GA-EDA: A New Hybrid Cooperative Search Evolutionary Algorithm

Victor Robles[1], Jose M. Peña[1], Pedro Larrañaga[2], María S. Pérez[1], Vanessa Herves[1]

[1] Department of Computer Architecture and Technology, Universidad Politécnica de Madrid, Madrid, Spain
{vrobles,jmpena,mperez,vherves}@fi.upm.es
[2] Department of Computer Science and Artificial Intelligence, University of the Basque Country, San Sebastián, Spain
ccplamup@si.ehu.es

Summary. Hybrid metaheuristics have received considerable interest in recent years. A wide variety of hybrid approaches have been proposed in the literature. In this paper a new hybrid approach, named GA-EDA, is presented. This new hybrid algorithm is based on genetic and estimation of distribution algorithms. The original objective is to benefit from both approaches and attempt to achieve improved results in exploring the search space. In order to perform an evaluation of this new approach, a selection of synthetic optimization problems have been proposed, together with some real-world cases. Experimental results show the competitiveness of our new approach.

1 Introduction

Over the last years, interest in hybrid metaheuristics has risen considerably among researchers. The best results found for many practical or academic optimization problems are obtained by hybrid algorithms. Combination of algorithms such as descent local search [32], simulated annealing [21], tabu search [12] and evolutionary algorithms have provided very powerful search algorithms.

Two competing goals govern the design of a metaheuristic [39]: exploration and exploitation. Exploration is needed to ensure every part of the search space is searched thoroughly in order to provide a reliable estimate of the global optimum. Exploitation is important since the refinement of the current solution will often produce a better solution. Population-based heuristics (where genetic algorithms [18] and estimation of distribution algorithms [23] are found) are powerful in the exploration of the search space, and weak in the exploitation of the solutions found.

With the development of our new approach, GA-EDA, a hybrid algorithm based on genetic algorithms (GAs) and estimation of distribution algorithms (EDAs), we aim to improve the exploration power of both techniques.

V. Robles et al.: *GA-EDA: A New Hybrid Cooperative Search Evolutionary Algorithm*, StudFuzz **192**, 187–219 (2006)
www.springerlink.com

This hybrid algorithm has been tested on combinatorial optimization problems (with *discrete* variables) as well as *real-valued* variable problems. Results of several experiments show that the combination of these algorithms is extremely promising and competitive.

This paper is organized in the following way: First, we will focus on different taxonomies of hybrid algorithms found in the literature; in Sect. 3, the new GA-EDA approach is proposed with a complete performance study presented in Sect. 4. Finally we close with our conclusions and further, future work.

2 Taxonomy of Hybrid Algorithms

The goal of the general taxonomies is to provide a mechanism to allow comparison of hybrid algorithms in a qualitative way. Additionally, taxonomies are useful to indicate areas in need of future work, as well as assist in classifying new hybrid approaches. In this section we include a survey of the current, most important hybrid taxonomies.

In [4] three different forms of hybridization are described:

- *Component Exchange Among Metaheuristics.*
  One of the most popular ways of hybridization concerns the use of trajectory methods, such as local search, Tabu Search or Simulated Annealing, in population-based algorithms. Most of the successful applications of Evolutionary Computation (EC) make use of local search algorithms. The reason for the success comes from the strengths of trajectory methods and population-based methods, finding a proper balance between diversification (exploration) and intensification (exploitation).
  The power of population-based methods is based on the concept of recombining solutions to obtain new ones. In EC algorithms, explicit recombinations are implemented by one or more recombination operations. In EDAs recombination is implicit because new solutions are generated using a distribution over the search space which is a function of earlier populations. This allows making guided steps in the search space which are usually larger than the steps done by trajectory methods.
  The strength of trajectory methods is found in the way they explore a promising region of the search space. A promising area in the search space is searched in a more structured way than in population-based methods. In this way, the danger of being close to good solutions but "missing" them is not as high as in population-based methods.
  In summary, population-based methods are better at identifying promising areas in the search space, whereas trajectory methods are better at exploring promising areas in the search space. Thus, metaheuristic hybrids that manage to combine the advantages of population-based methods with the strength of trajectory methods are often very successful.

Some examples of this trend are: GASAT [14] which incorporates local search within the genetic framework for solving the satisfiability problem or [45] a hybrid algorithm based on the combination of EDA with Guided Local Search for Quadratic Assignment Problems.

- *Cooperative Search.*
  A loose form of hybridization is provided by cooperative search [1, 8, 17, 38, 42, 43], which consists of a search performed by possibly different algorithms that exchange information about states, models, entire sub-problems, solutions or other search space characteristics. Typically, cooperative search algorithms consist of the parallel execution of search algorithms with a varying level of communication. The algorithms can be different or they can be instances of the same algorithm working on different models or running with different parameters settings.
  Presently, cooperative search receives more attention which, among other reasons, is due to the increasing research on parallel implementations of metaheuristics [3, 6, 24, 34, 35]. The aim of research on parallelization of metaheuristics is twofold. First, metaheuristics should be redesigned to make them suitable for parallel implementation in order to exploit intrinsic parallelism. Second, an effective combination of metaheuristics has to be found, both to combine different characteristics and strengths, and to design efficient communication mechanisms.
- *Integrating Metaheuristics and Systematic Methods.*
  This approach has recently produced very effective algorithms especially when applied to real-world problems. Discussions on similarities, differences and possible integration of metaheuristics and systematic search can be found in [11, 12, 15]. A very successful example of such an integration is the combination of metaheuristics and Constraint Programming [10].

Our hybrid GA-EDA algorithm, which is a completely new approach, can be classified in the second form; cooperative search, of Blum and Roli's classification.

Another excellent taxonomy can be found in [39]. In this hierarchical classification, at the first level, *low-level* and *high-level* hybridizations are distinguished. In low-level algorithms, a given function of a metaheuristic is replaced by another metaheuristic. In high-level algorithms, the different metaheuristics are self-contained; we have no direct relationship to the internal workings of a metaheuristic.

At the second level, *relay* and *co-evolutionary* hybridizations are distinguished. In relay hybridization, a set of metaheuristics is applied one after another, each using the output of the previous as its input, acting in a pipeline fashion.

Four classes are derived from this hierarchical taxonomy:

- *LRH (Low-level Relay Hybrid).*
  Algorithms in which a given metaheuristic is embedded into a single-solution metaheuristic. A few examples from the literature belong to this class. For instance in [28] a LRH hybrid which combines simulated annealing with local search to solve the travelling salesman problem, is introduced.
- *LCH (Low-level Co-evolutionary Hybrid).*

Algorithms in which population based heuristics have been coupled with local search heuristics such as hill-climbing, Simulated Annealing and Tabu Search. The local search algorithms will try to optimize locally, while the population based algorithms will try to optimize globally. It is exactly the same form as previously defined *component exchange among metaheuristics*.

- *HRH (High-level Relay Hybrid).*
  In HRH hybrid self-contained metaheuristics are executed in a sequence. For example, a HRH hybridization may use a greedy heuristic to generate a good initial population for an EC algorithm. Once the high performance regions are located, it may be useful to apply local search heuristics to these regions; thus, in this example, we have three pipelined algorithms. Many authors have used the idea of HRH hybridization for EC. In [25] the authors introduce simulated annealing to improve the population obtained by a GA. In [27] the proposed algorithm starts from simulated annealing and uses GAs to enrich the solutions found.
- *HCH (High-level Co-evolutionary Hybrid).*
  This schema is similar to the previously defined *cooperative search*. It involves several self-contained algorithms performing a search in parallel, and cooperating to find an optimum. Intuitively, HCH will ultimately perform at least as well as one algorithm alone, and more often perform better. Each algorithm provides information to the others to help them. An example of HCH based on parallel EDAs is the island model [34, 35].

In Talbi's taxonomy GA-EDA is heterogeneous because different metaheuristics are used; global because the algorithm search the whole state space, and general because both, GAs and EDAs, solve the same target optimization problem (HCH algorithm).

3 Hybrid GA-EDA Algorithm

Hybrid GA-EDA are new algorithms based on both techniques [33]. The original objective is to get benefits from both approaches. The main difference from these two evolutionary strategies is how new individuals are generated. These new individuals generated on each generation are called *offspring*. On the one hand, GAs use crossover and mutation operators as a mechanism to create new individuals from the best individuals of the previous generation. On the other, EDAs builds a probabilistic model with the best individuals and then sample the model to generate new ones.

3.1 Introduction

Our new approach generates two groups of offspring individuals, one generated by the GA mechanism and the other by EDA one. $Population_{p+1}$ is composed of the best overall individuals from (i) the past population ($Population_p$), (ii) the GA-evolved offspring, and (iii) EDA-evolved offspring.

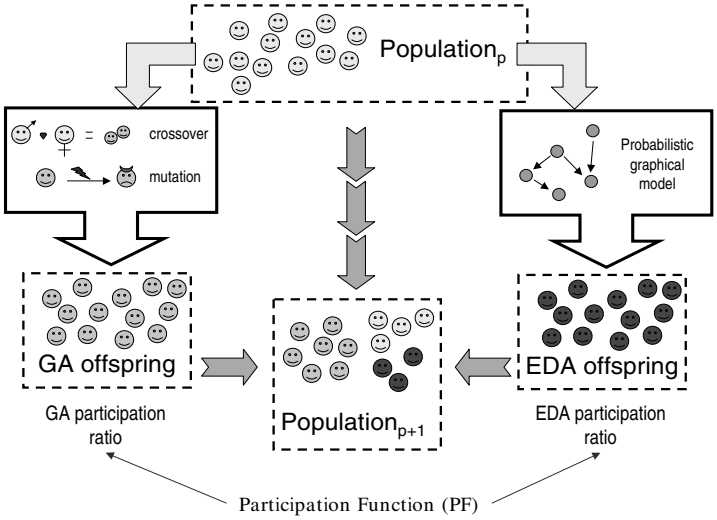

Fig. 1. Hybrid Evolutionary Algorithm Schema

The individuals are selected based on their fitness function. This evolutionary schema is quite similar to Steady State GA in which individuals from one population, with better fitness than new individual from the offspring, survive in the next one. In this case we have two offspring pools. Figure 1 shows how this model works.

