Multi-PBil: An Estimation Distribution Algorithm Applied to Multimodal Optimization Problems.

Rubens Barbosa Filho
Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul – UEMS.
Dourados – Mato Grosso do Sul – Brasil
rubens@uems.br

Abstract

The Estimation Distribution Algorithms (EDAs) compose an evolutionary metaheuristic whose main characteristic is the construction of solutions in randomly form, using a distribution of probabilities that evolves during the execution. The Population-Based Incremental Learning Algorithm (PBIL) is a type of EDA where the variables are independent, that is, they do not have significant interactions between themselves. The PBIL considers that the solutions can be represented as vectors of discrete variables, what makes it more adequate for combinatorial optimization problems. This paper presents a method called Multi-PBil that is an extension of PBIL with applications in multimodal problems. The Multi-PBil was developed with the goal to have an efficient and non expensive algorithm of search in multimodal spaces. From PBIL, it was implemented a routine that allows the Multi-PBil to create a probability model to act in the search space. A formula that allows initiating the probability models in regions of the search space next to the searched global points was applied in the process of the probability model initialization rule. The Multi-PBil method was tested and analyzed, presenting some experimental results that highlight its viability and characteristics. It is also shown a comparison of the performance between the Multi-PBil and a traditional Genetic Algorithm using the sharing method.

Keywords: Artificial Intelligence, Evolutionary Computation, Genetic Algorithms, Estimation Distribution Algorithms.

1 Introduction

The Evolutionary Computation (EC) relates to a great class of optimization algorithms, which are inspired in the evolutionary systems of the natural world [1]. Generally, each algorithm starts with a population of individuals that will multiply and compete for their survival [2]. Each individual has fitness value that indicates its potential inside the population. Between the techniques that are part of the evolutionary computation is [3]: Evolutionary programming (EP), Evolutionary Strategies (ES), Genetic Algorithms (GA) and Genetic Programming (GP).

Genetic algorithms are random optimization techniques that imitate the Darwinian Evolution through the process of modeling of natural selection and the genetic modifiers [4]. They are based on principles of genetic biology and operate analogous to the theory of evolution [5]. The genetic algorithms have been shown viable in a variety of unimodal domains [6]. In an environment of natural evolution it is common to have a variety between the species, each one occupying an ecological niche separately. However, when it's works with genetic algorithms, it can occur of quickly an artificial population to suffer a convergence, and all the individuals of the almost population to become identical [7].

However, this attractive fact can not be interesting in many problems, for example, in the case of the multimodal functions, where the algorithm runs the risk to converge to a point of premature convergence [8]. That is because of the loss of population diversity. To surpass this problem, the diversity must be kept during the process of generation of individuals in the population, preventing that the individuals becomes clones of the best individual [9].

The Estimation Distribution Algorithms (EDAs) [10] are heuristic optimization algorithms that base its search in the stochastic character. Like the genetic algorithms, the EDAs are based on populations that evolve. Instead of evolving the population directly, the EDAs evolve the parameters of a probability distribution in which is estimated a set of individuals. Amongst the main algorithms used for implementation of the EDAs it is distinguished the Population Based Incremental Learning (PBIL) [11] for its robustness and simplicity.

The Population Based Incremental Learning (PBIL) is one of the based evolutionary algorithms that use models. The PBIL is the result of a combination of two techniques, the GA and the competitive learning [11]. Originally, the PBIL was designed for binary search spaces [11]. Differently from GA that evolves its population of individuals, the PBIL evolves its probability model [12, 13, 14].

Baluja shows that PBIL is more efficient than GA and the hill-climbing algorithm in a set of problems [11], and since then the PBIL have been used in a variety of applications. Between the possible applications of PBIL there are, for example, researches using complex probability models [10, 13, 14].

The challenge of solving optimization problems, for example combinatorial problems, is a function not only related to combinatorial complexity, but also, related to the difficulty in reaching all the efficient

solutions that grows with the number of objectives of the problem. In this context, to resolve the multimodal optimization problems are welcome more flexible and easy implementation methods. The evolutionary algorithms, more specifically the EDAs present a set of characteristics that are applicable to the solution of this type of this problem.

