rules search, is presented. As shown, simulation results in static and dynamic plant control problems have verified the effectiveness and efficiency of HSEFC-F and HSEFC-R. Although in HSEFC solutions, the designed fuzzy controller structure is currently assigned in advance, further work on HSEFC intends to focus on its extension to automatic controller structure determination. A more accurate fitness assignment for each single rule in symbiotic evolution is another future research topic.

7. References

Anderson, C. W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, Vol. 9, 31-37.

Belarbi, K. & Titel, F. (2000). Genetic algorithm for the design of a class of fuzzy controllers: an alternative approach. *IEEE Trans. Fuzzy Systems*, Vol. 8, No. 4, 398-405.

Bonarini, A. (1993). ELF: learning incomplete fuzzy rule sets for an autonomous robot, *Proc. 1st European Congress on Intelligent Technologies and Soft Computing*, pp. 69-75, Aachen, Germany.

Booker, L.B.; Goldberg, D.E. & Holland, J.H. (1989). Classifier systems and genetic algorithms. *Artificial Intelligence*, Vol. 40, 235-282.

Chou, C.H. (2006). Genetic algorithm-based optimal fuzzy controller design in the linguistic space. *IEEE Trans. Fuzzy Systems*, Vol. 14, No. 3, 372-385.

Chung, IF.; Lin, C.J. & Lin, C.T. (2000). A GA-based fuzzy adaptive learning control network. *Fuzzy Sets and Systems*, Vol. 112, No. 1, 65-84.

Cordón, O.; Herrera, F.; Hoffmann, F. & Magdalena, L. (2001). *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*, World Scientific.

Cordón, O.; Gomide, F.; Herrera, F.; Hoffmann, F. & Magdalena, L. (2004). Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Systems*, Vol. 141, No. 1, 5-31.

Goldberg, D.E. (1989). *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley.

Grefenstette, J.J. (1988). Credit assignment in rule discovery systems based on genetic algorithms. *Machine Leaqrning*, Vol. 8, 225-246.

Furuhashi, T.; Nakaoka, K. & Uchikawa, Y. (1995). An efficient finding of fuzzy rules using a new approach to genetic based machine learning, *Proc. 4th IEEE Int. Conf. Fuzzy Systems*, pp. 715-722, Yokohama, Japan.

Homaifar, A. & McCormick, E. (1995). Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Trans. Fuzzy Systems*, Vol. 3, No. 2, 129-139.

Ishibuchi, H.; Nakashima, T. & Murata, T. (1999) Performance evaluation of fuzzy classfier systems for multidimentioinal pattern classification problems. *IEEE Trans. On Systems, Man and Cybernetics, Part B: Cybernetics*, Vol. 29, 601-618.

Jamei, M.; Mahfouf, M. & Linkens, D.A. (2004). Elicitation and fine-tuning of fuzzy control rules using symbiotic evolution. *Fuzzy Sets and Systems*, Vol. 147, No. 1, 57-74.

Juang, C.F. & Lin, C.T. (1999). A recurrent self-organizing neural fuzzy inference network. *IEEE Trans. Neural Networks*, Vol. 10, No. 4, 828-845.

Juang, C.F.; Lin, J.Y. & Lin, C.T. (2000). Genetic reinforcement learning through symbiotic evolution for fuzzy controller design. *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 30, No. 2, 290-302.

Juang, C.F. (2002). A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. *IEEE Trans. Fuzzy Systems,* Vol. 10, No. 2, 155-170.

Juang, C.F. (2004). Temporal problems solved by dynamic fuzzy network based on genetic algorithm with variable-length chromosomes. *Fuzzy Sets and Systems*, Vol. 142, No. 2, 199-219.

Juang, C.F. (2005a). Combination of on-line clustering and Q-value based GA for reinforcement fuzzy system design. *IEEE Trans. Fuzzy Systems*, Vo. 13, No. 3, 289-302.

Juang, C.F. (2005b). Genetic recurrent fuzzy system by coevolutionary computation with divide-and-conquer technique. *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews* , Vol. 35, No. 2, 249-254.

Karr, C.L. (1991). Design of an adaptive fuzzy logic controller using a genetic algorithm, *Proc. the Fourth Int. Conf. Genetic Algorithms*, pp. 450-457.

Kim, J.H.; College, D.T.; Gun, A. & DO, G. (1998). Fuzzy model based predictive control, *Proc. IEEE. Int. Conf. Fuzzy Systems*, pp. 405-409, Anchorage, AK, USA.

Kuo, H.C.; Chang, H.K. & Wang, Y.Z. (2004). Symbiotic evolution-based design of fuzzy-neural diagnostic system for common acute abdominal pain. *Expert Systems with Applications*, Vol. 27, No. 3, 391-401.

Lee, C.H. & Teng, C.C. (2000). Identification and control of dynamic systems using recurrent fuzzy neural networks. *IEEE Trans. Fuzzy Systems*, Vol. 8, No. 4, 349-366.

Lin, C.T. & Jou, C.P. (1999). Controlling chaos by GA-based reinforcement learning neural network. *IEEE Trans. Neural Networks*, Vol. 10, No. 4, 846-859.

Lin, C.T. & Lee, C.S.G. (1996). *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems*, Prentice Hall, USA.

Lin, C.J. & Xu, Y.J. (2006). A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications. *Fuzzy Sets and Systems*, Vol. 157, Issue 8, 1036-1056.

Mahfouf, M.; Jamei, M. & Linkens, D.A. (2001). Rule-base generation via symbiotic evolution for a Mamdani-type fuzzy control system, *Proc. the 10th IEEE Int. Conf. Fuzzy Systems*, pp. 396–399.

Miller, W.T.; Sutton, R.S. & Werbos, P.J. (1990). *Neural Networks for Control*, The MIT Press.

Moriarty, D.E. & Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, Vol. 22, 11-32.

Moriarity, D.E. & Miikkulainen, R. (1998). Hierarchical evolution of neural networks, *Proc. IEEE Conf. Evoulutionary Computation*, pp. 428-433, Anchorage, AK, USA.

Paredis, J. (1995). Coevolutionary computation. *Aftifical Life*, Vol. 2, No. 4, 355-375.

Peña -Reyes, C.A. & Sipper, M. (2001). Fuzzy CoCo: a cooperative-coevolutionary approach to fuzzy modelling. *IEEE Trans. Fuzzy Systems*, Vol. 9, No. 5, 727-737.

Potter, M.A.; Jong, K.D. & Grefenstette, J. (1995). A coevolutionary approach to learning sequential decision rules, *Proc. 6th Int. Conf. Genetic Algorithms*, pp. 366-372, Pittsburgh, USA.

Potter, M.A. & DeJong, K.A. (2000). Cooperative coevolution: An architectural for evolving coadopted subcomponents. *Evol. Computation*, Vol. 8, No. 1, 1-29.

Shi, Y.; Eberhart, R. & Chen, Y. (1999). Implementation of evolutionary fuzzy systems. *IEEE Trans. Fuzzy Systems*, Vol. 7, No. 2, 109-119.

Sutton, R.S. & Barto, A.G. (1998). *Reinforcement Learning*, The MIT Press.

Whitley, D.; Dominic, S.; Das, R. & Anderson, C.W. (1993). Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, Vol. 13, 259-284.

Valenzuela-Rendon, M. (1991). The fuzzy classifier system: A classifier system for continuously varying variables, *Proc. 4th Int. Conf. Genetic Algorithms*, pp. 346-353, San Diego, USA.

Zhao, Q. (1998). A general framework for cooperative co-evolutionary algorithms: a society model, *Proc. IEEE Conf. Evolutionary Computation*, pp. 57-62, Anchorage, AK, USA.

Zhang, J. & Morris, A.J. (1999). Recurrent neuro-fuzzy networks for nonlinear process modeling. *IEEE Trans. Neural Networks*, Vol. 10, No. 2, 313-326.

21

Evolutionary Computation Applied to Urban Traffic Optimization

Javier J. Sánchez Medina, Manuel J. Galán Moreno and Enrique Rubio Royo
Innovation Center for Information Society (CICEI)
University of Las Palmas de Gran Canaria
Spain

1. Introduction

At the present time, many sings seem to indicate that we live a global energy and environmental crisis. The scientific community argues that the global warming process is, at least in some degree, a consequence of modern societies unsustainable development. A key area in that situation is the citizens mobility. World economies seem to require fast and efficient transportation infrastructures for a significant fraction of the population.
The non-stopping overload process that traffic networks are suffering calls for new solutions. In the vast majority of cases it is not viable to extend that infrastructures due to costs, lack of available space, and environmental impacts. Thus, traffic departments all around the world are very interested in optimizing the existing infrastructures to obtain the very best service they can provide.
In the last decade many initiatives have been developed to give the traffic network new management facilities for its better exploitation. They are grouped in the so called Intelligent Transportation Systems.
Examples of these approaches are the Advanced Traveler Information Systems (ATIS) and Advanced Traffic Management Systems (ATMS). Most of them provide drivers or traffic engineers the current traffic real/simulated situation or traffic forecasts. They may even suggest actions to improve the traffic flow.
To do so, researchers have done a lot of work improving traffic simulations, specially through the development of accurate microscopic simulators. In the last decades the application of that family of simulators was restricted to small test cases due to its high computing requirements. Currently, the availability of cheap faster computers has changed this situation.
Some famous microsimulators are MITSIM(Yang, Q., 1997), INTEGRATION (Rakha, H., et al., 1998), AIMSUN2 (Barcelo, J., et al., 1996), TRANSIMS (Nagel, K. & Barrett, C., 1997), etc. They will be briefly explained in the following section.
Although traffic research is mainly targeted at obtaining accurate simulations there are few groups focused at the optimization or improvement of traffic in an automatic manner — not dependent on traffic engineers experience and "art".

One of the most important problems in traffic optimization is traffic light cycles[1] optimization. This is a hard Combinatorial Problem which seems not to have a known deterministic solution at the present time.
In our group we have been working on the optimization of traffic lights cycles for the better performance of urban traffic networks. As shown in (Brockfeld, Elmar, et al., 2001), traffic light cycles have a strong influence in traffic flow results. For that reason we decided to focused on that problem. We have combined a Genetic Algorithm (GA) as optimization technique with a traffic microscopic simulator running on a scalable MIMD multicomputer[2].
We have tested the fore mentioned three pillar model with some works (Sánchez, J. J. et al., 2004), (Sánchez, J. J. et al., 2005 A), (Sánchez, J. J. et al., 2005 B), (Sánchez, J. J. et al., 2006), (Sánchez, J. J. et al., 2007) and (Sánchez, J. J. et al., 2008).
The rest of this chapter is organized as follows. In section 2 we give a wide survey of the current State of the Art. In 2.4 we briefly expose our own contribution to the matter. In section 3 we explain with some detail the proposed methodology. In section 4 we outline the achieved goals obtained with the explained methodology. Finally, section 5 gives some ideas of research foreseeable trends.

2. State of the art

In this subsection we want to give a survey o some significant works in the area. We have categorized works in three classes: those mostly related to Advanced Traveler Information Services (ATIS); those mainly about Advanced Traffic Management Systems (ATMS), and in a third subset we have called Advanced Traffic Optimization Systems (ATOS), those where traffic is not just managed but optimized — or tried to be optimized — in an automatic manner, without human interaction.

2.1 Advanced traveler information services

Advanced Traveler Information Services are those services that can potentially help drivers to make better decisions in order to reduce their travel time. There are many initiatives in this area. Here we show some examples.
In (Florian, D. G, 2004), this thesis provides an empirical study of the impact of ATIS on transportation network quality of service using an application of DynaMIT (Dynamic network assignment for the Management of Information to Travelers). The main results are that the provision of dynamic route guidance can simultaneously benefit the individual performance of drivers, both guided and unguided, as well as the system performance of existing transportation infrastructure.
In (Hafstein, S. F., et al., 2004) a high resolution cellular automata freeway traffic simulation model applied to a Traffic Information System. They provide a simulation for current traffic zones without loop detectors, and 30 min. and 60 min. future traffic forecasts. They run a java applet in a web page in order to give the network users this useful information.

[1] Traffic light cycle: the finite sequence of states — e.g. green, orange, etc. — that a traffic light runs iteratively.
[2] MIMD: Multiple Instruction Multiple Data: A type of parallel computing architecture where many functional units perform different operations on different data. For example a network of PC's working in parallel.

2.2 Advanced traffic management systems.

Advanced Traffic Management Systems are those systems that help engineers to better manage traffic networks. There are many works around this topic, most of them focused on traffic simulation. Some examples are the following.

The INTEGRATION model has been used to simulate traffic for the Salt Lake Metropolitan Area (Rakha, H., et al., 1998).The objective of this paper is threefold. First, the feasibility of modeling a large-scale network at a microscopic level of detail is presented. Second, the unique data collection challenges that are involved in constructing and calibrating a large-scale network microscopically are described. Third, the unique opportunities and applications from the use of a microscopic as opposed to a macroscopic simulation tool are described.

The MITSIM model (Yang, Q., 1997) has been used to evaluate aspects of both the traffic control system and the ramp configurations of the Central Artery/Tunnel project in Boston. It explicitly incorporates traffic prediction, time variant traffic information, and dynamic route choice.

AIMSUN2 has been used to simulate the Rings Roads of Barcelona (Barcelo, J., et al., 1996). Uses parallel computers to shorten the execution time.

Traffic simulation using CA models has also been performed on vector supercomputers to simulate traffic in shortest possible time (Nagel, K. & Schleicher, A., 1994).

The INTELSIM model is used in (Aycin, M. F. & Benekohal, R. F., 1998) and (Aycin, M. F. & Benekohal, R. F., 1999). In those works a linear acceleration car-following model has been developed for realistic simulation of traffic flow in intelligent transportation systems (ITS) applications. The authors argue that the new model provides continuous acceleration profiles instead of the stepwise profiles that are currently used. The brake reaction times and chain reaction times of drivers are simulated. As a consequence, they say that the good performance of the system in car-following and in stop-and-go conditions make this model suitable to be used in ITS.

Moreover, in (Aycin, M. F. & Benekohal, R. F., 1999) they compare many car-following methods with their proposed method, and with field data.

In (Bham et al., 2004) they proposed a ``high fidelity'' model for simulation of high volume of traffic at the regional level. Their model uses concepts of Cellular Automata and Car-Following models. They propose the concept of Space Occupancy (SOC) used to measure the traffic congestion. Their aim is to simulate high volume of traffic with shorter execution time using efficient algorithms on a personal computer. Like in our case, they based their simulator on Cellular Automata concepts. Although their model could be more accurate than the one of ourselves, in our work we go further using our simulator inside a GA for optimizing the traffic — not just for simulating traffic.

In (Tveit, O., 2003), Dr. Tveit, a senior researcher with *SINTEF*[3], explains that a common cycle time[4] for a set of intersections is a worse approach than a distributed and

---

[3] SINTEF means The Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology.

[4] Common cycle time: This is a very simple way of programming traffic lights in an intersection or groups of intersections. All the traffic lights share a cycle length. The starting point of each one of the states or ***stages*** in the particular cycle of every traffic light may be different, but the cycle period is the same for all of them.

individualized one. His conclusions appear sound and convincing, so we consider them in our approach. In our system every intersection has independent cycles.
In (Smith, M. J., 1988) the use of responsive signals[5], with network capacity (rather than total travel cost) as a control criterion is argued. The capacity of the network is maximized if the signals operate to equalize traffic density on the most occupied parts of the network. This is another example of multiple local optimizations instead of a global optimization, like the one of ours.
In (Logi, F. & Ritchie, S.G., 2001) a knowledge based system is presented for traffic congestion management. The proposed model comprises a data fusion algorithm, an algorithm for selection the suitable control plan, and it presents the proposed plan with an explanation of the reasoning process for helping the traffic operators decisions. They presented also a validation example for displaying the ability of their system to reduce congestion. From our point of view, although this seems a very interesting approach to the matter, both the selection of control strategies and the estimation of future traffic are based on the experience of traffic engineers. In spite of this, in our methodology we use the combination of two widely accepted and trusted techniques. We use a more accurate estimation of future traffic — thought a microsimulator — and a genetic algorithm for the optimization of the traffic flow.

2.3 Advanced traffic optimization systems

TRANSIMS project used CA models to simulate traffic for the city of Fortworth-Dallas using parallel computers (Nagel, K. & Barrett, C., 1997). This paper presents a day-to-day re-routing relaxation approach for traffic simulations. Starting from an initial plan-set for the routes, the route-based microsimulation is executed. The result of the microsimulation is fed into a rerouter, which re-routes a certain percentage of all trips.
In (Wann-Ming Wey, et al., 2001), an isolated intersection is controlled applying techniques based on linear systems control theory to solve the linear traffic model problem. The main contribution of this research is the development of a methodology for alleviating the recurrent isolated intersection congestion caused by high transportation demand using existing technology. Again this work deals with very small scale traffic networks — one intersection.
In (Schutter, B. De & Moor, B. De, 1997) the authors present a single intersection — two two ways streets — model describing the evolution of the queue lengths in each lane as a function of time, and how (sub)optimal traffic switching schemes for this system can be determined.
In (Febbraro, A. Di, et al., 2002) Petri Nets are applied to provide a modular representation of urban traffic networks. An interesting feature of this model is the possibility of representing the offsets among different traffic light cycles as embedded in the structure of the model itself. Even though it is a very interesting work, the authors only optimize the coordination among different traffic light cycles. Our cycle optimization methodology is a complete flexible one because we implicitly optimize not only traffic light offsets but also every *stage length*.

[5] Responsive signals: Traffic signals capable o adapting their state to the current traffic situation near them.