3.2 Participation Functions

In this approach an additional parameter appears, this parameter has been called *Participation Function*. Participation Function provides a ratio of how many individuals are generated by each mechanism. In other words, the size of GA and EDA offspring sets. The size of these sets also represents how each of these mechanisms participates on the evolution of the population. These ratios are only a proportion for the number of new individuals each method generates, it is not a proportion of individuals in the next population, which is defined by the quality of each particular individual. If a method were better than the other in terms of how it combines the individuals, there would be more individuals from this offspring set than from the other.

The following alternatives for Participation Functions are introduced:

Constant Ratio (*x% EDA / y% GA*)

The percentage of individuals generated by each method is constant during all the generations.

Alternative Ratio (*ALT*)

On each generation it alternates either GA or EDA generation method. If the generation is an even number GA mechanism generates all offspring individuals, if it is an odd number, it is the EDA method.

Incremental Ratio (*EDA++* and *GA++*)

The partition ratio for one of the mechanism increases from one generation to the other. There are two incremental Participation Functions, GA Incremental Function and EDA Incremental Function. The ratio is defined by the formula[3]:

$$i - ratio = \frac{gen}{M + gen} \tag{1}$$

where,

$$i - ratio_{GA} = 1 - i - ratio_{EDA} \tag{2}$$

Dynamic Ratio (*DYNAMIC*)

The previous functions do not take into account the quality of the evolution methods they are merging. There is no simple method that outperforms the other in all the cases and a static Participation Function could lead toward the same problem. A constant ratio function, like *50% EDA / 50% GA* could balance the benefits and the problems from each approach, but if, for one problem, GAs do not provide good individuals, the former Participation Function would waste computational resources dealing with genetic operators and fitness evaluations for individuals that would not improve the overall population.

As a difference with the previous Participation Functions that are static and deterministic, we also propose a dynamic adaptative Participation Function. The idea is to have a mechanism that increases the participation ratio for the method which happens to generate better individuals. This function evaluates each generation considering the possibility to change the participation criterion as defined by the ratio array.

This function performs according to the algorithm in Fig. 2.

In Fig. 2 avg_score represents an array of the average fitness score of the top 25% of the individual generated by each of the offspring methods. $base$ is the average fitness of the first generation. dif represents the relative difference in terms of improvement that the best method has compared with the other. $ADJUST$ is a constant that defines the trade-off between these two methods when one of them performs better than the other (5% in our experimentation).

This algorithm starts with 50%/50% ratio distribution between the two methods. On each generation the best offspring individuals from each method are compared

[3] gen is the number of the current generation and M, called the Mid-point, represents at which generation the ratio is *50%/50%*. Participation Function is 0 at the first generation and never reaches 1

```
diff=(MAX(avg_score[GA],avg_score[EDA])-base) /
   (MIN(avg_score[GA],avg_score[EDA])-base);
if (avg_score[GA]>avg_score[EDA]) {
   ratio_inc=ratio[EDA] * ADJUST * dif;
   ratio[GA] += ratio_inc;
   ratio[EDA] = 1.0 - part[GA];
}
else if (avg_score[GA]<avg_score[EDA]) {
   ratio_inc=ratio[GA] * ADJUST * dif;
   ratio[EDA] += ratio_inc;
   ratio[EDA] = 1.0 - part[GA];
}
```

Fig. 2. Pseudocode of Dynamic Participation Function

and the wining method gets a 5% of the opposite method ratio (scaled by the amount of relative difference between the methods, dif variable). This mechanism provides a contest-based DYNAMIC function in which methods are competing to get higher ratios as they generate better individuals.

4 Binary-encoded Problems

Part of the experiments have been performed considering six different binary-encoded problems:

❶ The *MaxBit* problem.
❷ Two deceptive tramp functions.
❸ A Feature Subset Selection wrapper approach for a classification problem.
❹ The *Holland Royal Road* function.
❺ One Satisfiability (SAT) problem.

On the figures, which represent the experiments, it is shown the results using five different constant ratio functions: CONST 0.00 ($0\%GA$ / $100\%EDA$, pure EDA algorithm), CONST 0.25 ($25\%GA$ / $75\%EDA$), CONST 0.50 ($50\%GA$ / $50\%EDA$), CONST 0.75 ($75\%GA$ / $25\%EDA$) and CONST 1.00 ($100\%GA$ / $0\%EDA$, pure GA). The best of these five constant Participation Functions is included also in the second figure of the experiment, as well as the four variable Participation Functions: ALT (Alternative Function), GA++ (Incremental GA function), EDA++ (Incremental EDA function), and DYNAMIC (Dynamic Participation Function).

The proposed hybrid algorithm is composed of the simplest versions of both GA and EDA components. In this sense a single bit-string chromosome (for binary-encoded problems) and real string (for continuous problem) have been used to code all the problems. GA uses *Roulette Wheel* selector, one-point crossover, flip mutation (with probability 0.01) and uniform initializer. EDA uses the Univariate Marginal Distribution Algorithm (UMDA) [30] in discrete problems and the continuous

version (UMDA$_c$) [22] in continuous problems. The overall algorithms generate an offspring twice the size of the population. Depending on the ratios provided by the Participation Function, this offspring is then divided between the two methods. The composition of the new population is defined by a deterministic method, selecting the best fitness scores from the previous population and both offspring sets. The stopping criteria is quite straightforward, we stop when the difference of the sum of the fitness values of all individuals in two successive generations is smaller than a predefined value.

The experiments have been executed ten times and the average of these executions are presented. Several population sizes have been tested, but only the most representative size has been included. All the experiments have been performed in an 8-nodes cluster of bi-processors with Intel Xeon 2.4Ghz with 1GB of RAM and Gigabit network running Linux 2.4.

In most cases we have applied the Mann-Whitney statistical test to compare the results achieved by the algortihms. The fitness values of the best solutions found in the search are used for this purpose.

It is important to highlight that the use of different individual representations to the ones here used, can guide to very different results.

4.1 The MaxBit Problem

Definition

We try to obtain the maximum of the function defined as:

$$f_{M256}(\boldsymbol{x}) = \frac{\sum_{i=1}^{256} x_i}{n}$$
$$x_i \in \{0, 1\}$$
$$f_{M256}(\boldsymbol{x}^*) = max(f_{M256}(\boldsymbol{x}))$$

This problem is a typical benchmark function to evaluate the performance of evolutionary algorithms and the global maximum is found in 1.

Results

We have done this experiment using a population size of 100 individuals. Figure 3a shows that the performance of the pure genetic algorithm is very poor, while EDA outperforms all the constant Participation Functions, although these other functions also reach the optimum value.

Variable Participation Functions (see Fig. 3b) also succeed in finding the maximum but with few more iterations to converge. Dynamic Participation Function is the second best approach.

This problem shows a lineal independence among the genes of each of the individuals. EDA profits from this characteristic better than any other Participation Function. It should be considered that this feature is not quite realistic when considering real-world problems.

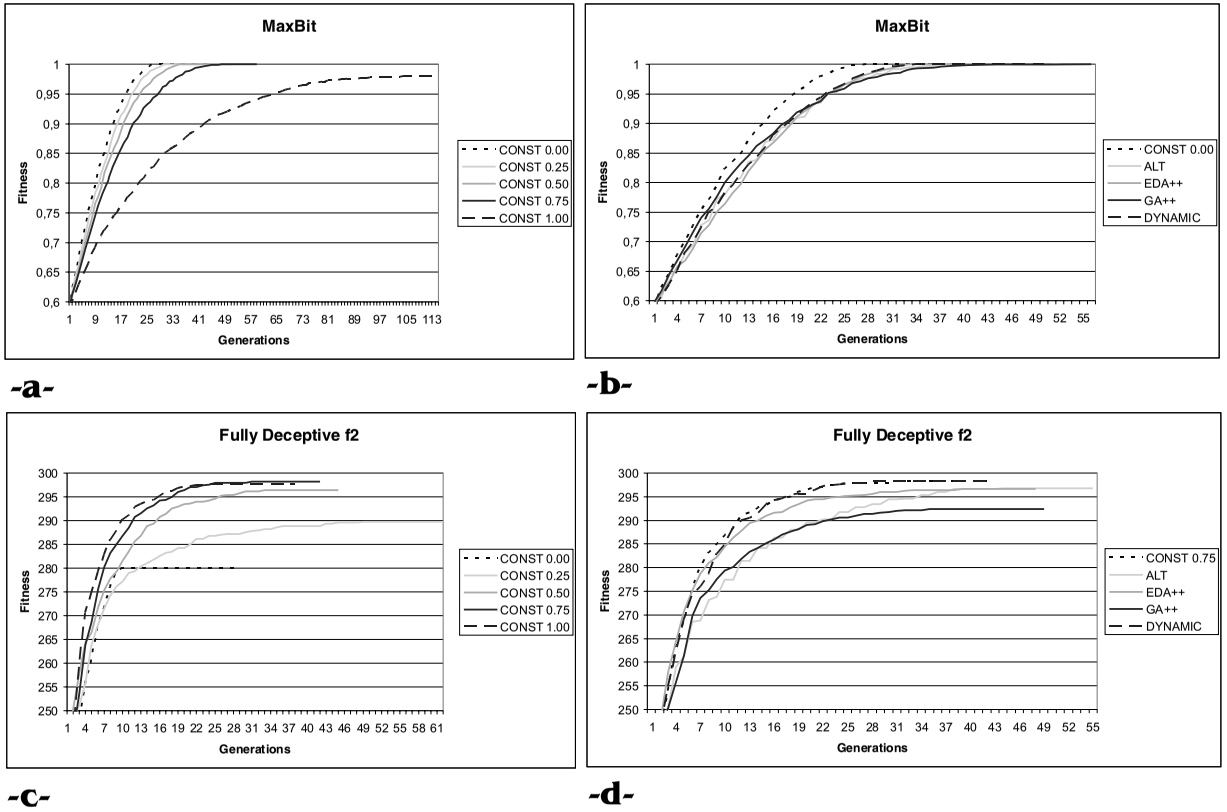

Fig. 3. MaxBit problem solved with -**a**- constant Participation Functions and -**b**- variable Participation Functions and Fully Deceptive f2 solved with -**c**- constant Participation Functions and -**d**- variable Participation Functions

4.2 4-bit Fully Deceptive Function

Definition

Deceptive trap functions are used in many studies of GAs because their difficulty is well understood and it can be regulated easily [7]. We have used the 4-bit fully deceptive functions of order 2 and order 3, defined in [44].