In this class of problem, the methods based on evolutionary concept privilege the individuals with better fitness, where the search procedure is based on the evolution of the probability vectors.

This work presents a study and implementation of an evolutionary method, more specifically, an Estimation Distribution Algorithm called Multi-PBil with the insertion of characteristics that allow the search in multimodal spaces.

The characteristics that make the Multi-PBil a robust and efficient method are two. The first one is the possibility to create how many probability models were necessary to an efficient search in a multimodal space. The second characteristic allows initiating the initiation rules of the probability models with values next to a 0 or 1; that means that de models can be initialized near to the searched objective.

Section 2 briefly reviews PBIL and its extensions to work with multimodal spaces. Section 3 presents the algorithm test environments that consist of the *Generalized Schwefel's Problem* and *Generalized Rastrigin's Function*. Section 4 provides the experimental results with analysis, and finally section 5 give out the conclusions with discussions on future work.

2. The Multi-PBil Method

2.1 Extensions from PBIL to Multi-PBil

PBIL is one of the simplest model-based EAs, which assumes no dependence among variables. The probabilistic model in use is a real-valued vector with each element independently representing the probability of generating a 1 in each corresponding bit.

PBIL allows representing the entire genetic population base through a probability vector rather than a myriad of chromosomes. A probability vector is simply a sequence of probabilities. Each probability value in the sequence represents the probability to generate a 1 or 0 at the gene position. By a scenario, the basic representation of a solution working with a traditional algorithm PBIL can be the same as a GA, but instead of to keep each possibility explicitly, the population is substituted by a probability distribution.

Learning in PBIL consists of using the current probability distribution to create N individuals. These individuals are evaluated according to the objective function. The "best" individual is used to update the probability vector, increasing the probability of producing solutions similar to the current best individual.

This work presents extensions to PBIL applied in multimodal problems with additional characteristics. One is the formula used to create many probability models instead of just one like PBIL does. With many probability models the algorithm can find more than one objective simultaneously. Second is a probability model initialization rule that starts with values on each entry set to values next to a 0 or 1, what is the contraire of PBIL that starts from an initial probability vector with values on each entry set to 0.5. The models with initialization next to a 0 or 1 allow locating the probability models next to the objectives searched.

2.2 Building Probability Models from Initial Population

A population member (individuals) is expressed as a binary string. Each individual has a fitness value which determines it's potential. The individuals are evaluated and ranked decreasingly according to a specific criterion related to the problem.

2.3 Probability Models.

The Multi-PBil method uses the initial population to create the probability models. Necessarily, the individual 1 of the population is used to create the first probability model. The probability models are initially configured with values generated by the formula **(1)**:

$$M_i = (1 - \alpha) * X_i + \alpha * (\textbf{AVERAGE HALF BEST INDIVIDUALS}) \quad (1)$$

Where:

- $\mathbf{M_i}$ is the created model;
- α is the learning rate;
- $\mathbf{X_i}$ is the binary value of the current position of the individual;

The criterion used to create the subsequent models follows a conditional structure. From the individual 2 to the least, a test must be done to verify if these individuals can present conditions to create the subsequent probability models. The conditional structure uses a parameter called **Factor Creation (Fc)**, to verify the probability of each individual be used to create a new probability model.

The **Fc** parameter represents a thermometer in the creation of few or many probability models. That is, as bigger as the value of **Fc**, bigger it will be the number of probability models created. And, inversely, as smaller as the value of **Fc**, less probability models will be created.

To calculate the possibilities of individual 2 to create the probability model 1, a calculation is used that measures the amount of positions that possess equal values between individual 2, and individual 1. This calculation compares all the positions filled by bits of individuals 1 and 2, and calculates the amount of position where the bits are equal (Hamming Distance). Then through the formula **(2)**:

$$Prob_n = (A * 100) / (T) \quad (2)$$

Where:

- **$Prob_n$** is the proximity of individual analyzed;
- **A** is the amount of position with existing equal bits between two individual;
- **T** is the total size of the individual in number of bits.