Another interesting work using Petri Nets is (Li, L et al., 2004) where they are applied to control a single intersection by means of programmable logic controllers (PLCs). They compare three methods for modeling the traffic lights at an intersection and found out that the more suitable is the one that combines Petri nets with PLCs. Again, in this research just one intersection is optimized, and not a whole traffic network.
In (Spall, J.C. & Chin, D.C., 1994) the author presented a neural network (NN) approach for optimizing traffic light cycles. A neural network is used to implement the traffic lights control function. The training process of the NN is fed exclusively with real data. This being so, it would only be useful in systems with an on-line data acquisition module installed. However, so far such systems are not common at all.
The "offset-time"[6] between two traffic lights is optimized using Artificial Neural Networks (ANNs) at (López, S., et al. 1999). Although our system does not treat explicitly the offset time parameter we think that our system faces traffic optimization in a much more flexible manner.
In (GiYoung L., 2001) a real-time local optimization of one intersection technique is proposed. It is based on fuzzy logic. Although an adaptive optimization may be very interesting — we checked out this in (Sánchez, J. J. et al., 2004) — we believe that a global optimization is a more complete approach to the problem.
In (You-Sik, H. et al., 1999) authors present a fuzzy control system for extending or shortening the fixed traffic light cycle. By means of electrosensitive traffic lights they can extend the traffic cycle when many vehicles are passing on the road or reduce the cycle if there are few vehicles passing. Through simulation they presented efficiency improvement results. This work performs a local adaptation for a single traffic light instead of a global optimization.
In (Rouphail, N., et al., 2000) an "ad hoc" architecture is used to optimize a 9 intersection traffic network. It uses Genetic Algorithms as an optimization technique running on a single machine. The CORSIM[7] model is used within the evaluation function of the GA. In this work scalability is not addressed. Authors recognize that it is a customized non scalable system. Our system has the scalability feature thanks to the intrinsic scalability of the Beowulf Cluster and the parallel execution of the evaluation function within the GA.
In (You Sik Hong, et al., 2001) the concept of the optimal green time algorithm is proposed, which reduces average vehicle waiting time while improving average vehicle speed using fuzzy rules and neural networks. Through computer simulation, this method has been proven to be much more efficient than using fixed time cycle signals. The fuzzy neural network will consistently improve average waiting time, vehicle speed, and fuel consumption. This work only considers a very small amount of traffic signals — two near intersections — in the cycle optimization. We do agree with them about the non-suitability of fixed cycles.
An interesting combination of Genetic Algorithms and Traffic Simulation is published in (Taniguchi, E. & Shimamoto, H., 2004). In this work a routing and scheduling system for freight carrier vehicles is presented. They use Genetic Algorithms as optimization technique. The objective of the GA is the minimization of the costs of travel. A dynamic vehicle routing

[6] Offset-time: the time since a traffic light turn green until the next traffic light — for example, in a boulevard — turns also green.
[7] CORSIM: Corridor Traffic Simulation Model (Halati A. et al., 1997).

algorithm is proposed and tested with a test road network. The implemented traffic simulation model is macroscopic.
Another very interesting work is presented in (Varia, H.R. & Dhingra, S.L., 2004). A dynamic system-optimal (DSO) traffic assignment model is formulated for a congested urban network with a number of signalized intersections. They also combine traffic simulation with Genetic Algorithms. The aim of this work is to assign any traveler a route. A GA is used to minimize the users total travel time. A macroscopic model is used for the estimation of traffic delays. The DSO problem is solved with fixed signal timings, and with the optimization of signal timings.
In (Vogel, A. et al., 2000) every intersection is optimized considering only local information. Moreover, it can be adapted to short and long term traffic fluctuations. In our case we perform a global optimization instead of multiple local optimizations. We think that our approach may be a more efficient exploitation of the traffic infrastructure.
A very interesting work is published in (Wiering, M. et al., 2004). In this work, traffic is regarded as formed by a set of intersections to be optimized in a stand alone manner. They proposed to use reinforcement learning algorithms to optimize what they consider a multi-agent decision problem. We do not agree with them. Although a local optimization can obviously reduce average waiting times of cars — as it seems to happen with simulated tests at this work — we think that a global optimization taking into account every intersection in a zone should be more profitable.

2.4 Own contribution.

In this subsection we have included our contribution to the art. In (Sánchez, J. J. et al., 2004) we presented our methodology for the optimization of Traffic Light Cycles in a Traffic Network. The very good results of a parallel speed-up study convinced us that it was advisable to use a “Beowulf Cluster” as parallel computing system.
In OPTDES IV[8] we shared a scalability study on that architecture. We ran tests using four networks from 80 up to 1176 cells. In that work we found out that our system had a very good performance for all cases.
In (Sánchez, J. J. et al., 2005 A) we compared two versions of our microscopic traffic simulator: a stochastic versus a deterministic traffic simulator. There were three differences between the stochastic and the deterministic version: The cells updating order; the new vehicle creation time and the acceleration probability. From that work we realized that the stochastic simulator is a suitable — convergent — statistical process to compare with; and we demonstrated that the deterministic simulator outputs are highly linearly correlated with the stochastic ones. Therefore, our deterministic simulator can arrange the population ranking in order of fitness at least as well as the stochastic simulator, but with a remarkably lower computing time.
In the research presented for CIMCA2005 (Sánchez, J. J., Galán, M. J., & Rubio, E., 2005 B) we described the difference between two sorts of encoding, yielding different crossover and mutation strategies. The main achievement in that work was to demonstrate — by means of a wide set of tests — that, at least for our particular case, a bit level crossover combined with a variable mutation probability means a great saving of computing time. Besides, we noticed

[8] Optimization and Design in Industry IV, Tokyo, Japan, (September, 26-30th, 2004)

how that choice lets the algorithm cover the solution space faster due to a bigger gene variability between generations. This combination seems to avoid premature convergence.
In ECT2006 we delivered a research (Sánchez, J. J. et al., 2006) that included two goals. First, we introduced a new methodology – such a visual one – helping those practitioners occupied tuning a GA by giving them much deeper knowledge of how the GA is doing than they had before. Furthermore, we tried this new methodology with a wide set of tests. We used it for tuning the genetic algorithm within our traffic optimization architecture applied to a particular network.
We presented another research in Eurocast 2007 (Sánchez, J. J. et al., 2007). In that communication we shared a study considering three candidate criteria as a first step toward extending our fitness function towards a multicriteria one. The criteria where related to the total number of vehicles that left the network, the occupancy of the network and greenhouse gases emissions. We performed a correlation study and, although conclusions where not definitive, we obtained some interesting conclusions about the relationship among those parameters.
Finally, soon we will publish an optimization research (Sánchez, J. J. et al., 2008) for another traffic network situated in Santa Cruz de Tenerife, Spain. Although the scale of that network is not as large as the one treated for the current paper, results are promising.

3. Methodology

3.1 Optimization model

The architecture of our system comprises thre items, namely a Genetic Algorithm (GA) as Non- Deterministic Optimization Technique, a Cellular Automata (CA) based Traffic Simulator inside the evaluation routine of the GA, and a Beowulf Cluster as MIMD multicomputer. Through this section we will give a wide description for the GA and the CA based Traffic Simulator used in our methodology. Finally, a brief description of the Beowulf Cluster sill also be provided.

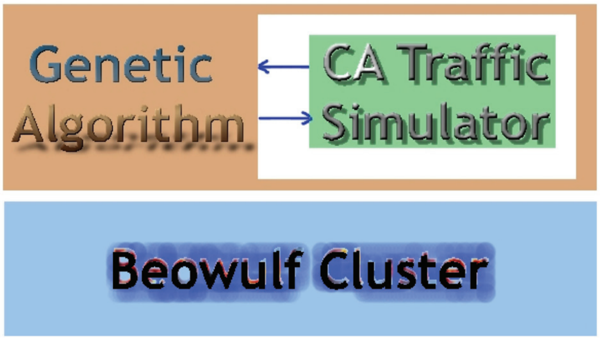


Fig. 1. Model Architecture

3.1.1 Genetic algorithm

In this subsection we will describe the genetic algorithm utilized.

3.1.1.1 Optimization criterion. Fitness function

After testing several criteria we found out that we obtained the better results just by using the absolute number of vehicles that left the traffic network once the simulation finishes.

During the traffic simulation many new vehicles are created as if they were arriving at the inputs of the network. Furthermore, during the simulation many vehicles reach their destination point and leave the network. The number of vehicles that reach their destination point easily illustrates how the simulation was, and consequently helps us to compare a particular cycle combination with another.

Other optimization criteria tested are the following:

- Mean time at the network – Mean Elapsed Time, MET. During the simulation, the arrival and departure time of every vehicle is stored. With these values we can easily calculate the number of iterations (or seconds) it takes any vehicle to leave the network. Once the simulation finishes the average time at the network is calculated.
- Standard Deviation values of vehicle times at the network.
- A linear combination between the MET and the Standard Deviation of vehicle times at the network.
- A linear combination between the MET and the total number of vehicles that have left the network during the simulation.
- The traffic network mean occupancy density. To calculate this parameter we divided the network into small sections and counted the number of vehicles inside every section.

As we search the optimization criteria for our system we encountered an unexpected problem. If we included the minimization of the MET in a multicriteria evaluation function we provoked a very undesirable effect. The chromosomes that blocked the network faster were the best marked. That is because only a few vehicles were able to leave the network (in a small amount of iterations) before it collapsed. Hence, we obtained very "good" values but caused by "false" optimal cycle combinations. Therefore, we resigned to include that criterion in our fitness function.

3.1.1.2 Chromosome encoding

In figure 1 we present the chosen encoding used in our methodology. In this figure we represent a chromosome example for a very simple traffic network. It consists of only two intersections and two traffic lights for each intersection.

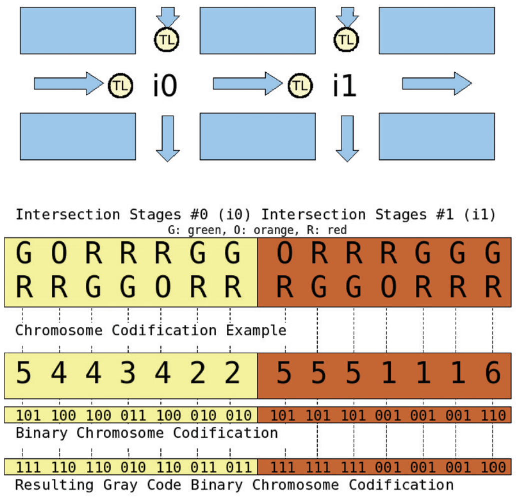

TL
TL
TL
i0
TL
i1
Intersection Stages #0 (i0) Intersection Stages #1 (i1)
G: green, O: orange, R: red
G O R R R G G O R R R G G G
R R G G O R R R G G O R R R
Chromosome Codification Example
5 4 4 3 4 2 2 5 5 5 1 1 1 6
101 100 100 011 100 010 010 101 101 101 001 001 001 110
Binary Chromosome Codification
111 110 110 010 110 011 011 111 111 111 001 001 001 100
Resulting Gray Code Binary Chromosome Codification

Fig. 2. Chromosome Codification

Below the traffic network we have put the *stages*[9] of each traffic light separated in two different color regions, one for each one of the two intersections. The traffic light state at each *stage* may be green (G), orange (O) or red (R).
This stages sequence is preestabilished, and wil cycle *ad infinitum* — or until we stop the corresponding simulation. The objective of our system is to optimize the duration of each *stage* (in seconds) in order to get the very best traffic behavior from the network under study.
In figure 1 a chromosome encoding example I included. It can be seen that through several translation steps we obtained a binary Gray Code encoding (Black, P. E., 2005). We have proven out this methodology to be very efficient for our case in (Sánchez, J. J. et al., 2005 B).
We use Gray Code because it is designed in such a manner that when a bit changes its value — when mutation occurs — the *stage length* value only increases or decreases one unit. This is a desirable feature because it makes the search space to *conform* with the "Hamming Distance Metric".

3.1.1.3 Initial population

Before the GA starts we created an initial population. Initially we set a time range for every preestablished *stage*. Each individual is created by choosing a random value within its corresponding range.

3.1.1.4 Random number generation

For the random number generation we have employed the MT19937 generator of Makoto Matsumoto and Takuji Nishimura, known as the "Mersenne Twister" generator. It has passed the DIEHARD statistical tests (Matsumoto, M. & Nishimura, T., 1998). The seeds for that algorithm were obtained from the ``/dev/urandom'' device provided by the Red Hat 9 operating system.

3.1.1.5 Selection strategy

We have chosen a Truncation and Elitism combination as selection strategy. It means that at every generation a little group of individuals — the best two individuals in our case — is cloned to the next generation. The remainder of the next generation is created by crossovering the individuals from a best fitness subset — usually a 66 percent of the whole population.
This combination seems to be the most fitted to our problem among a set of selection strategies tested. However, we do not discard to change it if better results seem attainable.
Other selection strategies previously tested — and discarded — for this problem are succinctly explained as follows:

- Elitism: The population is ordered by fitness and a small set with the best individuals (elite) is cloned to the next generation.
- Truncation: The population is ordered by fitness. Then the population is divided into two sets, one to survive and the another one is simply discarded.
- Tournament: Small groups of individuals are chosen at random. The best fitness individual of each one of them is selected.
- Random Tournament: Like the Tournament Selection but the best individual is not always selected. It will depend on a probability value.
- Roulette Linear Selection: Every individual has a survival probability proportional to its fitness value.
- Elitism plus Random Tournament.

[9] Stage: Every one of the states associated to an intersection, that contains a set of traffic lights.

3.1.1.6 Crossover operator

We have tested some different crossover operators: Uniform Crossover, Two Points Crossover at fixed points and Two Points Crossover at random points. We reached the conclusion that for our case the better one was the third one.

For a couple of parents, it simply chooses two random points at each one of the two chromosomes, cut them into three pieces and then interchanges the central chunk of them.

3.1.1.7 Mutation operator

The value of a randomly chosen bit in the chromosome is just flipped.

The mutation probability is not fixed. It starts with a very high mutation probability that will decrease multiplied by a factor value in the range (0,1) until it reaches probability values near to the inverse of the population size as approaching the end of the planned number of generations.

3.1.2 Traffic simulator

Traffic Simulation is known to be a very difficult task. There are mainly two different traffic simulations paradigms. The first one is the Macroscopic model. Macroscopic simulators are based on Fluid Dynamics, since they consider traffic flow as a continuous fluid. The second paradigm is the one that includes Microscopic simulators. For them, traffic is considered as a collection of discrete particles following some rules about their interaction. In the last decade there has been a common belief about the better performance of Microscopic simulators to do Traffic Modeling. One Microscopic model widely used is the Cellular Automata Model.

There has been a large tradition of macroscopic approaches for traffic modeling. In the 50's some “first order” continuum theories of highway traffic appeared. In the 70's and later on some other “second order” models were developed in order to correct the formers' deficiencies. References (Helbing, D., 1995); (Kerner, B. S., & Konhäuser, P., 1994); (Kühne, R. D., et al., 1991); (Kühne, R. D., 1991); (Payne, H. J., 1979) and (Witham, G. B., 1974) may illustrate some of these models. However, in (Daganzo, C. F., 1995) “second order” models are questioned due to some serious problems like negative flows predictions and negative speeds under certain conditions.

Nowadays the microscopic simulators are widely used. One reason for this fact is that macroscopic simulators can not model the discrete dynamics that arises from the interaction among individual vehicles (Benjaafar, S., et al., 1997). Cellular Automata are usually faster than any other traffic microsimulator (Nagel, K., & Schleicher, A., 1994), and, as said in (Cremer, M. & Ludwig, J., 1986) “the computational requirements are rather low with respect to both storage and computation time making it possible to simulate large traffic networks on personal computers”

3.1.2.1 The cellular automata as inspiring model

Cellular Automata Simulators are based on the Cellular Automata Theory developed by John Von Neumann (Neumann, J. von, 1963) at the end of the forties at the Logic of Computers Group of the University of Michigan. Cellular Automata are discrete dynamical systems whose behavior is specified in terms of local relation. Space is sampled into a grid, with each cell containing a few bits of data. As time advances, each cell decides its next state depending on the neighbors state and following a small set of rules.

In the Cellular Automata model not only space is sampled into a set of points, but also time and speed. Time becomes iterations. A relationship between time and iterations is set. For instance, 1(sec.) ≡ 1 (iteration). Consequently, speed turns into “cells over iterations”.

In (Brockfeld, E. et al., 2003) we can find a well described list of microscopic models and a comparative study of them. Although conclusions are not definitive, this work seems to demonstrate that models using less parameters have a better performance.
We have developed a traffic model based on the SK[10] model (Krauss, S., et al., 1997) and the SchCh[11] model (Schadschneider, A. et al., 1999). The SchCh model is a combination of a highway traffic model (Nagel, K. & Schreckenberg, M., 1992) and a very simple city traffic model (Biham et al., 1992). The SK model adds the "smooth braking" for avoiding abrupt speed changes. We decided to base our model in the SK model due to its better results for all the tests shown in (Brockfeld, E. et al., 2003).

3.1.2.2 Our improved cellular automata model

Based on the Cellular Automata Model we have developed a non-linear model for simulating traffic behavior. The basic structure is the same as the one used in Cellular Automata. However, in our case we add two new levels of complexity by creating two new abstractions: "Paths"and "Vehicles".
"Paths" are overlapping subsets included in the Cellular Automata set. There is one "Path" for every origin-destination pair. To do this, every "Path" has a collection of positions and, for each one of them, there exists an array of allowed next positions. In figure 2 we try to illustrate this idea.
"Vehicles" consists of an array of structures, each one of them having the following properties:

1. Position: the Cellular Automaton where it is situated. Note that every cell may be occupied by one and only one vehicle.
2. Speed: the current speed of a vehicle. It means the number of cells it moves over every iteration.
3. Path: In our model, every vehicle is related to a "path".