These deceptive functions (f_{D_2} and f_{D_3}) are 40 bit long maximization problems, and are comprised of 10 sub-problems, each 4 bits longs. The sub-problems evaluate 4 bits using the following lookup table shown in Table 1. Thus, the global maximum is 300.

Table 1. Evaluation of four bits for 4-bit fully deceptive function

	Evaluation			Evaluation	
Chromosome	**Order 2**	**Order 3**	**Chromosome**	**Order 2**	**Order 3**
1111	30	30	0000	28	10
1100	8	5	0101	16	5
1110	6	0	0001	26	25
1101	4	0	0110	14	5
1011	2	0	0010	24	26
0111	0	0	1001	12	5
0011	18	5	0100	22	27
1010	10	5	1000	20	28

Results

These problems have been solved with a population size of 250 individuals.

In both problems GAs get a performance better than EDAs, with p-value < 0.001. Nevertheless in f_{D_2} CONST 0.75 with p-value = 0.649 and Dynamic Participation Function with p-value = 0.649 slightly improve the results of pure GA algorithms. In f_{D_3} CONST 0.50 with p-value = 0.811 and Dynamic Participation Function with p-value = 0.257 also slightly improve to GAs.

Fully deceptive functions are problems designed to get GAs into trouble because the fitness values of the points in the neighborhood of the optimal values are worse than points located far away. This characteristic penalizes EDAs more severely, which are prone to fall into the deceptive tramps due to the combination method used to generate new individuals (distribution of single genes in the individual encoding).

The combination of both techniques performs better because once the mutation and crossover operators have reached the optimal value for a subproblem, no mutation will break this building block afterwards. The performance of the different constant or variable Participation Functions is not the same based on the particular

characteristics of the deceptive tramp. DYNAMIC seems to perfectly adapt the participation ratio in order to balance GA and EDA recombination techniques to deal with these problems.

4.3 Feature Subset Selection

Definition

Feature Subset Selection (FSS) [20] is a well-known task in the Machine Learning, Data Mining, Pattern Recognition and Text Learning fields. FSS formulates as follows: Given a set of candidate features, select the best subset under some learning algorithm. As the learning algorithm, we are going to use naïve Bayes [9,13]. A good review of FSS algorithm can be found in [26]. To test the FSS problem we will use the chess dataset from the UCI repository [31], which has a total of 36 features and 699 instances.

Results

We have done this experiment using a population size of 1000 individuals. Figure 4c shows that pure GAs are a better option than EDAs for the FSS problem, with p-value = 0.004. The other constant Participation Functions do not reach results as good as GAs. For instance, with respect to CONST 0.25, the best constant Participation Function, the Mann-Whitney p-value is 0.197.

On the other hand, Fig. 4d also shows that variable Participation Functions are close to the results of GAs, being DYNAMIC the best of these functions, with p-value = 0.819. Nevertheless, the results achieved by GAs are the best for this problem.

It is important to consider that the number of generations is quite low and the complexity of the problem is not very significant. A detailed study of more complex FSS scenarios should be addressed to confirm the performance of the different algorithms on this problem. The morphology of the problem, the dataset, its features and the relationships among them, is very relevant to evaluate the performance of the algorithms in them.

4.4 240 bit Holland Royal Road - JHRR

Definition

The Holland Royal Road functions were introduced in [29]. They were designed as functions that would be simple for a genetic algorithm to optimize, but difficult for a hillclimber. In [19], Holland presented a revised class of Royal Road functions that were designed to create insurmountable difficulties for a wider class of hillclimbers, and yet still admissible to optimization by a GA.

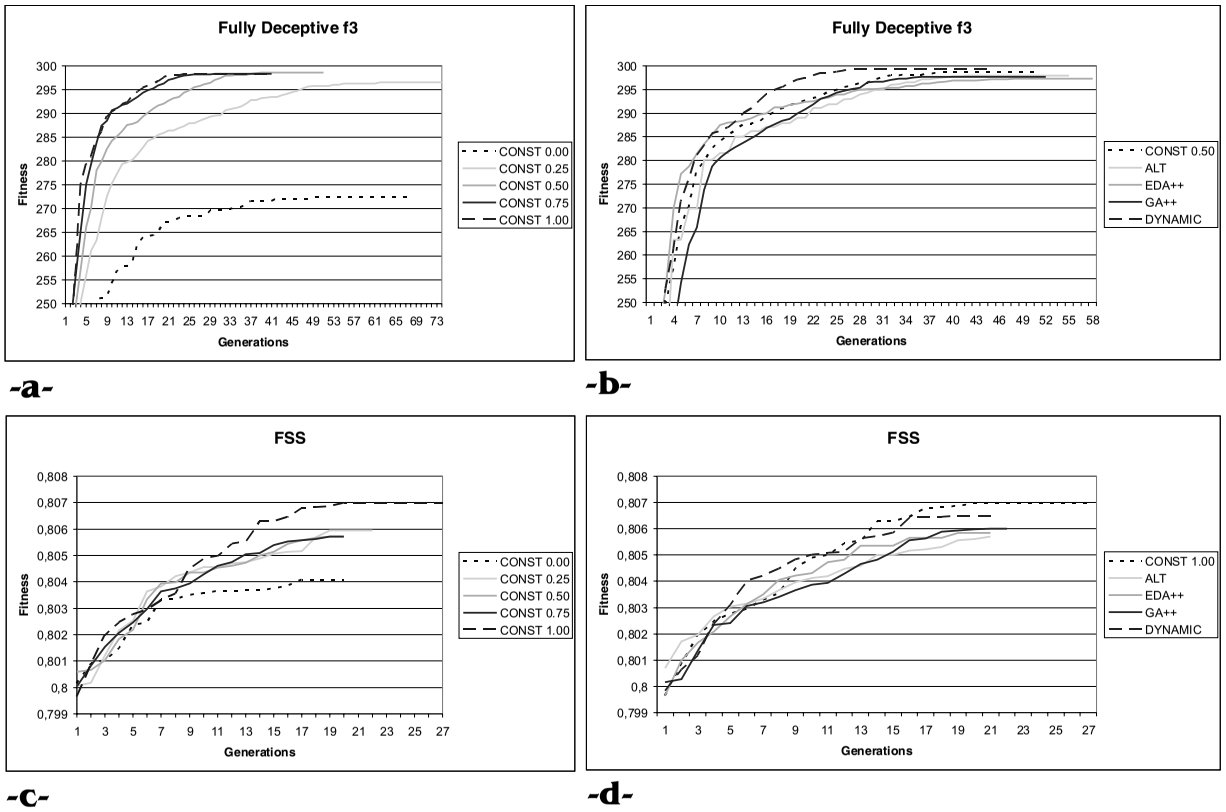

Fig. 4. Fully Deceptive f3 solved with -**a**- constant Participation Functions and -**b**- variable Participation Functions and Feature Subset Selection solved with -**c**- constant Participation Functions and -**d**- variable Participation Functions

The Holland Royal Road function takes a binary string as input and produces a real value. The function is used to define a search task in which one wants to locate strings that produce high function values. The string is composed of $2k$ non-overlapping continuous regions, each of length $b+g$. With Holland's defaults, $k = 4$, $b = 8$, $g = 7$, there are 16 regions of length 15, giving an overall string length of 240. Each region is divided into two non-overlapping pieces. The first, of length b, is called the block, and the second, of length g, is called the gap. In the fitness calculation, only the bits in the block part of each region are considered. The fitness calculation proceeds in two steps: the PART calculation, that considers each block individually and, the BONUS calculation, created to reward completed blocks and some combinations of completed blocks.

Results

We have done this experiment using a population size of 500 individuals. Holland Royal Road problem, as shown by Fig. 5a, is a very complex scenario for EDAs. This problem was designed to highlight and compare the benefits of GAs to hill climbers or other optimizers that are neighborhood-oriented search methods. Although GAs are well-suited for this problem, a combination of 25% EDAs and 75% GAs gets better results, with Mann-Whitney p-value = 0.0353. As one of the benefits mentioned in the introduction of this technique, hybrid algorithms improves the results by using two different exploratory techniques which increase the probability to find the optimal values as the range of possible movements is more complete.

In this case, ALT Participation Function outperforms all the other functions, including CONST 0.75 with p-value = 0.306. Figure 5b presents how GA++ also performs better than CONST 0.75 with less number of generations and a p-value = 0.3267. ALT gets more iterations to converge ($\sim 10\%$ more) which means that GA and EDA offsprings change a little more when they are near to the optimum value and then the exploration is more exhaustive.

4.5 SAT problem

Definition

The goal of the satisfiability (SAT) problem [36] is to attempt to find an assignment of truth values to the literals of a given Boolean formula, in its conjunctive normal form, that satisfies it. In theory, SAT is one of the basic core NP-complete problems. In practice, it has become increasingly popular in different research fields, given that several problems can be easily encoded into propositional logic formula such as planning, formal verification, knowledge representation and so on. In GAs and EDAs the SAT problem can be represented using binary strings of length n in which the $i\text{-}th$ bit represents the true value of the $i\text{-}th$ propositional variable in the formula. The fitness function used is the fraction of clauses satisfied.