If the value of $Prob_n$ is less than **Fc**, for example, then individual 2 will be used to create a new (second) probability model 2 using the formula **(1)**. On the contrary, if the value of $Prob_n$ is bigger or equal to **Fc**, then the individual analyzed also presents great probability to create the probability model 1. Thus, the individual 2 is discarded and the next individual to be analyzed is the individual 3. The analysis of the individuals follows until the last individual has been compared.

Generalizing, for each individual Ind_i of the population, with i > 1, it has been realizes a test to know if this individual in question can also create the last probability model created until then. Thus, to each execution with the goal to create the probability models, a number nearly always different of models can be had created, that is, the amount of probability models bred directly is related with the diversity of the initial population.

The algorithm **(1)** shows the responsible routine for the creation of the models.

Algorithm 1: Routine Creation of the Models

```
Sub-routine Initialization
  for individual Ind_i = 2 until N do
    If max (Prob_N(Ind_i | M ( k )), k = 1...N_i) < Fc then;
         Generate a new probability model; //using formula (1)
    else Ind_i++;
  end for.
end.
```

2.4 The Phases of Multi-PBil Algorithm

Each probability model created generates a population of individuals. Each probability model when sampled, reveal relatively high quality solution vectors with high probability. The probability model is expressed as a real-valued vector with elements in the range [0,1].

Initially, each entry of the probability model are set to values next to 0 or 1. Sampling from this vector reveals random solution vectors with different probability of generating a 1 or 0 in it position of the vector.

2.5 Evaluation and Fitness Function

Each probability model works with a proper population of individuals. So that, each model makes the independent update of itself from the others models. The fitness function allows evaluating the individuals of the population. After all the individuals of the population had been evaluated and ranked according to its fitness value, one of these is chosen to update the probability model. The way to perform the update is made using a selection method. The update of the models is made according to following formula **(3):**

$$P_{I+1} = P_I * (1.0 - \alpha) + X_i * (\alpha) \quad \textbf{(3)}$$

Where:

- P_{i+1} is the probability model position value *i* after the update;
- P_i is the probability model actual position value *i*;
- α is the learning rate;
- X_i is the individual binary value in position *i*.

The formula **(3)** is applied to all the probability models.

The phases of the Multi-PBil method are composed of the following steps:

1. Generate the individuals;
2. Evaluate and rank the individuals;
3. Create the probability models;
4. For each probability model created
    a. Initiate the probability models according to the formula (1);
    b. Initiate a population size N individuals for each probability model, and determine randomly the gene of each position of each individual (0,1);
    c. Evaluate and rank the individuals;
    d. Update the probability models, through an individual chosen by a selection method, and using a learning rate;
    e. Verify if the probability model converged. Else, repeat the steps (b) to (e).

After the probability model is updated a new set of solutions is generated by sampling from the new probability model and this cycle is repeated. The phases of the Multi-PBil method continue until a stop criterion is satisfied. The stop criterion for each probability model can be:

- The algorithm executes until that one objective is founded; or
- The algorithm executes until the end of generations; or
- The algorithm executes until the elimination of the probability model.

2.6 The Mutation Operator

The mutation operator is applied in the probability models. The mutation helps to prevent premature convergence of the probability model. Thus, the application of the operator in the probability model will cause a movement of all the population through the search space, creating this way conditions so that more than one individual can find the global point [11].

The application of the mutation operator follows the following rule. For all the elements of the probability model, if a value randomly generated in the interval (0, 1] were less them **Mut_prob**, then the position where occurred this truth it suffers a movement according to the formula **(4):**

$$P_i = P_i * (1 - \text{MUT\_SH}) + \text{RAND}(0 \text{ OR } 1) * (\text{MUT\_SH}) \quad (4)$$

Where:

- Rand() is a mathematical function that generate numbers randomly;
- Mut_Sh is a constant defined by the user that indicates the motion rate of the mutation operator when it's applied to the probability model.