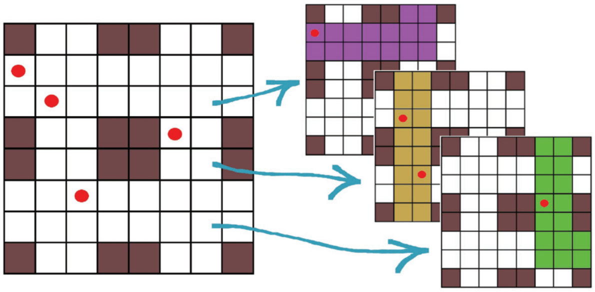


Fig. 3. Paths in our Improved Cellular Automata Model

These are the rules applied to every vehicle:

1. A vehicle ought to accelerate up to the maximum speed allowed. If it has no obstacle in its way (another vehicle, or a red traffic light), it will accelerate at a pace of 1 point per iteration, every iteration.
2. If a vehicle can reach an occupied position, it will reduce its speed and will occupy the free position just behind the preceding.

[10] Stephan Krauss, the author.
[11] Andreas Schadschneider and Debashish Chowdhury, the authors.

3. If a vehicle has a red traffic light in front of, it will stop.
4. Smooth Braking: Once the vehicle position is updated, then the vehicle speed is updated too. To do this, the number of free positions from the current position ahead is taken into account. If there is not enough free space for the vehicle to move forward on the next iteration going at its current speed (hypothetically, since in the next iteration the traffic situation may change), it will reduce its speed in one unit.
5. Multi-lane Traffic: When a vehicle is trying to move on, or update its speed, it is allowed to consider positions on other parallel lanes. For every origindestination couple (path), at every point there exists a list of possible next positions. The first considered is the one straight forward. If this one is not free, there may be more possible positions in parallel lanes that will be considered. Of course, this list of possible next positions is created taking the basic driving rules into account.

By means of these rules we can have lots of different path vehicles running in the same network. This model may be seen as a set of N_{paths} traditional Cellular Automata networks working I parallel over the same physical points.

Note that, so far, we are not considering a different behavior for the green and the orange state. However, our architecture is designed in such a manner that we can modify this whenever we want to, with a small effort.

3.1.3 Beowulf cluster

The Architecture of our system is based on a five node Beowulf Cluster, due to its very interesting price/performance relationship and the possibility of employing Open Software on it. On the other hand, this is a very scalable MIMD computer, a very desirable feature in order to solve all sort — and scales — of traffic problems.

Every cluster node consists of a Pentium IV processor at 3.06 GHz with 1 GB DDR RAM and 80GB HDD. The nodes are connected through a Gigabit Ethernet Backbone. Every node has the same hardware, except the master node having an extra Gigabit Ethernet network card for “out world” connection.

Every node has installed Red Hat 9 on it — Kernel 2.4.20-28.9, glibc ver. 2.3.2 and gcc ver. 3.3.2. It was also necessary for parallel programming the installation of LAM/MPI (LAM 6.5.8, MPI 2).

In our application there are two kinds of processes, namely *master* and *slave* process. There is only one master process running on each test. At every generation it sends the chromosomes (*MPI_Send*) to slave processes, receives the evaluation results (*MPI_Recv*) and creates the next population. Slave processes are inside an endless loop, waiting to receive a new chromosome (*MPI_Recv*). Then they evaluate it and send the evaluation result (*MPI_Send*).

4. Achieved goals and future aims

The main goal obtain with this methodology is its application to two real world test cases in a simulated environment. To do so we have earned both collaboration agreements with Saragossa and Santa Cruz de Tenerife local governments.

4.1 La Almozara

In figures 4 and 5 the Saragossa district number 7 — “La Almozara” — is shown. We want to remark the large scale of the zone.

In our simulated environment we improve 10% fitness, in comparison with results obtained with the times currently used in the zone.

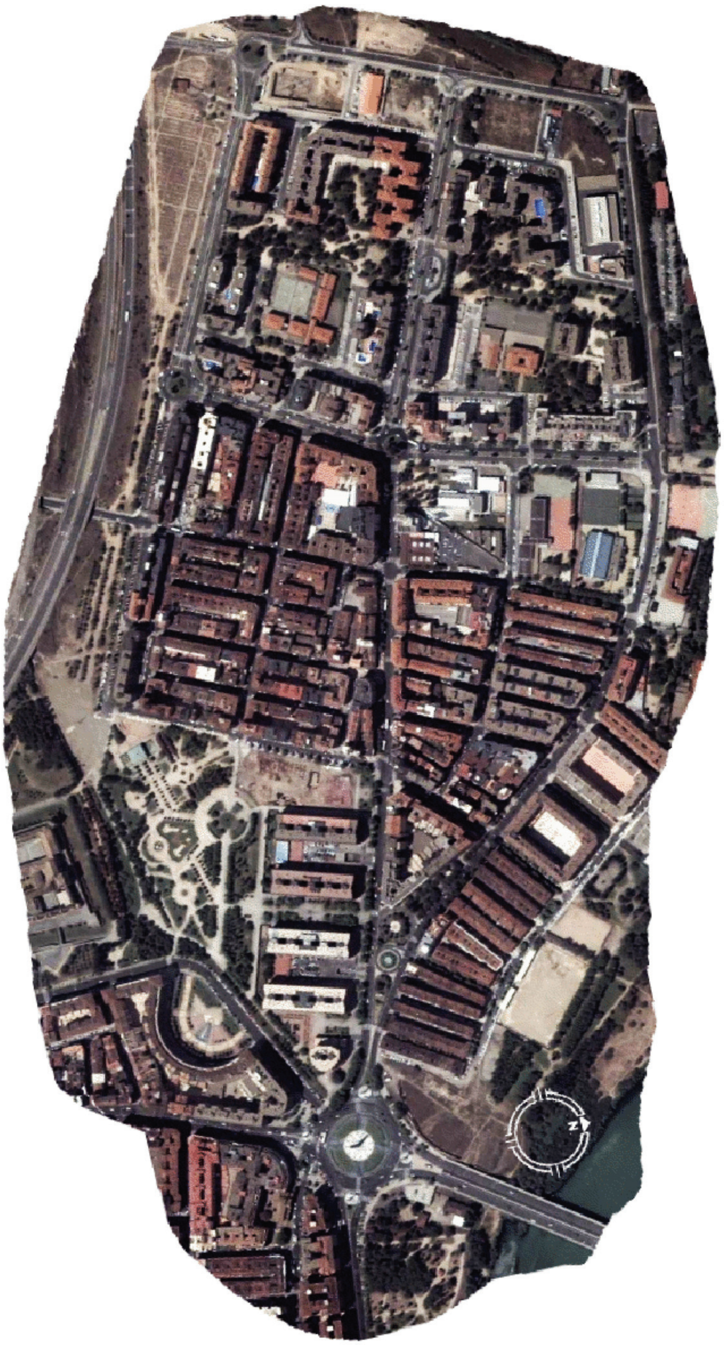


Fig. 4. Eye view of "La Almozara" in Saragossa (from Google Maps).

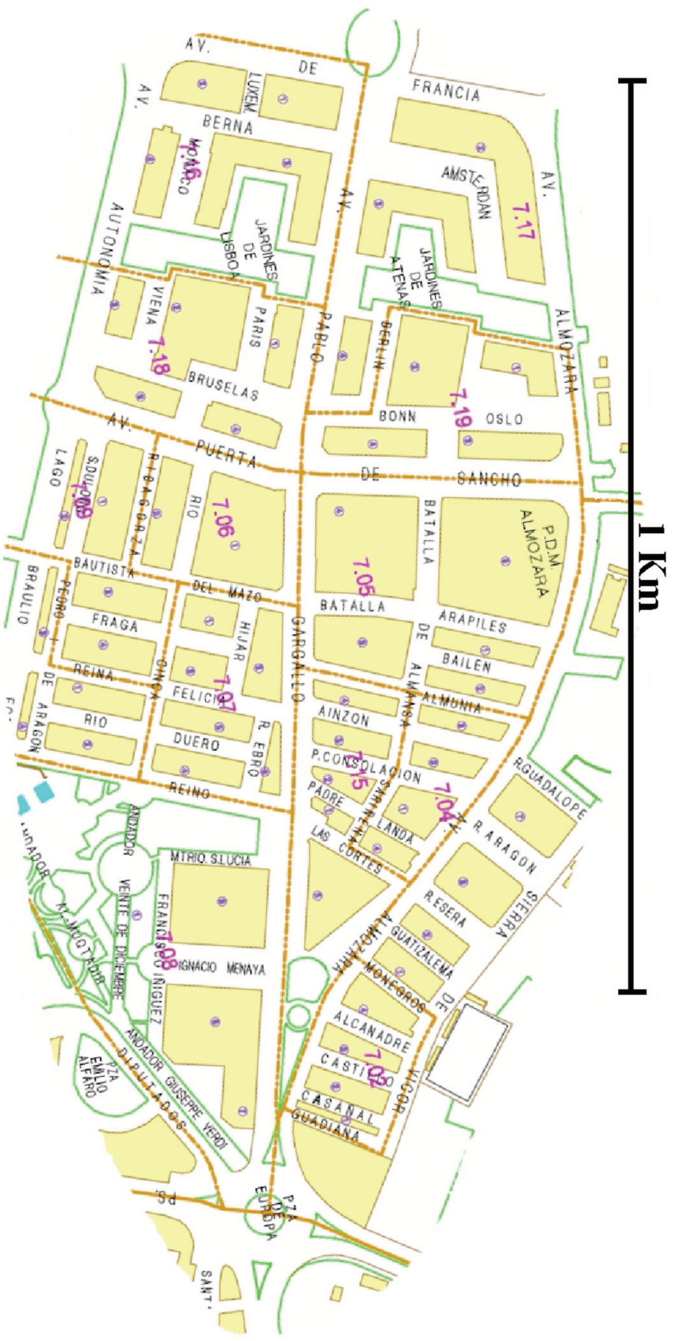


Fig. 5. "La Almozara" Zone Scale.

The statistics provided reflected a scarcely occupied network (under 10%). Everything seems to indicate that when the traffic zone is that empty, no matter what is the combination of traffic light times, the network would have similar outputs. In other words, a nearly empty traffic network is not likely to be improved just by optimizing traffic light times.
Nevertheless, we are carrying out a stud increasing the network occupation, and results seem really promising.

4.2 Las Ramblas

Illustrations 6 and 7 show the treated zone in Santa Cruz de Tenerife — Canary Islands.
In figure 8 we represent the performance results using the solutions given to us by the Local Government — the first 9 points. The rest of the points represent the performance obtained using the solutions yielded by our method. One may observe that there is an obvious improvement using our times. Likewise, our 150 solutions seem to be more stable than theirs.
Figure 9 shows the improvement — as a percentage — of the mean, best and worst values of our 150 solutions against the 9 supplied.
This improvement (%) stays within a range fro 0.53 to 26.21. The smallest difference between the optimized results and the supplied simulated results is 12 vehicles — solution 43 with respect to supplied 'R1'. The biggest difference is 521 vehicles — distance from solution 69 to supplied 'R6'.
The improvement stays within a range from 0.53 to 26.21. The smallest difference between the optimized results and the supplied simulated results is 12 vehicles — solution 43 with respect to supplied 'R1'. The biggest difference is 521 vehicles — distance from solution 69 to supplied 'R6'.
One important conclusion is that we can clearl improve the supplied times in our simulated environment. So, we can seek optimal cycle time combinations for the traffic lights programming using our architecture with an appropriate amount of statistics. We have proven this with a real world test case (nevertheless, using a simulated environment).
This is useful as reducing travel times in a city clearly means saving money and reducing environmental impact.
It is important to note that our system is intrinsically adaptable to particularized requirements, such as "Path" preferences, minimum and maximum *stage length*, etc. In this sense, our system is flexible and adaptable.

4.3 Future work aims

Currently, we are planning to extend the model to a dynamic version. To do so we will need new agreements with traffic departments in order to obtain real time data.
On a second step we plan to validate our model running real traffic lights with times provided by us. This will require real commitment from any public institution, and we are convinced that we will earn that confidence soon.
Finally, we are considering the possibility of extending our model to take into account the "Pedestrians' Interaction" and including environmental aspects in the optimization criteria using a multiobjective approach.

5. Research trends

Forecasting research trends is always tricky. Fortunately, new discoveries surprise the scientific community every day, discarding common places and settled ideas.

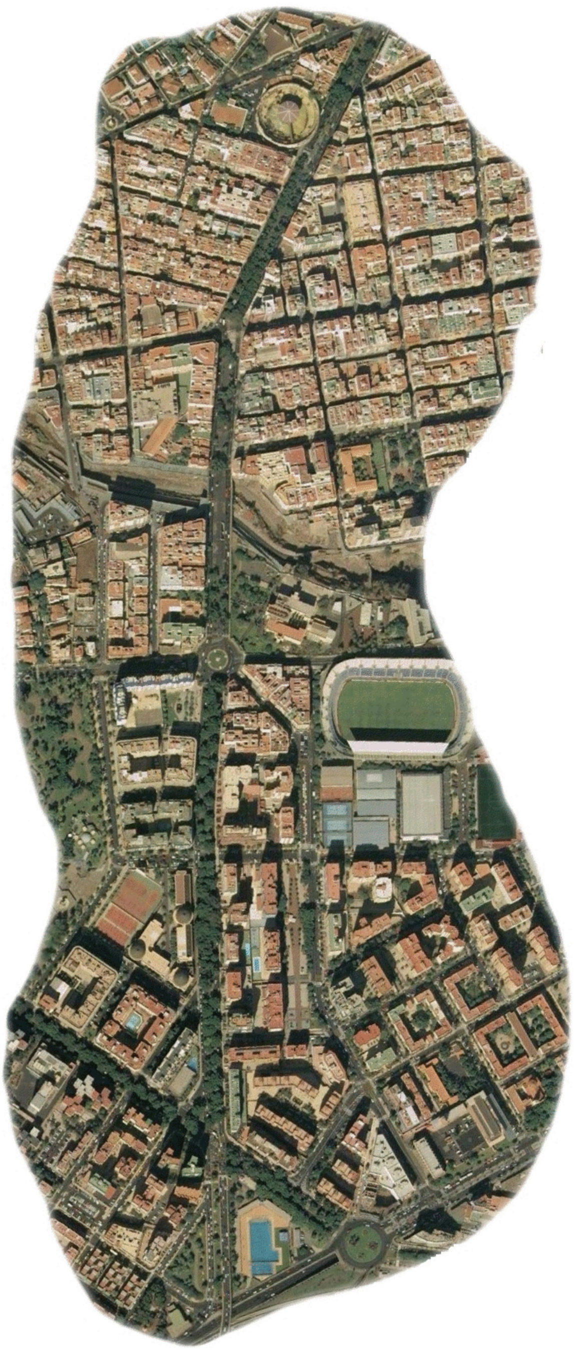


Fig. 6. Eye View of "Las Ramblas", in Santa Cruz de Tenerife (from Google Maps)

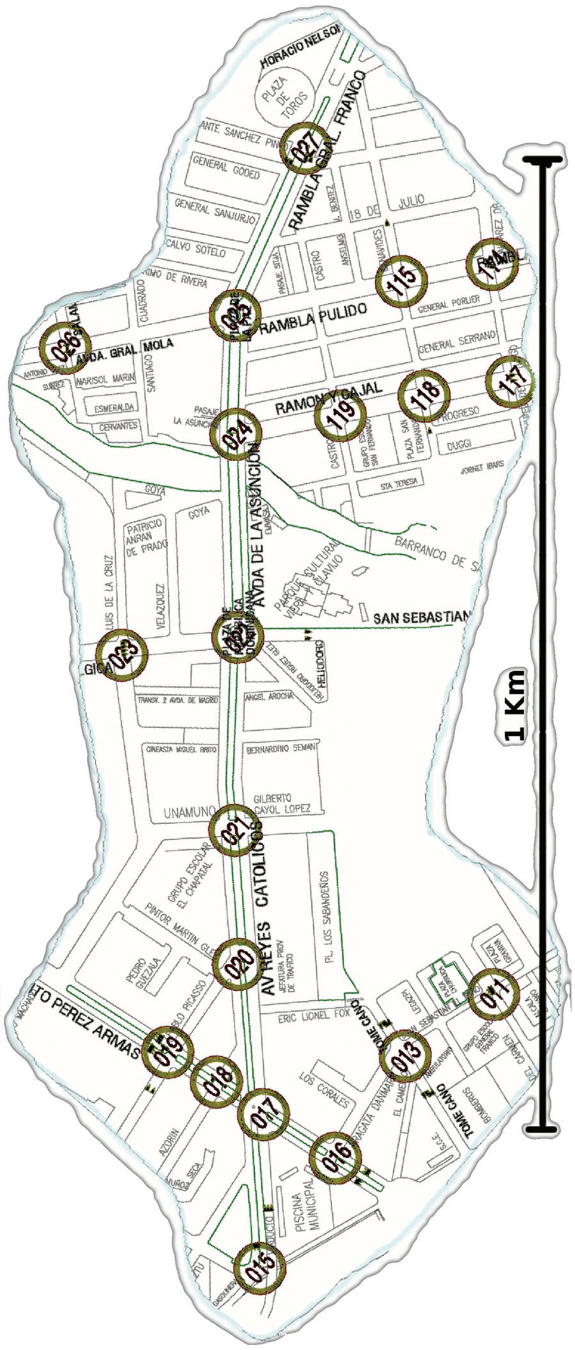


Fig. 7. "Las Ramblas" Zone Scale

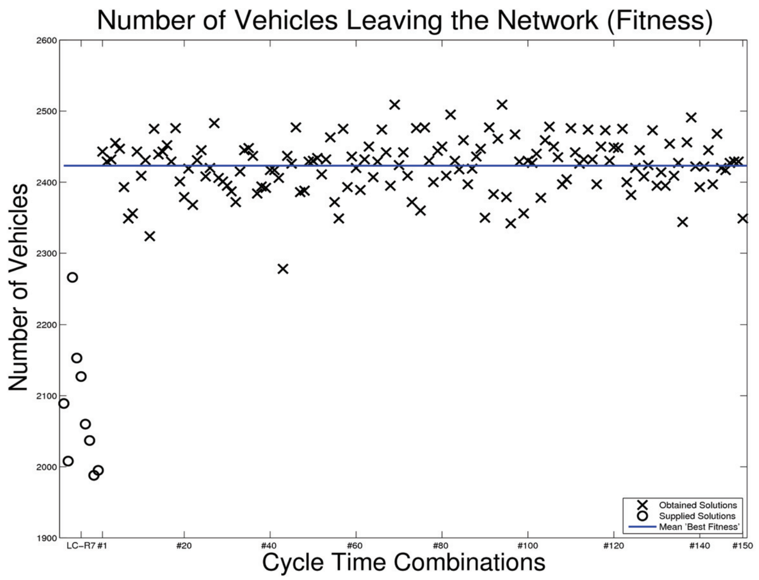


Fig. 8. Number of Vehicles Leaving the Network for the 9 Solutions Provided (on the Left) and the 50 Solutions Calculated by the System

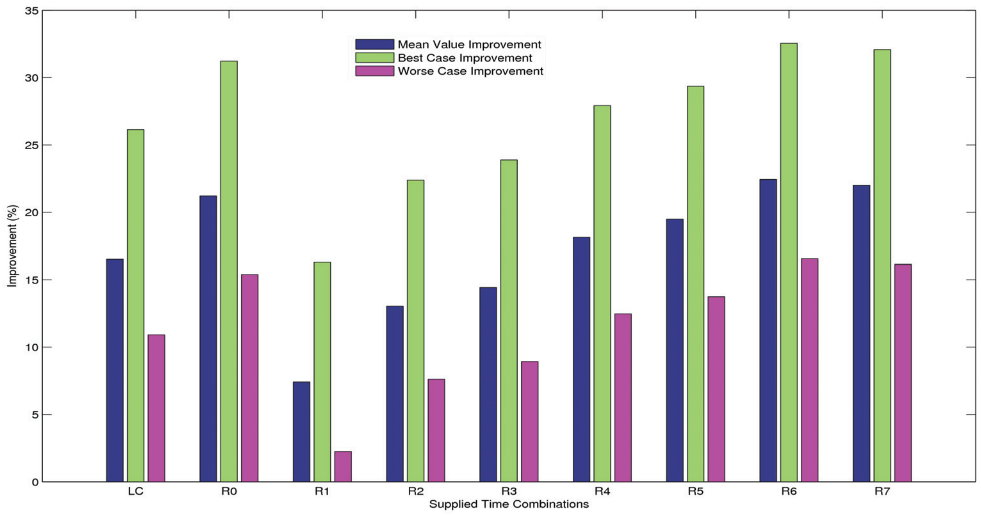


Fig. 9. Improvement of Fitness

However, everything seems to indicate that human control of traffic will be progressively replaced by automatic control systems, at least in crowded scenarios.