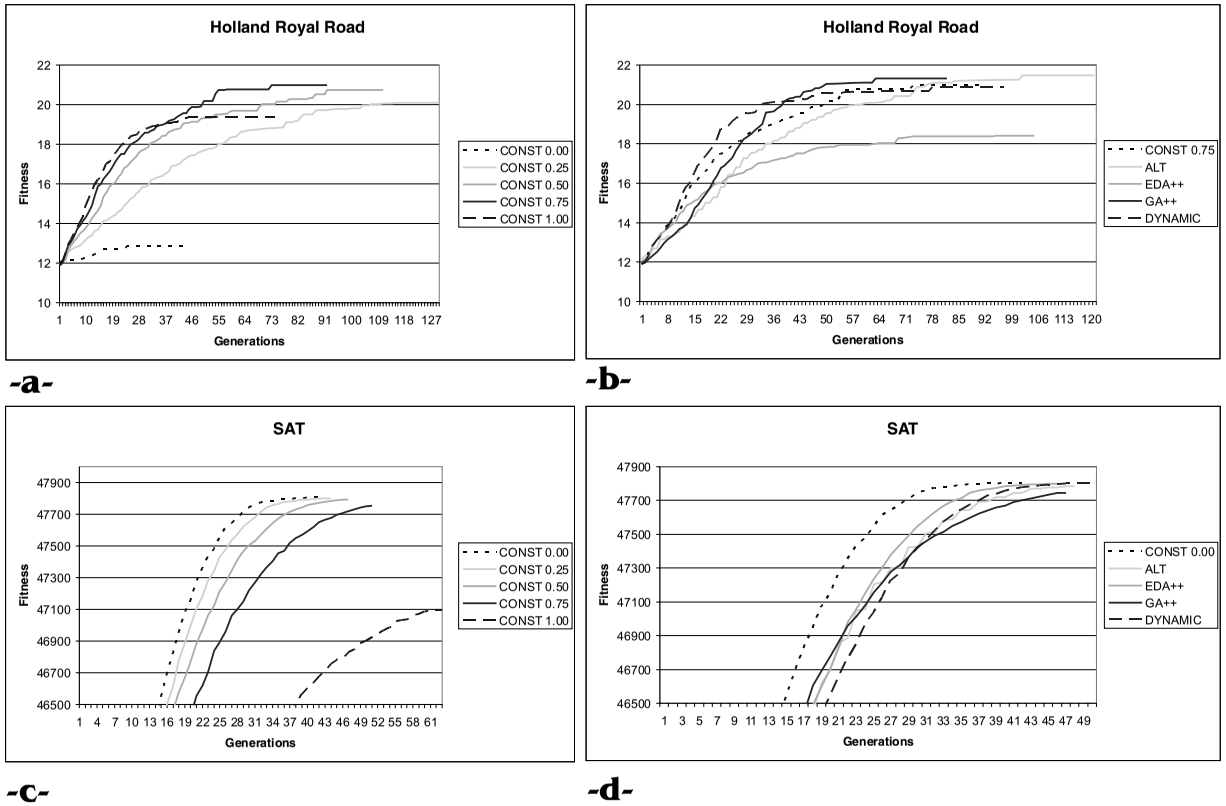

Fig. 5. Holland Royal Road solved with -**a**- constant Participation Functions and -**b**- variable Participation Functions and SAT problem solved with -**c**- constant Participation Functions and -**d**- variable Participation Functions

$$f_{SAT}(\boldsymbol{x}) = \frac{1}{C}\sum_{c=1}^{C} s(x_c)$$

$$s(x_c) = \begin{cases} 1 & \text{if the clause } c \text{ is satisfied} \\ 0 & \text{in other case} \end{cases}$$

In previous equation C denotes the number of clauses that the formula has. To test the developed algorithm, the SAT instances *4blocksb.cnf* was used since they are widely-known and easily available from the SATLIB benchmark[4]. *4blocksb.cnf* contains 24758 clauses, 410 propositional variables and is satisfiable.

Results

SAT problem is one of the best scenarios for EDAs (Fig. 5c) by getting the top results using less number of generations. The difference, when compared with GAs, are very significant with a p-value < 0.001. This problem has been solved with a population size of 1000 individuals.

A very interesting issue is that as the constant ratio varies the progression of the algorithm seems to be the same with a gap between each of the graphs. This means that the lack of accuracy shown by the GA components is located on the earliest stages of the evolutionary process (the first iterations). The evolution curves are quite similar after these first generations.

None of the constant or variable Participation Functions, see Figs. 5c and 5d, reach the same fitness value as pure EDAs. Although their performance is not as bad as GAs, their p-values are in all the cases smaller than 0.001.

Dynamic Participation Function goes quite slowly on the first generation. This could drive to a new definition of this Dynamic Participation Function, with more aggressive behavior in early generations and more conservative changes later.

5 Continuous Problems

The other part of the experiments have been performed considering ten continuous problems:

① Five well-known continuous optimization problems: *Branin* RCOS function, *Griewank* function, *Rastrigin* function, *Rosenbrock* function and *Schwefel's* problem. [16, 41]
② A new synthetic problem has been also defined (proportional Participation Function).
③ A continuous version of the *MaxBit* problem.
④ A real-coded solution for three different TSP problems.

[4] http://www.satlib.org/benchm.html

5.1 Branin RCOS Function

Definition

Results

This problem is a two-variable continuous problem with three global minimum and no local minimum. The problem is defined as follows [5]:

$$f_B(x_1, x_2) = \left(x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$
$$-5 < x_1 < 10$$
$$0 < x_2 < 15$$

The global optimum for this problem is 0.397887 that is reached in the points $(x_1, x_2) = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)$.

This problem is considered easy not only because of the number of variables, but the small chance to miss the basin of the global minimum in a global optimization procedure. This is due to the probability of reaching the global optimum using local optimization methods, started with a small number of random points, is quite high.

This problem was solved using a population size of 150 individuals.

Branin is a very simple problem where in few generations all the algorithms converge. Figure 6 shows CONST 0.25 is the best function, and GAs a very poor option to solve this problem. CONST 0.25 aheads EDA with p-value = 0.063 and CONST 0.50 with p-value = 0.339.

In this problem GA++ performs similarly to EDA. This is due to the reduced number of generations which represents the participation share of GAs which do not increase too fast to recover the majority of individuals generated by the EDA mechanism. See Fig. 6b.

The simplicity of this function biases the performance of the algorithm towards the trend addressed by the very first generations. Few modifications are achieved after these generations. For the DYNAMIC function, this could be a good justification to define a more radical variations of the first iterations of the algorithm.

5.2 Griewank Function

Definition

This problem has ten variables with a unique global optimum with many ($O(10^3)$) local minima nearby.

$$f_{G10}(\boldsymbol{x}) = 1 + \frac{1}{d}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$$
$$d = 4000; n = 10$$
$$-500 < x_i < 500$$

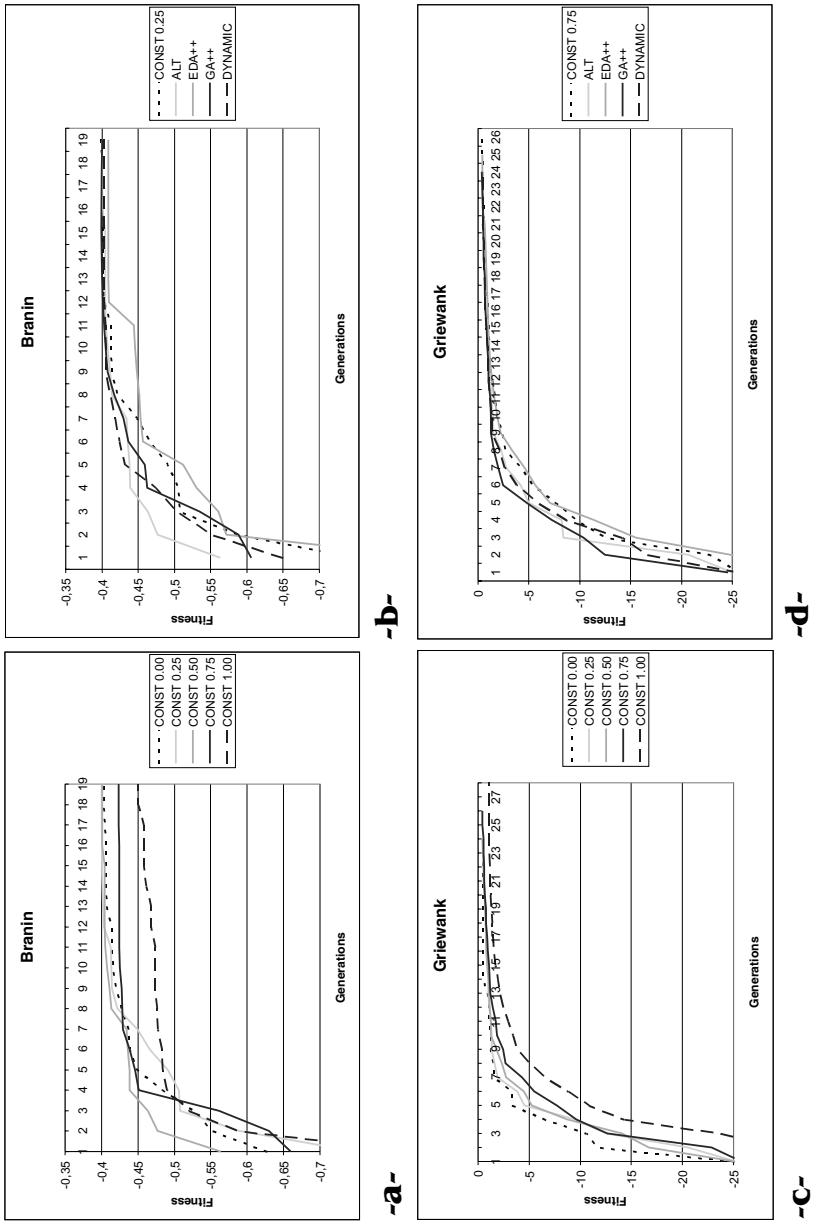

Fig. 6. Branin function solved with **-a**- constant Participation Functions and **-b**- variable Participation Functions and Griewank function solved with **-c**- constant Participation Functions and **-d**- variable Participation Functions

The global minimum for this problem can be found in the solution $x_i = 0, i = 1, \ldots, n$ with a fitness value of 0.