The Algorithm **(2)** shows a routine responsible for the applying of mutation operator.

```
Algorithm 2: Routine Mutation

For each position i of probability model do
          If (rand(0, 1] < Mut_prob) them
          P_i = P_i * (1 - Mut_Sh) + rand(0or1) * (Mut_Sh)
          end if;
  end for;
end.
```

Where:

- Mut_prob is a constant defined by the user, that indicates the mutation operator probability occurs in each position;

3. Experiments

The Multi-PBil method is designed to solve multimodal problems. It was used the multimodal function *Generalized Schwefel's Problem* and *Generalized Rastrigin's function* [15] in the experimental studies. Table **(1)** lists all the test functions. Each problem was run 50 times for the proposed method and GA-sharing[16], from k=1 to k=30. The results are compared with a GA-sharing. These functions are examples of non linear multimodal functions with many local minima points, where these points grow up exponentially as the dimension of the function grows. The functions are:

Table 1: Test Funcions.

"Generalized Schwefel's Problem":	*"Generalized Rastrigin's Function":*
Min $f_8(y) = -\sum_{i=1}^{k}(yi \sin(\sqrt{\lvert yi \rvert}))$, where: • K = 1, 2, ..., 30 • -500 < yi < 500	min $f_9(y) = \sum_{i=1}^{k}(y_i^2 - 10 * Cos(\pi y_i) + 10)$, where: • k = 1, 2, ..., 30 • $-5.12 < y_i < 5.12$

The table 2 shows the parameters used in the tests.

Table 2: Parameters used in Method Multi-PBil and GA-Sharing.

Multi-PBil		**Genetic Algorithm – Sharing**	
Parameters	**Value**	**Parameters**	**Value**
Individuals size	50 bits	Chromosome size	50 bits
Amount of individuals	500 individuals	Amount of individuals	500 individuals
Learning rate	0.005	Prob. Crossover	0.6
Tournament selection (k)	10%	Prob. Mutation	0.1
Factor criation (Fc)	80	Crossover Rate - 1 point	1
Mut_Prob	0.02	Crossover Rate - 2 ponts	1
Mut_Sh	0.05	Uniform Crossover	2

4 Discussions

The functions used in the experimental test are multimodal functions with many local minima. The number of local minima increases exponentially as the function dimensions increases. These functions appear to be very "rugged" and difficult to optimize. The figures 1, 2 and 3 shows the results of Generalized Schwefel's Problems, and figures 4 and 5 treats on Generalized Rastrigin's Functions. The average of probability models created during the runs for boths test functions were 380 probability models.

Figure 1 compare the performance of the Multi-PBil method driven by on the best local search versus the performance of the GA using sharing driven by the other local search procedures working with Generalized Schwefel's Problem. It plots the performance measure value found, expressed as a percentage of the optimal value. It's possible to realize that with the growing of the dimension of the function the Multi-PBil method presents a consistent and a higher value on the best local search points. At the end of the higher dimension both methods presents less performance, but it is still noted a better behavior of the Multi-PBil.

Figure 2 shows the mean function value where the lateral coordinates is the dimension of the test function. Figure 3 shows the standard deviation of the function value f. They can be calculated, as follow:

$$f = \frac{1}{50}\sum_{i=1}^{50}(fi/k), \text{ and } \delta_f = \sqrt{\frac{1}{50}\sum_{i=1}^{50}(\frac{fi}{k} - f)^2} \quad .$$

According to these results, we have that when the dimension of the function is large; the mean function value of the Multi-PBil method is smaller than that of the GA-sharing. It means that the search ability of Multi-PBil method is strongest compared with the GA-sharing. The standard deviation on Multi-PBil is smaller than that of GA-sharing. It means that the Multi-PBil can increase the robustness against uncertainty of GA. When the dimension of function is small, the mean function value and the standard deviation of the proposed method are larger than GA.

Figure 4 compare the performance of the Multi-PBil method driven by on the best local search versus the performance of the GA using sharing driven by the other local search procedures working with Generalized Rastrigin's Function. It plots the performance measure value found, expressed as a percentage of the optimal value. It's possible to realize that with the growing of the dimension of the function the Multi-PBil method presents a consistent and a higher value on the best local search points. But the difference on the percentage of success for both methods becomes closer. From dimension 1 to 12 both methods was able to reach approximately the same solutions.