First, public traffic facilities, and then private vehicles, could be controlled by safe and automatic systems, maximizing the use of infrastructures, the safety of passengers, and minimizing the environmental impact of mobility.

6. Acknowledgements

The authors would like to acknowledge th Saragossa Town Hall Traffic Department for their kind help and assistance.
We would like to thank Mrs. María Luisa Sein- Echaluce Lacleta too, from the University of Saragossa for her kind help to obtain the data from the Saragossa City Traffic Department.
Finally, the authors would like to acknowledge the Santa Cruz de Tenerife City Council Traffic Department for their kind help. We specially would like to thank Mr. Hilario Rodríguez González for his willingness to collaborate with us, making this work feasible.

7. References

Aycin, M. F. and Benekohal, R. F., (1998). Linear acceleration car-following model development and validation, Transportation research record, 1644, 10—19

Aycin, M. F. and Benekohal, R. F., (1999). Comparison of car-following models for simulation, Transportation research record, 1678, 116—127

Barcelo, J., et al., (1996). The Parallelization of AIMSUN2 microscopic simulator for ITS applications, Proceedings of The 3rd World Congress on Intelligent Transportation Systems

Benjaafar, S., et al., (1997). Cellular Automata for Traffic Flow Modeling, CTS 97-09, Intelligent Transportation Systems Institute

Bham, Ghulam H. and Benekohal, Rahim F., (2004). A high fidelity traffic simulation model based on cellular automata and car-following concepts, Transportation Research Part C, 12, 1 —32

Biham, Ofer, et al., (1992). Self-organization and a dynamical transition in traffic-flow models, Phys. Rev. A, 46, 10, R6124—R6127

Black, P. E., (2005). ``Gray code'', from Dictionary of Algorithms and Data Structures, Paul E. Black, ed., NIST

Brockfeld, Elmar, et al., (2001). Optimizing traffic lights in a cellular automaton model for city traffic, Phys. Rev. E, 64., 056132

Brockfeld, E. et al., (2003). Towards Benchmarking Microscopic Traffic Flow Models, Transportation Research Record, 1852, 124—129, Washington, DC; National Academy Press

Cremer, M. & Ludwig, J., (1986). A fast simulation model for traffic flow on the basis of Boolean operations, Mathematics and Computers in Simulation, 28, 297—303

Daganzo, C. F., (1995). Requiem for second order fluid approximations of traffic flow, Transportation Research B

Febbraro, A. Di, et al., (2002) On applying Petri nets to determine optimal offsets for coordinated traffic light timings, On applying Petri nets to determine optimal offsets for coordinated traffic light timings, The IEEE 5th International Conference on Intelligent Transportation Systems, 773 — 778

FHWA, (1980). Development and Testing of INTRAS, a Microscopic Freeway Simulation Model, Program Design, Parameter Calibration and Freeway Dynamics Component Development, I, FHWA/RD-80/106

Florian, D. G, (2004). Simulation-based evaluation of Advanced Traveler Information Services, PhD Thesis, Massachusetts Institute of Technology. Technology and Policy Program

GiYoung L. et al., (2001). The optimization of traffic signal light using artificial intelligence, Fuzzy Systems, 2001. The 10th IEEE International Conference on, 3, 1279—1282

Hafstein, S. F., et al., (2004). A High-Resolution Cellular Automata Traffic Simulation Model with Application in a Freeway Traffic Information System, Computer-Aided Civil and Infrastructure Engineering, 19, 338—350

Halati A. et al., (1997). CORSIM - Corridor Traffic Simulation Model, The 76th Annual Meeting of the Transportation Research Board, Washington, D. C.

Helbing, D., (1995). An improved fluid dynamical model for vehicular traffic, Physical Review. E, 51, 3164

Kerner, B. S., & Konhäuser, P., (1994). Structure and Parameters of Clusters in Traffic Flow, Physical Review E, 50:54

Kühne, R. D., et al., (1991). Macroscopic simulation model for freeway traffic with jams and stop-start waves, WSC '91: Proceedings of the 23rd conference on Winter simulation, 762 —770, Phoenix, Arizona, USA

Kühne, R. D., (1991). Verkehrsablauf auf Fernstraßen, Phys. Bl., 47, 3:201

Krauss, S., et al., (1997). Metastable states in a microscopic model of traffic flow, Phys. Rev. E, 55, 5597 — 5605

Li, L. et al., (2004). Implementation of Traffic Lights Control Based on Petri Nets, Intelligent Transportations Systems, IEEE 2003, 3, 1749— 1752

Logi, F. and Ritchie, S.G., (2001). Development and evaluation of a knowledge-based system for traffic congestion management and control, Transportation Research Part C: Emerging Technologies, 9, 6, 433—459

López, S., et al., (1999). Artificial Neural Networks as useful tools for the optimization of the relative offset between two consecutive sets of traffic lights. LNCS. Foundations and Tools for Neural Modeling. Springer-Verlag, 795—804

Matsumoto, M. & Nishimura, (1998). T., Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator, ACM Transactions on Modeling and Computer Simulation, 8, 3—30

Nagel, K. & Schreckenberg, M., (1992). A Cellular Automaton Model for Freeway Traffic, Journal de Physique I France, 33, 2, 2221—2229

Nagel, K., & Schleicher, A., (1994). Microscopic traffic modeling on parallel high performance computers, Parallel Computing, 20, 1, 125— 146

Nagel, K. & Barrett, C., (1997). Using microsimulation feedback for trip adaptation for realistic traffic in Dallas, Int.J.Mod.Phys.C, 505

Neumann, J. von, (1963). The General and Logical Theory of Automata. John von Neumann— Collected Works, 5. A. H. Taub (ed.), Macmillan, New York, 288—328.

Payne, H. J., (1979). Freflo: A macroscopic simulation model of freeway traffic, Transportation Research Record, 722:68

Rakha, H., et al., (1998). Construction and Calibration of a Large Scale Microsimulation Model of the Salt Lake Area, Transportation Research Record, 1644, 93—102

Rouphail, N., et al., (2000). Direct Signal Timing Optimization: Strategy Development and Results, In XI Pan American Conference in Traffic and Transportation Engineering

Sánchez, J. J., Galán, M. J., & Rubio, E., (2004). Genetic Algorithms and Cellular Automata: A New Architecture for Traffic Light Cycles Optimization, Proceedings of The Congress on Evolutionary Computation 2004 (CEC2004), 2, 1668—1674, Portland, Oregon (USA)

Sánchez, J., Galán, M. and Rubio, E. (2005A). Stochastic Vs Deterministic Traffic Simulator. Comparative Study for its Use within a Traffic Light Cycles Optimization Architecture, Proceedings of The International Workconference on the Interplay between Natural and Artificial Computation (IWINAC), 2, 622— 631

Sánchez, J. J., Galán, M. J., & Rubio, (2005B). E., Bit Level Versus Gene Level Crossover in a Traffic Modeling Environment, International Conference on Computational Intelligence for Modelling Control and Automation - CIMCA'2005, 1, 1190 - 1195, Vienna, Austria

Sánchez, J. J., Galán, M. J. and Rubio, E., (2006). A Visual and Statistical Study of a Real World Traffic Optimization Problem, Proceedings of the Fifth International Conference on Engineering Computational Technology, Civil- Comp Press, paper 147, Stirlingshire, United Kingdom

Sánchez, J. J., Galán, M. J. and Rubio, E., (2007). Study of Correlation Among Several Traffic Parameters Using Evolutionary Algorithms: Traffic Flow, Greenhouse Emissions and Network Ocuppancy, Proceedings of the EUROCAST 2007 conference. 1134-1141

Sánchez, J. J., Galán, M. J. and Rubio, E., (2008). Applying a Traffic Lights Evolutionary Optimization Technique to a Real Case: “Las Ramblas” Area in Santa Cruz de Tenerife, Evolutionary Computation, IEEE Transactions on, Volume 12, Issue 1, Feb. 2008 Page(s):25 - 40

Schadschneider, A., Chowdhury, D., Brockfeld, E., Klauck, K., Santen, L., & Zittartz, J., (1999). A new cellular automata model for city traffic, Traffic and Granular Flow '99: Social, Traffic, and Granular Dynamics, Berlin

Schutter, B. De & Moor, B. De, (1997). Optimal Traffic Light Control for a Single Intersection, Proceedings of the 1997 International Symposium on Nonlinear Theory and its Applications (NOLTA'97), 1085—1088

Smith, M. J., (1988). Optimum network control using traffic signals, UK Developments in Road Traffic Signalling, IEE Colloquium on, 8/1 — 8/3

Spall, J.C. & Chin, D.C., (1994). A model-free approach to optimal signal light timing for system-wide traffic control, 33rd IEEE Conference on Decision and Control, 1994, 1868 — 1875

Taniguchi, E. and Shimamoto, H., (2004). Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times, Transportation Research Part C: Emerging Technologies, 12, 3—4, 235— 250

Tveit, O., (2003) . Common cycle time - a strength or barrier in traffic light signaling, Traffic Engineering and Control (TEC) Magazine, 1, 44, 19—21

Varia, H.R. and Dhingra, S.L., (2004). Dynamic Optimal Traffic Assignment and Signal Time Optimization Using Genetic Algorithms, Computer-Aided Civil and Infrastructure Engineering, 19, 260—273

Vogel, A. et al., (2000). Evolutionary Algorithms for Optimizing Traffic Signal Operation, ESIT 2000, 83—91, Aachen, Germany

Wann-Ming Wey, et al., (2001). Applications of linear systems controller to a cycle-based traffic signal control, Intelligent Transportation Systems, 2001, 179 — 184

Wiering, M. et al., (2004). Simulation and Optimization of Traffic in a City, Intelligent Vehicles Symposium, IEEE 2004, 453—458

Witham, G. B., (1974). Linear and Nonlinear Waves, Wiley, New York

Yang, Q., (1997). A Simulation Laboratory for Evaluation of Dynamic Traffic Management Systems, PhD Thesis, Massachusetts Institute of Technology

You-Sik, H. et al., (1999). New Electrosensitive Traffic Light Using Fuzzy Neural Network, IEEE Transactions on Fuzzy Systems, VII, 6, 759— 767

You Sik Hong, et al., (2001). Estimation of optimal green time simulation using fuzzy neural network, Fuzzy Systems Conference Proceedings, 1999. FUZZ-IEEE '99, 761 — 766

Evolutionary Algorithms in Decision Tree Induction

Francesco Mola[1], Raffaele Miele[2] and Claudio Conversano[1]
[1]University of Cagliari,
[2]University of Naples Federico II
Italy

1. Introduction

One of the biggest problem that many data analysis techniques have to deal with nowadays is Combinatorial Optimization that, in the past, has led many methods to be taken apart. Actually, the (still not enough!) higher computing power available makes it possible to apply such techniques within certain bounds. Since other research fields like Artificial Intelligence have been (and still are) dealing with such problems, their contribute to statistics has been very significant.
This chapter tries to cast the Combinatorial Optimization methods into the Artificial Intelligence framework, particularly with respect Decision Tree Induction, which is considered a powerful instrument for the knowledge extraction and the decision making support. When the exhaustive enumeration and evaluation of all the possible candidate solution to a Tree-based Induction problem is not computationally affordable, the use of Nature Inspired Optimization Algorithms, which have been proven to be powerful instruments for attacking many combinatorial optimization problems, can be of great help.
In this respect, the attention is focused on three main problems involving Decision Tree Induction by mainly focusing the attention on the Classification and Regression Tree-CART (Breiman et al., 1984) algorithm. First, the problem of splitting complex predictors such a multi-attribute ones is faced through the use of Genetic Algorithms. In addition, the possibility of growing "optimal" exploratory trees is also investigated by making use of Ant Colony Optimization (ACO) algorithm. Finally, the derivation of a subset of decision trees for modelling multi-attribute response on the basis of a data-driven heuristic is also described. The proposed approaches might be useful for knowledge extraction from large databases as well as for data mining applications. The solution they offer for complicated data modelling and data analysis problems might be considered for a possible implementation in a Decision Support System (DSS).
The remainder of the chapter is as follows. Section 2 describes the main features and the recent developments of Decision Tree Induction. An overview of Combinatorial Optimization with a particular focus on Genetic Algorithms and Ant Colony Optimization is presented in section 3. The use of these two algorithms within the Decision Tree Induction Framework is described in section 4, together with the description of the algorithm for modelling multi-attribute response. Section 5 summarizes the results of the proposed

method on real and simulated datasets. Concluding remarks are presented in section 6. The chapter also includes an appendix that presents J-Fast, a Java-based software for Decision Tree that currently implements Genetic Algorithms and Ant Colony Optimization.

2. Decision tree induction

Decision Tree Induction (DTI) is a tool to induce a classification or regression model from (usually large) datasets characterized by *N* observations (records), each one containing a set **x** of numerical or nominal variables, and a variable *y*. Statisticians use the terms "splitting predictors" to identify **x** and "response variable" for *y*. DTI builds a model that summarizes the underlying relationships between **x** and y. Actually, two kinds of model can be estimated using decision trees: classification trees if *y* is nominal, and regression trees if *y* is numerical. Hereinafter we refer to classification trees to show the main features of DTI and briefly recall the main characteristics of regression trees at the end of the section.
DTI proceeds by inducing a series of follow-up (usually binary) questions about the attributes of an unknown observation until a conclusion about what is its most likely class label is reached. Questions and their alternative answers can be represented hierarchically in the form of a decision tree. It contains a root node and some internal and terminal nodes. The root node and the internal ones are used to partition observations of the dataset into smaller subsets of relatively homogeneous classes. To classify a previously unlabelled observation, say *i* (i*=1,…..,N)*, we start from the test condition in the root node and follow the appropriate pattern based on the outcome of the test. When an internal node is reached a new test condition is applied, and so on down to a terminal node. Encountering a terminal node, the modal class of the observations in that node is the class label of *y* assigned to the (previously) unlabeled observation. For regression trees, the assigned class is the mean of *y* for the observations belonging to that terminal node.
Because of their top-down binary splitting approach, decision trees can easily be converted into IF-THEN rules and used for decision making purposes.
DTI is useful for knowledge extraction from large databases and data mining applications because of the possibility to represent functions of numerical and nominal variables as well as of its feasibility, predictive ability and interpretability. It can effectively handle missing values and noisy data and can be used either as an explanatory tool for distinguishing observations of different classes or as a prediction tool to class labels of previously unseen observations.
Some of the well-known DTI algorithms include ID3 (Quinlan, 1983), CART (Breiman et al., 1984), C4.5 (Quinlan, 1993), SLIQ (Metha et al., 1996), FAST (Mola & Siciliano, 1997) and GUIDE (Loh, 2002). All these algorithms use a greedy, top-down recursive partitioning approach. They primarily differ in terms of the splitting criteria, the type of splits (2-way or multi-way) and the handling of the overfitting problem.
DTI uses a greedy, top-down recursive partitioning approach to induce a decision tree from data. In general, DTI involves the following tasks: decision tree growing and decision tree pruning.

2.1 Tree growing

As for the growing of a decision tree, DTI use a greedy heuristic to make a series of locally optimum decisions about which value of a splitting predictor to use for data partitioning. A test condition depending on a splitting method is applied to partition the data into more homogeneous subgroups at each step of the greedy algorithm.