It is considered a moderately difficult optimization problem, because of its non-separable characteristic. Non-separable means that there is non-linear interaction among variables.

Results

This problem was solved using a population size of 250 individuals.

In this problem the results achieved by EDAs are better than the ones provided by GAs with p-value < 0.001. This feature is emphasized when it is shown even a small participation ratio of EDAs increases the fitness obtained by the overall algorithm, as shown by Fig. 6c.

In Fig. 6d, a very similar performance of the variable Participation Functions can be seen. There is almost no significant difference on any of the variable Participation Function. For instance, between CONST 0.75 and EDA++ we have p-value = 0.853 and between CONST 0.75 and DYNAMIC we have p-value = 0.795.

It is interesting that this problem shows an interaction between variables which could lead EDA evolution, when using UMDA, to suboptimal values, with higher probability than GAs. Actually the dependence factor is not very significant as the sum of the quadratic terms of the first part of the function, which is lineal-independent, is much more significant than the product of values between $[-1, 1]$.

5.3 Rastrigin Function

Definition

It is a scalable, continuous, and multimodal function that must be minimized. It is the result of modulating n-dimensional sphere equation with $a \cdot \cos(\omega x_i)$.

$$f_{Ra5}(\boldsymbol{x}) = a \cdot n + \sum_{i=1}^{n} \left(x_i^2 - a \cdot \cos(\omega x_i)\right)$$
$$a = 10; \omega = 2\pi; n = 5$$
$$-5.12 < x_i < 5.12$$

The global minimum for this problem can be found in the solution $x_i = 0, i = 1, \ldots, n$ with a fitness value of 0.

Results

This problem was solved using a population size of 1000 individuals.

Rastrigin (Fig. 7a) function has no lineal dependency among the variables, but the performance of EDAs is very poor. Near the optimum value there are many local optimum and EDAs seems to be very sensitive to this characteristic. The best

constant ratio function is CONST 0.50, with a p-value < 0.001 respect to GA. This means that pure GA could be improved by the help of EDAs even if this method is not very well-suited by itself.

In Fig. 7b, we can see that DYNAMIC is able to provide the best participation ratio to outperform CONST 0.50 with a p-value = 0.006 and the other variable Participation Functions.

5.4 Rosenbrock Function

Definition

It is a continuous, non-separable, and unimodal function. It has the global minimum located in a steep parabolic valley with a flat bottom [37]. This issue represents a big challenge to the optimization process.

$$f_{Ro10}(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left(100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$$
$$n = 10$$
$$-500 < x_i < 500$$

The global minimum for this problem can be found in the solution $x_i = 0, i = 1, \ldots, n$ with a fitness value of 0.

Results

This problem was solved using a population size of 1000 individuals.

In Fig. 7c, EDAs perform much better than the other approaches, and GAs are far worse than any Participation Function with at least a small EDA ratio.

Variable Participation Functions on this problem are not better than pure EDAs. It is very significant, as shown by Figure 7d, that performance of GA++ is much better than EDA++. This is due to the small number of generations that represents a more intensive participation of EDAs.

DYNAMIC does not perform very well. This can be explained in the same terms previously used. In problems with few generations our Dynamic Participation Functions has no time to balance the participation ratios of the algorithms.

5.5 Schwefel's Problem

Definition

It is also a continuous unimodal function. Its difficulty also concerns the fact of searching along the coordinate axes only gives a poor rate of convergence because

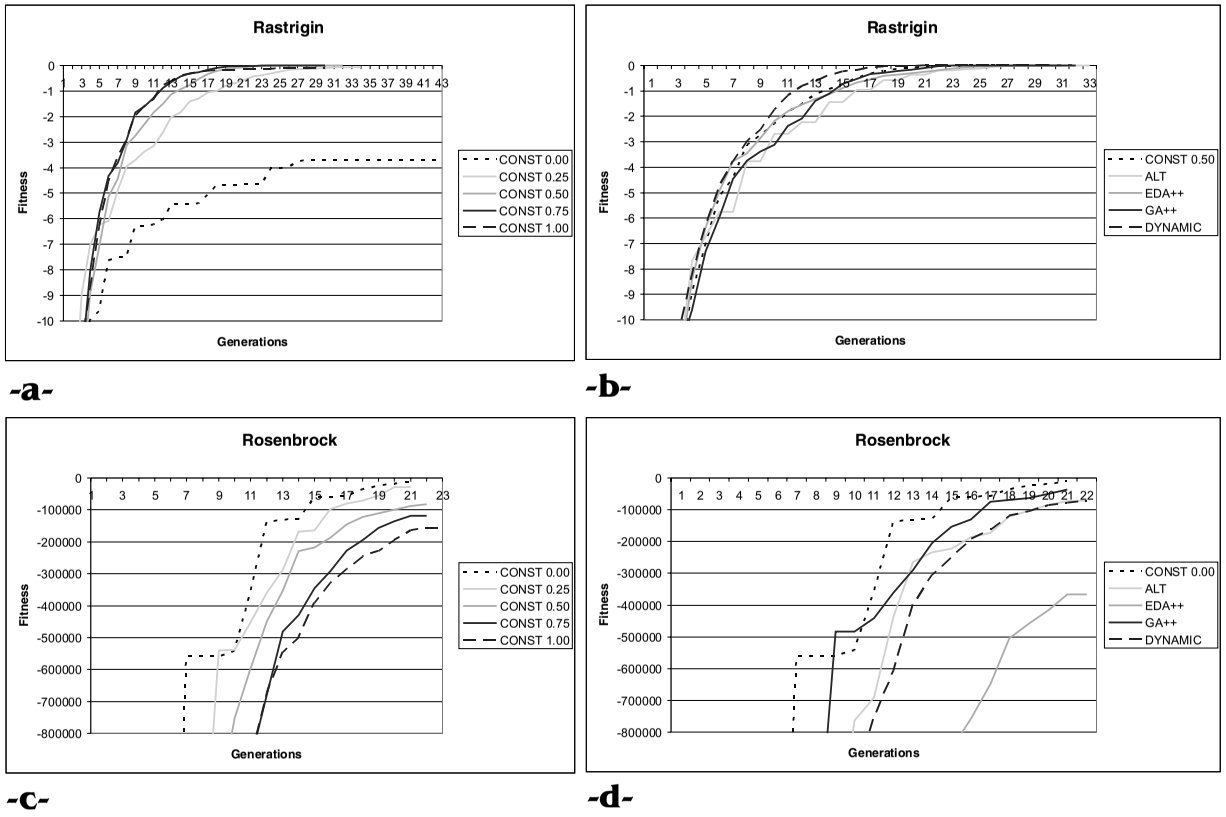

Fig. 7. Rastrigin function solved with -**a**- constant Participation Functions and -**b**- variable Participation Functions and Rosenbrock function solved with -**c**- constant Participation Functions and -**d**- variable Participation Functions

function gradient is not oriented along the axes. As in the previous case global optimum is surrounded by several local optimum in the neighborhood.

$$f_{S10}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i \cdot \sin(\sqrt{|x_i|})$$
$$n = 10$$
$$-500 < x_i < 500$$
$$f_{S10}(\boldsymbol{x}^*) = \min(f_{S10}(\boldsymbol{x}))$$

The global minimum for this problem can be found in the solution $x_i = 420.9687, i = 1, \ldots, n$ with a fitness value of 0.

Results

This problem has been solved with a population of 250 individuals.

Schwefel's problem is very difficult due to the large number of suboptimal points, especially those near the global optimum. This feature drives EDAs to a very poor performance, also due to the non-lineal relationships among the variables.

In Fig. 8a GAs is the best approach, much better than any other of the constant ratio functions, although they are not able to find the optimal value in all of the cases.

All the results achieved by other than pure GA algorithms are not able to improve the results reached after the first generations. Even GA++, which increments the ratio of GA-based individuals, are very poor on this problem (see Fig. 8b). Many studies have proved that significantly high mutation rates could help improve the results of this problem.

5.6 Proportion Problem

Definition

This new function represents a model of similar real-world problems that deal with the search of the correct proportions that should make it true that:

$$\sum_{i=1}^{n} x_i = 1 \tag{3}$$

The fitness function is:

$$f_{P128}(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \left(1 - |x_i - x_i^*|^{1/p}\right)$$
$$n = 128; p = 2; x_i^* = \frac{i}{\frac{n \cdot (n+1)}{2}}$$
$$0 \leq x_i \leq 1$$
$$f_{P128}(\boldsymbol{x}^*) = \max(f_{P128}(\boldsymbol{x}))$$