Splitting methods differ with respect to the type of splitting predictor: for nominal splitting predictors the test condition is expressed as a question about one or more of its attributes, whose outcomes are "Yes"/"No". Grouping of splitting predictor attributes is required for algorithms using 2-way splits. For ordinal or continuous splitting predictors the test condition is expressed on the basis of a threshold value υ such as *($x_i \le \upsilon$?)* or *($x_i > \upsilon$?)*. By considering all the possible split points υ, the best one υ^* partitioning the instances into homogeneous subgroups is selected.

In the classification problem, the sample population consists of *N* observations deriving from *C* response classes. A decision tree (or classifier) will break these observations into *k* terminal groups, and to each of these a predicted class (being one of the possible attributes of the response variable) is assigned. In actual application, most parameters are estimated from the data. In fact, denoting with *t* some node of the tree (*t* represents both a set of individuals in the sample data and, via the tree that produced it, a classification rule for future data) from the binary tree it is possible to estimate *P(t)* and *P(i | t)* for future observations as follows:

$$P(t) = \sum_{i=1}^{C} \pi_i P\left\{x \in t \middle| \tau(x) = i\right\} \approx \sum_{i=1}^{C} \pi_i \left(n_{iA}/n_i\right) \tag{1}$$

$$P(i|t) = P\left\{\tau(x) = i \middle| x \in t\right\} = \pi_i P\left\{x \in t \middle| \tau(x) = i\right\} \Big/ P\left\{x \in t\right\} \approx \pi_i \left(n_{it}/n_i\right) \Big/ \sum_{i=1}^{C} \pi_i \left(n_{it}/n_i\right) \tag{2}$$

where π_i is the prior probability of each class *i* ($i \in 1,2,\ldots,C$), $\tau(\mathbf{x})$ is the true class of an observation x_i (**x** is the vector of predictor variables), n_i and n_t are the number of observations in the sample that respectively are class *i* and node *t*, and n_{it} is the number of observations in the sample that are class *i* and node *t*.

In addition, by denoting with *R* the risk of misclassification, the risk of *t* (denoted with *R(t)*) and the risk of a model (or tree) *T* (denoted with *R(T)*) are measured as follows:

$$R(t) = \sum_{i=1}^{C} P(i|t) L(i, \tau(t)) \tag{3}$$

$$R(T) = \sum_{j=1}^{k} P(t_j) R(t_j) \tag{4}$$

where *L(i,j)* is the loss matrix for incorrectly classifying an *i* as a *j* (with *L(i,i)=0*), and $\tau(t)$ is the class assigned to *t* once that *t* is a terminal node and $\tau(t)$ is chosen to minimize *R(t)* and t_j are terminal nodes of the tree *T*. If *L(i,i)=1* for all $i \neq j$, and the prior probabilities τ are set to be equal to the observed class frequencies in the sample, then $P(i|t)=n_{it}/n_t$ and *R(T)* is the proportion of misclassified observations.

When splitting a node *t* into t_r and t_l (left and right sons), the following relationship holds: $P(t_l)\ R(t_l) + P(t_r)\ R(t_r) \le P(t)\ R(t)$. An obvious way to build a tree is to chose that split maximizing ΔR, i.e., the decrease in risk. To this aim, several measures of impurity (or diversity) of a node are used. Denoting with *f* some impurity function, the local impurity of a node *t* is defined as:

$$\varepsilon(t)=\sum_{i=1}^{C} f\left(p_{it}\right) \tag{5}$$

where p_{it} is the proportion of those in t that belong to class i for future samples. Since $\varepsilon(t)=0$ when t is pure, f must be concave with $f(0)=f(1)=0$. Two candidates for f are the information index $f(p) = -p \log(p)$ and the Gini index $f(p)= -p(1-p)$, that slightly differ for the two class problem where nearly always choose the same split point. Once that f has been chosen, the split maximizing the impurity reduction is:

$$\Delta\varepsilon = p(t)\varepsilon(t)-p(t_l)\varepsilon(t_l)-p(t_r)\varepsilon(t_r) \tag{6}$$

Data partitioning proceeds recursively until a stopping rule is satisfied: this usually happens when the number of observations in a node is lower than a previously-specified minimum number necessary for splitting, as well as when the same observations belong to the same class or have the same response class.

2.2 FAST splitting algorithm

The goodness of split criterion based on (6) expresses in different way some equivalent criteria which are present in most of the tree-growing procedures implemented in specialized software; such as, for instance, CART (Breiman et al., 1984), ID3 and C4.5 (Quinlan, 1993).
In many situations the computational time required by a recursive partitioning algorithm is an important issue that can not be neglected. In this respect, a fast algorithm is required to speed up the procedure. In view of that, it is worth considering a two-stage splitting criterion which takes into account of the global role played by a splitting predictor in the partitioning step. A global impurity reduction factor of any predictor x_i is defined as:

$$\mathrm{E}_{y|x_s}(t)=\sum_{g\in G_s}\varepsilon_{y|g}(t)\,p(g\,|\,t) \tag{7}$$

where $\varepsilon_{y|g}(t)$ is the impurity of the conditional distribution of y given the s-th attribute of x_s and G is the number of attributes of x_s $(g \,\varepsilon\, G)$. The two-stage criterion finds the best splitting predictor(s) as the one (or those) minimizing (7) and, consequently, the best split point among the candidate splits induced by the best predictor(s) minimizing the (6) by taking account only the partitions or splits generated by the best predictor. This criterion can be applied either *sic et simpliciter* or by considering alternative modelling strategies in the predictor selection (an overview of the two-stage methodology can be found in Siciliano & Mola, 2000).
The FAST splitting algorithm (Mola & Siciliano, 1997) can be applied when the following property holds for the impurity measure:

$$\mathrm{E}_{y|x_s}(t)\le \mathrm{E}_{y|h}(t) \qquad \forall\ h\neq g;\ h\in G \tag{8}$$

and it consists of two basic rules:

- iterate the two-stage partitioning criterion by using (7) and (6): select one splitting predictor at a time and consider, at each time, the previously unselected splitting predictors;

- stop the iterations when the current best predictor in the order *x(k)* at iteration *k* does not satisfy the condition $\mathrm{E}_{y|x_k}(t) \le \mathrm{E}_{y|h^*_{(k-1)}}(t)$, where $s^*_{(k-1)}$ is the best partition at the iteration *(k − 1)*.

The algorithm finds the optimal split with substantial time savings in terms of the reduced number of partitions or splits to be tried out at each node of the tree. Simulation studies show that the relative reduction in the average number of splits analyzed by the FAST algorithm with respect to the standard approaches in binary trees increases as a function of both the number of attributes of the splitting predictor and of the number of observations at a given node. Further theoretical results about the computational efficiency of FAST-like algorithms can be found in Klaschka et al. (1998).

2.3 Tree pruning

As for the pruning step, it is usually required in DTI in order to control for the size of the induced model and to avoid in this way data overfitting. Typically, data is partitioned into a training set (containing two-third of the data) and a test set (with the remaining one-third). Training set contains labelled observations and it is used for the tree growing. It is assumed that the test set contains unlabelled observations and it is used for selecting the final decision tree: to check whether a decision tree, say *T*, is generalizable, it is necessary to evaluate its performance on the test set in terms of misclassification error by comparing the true class labels of the test data against those predicted by *T*. Reduced-size trees perform poorly on both training and test sets causing underfitting. Instead, increasing the size of *T* improves both the training and test errors up to a "critical size" from which the test errors increase even though the corresponding training errors decrease. This means that *T* overfits the data and cannot be generalized to class prediction of unseen observations. In the machine learning framework, the training error is named resubstitution error and the test error is known as the generalization error.

It is possible to prevent overfitting by haltering the tree growing before it becomes too complex (pre-pruning). In this framework, one can assume the training data is a good representation of the overall data and use the resubstitution error as an optimistic estimate of the error of the final DTI model (optimistic approach). Alternatively, Quinlan (1987) proposed a pessimistic approach that penalizes complicated models by assigning a cost penalty to each terminal node of the decision tree: for C4.5, the generalization error is $R(t)/n_t+\varepsilon$, where, for a node *t*, n_t is the number of observations and *R(t)* is the misclassification error. It is assumed that *R(t)* follows a Binomial distribution and that ε is the upper bound for *R(t)* computed from such a distribution (Quinlan, 1993).

An alternative pruning strategy is based on the growing of the entire tree and the subsequent retrospective trimming of some of its internal nodes (post-pruning): the subtree departing from each internal node is replaced with a new terminal node whose class label derives from the majority class of observations belonging to that subtree. The latter is definitively replaced by the terminal node if such a replacement induces an improvement of the generalization error. Pruning stops when no further improvements can be achieved. The generalization error can be estimated through either the optimistic or pessimistic approaches.

Other post-pruning algorithms, such as CART, use a complexity measure that accounts for both the tree size and the generalization error. Once the entire tree is grown using training observations, a penalty parameter expressing the gain/cost trade off for trimming each subtree is used to generate a sequence of pruned trees, and the tree in the sequence presenting the lowest generalization error (0-SE rule) or the one with a generalization error within one standard error of its minimum (1-SE rule) is selected. Let α be a number in $[0,+\infty]$, called complexity parameter, measuring the "cost" of adding another variable to the model. Let $R(T_0)$ be the risk for the zero split tree. Define:

$$R_\alpha(T) = R(T) + \alpha|T| \tag{9}$$

to be the cost for the tree, and define T_α to be that subtree of the entire tree having the minimal cost. Obviously, T_0 is the entire tree and T_∞ is the zero splits model. The idea is to find, for each α, the subtree $T_\alpha \subseteq T_0$ minimizing $R_\alpha(T)$. The tuning parameter $\alpha \geq 0$ governs the trade off between the tree size and its goodness of fit to the data. Large values of α result in small trees, and conversely for smaller values of α. Of course, with $\alpha=0$ the solution is the full tree T_0. It is worth noticing that, by adaptively choosing αI, it exists a unique smallest subtree T_α minimizing $R_\alpha(T)$. A weakest link pruning approach is used to find T_α: it consists in successively collapsing the internal node producing the smallest per-node increase in $R(T)$, continuing this way until the single-node (root) tree is produced. This gives a (finite) sequence of subtrees, and it is easy to show that this sequence must contains T_α (see Breiman et al (1984) for details).

Usually, pruning algorithms can be combined with V-fold cross-validation when few observations are available. Training data is divided into V disjoint blocks and a tree is grown V times on V-1 blocks estimating the error by testing the model on the remaining block. In this case, the generalization error is the average error made for the V runs. The estimation of αI is achieved by V-fold cross-validation: the final choice is the $\hat{\alpha}$ minimizing the cross-validated $R(T)$ and the final tree is $T_{\hat{\alpha}}$.

Cappelli et al. (2002) improved this approach introducing a statistical testing pruning to achieve the most reliable decision rule from a sequence of pruned trees.

2.4 Regression tree

In the case the response variable is numeric, the outcome of a recursive partitioning algorithm is regression tree. Here, the splitting criterion is SS_t- $(SS_l - SS_r)$, where SS_t is the residual sum of squares for the parent node, and SS_l and SS_r are the residual sum of squares for the left and right son, respectively. This is equivalent to choosing the splits maximizing the between-groups sum-of-squares in a simple analysis of variance. In each terminal node, the mean value of the response variable μ_y of cases belonging to that node is considered as the fitted value whereas the variance is considered as an indicator of the error of a node. For a new observation y_{new} the prediction error is $(y_{new} - \mu_y)$. In the regression tree case, cost-complexity pruning is applied with the sum of squares replacing the misclassification error.

2.5 DTI enhancements

A consolidated literature about the incorporation of parametric and nonparametric models into trees appeared in recent years. Several algorithms have been introduced as hybrid or functional trees (Gama, 2004), among the machine learning community. As an example, DTI is used for regression smoothing purposes in Conversano (2002): a novel class of semiparametric models named Generalized Additive Multi-Mixture Models (GAM-MM). Other hybrid approaches are presented in Chan and Loh (2004), Su et al. (2004), Choi et al. (2005) and Hothorn et al. (2006). Nevertheless, relatively simple procedures combining DTI models in different ways have been proposed in the last decade in the statistics and machine learning literature and their effectiveness in improving the predictive ability of the traditional DTI method has been proven in different fields of application.

The first, rather intuitive, approach is Tree Averaging. It is based on the generation of a set of candidate trees and on their subsequent aggregation in order to improve their generalization ability. It requires the definition of a suitable set of trees and their associated weights and classifies a new observation by averaging over the set of weighted trees (Oliver and Hand, 1995). Either a compromise rule or a consensus rule can be used for averaging.

An alternative method consists in summarizing the information of each tree in a table cross-classifying terminal nodes outcomes with the response classes in order to assess the generalization ability through a statistical index and select the tree providing the maximum value of such index (Siciliano, 1998).

Tree Averaging is very similar to Ensemble methods. These are based on a weighted or non weighted aggregation of single trees (the so called weak learners) in order to improve the overall generalization error induced by each single tree. They are more accurate than a single tree if they have a generalization error that is lower than random guessing and if the generalization errors of the different trees are uncorrelated (Dietterich, 2000).

A first example of Ensemble method is Bootstrap Aggregating, which is also called Bagging (Breiman, 1996). It works by randomly replicating the training observations in order to induce single trees whose aggregation by majority voting provides the final classification. Bagging is able to improve the performance of unstable classifiers (i.e. trees with high variance). Thus, bagging is said to be a reduction variance method.

Adaptive Boosting, also called AdaBoost (Freud & Schapire, 1996) is an Ensemble method that uses iteratively bootstrap replication of the training instances. At each iteration, previously-misclassified observations receive higher probability of being sampled. The final classification is obtained by majority voting. Boosting forces the decision tree to learn by its error, and is able to improve the performance of trees with both high bias (such as single-split trees) and variance.

Finally, Random Forest (Breiman, 2001) is an ensemble of unpruned trees obtained by randomly resampling training observations and variables. The overall performance of the method derives from averaging the generalization errors obtained in each run. Simultaneously, suitable measures of variables importance are obtained to enrich the interpretation of the model.

3. Combinatorial optimization

Combinatorial Optimization can be defined as the analysis and solution of problems that can be mathematically modelled as the minimization (or maximization) of an objective function over a feasible space involving mutually exclusive, logical constraints. Such logical constraints can be seen as the arrangement of a bunch of given elements into sets. In a mathematical form:

$$\min_{T \in F} \{\alpha(T)\} \quad \text{or} \quad \max_{T \in F} \{\alpha(T)\} \tag{10}$$

where *T* can be seen as an arrangement, *F* is the collection of feasible arrangements and $\alpha(T)$ measures the value of the members of *F*.
Combinatorial Optimization problems are of great interest because many real life decision-making situations force people to choose over a set of possible alternatives with the aim of maximizing some utility function. On the one hand, the discreteness of the solutions space offers the great advantage of concreteness and, indeed, elementary graphs or similar illustrations can often naturally be used to represent the meaning of a particular solution to a problem. On the other end, those problems carry a heavy burden in terms of dimensionality. If more than few choices are to be made, the decision-making process has to face with the evaluation of a terribly big expanse of cases. This dualism (intuitive simplicity of presentation of a solution versus complexity of solutions search) has made this area of combinatorics attractive for researchers from many fields, ranging from engineering to management sciences.
Elegant procedures to find optimal solutions have been found for some problems, but for most of them only a bunch of properties and algorithms have been developed that still do not allow to reach a complete resolution. This is the case of Computational Statistics, in which computationally-intensive methods are used to "mine" large, heterogeneous, multi-dimensional datasets in order to discover knowledge in the data.
To give an example, the objective of Cluster Analysis is to find the "best" partition of the dataset according to some criterion, which is always expressed as an objective function. This means that all possible and coherent partitions of the dataset should be generated and the objective function has to be calculated for each of them. In many cases, the number of possible partitions grows too rapidly with respect to the number of units, making such strategy practically unfeasible. Another example is the apparently simple problem of calculating the variance for interval data, for which the maximum and the minimum of the variance function have to be searched over the multidimensional cube defined by all the intervals in which the statistical units are defined.
These are examples of statistical problems that cannot be faced with the total enumeration and evaluation of the solutions. In order to try to tackle with this kind of problems, a lot of theory has been developed. One case is when some properties about the objective function are available. These allow to calculate some kind of upper (or lower) bound that a set of possible solutions could admit. In this case, the search could be performed just on the set of possible solutions whose upper bound is higher. If one solution whose effective value is higher than the bounds of all the other sets is found, it would not be necessary to continue the search, being all the other subsets not able to provide better solutions. This is the case of the aforementioned problem of finding the upper bound of variance for interval data, because it can be verified that the maximum is necessarily reached in one of the vertices of the multidimensional cube, so that exploring the whole cube is not necessary. Such a situation allows to restrict the solutions space to a set of 2^n possible solutions, where *n* is the number of statistical units. Unfortunately, this does not solve the problem because the solutions space becomes enormous even in presence of small datasets (with just 30 units the number of solutions to evaluate is greater than one thousand millions).
The FAST algorithm is another example of a partial enumeration approach, in which a measure of the upper bound of the predictive power of a solutions set is defined and exploited in order to get the same results of the CART greedy approach by using a reduced amount of computations.

Another way to proceed is to make use of non exact procedures, often called heuristics. Those algorithms do not claim to find the global optimum, but are able to converge rapidly towards a local one. Non exact algorithms (that will be called heuristics in the rest of this chapter) are certainly not recent. What has changed, in time, is the respectability associated to them, due to the fact that many heuristics have been proved to rival their counterparts in elegance, sophistication and, particularly, usefulness. Many heuristics have been proposed in the literature, but only two kinds of them will be briefly described in this context due to their role in the problems that will be faced in the next sections. These are: Greedy procedures and Nature Inspired optimization algorithms. In Greedy procedures the optimization process selects, at each stage, an alternative that is the best among all the feasible alternatives without taking into account the impact that such choice will have on the subsequent decisions. The CART algorithm makes use of a greedy procedure to grow a tree in which the optimality criterion is maximised just locally, that is, for each node of the tree but not considering the tree as a whole. This approach clearly results in a suboptimal tree but allows, at least, to obtain a tree in a reasonable amount of time. Whereas, the so-called Nature Inspired heuristics, which are also called "Heuristics from Nature" (Colorni et al., 1993), are Inspired by natural phenomena or behaviour such as Evolution, Ants, Honey-Bees, Immune systems, Forests, etc. Some important Nature Inspired heuristics are: Simulated Annealing (SA), TABU Search (TS) algorithms, Ant Colony Optimization (ACO) and Evolutionary Computation (EC). ACO and EC are described in the following since they are used throughout the chapter.

Ant Colony Optimization represents a class of algorithms that were inspired by the observation of real ant colonies. Observation shows that a single ant only applies simple rules, has no knowledge and it is unable to succeed in anything when it is alone. However, an ant colony benefits from the coordinated interaction of each ant. Its structured behaviour, described as a "social life", leads to a cooperation of independent searches with high probability of success. ACO were initially proposed by Dorigo (1992) to attack the Traveling Salesman Problem. A real ant colony is capable of finding the shortest path from a food source to its nest by using pheromone information: when walking, each ant deposits a chemical substance called pheromone and follows, in probability, a pheromone trail already deposited by previous ants. Assuming that each ant has the same speed, the path which ends up with the maximum quantity of pheromone is the shortest one.

Evolutionary computation (Fogel and Fogel, 1993) incorporates algorithms that are inspired from evolution principles in nature. The methods of evolutionary computation algorithms are stochastic and their search methods imitate and model some natural phenomena, namely:

1. the survival of the fittest
2. genetic inheritance

Evolutionary computing can be applied to problems when it is difficult to apply traditional methods (e.g., when gradients are not available) or when traditional methods lead to unsatisfactory solutions like local optima (Fogel, 1997). Evolutionary algorithms work with a population of potential solutions (i.e. individuals). Each individual is a potential solution to the problem under consideration and it is encoded into a data structure suitable to the problem. Each encoded solution is evaluated by an objective function (environment) in order to measure its fitness. The bias on selecting high-fitness individuals exploits the acquired fitness information. The individuals will change and evolve to form a new population by applying genetic operators. Genetic operators perturb those individuals in order to explore the search space. There are two main types of genetic operators: Mutation and Crossover. Mutation type operators are asexual (unary) operators, which create new individuals by a small change in a single individual. On the other hand, Crossover type operators are multi-sexual (multary) operators, which create new individuals by combining parts from two or more individuals. As soon as a number of generations have evolved, the process is terminated according to a termination criterion. The best individual in the final step of the process is then proposed as a (hopefully suboptimal or optimal) solution for the problem.

Evolutionary computing are further classified into four groups: Genetic Algorithms (GA), Evolutionary Programming, Evolution Strategies and Genetic Programming. Although there are many relevant similarities between these evolutionary computing paradigms, profound differences among them also emerge (Michalewicz, 1996). These differences generally involve the level in the hierarchy of the evolution being modelled, that is: the chromosome, the individual or the species. There are also many hybrid methods that combine various features from two or more of the methods described in this section.

Genetic Algorithms (GAs), that will be used in the follwing, are part of a collection of stochastic optimization algorithms inspired by the natural genetics and the theory of the biological evolution. The idea behind genetic algorithms is to simulate the natural evolution when optimizing a particular objective function. GAs have emerged as practical, robust optimization and search methods in the last three decades. In the literature, Hollands' genetic algorithm is called Simple Genetic Algorithm (Vose, 1999). It works with a population of individuals (chromosomes), which are encoded as binary strings (genes).

4. Genetic algorithms and heuristics in DTI

4.1 Genetic algorithm for complex predictors

The CART methodology looks for the best split by making use of a brute-force (enumerative) procedure. All the possible splits from all the possible variables are generated and evaluated. Such a procedure must be performed anytime a node has to be split and can lead to computational problems when the number of modalities grows.

Let us first consider how a segmentation procedure generates and evaluates all possible splits. Nominal unordered predictors (Nup) are more complicated to handle than ordered ones because the number of possible splits that can be generated grows exponentially with the number of attributes *m*. The number of possible splits is $(2^{m-1}-1)$. The computational complexity of a procedure that generates and evaluates all the splits from a nominal unordered predictor is $O(2^n)$. In this respect, it is evident that such enumerative algorithm becomes prohibitive when the number of attributes is high. This is one of the reasons why some software do not accept Nups with a number of attributes higher than a certain threshold (usually between 12 and 15).

One of the possible way to proceed is to make use of a heuristic procedure, like the one proposed in this section. In order to design a Genetic Algorithm to solve such a combinatorial problem, it is necessary to identify:

- a meaningful representation (coding) for the candidate solutions (the possible splits)
- a way to generate the initial population
- a fitness function to evaluate any candidate solution

- a set of useful genetic operators that can efficiently recombine and mutate the candidate solutions
- the values of the parameters used by the GA (population size, genetic operators parameters values, selective pressure, etc.);
- a stopping rule for the algorithm.

The aforementioned points have been tackled as follows. As for the coding, it has been chosen the following representation: a solution is coded in a string of bits (chromosomes) called x, where each bit (gene) is associated to an attribute of the predictor according to the following rule:

$$x_i = \begin{cases} 0 & \textit{if} \quad i \quad \textit{goes} \quad \textit{to} \quad \textit{left} \\ 1 & \textit{if} \quad i \quad \textit{goes} \quad \textit{to} \quad \textit{right} \end{cases} \tag{11}$$

The choice of the fitness function is straightforward: the split evaluation function of the standard recursive partitioning algorithm is used (i.e. the maximum decrease in node impurity). Since the canonical (binary) coding is chosen, the corresponding two parents single-point crossover and mutation operators and, as a stopping rule can be used. In addition, a maximum number of iterations is chosen on the basis of empirical investigations. The rest of the GA features are similar to the classic ones: elitism is used (at each iteration the best solution is kept in memory) and the initial population is chosen randomly.

4.2 An ACO algorithm for exploratory DTI

When growing a Classification or a Regression Tree, CART first grows the so-called exploratory tree. Such tree is grown using data of the training set. Then, it is validated by using the test set or by cross-validation.

In this section, the attention is focused on the exploratory tree-growing procedure. In this phase, in theory, the best possible tree should be built, which is the tree having the lowest global impurity measure among all the generable trees. It has been shown (Hyafil and Rivest, 1976) that constructing the optimal tree is a NP-Complete problem. In other words, in order to use a polynomial algorithm, it is only possible to get suboptimal trees. For such a reason, the recursive partitioning algorithms make use of greedy heuristics to reach a compromise between the tree quality and the computational effort. In particular, most of the existing methods for DTI use a greedy heuristic, which is based on a top-down recursive partitioning approach in which, any time, the split that maximizes the one step impurity decrease is chosen. This kind of greedy approach, that splits the data locally (i.e., in a given node) and only once for each node, allows to grow a tree in a reasonable amount of time. On the other hand, this rule is able to generate only a suboptimal tree because anytime a split is chosen a certain subspace of possible trees is not investigated anymore by the algorithm. If the optimal tree is included in one of those subspaces there is no chance for the algorithm of finding it.

Taking these considerations into account, we propose an Ant Colony Optimization algorithm to try to find best exploratory tree. In order to attack a problem with ACO the following design task must be performed:

1. Represent the problem in the form of sets of components and transitions or by means of a weighted graph, on which ants build solutions

2. Appropriately define the meaning of the pheromone trails: that is, the type of decision they bias.
3. Appropriately define the heuristic reference for each decision an ant has to take while constructing a solution.
4. If possible, implement an efficient local search algorithm for the problem to be solved. The best results from the application of the ACO algorithms to NP-hard combinatorial optimization problems are achieved by coupling ACO with local optimizers (Dorigo and Stutzle, 2004)
5. Choose a specific ACO algorithm and apply it to the problem to be solved, taking the previous issues into account
6. Tune the parameters of the ACO algorithm. A good starting point is to use parameter settings that were found to be good when applying the same ACO algorithm to similar problems or to a variety of other problems

The most complex task is probably the first one, in which a way to represent the problem in the form of a weighted graph must be found. We use a representation based on the following idea: let us imagine having two nominal predictors $P_1 = \{a_1, b_1, c_1\}$ and $P_2 = \{a_2, b_2\}$ with, respectively, two and three attributes. Such simple predictors are considered only to explain the idea, because of the combinatorial explosion of the phenomenon. In this case, the set of all possible splits, at a root node, is the following:

- $S_1 = [a_1] - [b_1, c_1]$
- $S_2 = [a_1, b_1] - [c_1]$
- $S_3 = [a_1, c_1] - [b_1]$
- $S_4 = [a_2] - [b_2]$

Any time a split is chosen, it generates two child nodes. For such nodes, the set of possible splits is, in the worst case, equal to 3 (the same as the parent node except the one that was chosen for splitting). This consideration leads to the representation shown in Figure 1 in which, for simplicity, only the first two levels of the possible trees are considered.

It is easy to imagine how the complexity grows when we deal with predictors that generate hundreds or even thousands of splits (which is a common case).

In Figure 1, the space of all possible trees is represented by a connected graph. Moving from a level to another one corresponds to split a variable. The arcs of such a graph have the same meaning of the arcs of the TSP graph (transition from a state to another one or, even better, addition of a component to a partial solution). In this view, it would be correct to deposit pheromone on them. The pheromone trails meaning, in this case, corresponds to the desirability to choose the corresponding split from a certain node.

As for the heuristic information, it is possible to refer to the decrease in impurity deriving from adding the corresponding node to the tree. Such a measure has a meaning which is similar, in some way, to the one that visibility has in the TSP . An arc is much more desirable as higher the impurity decrease is. As a result, to make analogies with the TSP, such impurity decrease can be seen as an inverse measure of the distance between two nodes.

Once the construction graph has been built, and pheromone trails meaning and heuristic function have been defined, it is possible to attack that problem using an ACO algorithm. It is important to note that, because of the specificity of the problem to be modelled (ants can move into a connected graph and there is a measure of "visibility"), the search of the best tree can be seen as a shortest path research, like in TSP. In the latter, ants are forced to pass only one time for each city while, in our case, ants are forced to choose paths that correspond to binary trees, since the solutions to build must be in the form of tree structures. All the ants will start from the root node and will be forced to move from one node to another in order to build a tour that corresponds to a tree.

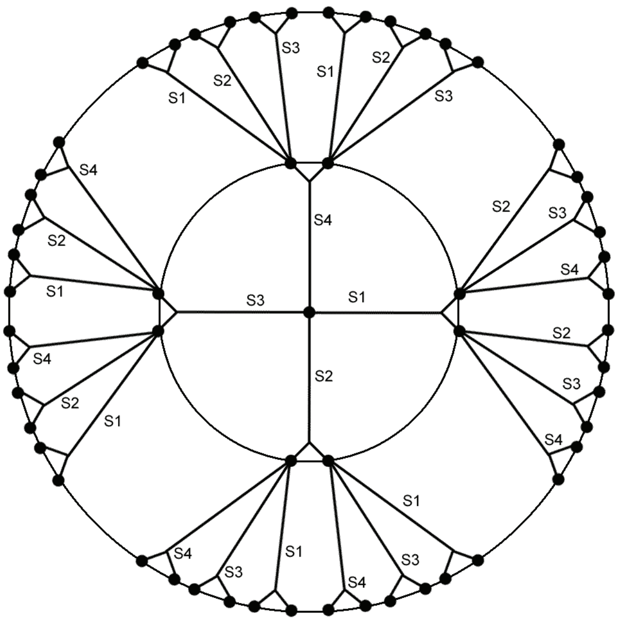


Fig. 1. An example of ACO algorithm for exploratory DTI: each path corresponds to a 2-levels tree.

It is important to notice the basics of the ant moves in the graph shown in Figure 1. At each step, the ant looks for the heuristic information (impurity decrease) and the pheromone trail of any possible direction and decides for the one to choose (and, therefore, the associated split) on the basis of the selected ACO algorithm. Once the ant arrives to a terminal node, it recursively starts to move back to the other unexplored nodes.
In different ACO algorithms, pheromone trails are initialized to a value obtained by manipulating the quality measure (the path's length for the TSP case) of a solution obtained with another heuristic (Dorigo suggests the nearest-neighbour heuristic). In our case, the greedy tree induction rule solution quality is used. Elitism will also be implemented and the chosen parameters (due to the strong similarity with TSP) are the same that have been used successfully for the TSP problem.

4.3 Identification of a parsimonious set of decision trees in multi-class classification

In many situations, the response variable used in classification tree modelling rarely presents a number of attributes that allow to apply the recursive partitioning algorithm in the most accurate manner.
It is well known that:

a) a multi-class response, namely a nominal variables with several classes, usually causes prediction inaccuracy;

b) multi-class and numeric predictors play often the role of splitting variables in the tree growing process in disadvantage of two-classes ones, causing selection bias.

To account for the problems deriving from the prediction inaccuracy of tree-based classifiers grown for multi-class response, as well as to reduce the drawback of the loss of interpretability induced by ensemble methods in these situations, Mola and Conversano (2008) introduced an algorithm based on a *Sequential Automatic Search of a Subset of Classifiers* (SASSC). It produces a partition of the set of the response classes into a reduced number of disjoint subgroups and introduces a parameter in the final classification model that improves its prediction accuracy, since it allows to assign each new observation to the most appropriate classifier in a previously-identified reduced set of classifiers. It uses a data-driven heuristic based on cross-validated classification trees as a tool to induce the set of classifiers in the final classification model.

SASSC produces a partition of the set of the response classes into a reduced number of *super-classes*. It is applicable to a dataset **X** composed of N observations characterized by a set of J (numeric or nominal) splitting variables x_j $(j=1,\ldots,J)$ and a response variable y presenting K classes. Such response classes identify the initial set of classes $C^{(0)} = (c_1, c_2, \ldots, c_K)$. Partitioning **X** with respect to $C^{(0)}$ allows to identify K disjoint subsets $\mathbf{X}^{(0)}_k$, such that: $\mathbf{X}^{(0)}_k = \{\mathbf{x}_s : y_s \in c_k\}$, with $s=1,\ldots,N$. In practice, $\mathbf{X}^{(0)}_k$ is the set of observations presenting the k-th class of y. The algorithms works by aggregating the K classes in pairs and learns a classifier to each subset of corresponding observations. The "best" aggregation (*super-class*) is chosen as the one minimizing the generalization error estimated using V-fold cross-validation.

Suppose that, in the ℓ-th iteration of the algorithm such a best aggregation is found for the pair of classes c_{i^*} and c_{j^*} (with $i^* \neq j$ and $i^*, j^* \in (1,\ldots,K)$) that allows to aggregate the subsets $\mathbf{X}_{i^*}$ and $\mathbf{X}_{j^*}$. Denoting with $T_{(i^*,j^*)}$ the decision tree minimizing the cross-validated generalization error $\delta^{()}_{cv}$, the heuristic for selecting the "best" decision tree can be formalized as follows:

$$(i^*, j^*) = \underset{(i,j)}{\arg\min} \left\{ \delta^{(\ell)}_{cv} \left(T_{(i,j)} \mid \mathbf{X}_i \cap \mathbf{X}_j \right) \right\} \tag{12}$$

The SACCS algorithm is analytically described in Table 1. It proceeds by learning all the possible decision trees obtainable by joining in pairs the K subgroups, and by retaining the one satisfying the selection criteria introduced in (12). After the ℓ-th aggregation, the number of subgroups is reduced to $K^{(-1)} - 1$, since the subgroups of observations presenting the response classes c_{i^*} and c_{j^*} are discarded from the original partition and replaced by the subset $\mathbf{X}^{()}_{(i^*,j^*)} = \mathbf{X}_{(i^*)} \cap \mathbf{X}_{(j^*)}$ identified by the super-class $c^{()} = (c_{(i^*)} \cap c_{(j^*)})$. The initial set of classes C is replaced by $C^{()}$, the latter being composed of a reduced number of classes since some of the original classes form the superclasses coming out from the ℓ aggregations. Likewise, also $\mathbf{X}^{()}_k$ is formed by a lower number of subsets as a consequence of the ℓ aggregations.

The algorithm proceeds sequentially in the iteration $\ell+1$ by searching for the most accurate decision tree over all the possible ones obtainable by joining in pairs the $K^{()}$ subgroups. The sequential search is repeated until the number of subgroups reduces to one in the K-th iteration. The decision tree learned on the last subgroup corresponds to the one obtainable applying the recursive partitioning algorithm on the original dataset.