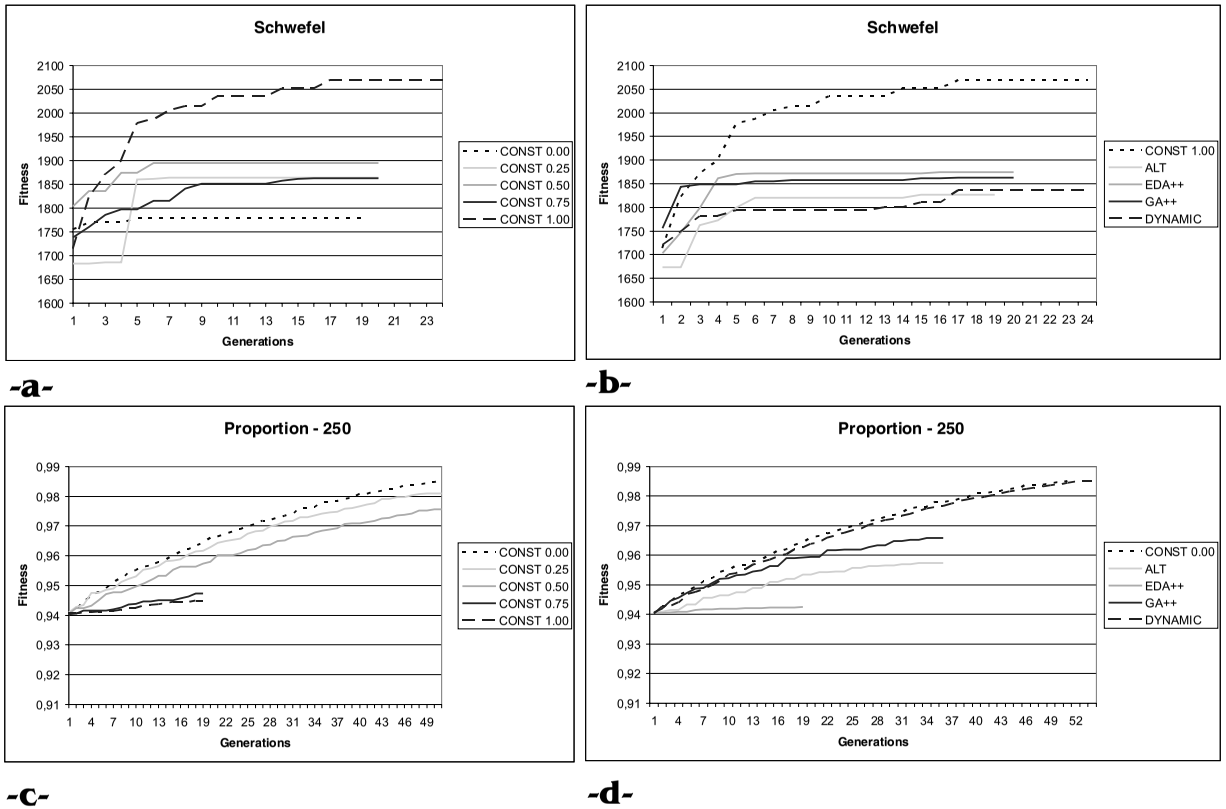

Fig. 8. Schwefel function solved with -**a**- constant Participation Functions and -**b**- variable Participation Functions and Proportion problem solved with -**c**- constant Participation Functions and -**d**- variable Participation Functions

The objective of the function is to find the right proportion (represented by x^*), with p-order distance function.

This problem is not difficult, as the distance function is separable and lineal. In order to be compliant with the restriction expressed by Eq. 3 a Lamarckian correction is performed to the individual represented by $\boldsymbol{x}$, instead of dropping malformed elements.

$$x_i^c = \frac{x_i}{\sum_{i=1}^{n} x_i} \tag{4}$$

The individual $\boldsymbol{x^c}$ substitutes the individual $\boldsymbol{x}$ in the population, before fitness calculation is performed.

Results

This problem has been solved with a population of 250 individuals.

EDAs deal with this problem much better than any other constant functions. Between EDAs and CONST 0.25, the best constant function, the p-value is p-value $<$ 0.001.

DYNAMIC slightly outperforms EDAs (with p-value = 0.185) and seems to adapt perfectly to the characteristic of this problem (see Fig. 8d). EDA++ converges prematurely due to the heavy ratio of the GA-based individuals.

5.7 The MaxBit Continuous Problem

Definition

This problem is a redefinition of the binary MaxBit problem previously presented. The aim is to maximize:

$$f_{M12}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} x_i}{n}$$
$$x_i \in \{0, 1\}; n = 12$$

In the continuous domain this problem is more complex, as the optimum value of the function is located on the boundary of the search space.

Results

This problem has been solved with a population of 250 individuals.

Figures 9a and 9b contain the obtained results. All the constant and variable Participation Functions perform in a very similar way reaching the global optimum in almost the same number of generations. However, pure GAs, which converge after more generation, only reach a suboptimal value.

MaxBit performance, as we can see, is very similar in both cases with continuous and with bit-string individuals.

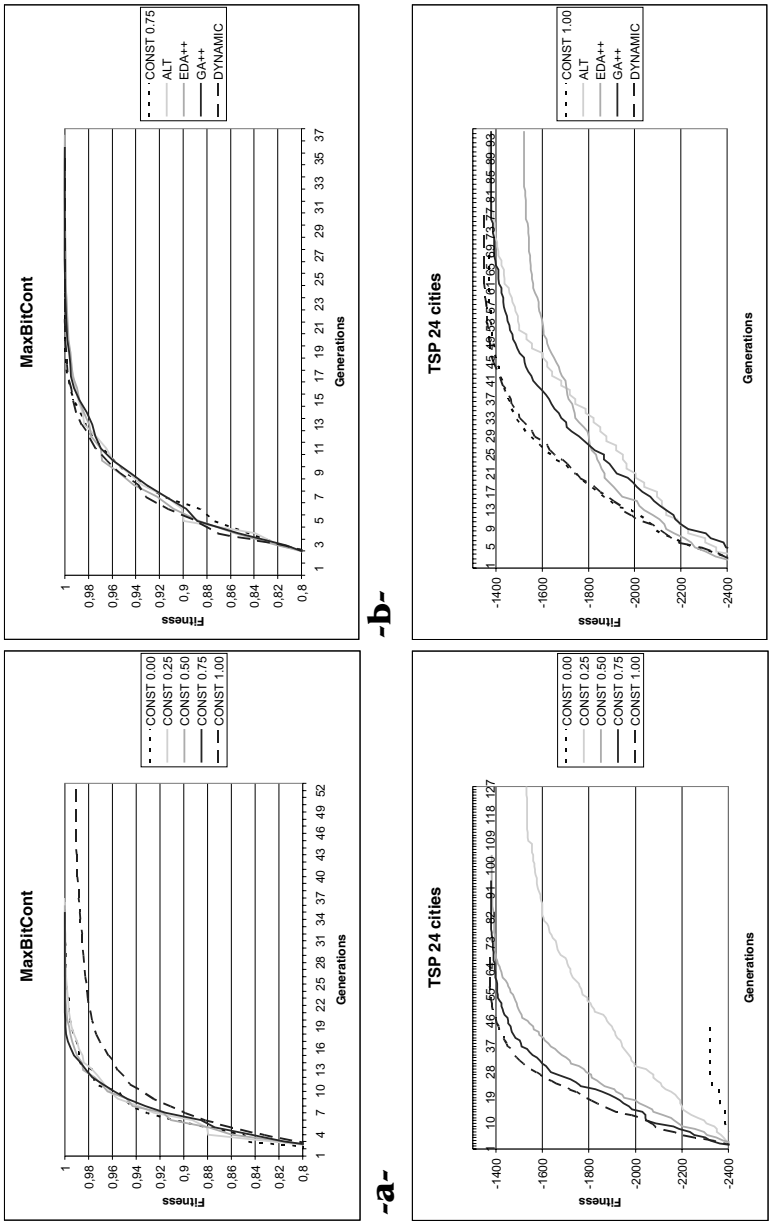

Fig. 9. MaxBit continuous problems solved with **-a-** constant Participation Functions and **-b-** variable Participation Functions and TSP problem with 24 cities solved with **-c-** constant Participation Functions and **-d-** variable Participation Functions

5.8 TSP Continuous

Definition

The Travelling Salesman Problem (TSP) objective is to find the shortest route for a travelling salesman who, starting from his home city, has to visit every city on a given list precisely once and them return to his home city. The main difficult of this problem is the immense number of possible tours: $(n-1)!/2$ for n cities.

The TSP is a relatively old problem. It was documented as early as 1759 by Euler, however not using that name, whose interest was in solving the knights' tour problem in chess. A correct solution would have a knight visit each of the 64 squares of a chessboard exactly once on its tour. The term "traveling salesman" was first used in 1932, in a German book written by a veteran traveling salesman. The RAND CORPORATION introduced the TSP in 1948. The corporation's reputation helped to make the TSP a well-known and popular problem.

Although there are different alternatives to encode this problem, in this paper individuals of population are represented by using vectors with real numbers. Thus, we need a method to translate these real vectors to a valid tour for the TSP. In the following table we see one of these translations.

In Table 2 we can see a 6-city example. In the original vector the generated real numbers are between 3 and -3. The obtained tour will be an integer vector in which each of the elements is the index after the values of the original vector are sorted. Thus, the calculus of the fitness function of individuals is more complex to compute.

Table 2. Translation of an individual to a correct tour

Original vector:	1.34	2.14	0.17	0.05	–1.23	2.18
Resulting tour:	4	5	3	2	1	6

The following files have been used in the empirical study: The well known Gröstel24, Gröstel48 and Gröstel120. These are files that can be obtained via web or ftp in many sites. They represent the distances between 24, 48 and 120 imaginary cities. They are often used in TSP problems to know the fitness of the algorithm we use, and can be defined as a classical experiment in the TSP.

Results for Gröstel24 Problem

This problem has been solved with a population of 1000 individuals.

In the TSP problem of 24 cities (see Fig. 9c) the best algorithms are pure GA, CONSTANT 0.75 and CONSTANT 0.50 (p-values > 0.9 between them). The worst algorithm is EDA which presents a very poor performance.

In the variable Participation Functions (see Fig. 9d) obtained results are excellent, being the DYNAMIC approach being better than GA with a p-value = 0.161.

Although it has a very good beginning because of the number of GA individuals created in the first generations, the EDA++ approach presents bad results.

Results for Gröstel48 Problem

This problem has been solved with a population of 1500 individuals.

The results obtained for 48 cities with the constant Participation Functions (see Fig. 10a) are very similar to the previous ones, CONSTANT 0.75 and pure GAs being the best constant approaches, without statistical significance difference between them.

However, in this case, in the variable Participation Functions (Fig. 10b), GA++ is similar to the DYNAMIC approach, with p-value = 0.722, but worse than CONST 0.75, with p-value = 0.147.

Results for Gröstel120 Problem

This problem has been solved with a population of 1500 individuals.

TSP with 120 cities is a very hard problem for heuristic optimization approaches such as GAs and EDAs without the help of local optimization techniques. However, the obtained results are quite similar to the previous ones, GA being the best approach with constant Participation Function, and GA++ the best with variable Participation Function, without significant difference respect to GA (p-value = 0.7393).