The output of the procedure is a sequence of sets of response classes $C^{(1)},\ldots,C^{(K-1)}$ with the associated sets of decision trees $T_{(1)},\ldots,T^{(K-1)}$. The latter are derived by learning $K-k$ trees $(k = 1, \ldots, K-1)$ on disjoint subgroups of observations whose response classes complete the initial set of classes $C^{(0)}$: these response classes identify the super-classes relating to the sets of classifiers $T_{(k)}$. An overall generalization error is associated to each $T_{(k)}$: such an error is also based on V-fold cross-validation and it is computed as a weighted average of the generalization errors obtained from each of the $K-k$ decision trees composing the set. In accordance to the previously introduced notation, the overall generalization errors can be denoted as $\Theta^{(1)}_{cv}, \ldots, \Theta^{(k)}_{cv}, \ldots, \Theta^{(K-1)}_{cv}$. Of course, by decreasing the number of trees composing a sequence $T_{(k)}$ (that is, when moving k from 1 to $K-1$) the corresponding $\Theta^{(k)}_{cv}$ increases since the number of super-classes associated to $T_{(k)}$ is also decreasing. This means that a lower number of trees are learned on more heterogeneous subsets of observations, since each of those subsets pertain to a relatively large number of response classes.

```
Input:        $C = \{c_1,\ldots,c_K\}_{c_i \cap c_j = \varnothing; i \neq j; i,j \in (1,\ldots,K)}$
Set:          $C^{(0)} = C;\quad K^{(0)} = K;\quad \mathbf{X}_k^{(0)} = \{\mathbf{x}_s : y_s \in c_k\}_{s=1,\ldots,N; k=1,\ldots,K}$
For: $\ell$ in 1 to $K$
              $c^{(\ell)} = \{c_{i^*} \cap c_{j^*}\} : \theta_{cv}^{(\ell)}\left(T_{(i^*,j^*)} \mid \mathbf{X}_{i^*} \cap \mathbf{X}_{j^*}\right) = \min$
              $K^{(\ell)} = K^{(\ell-1)} - 1$
              $C^{(\ell)} = \left\{c_1,\ldots,c_{K^{(\ell)}-2+1} = c^{(\ell)}\right\}$
              $\mathbf{X}_k^{(\ell)} = \{\mathbf{x}_s : y_s \in c_k\}_{k=1,\ldots,K^{(\ell)}-1}$
end For
Output:       $C^{(1)},\ldots,C^{(K-1)};\quad T_{(1)},\ldots,T_{(K-1)};\quad \Theta_{cv}^{(1)},\ldots,\Theta_{cv}^{(K-1)}$
```

Table 1. The SASSC algorithm

Taking this inverse relationship into account, the analyst can be aware of the overall prediction accuracy of the final model on the basis of the relative increase in $\Theta^{(k)}_{cv}$ when moving from 1 to $K-1$. In this respect, he can select the suitable number of decision trees to be included in the final classification model accordingly. Supposing that a final subset of g decision trees has been selected $(g<<K-1)$, the estimated classification model can be represented as:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{g-1} \sum_{m_i=1}^{M_i} \hat{\psi}_i \hat{c}_{k,i} I\left((x_1,\ldots,x_p) \in R_{m_i}\right) \tag{13}$$

The parameter ψ is called "vehicle parameter". It allows to assign a new observation to the most suitable decision tree in the subset g. It is defined by a set of $g-1$ dummy variables. Each of them equals 1 if the object belongs to the i-th decision tree ($i = 1,\ldots, g-1$) and zero otherwise. The M_i regions, corresponding to the number of terminal nodes of the decision tree *i*, are created by splits on predictors $(x_1,....,x_p)$. The classification tree *i* assigns a new observation to the class $\hat{c}_{k,i}$ of *y* according to the region R_{m_i}. *I* is an indicator function with value *1* if an observation belongs to R_{m_i} and value *0* if not. R_{m_i} is defined by the inputs used in the splits leading to that terminal node. The modal class of the observations in a region R_{m_i} (also called the *m*-th terminal node of the *i*-th decision tree) is usually taken as an estimate for $\hat{c}_{k,i}$. This notation is consistent with that used in Hastie et al. (2001).

The estimation of τ_i is based on the prediction accuracy of each decision tree in the final subset *g*. A new observation is slipped into each of the *g* trees. The assigned class $\hat{c}_{k,i}$ is found with respect to the tree whose terminal node better classifies the new observation. In other words, a new observation is assigned to the purest terminal node among all the *g* decision trees.

Another option of the algorithm is the possibility to learn decision trees to select the suitable pair of response classes satisfying (12) using alternative splitting criteria. As for CART, it is possible to refer to both the Gini index and Twoing as alternative splitting rules. It is known that, unlike Gini rule, Twoing searches for the two classes that make up together more than 50% of the data and allows us to build more balanced trees even if the resulting recursive partitioning algorithm works slower. As an example, if the total number of classes is equal to *K*, Twoing uses 2^{K-1} possible splits. Since it has been proved (Breiman et al., 1984, pag.95) that the decision tree is insensitive to the choice of the splitting rule, it can be interesting to see how it works in a framework characterized by the search of the most accurate decision treers like the one introduced in SASSC.

5. Application on real and simulated datasets

Genetic Algorithm. The proposed GA has been applied on two datasets for which the optimal best split could be calculated and for a more complex one, for which it is not possible to proceed with such a brute force strategy.

The first test has been done on the "Mushroom" dataset, available from the UCI Machine Learning Repository (source http://archive.ics.uci.edu/ml/). This dataset has a two-class response variable ("is the mushroom poisonous?") and set of categorical and numerical predictors. One of them (gill colour) has 12 categories (attributes), which can be evaluated exhaustively. The GA algorithm could find the global best solution (which was extracted by using the Rpart package of the R software) in less than 10 iterations. The algorithm has then been tested on a simulated dataset which was obtained by uniformly generating a response variable with 26 modalities and a nominal unordered predictor with 16 modalities for 20,000 observations. By letting be 16 the number of modalities of the splitting predictor it was possible, also in this case, to find the (global) best split by making use of the exhaustive enumeration. Such experimental studies showed that the most efficient configuration of the GA was the following:

- By randomly selecting the initial population (no other solutions have been tried, in fact).
- By setting the number of solutions building the population to be equal to the number of necessary genes (the number of categories of the predictor).
- By setting a crossover proportion of 0.80.
- By setting a mutation probability equal to 0.10.
- By selecting the rank for choosing the solutions to be recombined.

For this kind of problem (20,000 units, 16 categories for the response variable and 26 categories for the splitting predictor) the global optimum was reached in less than 30 iterations.

When the complexity of the problem grows many iterations seems to be required, though such number never appeared to grow exponentially.

The GA has been tested also on the "Adult" dataset available from the UCI Machine Learning website. This dataset has been extracted from the US Census Bureau Database (source: http://www.census.gov/) with the aim of predicting whether a person earns more than 50,000 dollars per year. Such dataset has 325,614 observations and some categorical unordered splitting predictors with many attributes. In particular, the native-country predictor has 42 attributes.

State	Goes to	State	Goes to
United-States	Left	Cuba	Left
Jamaica	Right	India	Left
Unknown Country	Left	Mexico	Right
South	Left	Puerto-Rico	Right
Honduras	Right	England	Left
Canada	Left	Germany	Left
Iran	Left	Philippines	Left
Italy	Left	Poland	Left
Columbia	Right	Cambodia	Left
Thailand	Left	Ecuador	Right
Laos	Right	Taiwan	Left
Haiti	Right	Portugal	Right
Dominican-Republic	Right	El-Salvador	Right
France	Left	Guatemala	Right
China	Left	Japan	Left
Yugoslavia	Left	Peru	Right
Outlying-US	Right	Scotland	Left
Trinadad-Tobago	Right	Greece	Left
Nicaragua	Right	Vietnam	Right
Hong	Left	Ireland	Left
Hungary	Left	Holland-Netherlands	Right

Table 2. The split provided by the GA for the native-country in the Adult dataset

The GA has been run with the aim of trying to find a good split by making use of the native-country splitting predictor that both R and SPSS, for instance, refused to process. As previously mentioned, 30 iterations seemed to be not enough because, in many runs of the algorithm, the "probably best" solution appeared after iteration 80. The solution provided by the algorithm is shown in Table 2. It gives an idea of the complexity of the problem.

The corresponding decrease in the node impurity is 0.3628465. The algorithm has been tested over many simulated dataset and the number of required iterations for the algorithm to reach convergence has been shown to linearly grow as a function of the number of attributes of the splitting predictor (the number of observations in the dataset appeared to be uninfluential).

Ant System. The strong complexity of the decision tree growing procedure (Hyafil & Rivest, 1976) does not allow to exhaustively enumerate and evaluate all the possible generable trees, even from very small datasets. In this respect, it is not possible to check whether the chosen heuristic is able to find the global optimum (in the same manner as it has been previously done for the genetic algorithm).
In the first experiment the algorithm has been tested on a simulated dataset of 500 observations with 11 nominal unordered predictors (with a number of attributes that ranges between 2 and 9) and 2 numeric (continuous) predictors. It could be seen that, when the required tree depth increases, the differences between the global impurity of the tree obtained by the CART greedy heuristic and the one obtained by the Ant System tend to increase. Table 3 reports such results.

Tree Depth	**CART**	**Ant System**
4	0.158119	0.153846
5	0.147435	0.121794
6	0.100427	0.085477
7	0.079059	0.059829
8	0.044871	0.029911

Table 3.Global impurity of the decision trees extracted by the proposed algorithm on a simulated dataset

Figure 2 shows the result obtained on the "Credit" dataset that can be found in the SPAD software (source: www.spadsoft.com). This dataset has 468 observations on which 11 nominal variables have been observed together with a two-class response variable. The aim would be to predict such response variable ("is a customer good or bad?).
The first decision tree is the one found by the CART heuristic and the second one has been extracted after 200 iterations of the Ant System algorithm.
Table 4 shows the global impurity of the trees extracted by the CART and Ant heuristics.

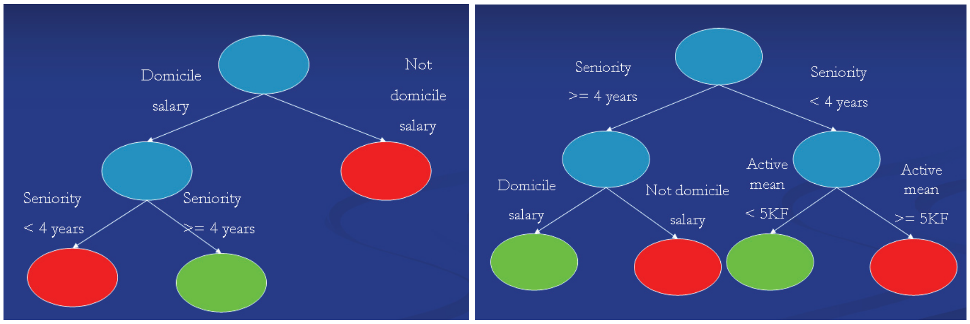


Fig. 2. Decision Trees for the Credit dataset obtained using the CART heuristic (left panel) and after 200 iterations of the Ant System algorithm (right panel).

The algorithms presented here are in an early stage of development. In these examples, an Ant System has been proposed to attack the problem of finding the best exploratory decision tree and it came out that the Ant System-based decision trees performed better than the ones found by the CART greedy heuristic. Even if the improvements weren't too large (from 2% to 5% in all of the simulation studies) such algorithm could be still useful for the situations in which high accuracy is required from the decision tree would. Ant System, on the other hand, is the simplest (yet less efficient) ACO technique, so that the use of more powerful ACO algorithms (which is currently under development) would reasonably bring better results. It is well known that ACO algorithms reach their maximum efficiency when coupled with local search techniques or can improve their efficiency by making use of candidate lists.

Tree Depth	CART	Ant System
2	0.2948	0.2734
3	0.2435	0.2301
4	0.2029	0.1816
5	0.1773	0.1517
6	0.1645	0.1539

Table 4. Global impurity of the decision trees extracted by the proposed algorithm on the Credit dataset

SASSC algorithm. In the following, the SASSC algorithm is applied on the "Letter Recognition" dataset from the UCI Machine Learning Repository (source http://archive.ics.uci.edu/ml/). This dataset is originally analyzed in Frey & Slate (1991), who did not achieve a good performance since the correct classified observations did never exceed 85%. Later on, the same dataset is analyzed in Fogarty(1992) using nearest neighbours classification. Obtained results give over 95.4% accuracy compared to the best result of 82.7% reached in Frey & Slate(1991). Nevertheless, no information about the interpretability of the nearest neighbour classification model is provided and the computational inefficiency of such a procedure is deliberately admitted by the authors.

In the Letter Recognition analysis, the task is to classify 20, 000 black-and-white rectangular pixel displays into one of the 26 letters in the English alphabet. The character images are based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 numerical attributes that have to be submitted to a decision tree. Dealing with $K = 26$ response classes, SASSC provides 25 sequential aggregations. Classification trees aggregated at each single step were chosen according to 10-fold cross validation. A tree was aggregated to the sequence if it provided the lowest cross validated generalization error with respect to the other trees obtainable from different aggregations of (subgroups of) response classes.

The results of the SASSC algorithm are summarized in Figure 3. It compares the performance of the SASSC model formed by g=2 up to g=6 superclasses with that of CART using, in all cases, either Gini or Twoing as splitting rules. Bagging (Brieman, 1996) and Random Forest (Breiman, 2001) are used as benchmarking methods as well. Computations have been carried out using the R software for statistical computing.

The SASSC model using 2 superclasses consistently improves the results of CART using the Gini (Twoing) splitting rule since the generalization error reduces to 0.49 (0.34) from 0.52 (0.49). As expected, the choice of the splitting rule (Gini or Twoing) is relevant when the number of superclasses g is relatively small $(2 \leq g \leq 4)$, whereas it becomes negligible for higher values of g (results for $g \geq 5$ are almost identical). Focusing on the Gini splitting criterion, the SASSC's generalization error further reduces to 0.11 when the number of subsets increases to 6. For comparative purposes, Bagging and Random Forest have been trained using 6 and 10 classifiers respectively and, in these cases, obtained generalization errors are worse than those deriving from SASSC with g = 6. As for Bagging and Random Forest, increasing the number of trees used to classify each subset of randomly drawn objects improves the performance of these two methods in terms of prediction accuracy. The reason is that their predictions derive form ("in-sample") independent bootstrap replications. Instead, cross-validation predictions in SASSC derives from aggregations of classifications made on "out-of-sample" observations that are excluded from the tree growing procedure. Thus, it is natural to expect that cross-validation predictions are more inaccurate than bagged ones. Of course, increasing the number of subsets of the response classes in SASSC reduces the cross-validated generalization error but, at the same time, increases the complexity of the final classification model. In spite of a relatively lower accuracy, interpretability of the results in SASSC with *g = 6* is strictly preserved.

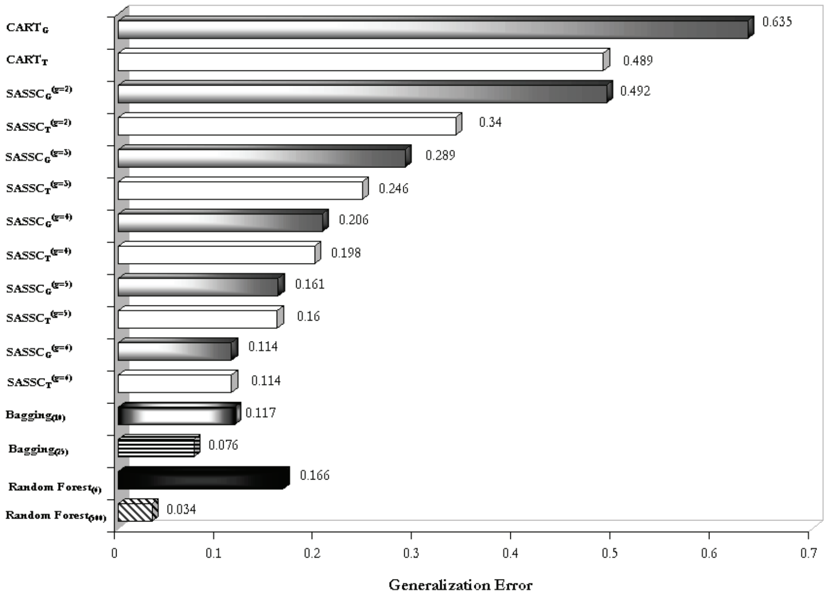


Figure 3. The generalization errors for the Letter Recognition dataset provided by alternative approaches: as for SASSC, subscript *G(T)* indicates the Gini (Twoing) splitting rule, whereas apex *g* indicates the number of superclasses (i.e., decision trees) identified by the algorithm. The subscript for Bagging and Random Forest indicates the number of trees used to obtain the classification by majority voting.

6. Discussion and conclusions

In the last two decades, computational enhancements highly contributed to the increase in popularity of DTI algorithms. This cause the successful use of Decision Tree Induction (DTI) using recursive partitioning algorithms in many diverse areas such as radar signal classification, character recognition, remote sensing, medical diagnosis, expert systems, and speech recognition, to name only a few. But recursive partitioning and DTI are two faces of the same medal. While the computational time has been rapidly reducing, the statistician is making more use of computationally intensive methods to find out unbiased and accurate classification rules for unlabelled objects. Nevertheless, DTI cannot result in finding out simply a number (the misclassification error), but also an accurate and interpretable model. Software enhancements based on interactive user interface and customized routines should empower the effectiveness of trees with respect to interpretability, identification and robustness. The latter considerations have been the inspiration for the algorithms presented in this chapter aimed at the improvement of DTI effectiveness. They lead to easily interpretable solutions for rather complicated data analysis problems and can be fruitfully used in different fields of Knowledge Discovery from Databases (KDD) and data mining such as, for example, web mining and Customer Relationship Management (CRM).

A Genetic Algorithm for multi-attribute predictor splitting is proposed in this chapter. It can be said that the proposed GA works very well in presence of treatable splitting predictors, for which the exhaustive enumeration is affordable. The algorithm always reaches the global optimum very quickly. This makes possible to think positively, even if nothing can be said, of course, about the case in which the number of attributes gets too large for the exhaustive enumeration and evaluation. Obtained results can be considered definitely useful in those cases where there are no other ways to attack the problem. Future research directions will include exhaustive enumerations on bigger datasets on a grid computing infrastructure.

In addition an Ant Colony Optimization algorithm is also proposed for exploratory tree growing. Such algorithm could be useful for the situations in which high accuracy is required from the decision tree would. Ant System, on the other hand, is the simplest (yet less efficient) ACO technique, so that the use of more powerful ACO algorithms (which is currently under development) would reasonably bring better results. It is well known that ACO algorithms reach their maximum efficiency when coupled with local search techniques or can improve their efficiency by making use of candidate lists.

Finally, a sequential search algorithm for modelling multi-attribute response through DTI has also been introduced. The motivation underlying the formalization of the SASSC algorithm derives from the following intuition: basically, since standard classification trees unavoidably lead to prediction inaccuracy in the presence of multi-class response, it would be favourable to look for a relatively reduced number of decision trees each one relating to a subset of classes of the response variable, the so called super-classes. Reducing the number of response classes for each of those trees naturally leads to improve the overall prediction accuracy. To further enforce this guess, an appropriate criterion to derive the correct number of super-classes and the most parsimonious tree structure for each of them has to be found. In this respect, a sequential approach that automatically proceeds through subsequent aggregations of the response classes might be a natural starting point.

The analysis of the Letter Recognition dataset demonstrated that the SASSC algorithm can be applied pursuing two complementary goals: 1) a content-related goal, resulting in the specification of a classification model that provides a good interpretation of the results without disregarding accuracy; 2) a performance-related goal, dealing with the development of a model resulting effective in terms of predictive accuracy without neglecting interpretability. Taking these considerations into account, SASSC appears as a valuable alternative to evaluate whether a restricted number of independent classifiers improves the generalization error of a classification model.

7. References

Breiman, L., Friedman, J.H., Olshen, R.A., & Stone C.J. (1984) *Classification and Regression Trees*, Wadsworth, Belmont CA.

Breiman, L. (1996) Bagging Predictors, *Machine Learning*, 24, 123-140.

Breiman,, L. (2001). Random Forests, *Machine Learning*, 45, 5-32.

Cappelli, C., Mola, F., & Siciliano, R. (2002), A Statistical Approach to Growing a Reliable Honest Tree, *Computational Statistics and Data Analysis*, 38, 285-299.

Chan, K. Y. & Loh, W. Y. (2004). LOTUS: An algorithm for building accurate and comprehensible logistic regression trees. *Journal of Computational and Graphical Statistics*, 13, 826–852.

Choi, Y., Ahn, H. & Chen, J.J. (2005). Regression trees for analysis of count data with extra Poisson variation. *Computational Statistics and Data Analysis,* 49, 893–915.

Colorni, A., Dorigo, M., Maffioli, F., Maniezzo, V., Righini, G., & Trubian, M. (1996). Heuristics from nature for hard combinatorial problems. *International Transactions in Operational Research*, March, 1-21.

Conversano, C. (2002) Bagged mixture of classifiers using Model Scoring Criteria. *Patterns Analysis & Applications*, 5, 4, 351-362.

Dietterich, T.G. (2000) Ensemble methods in machine learning. In J.Kittler and F.Roli, (Eds.), *Multiple Classifier System*. First International Workshop, MCS 2000, Cagliari, vol. 1857 of Lecture notes in computer science. Springer-Verlag.

Dorigo, M. & Stutzle, T. (2004). *Ant Colony Optimization*. The MIT Press, London. 1-15

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy.

Fogarty, T. (1992) First Nearest Neighbor Classification on Frey and Slate's Letter Recognition Problem (Technical Note). *Machine Learning*, 9, 387-388 .

Fogel, L. J. (1997). A retrospective view and outlook on evolutionary algorithms. In *Fuzzy Days*, 337–342.

Fogel, D. B. & Fogel, L. (1993). Evolutionary computation. *IEEE Transactions on Neural Networks*, 5(1):1–2.

Freund, Y., & Schapire, R. (1996), Experiments with a new boosting algorithm, *Machine Learning: Proceedings of the Thirteenth International Conference*, 148-156.

Frey, P.W. & Slate, D.J. Letter Recognition Using Holland-style Adaptive Classifiers. *Machine Learning*, 6, 161-182.

Gama, J. (2004), Functional trees, *Machine Learning*, 55, 219–250.

Hastie, T., Friedman, J. H., & Tibshirani, R., (2001). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer.

Hothorn, T., Hornik, K. & Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework, *Journal of Computational and Graphical Statistics*, 15, 651–674.

Hyafil & Rivest (1976). Constructing optimal binary decision trees is NPcomplete. *IPL: Information Processing Letters*, 15-17.

Klaschka, J., Siciliano, R., & Antoch, J. (1998): Computational Enhancements in Tree-Growing Methods, in: Rizzi, A., Vichi, M. & Bock, H.H. (Eds.), *Advances in Data Science and Classification: Proceedings of the 6th Conference of the International Federation of Classification Society,* Springer-Verlag, Berlin Heidelberg. 295-302

Loh, W.Y. (2002). Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, 12, 361-386.

Mehta, M., Agrawal, R. & Rissanen J. (1996). SLIQ. A Fast Scalable Classifier for Data Mining. In *Proceedings of the International Conference on Extending Database Technology EDBT*, 18-32.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, third edition.

Miele, R., Mola, F., Siciliano, R. (2005). J-Fast: An Interactive Software for Classification and Regression Trees. In *Proceedings of the Classification and Data Analysis Group (CLADAG) of the Italian Statistical Society.* Parma, Italy, 437-440

Miele, R. (2007). Nature Inspired Optimization Algorithms for Classification and Regression Trees. Ph.D. Thesis. Univeristy of Naples "Federico II".

Mola, F., & Conversano, C. (2008) Sequential Automatic Search of a Subset of Classifiers in Multiclass Learning, in: Brito P. & Aluja-Banet T. (Eds.) *COMPSTAT 2008 Proceedings in Computational Statistics,* Physica-Verlag, to appear.

Mola, F., & Siciliano, R. (1997). A fast splitting algorithm for classification trees. *Statistics and Computing*, 7, 209–216.

Oliver, J.J., & Hand, D. J. (1995). On Pruning and Averaging Decision Trees, in *Machine Learning: Proceedings of the 12th International Workshop*,430-437.

Quinlan, J.R., (1983). Learning Efficient Classification Procedures and Their Application to Chess and Games. In Michalski R.S., Carbonell J.G. & Mitchell T.M. (ed.): *Machine Learning: An Artificial Intelligence Approach*, 1, Tioga Publishing, 463-482.

Quinlan, J.R., (1987). Simplifying decision tree. *International Journal of Man-Machine Studies*, 27, 221–234.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Siciliano, R., (1998). Exploratory versus decision trees. In: Payne, R., Green, P. (Eds.), *COMPSTAT 1998 Proceedings in Computational Statistics*. Physica-Verlag, 113–124.

Siciliano, R. & Mola, F. (2000). Multivariate Data Analysis through Classification and Regression Trees, *Computational Statistics and Data Analysis*, 32, 285-301, Elsevier Science, 2000.

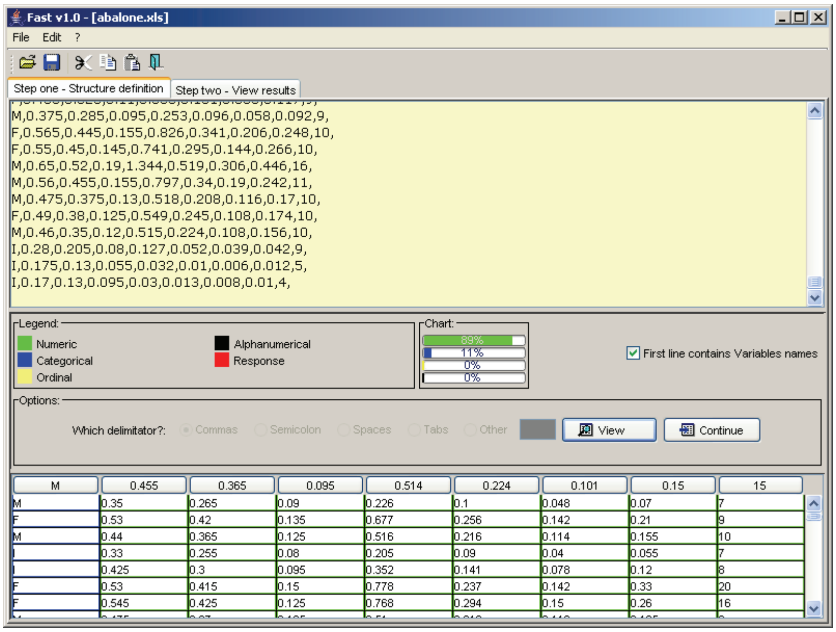
Su, X., Wang, M. & Fan, J. (2004). Maximum likelihood regression trees. *Journal of Computational and Graphical Statistics*, 13, 586–598.

Vose, M. D. (1999). The simple genetic algorithm: foundations and theory. MIT Press, Cambridge, MA.

Appendix: The J-FAST software

The algorithms presented in this chapter have been implemented in the Java language. In order to make it possible to test them on real datasets a Java segmentation framework, called J-FAST, has been developed. The first aim of this software is to take care of all the necessary operations to perform before and after running the recursive partitioning algorithm. These can be summarized as follows: reading data from text files and spreadsheets; processing data before carrying out the tree growing process; specifying the type of recursive partitioning algorithm to be applied (i.e., classification or regression tree) ; interpretation of the results.

The J-FAST program is a Java-based recursive partitioning software, which is particularly research oriented. It mainly consists of a flexible, efficient and transparent cross-platform application for building classification and regression trees using any kind of heuristic in the tree growing process (like the CART greedy algorithm or the FAST branch and bound heuristic or any other one written by the user). It also allows to interactively visualize and compare the results. J-FAST divides the recursive partitioning procedure into three main sections. The data-importing Graphical User Interface (see Figure 4) allows to read data from Excel-like spreadsheets and plain text files and automatically recognises the nature of the variables by distinguishing the categorical, numerical or alphanumerical columns of a data matrix. J-Fast also allows the user to specify the Decision Tree Induction model by choosing the response variable, as well as which predictor(s) should be treated as ordinal, nominal or as excluded from the analysis.



```
M,0.375,0.285,0.095,0.253,0.096,0.058,0.092,9,
F,0.565,0.445,0.155,0.826,0.341,0.206,0.248,10,
F,0.55,0.45,0.145,0.741,0.295,0.144,0.266,10,
M,0.65,0.52,0.19,1.344,0.519,0.306,0.446,16,
M,0.56,0.455,0.155,0.797,0.34,0.19,0.242,11,
M,0.475,0.375,0.13,0.518,0.208,0.116,0.17,10,
F,0.49,0.38,0.125,0.549,0.245,0.108,0.174,10,
M,0.46,0.35,0.12,0.515,0.224,0.108,0.156,10,
I,0.28,0.205,0.08,0.127,0.052,0.039,0.042,9,
I,0.175,0.13,0.055,0.032,0.01,0.006,0.012,5,
I,0.17,0.13,0.095,0.03,0.013,0.008,0.01,4,
```

Legend:
🟩 Numeric
🟦 Categorical
🟨 Ordinal
⬛ Alphanumerical
🟥 Response

Chart: 89% 11% 0% 0%

✓ First line contains Variables names

Options:
Which delimitator?: [● Commas] Semicolon Spaces Tabs Other

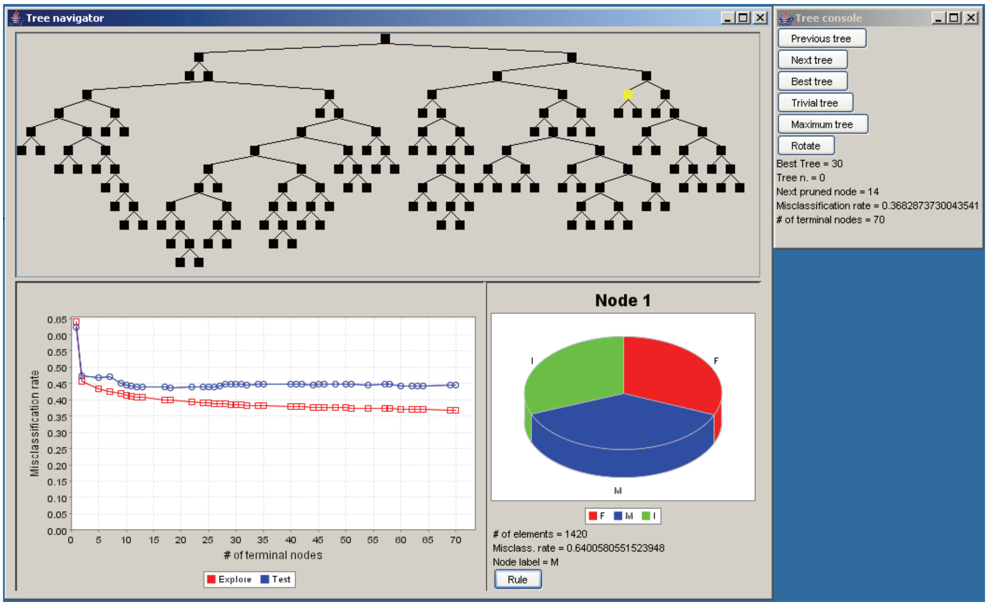
M	0.455	0.365	0.095	0.514	0.224	0.101	0.15	15
M	0.35	0.265	0.09	0.226	0.1	0.048	0.07	7
F	0.53	0.42	0.135	0.677	0.256	0.142	0.21	9
M	0.44	0.365	0.125	0.516	0.216	0.114	0.155	10
I	0.33	0.255	0.08	0.205	0.09	0.04	0.055	7
I	0.425	0.3	0.095	0.352	0.141	0.078	0.12	8
F	0.53	0.415	0.15	0.778	0.237	0.142	0.33	20
F	0.545	0.425	0.125	0.768	0.294	0.15	0.26	16

Fig. 4. J-Fast data importing Graphical User Interface

A second GUI visualizes some information about the chosen DTI model and provides some descriptive statistics about the analyzing data. It also allows the user to specify which are the features of the DTI model under specification, such as the learning sample rate, the stopping conditions, the possibility of obtaining a verbose output. It also asks the user to choose between all the recursive partitioning heuristics that are present into the class path. Then, the software starts the tree growing procedure.

The third component of the J-FAST software is the results navigator. It allows the user to interactively display and navigate into the results of the analysis.

The results navigator GUI (see Figure 5) consists of two windows. The first one is the main results window. It visualises the obtained decision tree, charts the misclassification rates and the selected node's information panel (there is a button for visualizing the splitting rule to reach the node, the misclassification rate for the node, etc.). The second component is the Tree Console Window (Figure 6). It contains buttons that allow the user to navigate through the pruning sequence and access directly the best, the trivial and the maximal tree. For each tree in the pruning sequence, the node that is going to be pruned is highlighted. By clicking on the node, the interface allows to get the data units which fall in that node and to write them into a file in order to continue the analysis of such units using another software. It is also possible, from the second step GUI, to simultaneously start more than one analysis in order to obtain different tree navigators simultaneously on the screen. This feature is particularly useful for comparing trees grown from different datasets or on the same dataset but with using different DTI specifications.



```
Best Tree = 30
Tree n. = 0
Next pruned node = 14
Misclassification rate = 0.3682873730043541
# of terminal nodes = 70
```

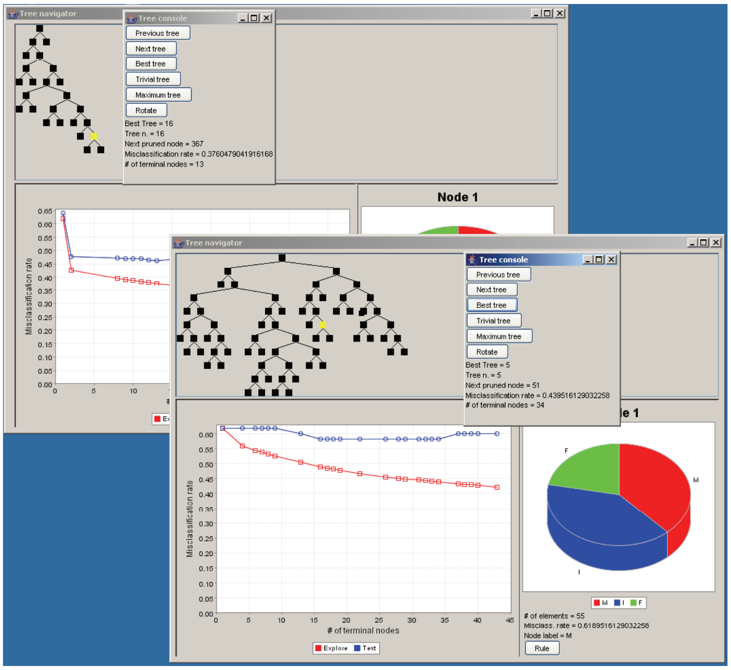
```
# of elements = 1420
Misclass. rate = 0.6400580551523948
Node label = M
```

Fig. 5. J-Fast data results navigator Graphical User Interface

J-FAST is more than a simple recursive partitioning software. Because of the fact that it has been mainly designed to support the research activity, it offers many useful functions like the possibility of saving created objects (trees, datasets, nodes, etc.) via the Java serialization mechanism in order to better analyze using other ad-hoc written Java programs (some of them have already been implemented, like a different tree interface called "TreeSurfer").
Interactivity with the R statistical software is also provided: by right-clicking on a node it is possible to send the corresponding data to R in order to continue the analysis. This is particularly useful if another statistical analysis (i.e. a logit model) has to be made on a particular segment (node) extracted from the obtained decision tree.
J-FAST has to be also considered as a Java objects Library (or API - Application Program Interface), for building Classification and Regression Trees. Any researcher which is able to program in Java could use the classes from the J-FAST API in order to get trees without having to write all the necessary code. In addition, the J-FAST platform offers many useful objects. The most important ones are:

- Statistics: it provides univariate and bivariate descriptive statistics.
- DataSet: it stores data for recursive partitioning purposes (response variable, predictors, etc.).
- Split: it specifies the type of split (binary, ternary,etc.)

- TreeGrower: it is a class for growing decision trees
- Pruner: it is class that for decision tree pruning
- TreeViewer: it is a interactive interface class
- Utility: it encompasses many useful function like reading data from plain text files, Excel-like spreadsheets, etc.
- TreeBuild interface: it defines all the rules to follow for the programmer to write his own heuristic.



```
Best Tree = 16
Tree n. = 16
Next pruned node = 367
Misclassification rate = 0.3760479041916168
# of terminal nodes = 13
```

```
Best Tree = 5
Tree n. = 5
Next pruned node = 51
Misclassification rate = 0.439516129032258
# of terminal nodes = 34
```

```
# of elements = 55
Misclass. rate = 0.6189516129032258
Node label = M
```

Fig. 6. J-Fast tree console window Graphical User Interface