6 Conclusion and Further Work

In this chapter we have proposed a new hybrid algorithm based on genetic and estimation of distribution algorithms. This new algorithm has been tested on a set of different problems. Although the hybrid algorithm proposed is composed by the simplest versions of both GA and EDA components, the experimentation shows it is really promising and competitive. In most of the experiments we reach the best of the values found by GAs or EDAs or we even improve them. There is still a lot of further future work. Here are some possibilities: Extend the implementation to support more sophisticated individual representations, make new Participation Functions based on statistical tests, implement a parallel version based on the island model or use more complex GAs and EDAs in the hybrid solution.

6.1 Evolution of the Dynamic Participation Function

One interesting issue is to survey the evolution of the dynamic Participation Function in the series of different experiments. This function, as we have seen, adjusts the participation ratio depending on the quality of the individuals each of the methods is providing. Indirectly, this measure could be used to evaluate the quality of each of the methods across the continuous generations of an algorithm.

As we see in Fig. 11 the evolution of the dynamic functions are able to guide the hybrid algorithms towards the best option, either GA, EDA or other constant ratio Participation Functions. For example, in TSP, GAs outperforms clearly all the other

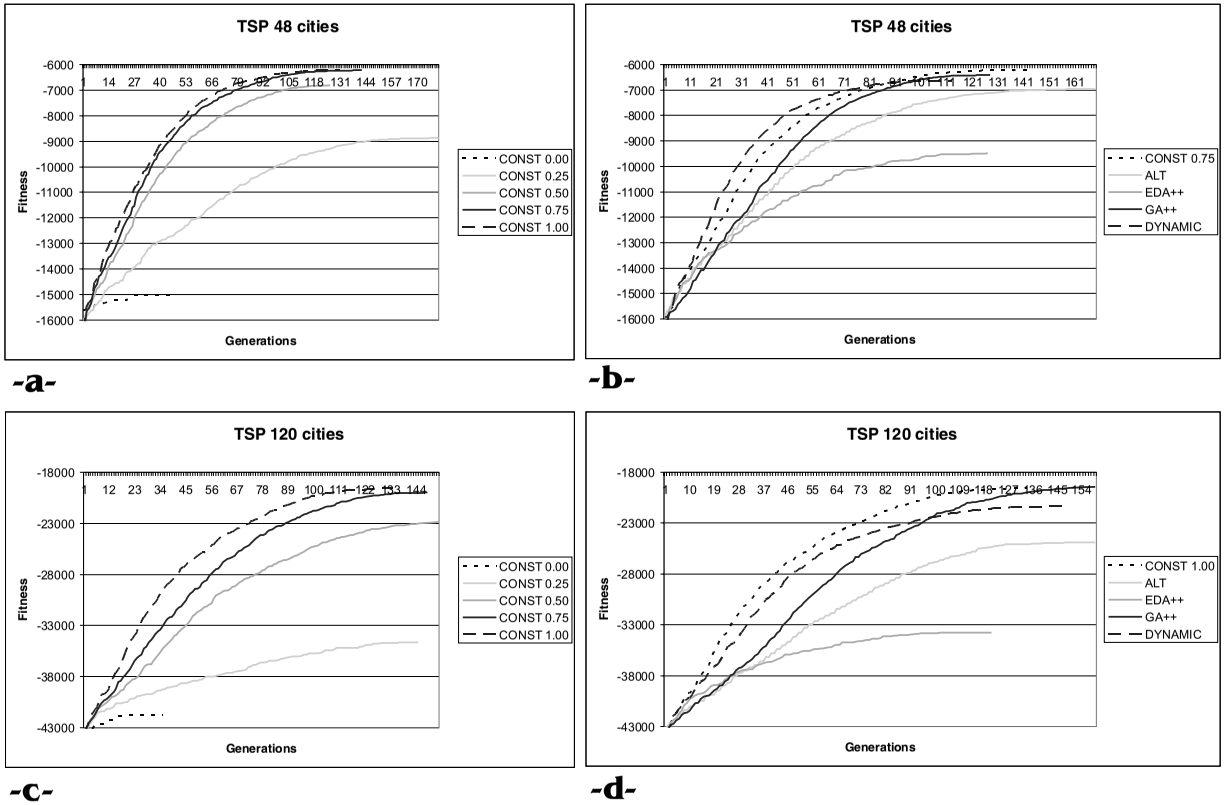

Fig. 10. TSP problems solved with -**a**- constant Participation Functions and -**b**- variable Participation Functions and TSP problem with 24 cities solved with -**c**- constant Participation Functions and -**d**- variable Participation Functions

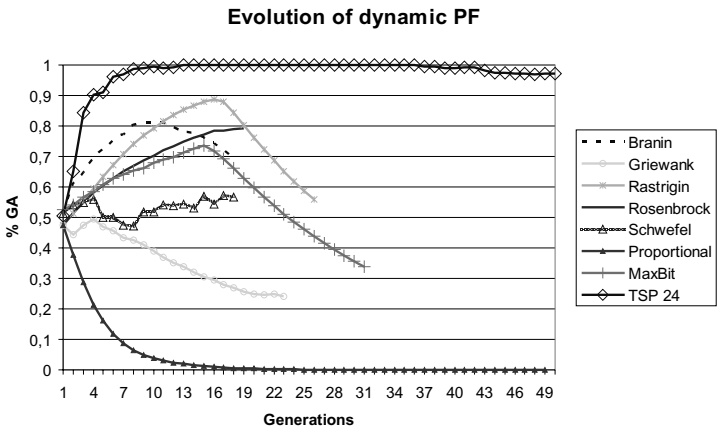

Fig. 11. Evolution of Dynamic Participation Function (Continuous)

constant ratio approaches, and DYNAMIC (with similar results) changes the participation ratio in this direction. A similar case is shown by Proportion. However, in this case EDAs are the best algorithm. In other problems, with a best option between pure EDAs and pure GAs, the Participation Function moves to find the correct balance in order to improve the results of the overall algorithm.

On the other hand, for bit-string problems in Fig. 12, the dynamic Participation Function has this general trend:

- On early generations GA performs better than EDAs, the exploratory technique is able to find better individuals using the same input information (the last generation).

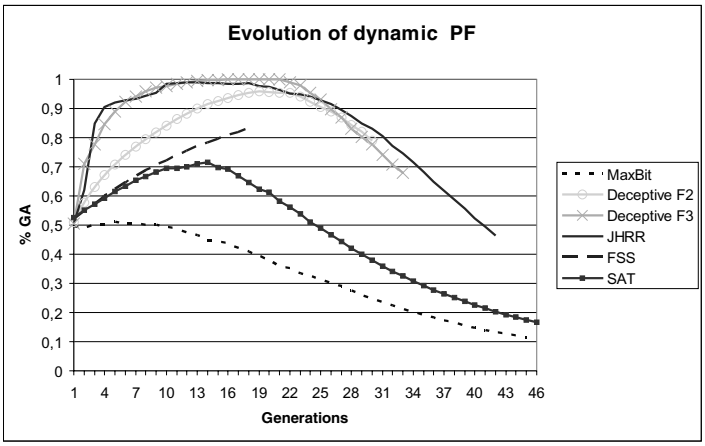

Fig. 12. Evolution of Dynamic Participation Function (Bit-string)

- When the algorithm is close to the optimum, EDAs generate the best solutions. That is probably due to the mutation ratio, which is very useful to avoid local optimum, but once the environment of the global solution is reached it drives towards malformed individuals, far from the local optimum.

This trend is also shown, with small variations, by the continuous problems shown before, in the case neither pure GAs nor pure EDAs are clear options. Although special abnormalities are present, for example Rosenbrock is best solved by EDAs, but DYNAMIC trend increases the participation ratio of GAs instead.

These trends of the fitness, provided by each of the methods, could be useful in order to tune up either genetic and estimation of distribution algorithms by themselves. Updating mutation rate is one of the issues already considered by works such as [2, 40].

6.2 Experiments Summary

On Tables 3 and 4, the summary of the results obtained by these experiments show that hybrid algorithms, in most of the cases, are a better option than the pure EDA or GA algorithms by themselves. Although there are specific problems in which EDA (SAT, Proportion and Rosenbrock) or GA (FSS and Schwefel) are the best options, hybrid algorithms show a competitive behavior. The opposite is not as common, as there are experiments (like JHRR, Branin, Griewank, Rastrigin, and MaxBit Continuous) in which neither EDAs nor GAs present good results compared to most of the hybrid approaches.

Table 3. Result Summary Table

	EDA		*CONST 0.25*		*CONST 0.50*		*CONST 0.75*		*GA*	
Problem	Mean	Gen	Mean	Gen	Mean	Gen	Mean	Gen	Mean	Gen
Max Bit	1	45	1	48	1	51	1	59	0,9793	115
Deceptive f2	280,0	28	289,8	62	296,4	45	298,2	42	297,6	38
Deceptive f3	272,4	68	296,5	74	298,6	51	298,3	41	298,2	40
FSS	0,8041	20	0,8059	22	0,8057	20	0,8057	20	0,8070	27
JHRR	12,85	43	20,09	130	20,75	111	20,99	92	19,37	74
SAT	47803,6	42	47800,1	44	47790,9	47	47752,7	51	47096,5	63
Branin	–0,4035	19	–0,3999	19	–0,4006	19	–0,4235	19	–0,4513	19
Griewank	626,38	24	626,45	25	626,51	25	626,42	25	625,45	32
Rastrigin	–3,69683	43	–0,06368	34	–0,00054	32	–0,00473	30	–0,10823	30
Rosenbrock	–12403	21	–29197	21	–81222	22	–118522	22	–157107	23
Schwefel	1778,36	19	1863,85	20	1894,66	20	1862,60	20	2068,70	24
Proportion	0,9851	51	0,9810	51	0,9756	51	0,9473	19	0,9449	19
MaxBit Cont	0,9999	37	0,9999	37	1	36	1	35	0,9909	53
TSP 24	–2324,7	45	–1531,3	128	–1381,3	104	–1378,9	95	–1372,2	69
TSP 48	–15037,2	45	–8873	181	–6814,2	126	–6222,8	142	–6227,3	131
TSP 120	–41809,1	35	–34666,9	145	–22859,9	154	–19983,7	149	–19640,5	134

Table 4. Result Summary Table

	ALT		*GA++*		*EDA++*		*DYNAMIC*	
Problem	Mean	Gen	Mean	Gen	Mean	Gen	Mean	Gen
Max Bit	1	52	1	56	1	51	1	50
Deceptive f2	296,8	55	292,4	49	296,6	48	298,2	42
Deceptive f3	298,0	55	297,7	52	297,3	58	299,2	45
FSS	0,8057	21	0,8060	22	0,8059	21	0,8065	21
JHRR	21,48	121	21,33	82	18,40	105	20,88	97
SAT	47783,0	48	47745,9	47	47799,4	46	47798,7	50
Branin	–0,4006	19	–0,4001	19	–0,4085	19	–0,4033	19
Griewank	626,42	25	626,47	25	626,49	26	626,42	25
Rastrigin	–0,00159	33	–0,00027	32	–0,00472	31	–0,00005	30
Rosenbrock	–70688	22	–36544	21	–366490	22	–73786	22
Schwefel	1826,47	19	1863,35	20	1874,48	20	1836,26	24
Proportion	0,9574	36	0,9659	36	0,9426	19	0,9851	54
MaxBit Cont	1	36	1	36	0,9999	37	1	35
TSP 24	–1371,6	93	–1378	95	–1522,5	95	–1351,3	78
TSP 48	–6962,8	169	–6413,1	128	–9484,5	127	–6641,9	113
TSP 120	–24875,1	159	–19457,4	159	–33759,5	121	–21327,8	147

In order to compare the results, for the experiments carried out, the relative position (ranked-based) has been computed. This ranking has been developed using fitness-driven criteria. The best fitness is #1, next one #2, and so on. Using this method, the average ranking has also been computed:

$$avg\_rank(PF) = \frac{\sum_{i=1}^{N} rank(PF, i)}{N} \tag{5}$$

being $rank(PF, P)$ the relative ranking of Participation Function PF in the problem P.

DYNAMIC is the best Participation Function, as can be seen on Table 5. Another interesting result is that CONST 0.50 also behaves quite well. Among the worst results are both pure EDAs and pure GAs. Of course, the set of experiments is not representative of all the possible optimization problems, but have been selected to cover a wide spectrum of possible real-world scenarios. In the performance of EDA++ and GA++, we should consider that several experiments do not last for many generations, thus the influence of the first generations biases the results achieved by these approaches.

The experimentation and research perspective of hybrid methods is very promising, and several issues are still open in terms of alternatives of the presented Participation Functions, using of more complex EDAs approaches, hybridization with local heuristics, three or multi hybrid algorithms (using more than one GA or one EDA algorithm), and parallel definition of the hybrid algorithms.

Table 5. Average ranking of the Participation Functions

Participation Function	avg_rank
EDA	6,625
CONST 0.25	4,9375
CONST 0.50	3,8125
CONST 0.75	4,4375
GA	5,4375
ALT	4,25
GA++	3,5
EDA++	5,6875
DYNAMIC	3,0625

References

1. V. Bachelet and E. Talbi. Cosearch: A co-evolutionary metaheuritics. In *Proceedings of Congress on Evolutionary Computation – CEC2000*, pp. 1550–1557, 2000.
2. T. Back and M. Schutz. Intelligent mutation rate control in canonical genetic algorithms. In *International Syposium on Methodologies for Intelligent Systems*, 1996.
3. T.C. Belding. The distributed genetic algorithm revisited. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 114–121, 1995.
4. C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, September 2003.
5. F.K. Branin. A widely convergent method for finding multiple solutions of simultaneous nonlinear equations. *IBM Journal of Research and Development*, pp. 504–522, 1972.
6. E. Cantú-Paz. *Efficient and accurate parallel genetic algorithms*. Kluwer Academic Publisher, 2001.
7. K. Deb and D.E. Golberg. Analyzing deception in trap functions. In L.D. Withley, editor, *Foundation of Genetic Algorithms 2*, pp. 93–108, San Mateo, CA, 1993. Morgan Kaufmann.
8. J. Denzinger and T. Offerman. On cooperation between evolutionary algorithms and other search paradigms. In *Proceedings of Congress on Evolutionary Computation – CEC1999*, pp. 2317–2324, 1999.
9. R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.
10. F. Foccaci, F. Laburthe, and A. Lodi. Local search and constraint programming. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research and Management Science*, Kluwer Academic Publishers, Norwell, MA, 2003.
11. E.C. Freuder, R. Dechter, M.L. Ginsberg, B. Selman, and E.P.K. Tsang. Systematic versus stochastic constraint satisfaction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI'1995*, pp. 2027–2032. Morgan-Kaufmann, 1995.
12. F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
13. D.J. Hand and K. Yu. Idiot's Bayes – not so stupid after all? *International Statistical Review*, 69(3):385–398, 2001.
14. J. Hao, F. Lardeux, and F. Saubion. A hybrid genetic algorithm for the satisfiability problem. In *Proceedings of the First International Workshop on Heuristics*, Beijing, 2002.

15. W.D. Harvey. *Nonsystematic backtracking search*. PhD thesis, University of Oregon, 1995.
16. F. Herrera and M. Lozano. Gradual distributed real-coded genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1), 2000.
17. T. Hogg and C. Williams. Solving the really hard problems with cooperative search. In *Proceedings of AAAI'1993*, pp. 213–235. AAAI Press, 1993.
18. J.H. Holland. *Adaption in natural and artificial systems*. The University of Michigan Press, Ann Harbor, MI, 1975.
19. J.H Holland. Royal road functions. *Internet Genetic Algorithms Digest*, 7(22), 1993.
20. I. Inza, P. Larrañaga, and B. Sierra R. Etxeberria. Feature subset selection by Bayesian networks based optimization. *Artificial Intelligence*, 123(1–2):157–184, 2000.
21. S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.
22. P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Optimization in continuous domains by learning and simulation of Gaussian networks. In A. S. Wu, editor, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*, pp. 201–204, 2000.
23. P. Larrañaga and J.A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publisher, 2002.
24. D. Levine. *A Parallel Genetic Algorithm for the Set Partitioning Problem*. PhD thesis, Illinois Institute of Technology, Mathematics and Computer Science Division, Argonne National Laboratory, 1994.
25. F.T. Lin, , C.Y. Kao, and C.C. Hsu. Incorporating genetic algorithms into simulated annealing. *Proceedings of the Fourth International Symposium on Artificial Intelligence*, pp. 290–297, 1991.
26. H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publisher, 1998.
27. S.W. Mahfoud and D.E. Golberg. Parallel recombinative simulated annealing: A genetic algorithm. *Parallel computing*, 21:1–28, 1995.
28. O.C. Martin and S.W. Otto. Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63:57–75, 1996.
29. M. Mitchell, S. Forrest, and J.H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In *Proceedings of the First European Conference on Artificial Life*, Cambridge, 1992. MIT Press/Bradford Books.
30. H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5:303–346, 1998.
31. P.M. Murphy and D.W. Aha. UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/, 1995.
32. C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.
33. J.M. Peña, V. Robles, P. Larrañaga, V. Herves, F. Rosales, and M.S. Pérez. GA-EDA: Hybrid evolutionary algorithm using genetic and estimation of distribution algorithms. In *Innovations in Applied Artificial Intelligence: Proceeding of IEA/AIE 2004*, 2004.
34. V. Robles. *Clasificación supervisada basada en redes Bayesianas. Aplicación en biología computacional*. PhD thesis, Universidad Politécnica de Madrid, Departamento de Arquitectura de Computadores, Junio, 2003.
35. V. Robles, M.S. Pérez, V. Herves, J.M. Peña, and P. Larrañaga. Parallel stochastic search for protein secondary structure prediction. In *Lecture Notes in Computer Science 2724*, pp. 1162-1170, Czestochowa, Poland. 5th International Conference on Parallel Processing and Applied Mathematics, 2003.

36. E. Rodriguez-Tello and J. Torres-Jimenez. ERA: An algorithm for reducing the epistasis of SAT problems. Number 2724 in Lecture Notes in Computer Science, pp. 1283–1294. GECCO 2003, Genetic and Evolutionary Computation Conference, Springer Verlag, 2003.
37. H.H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *Computational Journal*, pp. 175–184, 1960.
38. L. Sondergeld and S. Voß. Cooperative intelligent search using adaptive memory techniques. In I. Osman S. Voß, S. Martello and C. Roucairol, editors, *In MetaHeuristics: Advances and Trends in Local Search Paradigms for Optimization*, chapter 21, pp. 297–312. Kluwer Academic Publishers, 1999.
39. E-G. Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–564, 2002.
40. D. Thierens. Adaptive mutation rate control schemes in genetic algorithms. In *Proceedings of the 2002 IEEE World Congress on Computational Intelligence: Congress on Evolutionary Computation*, 2002.
41. A. Törn, M. M. Ali, and S. Viitanen. Stochastic global optimization: Problem classes and solution techniques. *Journal of Global Optimization*, 14:437-47, 1999.
42. M. Toulouse, T. Crainic, and B. Sansó. An experimental study of the systemic behavior of cooperative search algorithms. In I. Osman S. Voß, S. Martello and C. Roucairol, editors, *In Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, chapter 26, pp. 373–392. Kluwer Academic Publishers, 1999.
43. M. Toulouse, K. Thulasiraman, and F. Glover. Multi-level cooperative search: A new paradigm for combinatorial optimization and application to graph partitioning. In *Proceedings of the 5th International Euro-Par Conference on Parallel Processing. Lecture Notes in Computer Science. Springer-Verlag*, pp. 533–542, New York, 1999.
44. D. Withley and T. Starkweather. Genitor II: a distributed genetic algorithm. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:189–214, 1990.
45. Q. Zhang, J. Sun, E. Tsang, and J. Ford. Combination of guided local search and estimation of distribution algorithm for quadratic assignment problems. In *Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference, in GECCO 2003*, pp. 42–48, 2003.