A bigger distance is noted when k = 16 to k = 30. At the end of the higher dimension both methods presents less performance, but it is still noted a better behavior of the Multi-PBil. Even if the Multi-PBil method outperforms GA sharing on this apparently difficult function.

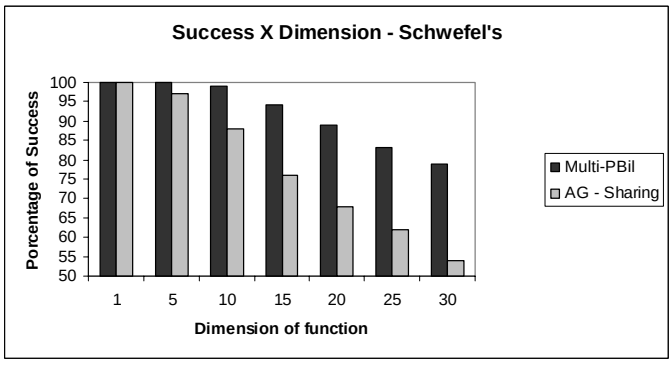

Figure 1: Success relation between the Multi-PBil method and the GA-sharing.

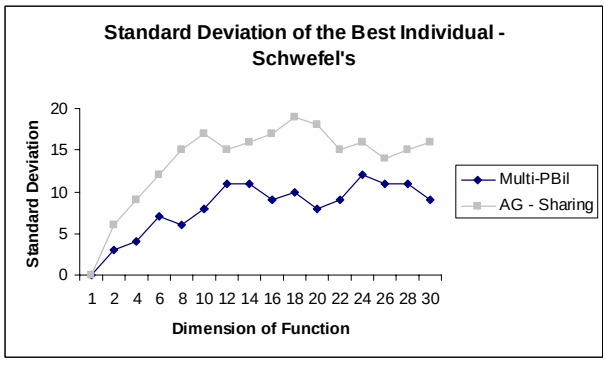

Figure 2: Performance Comparison average of the best individual between the Multi-PBil and the GA-sharing

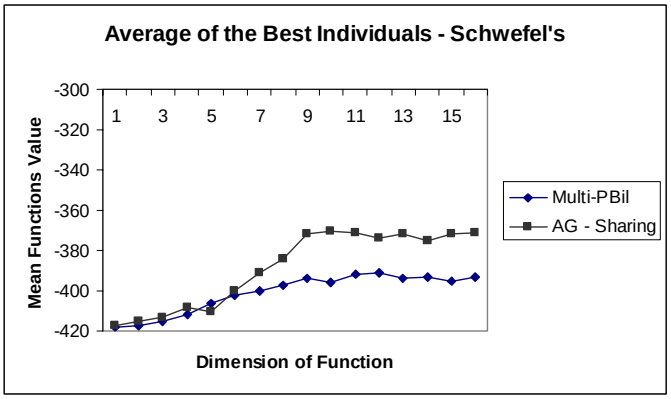

Figure 3: Standard Deviation comparison.

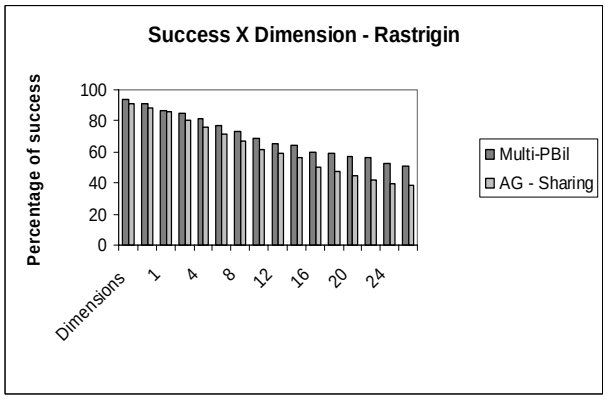

Figure 4: Success Multi-PBil and GA-sharing.

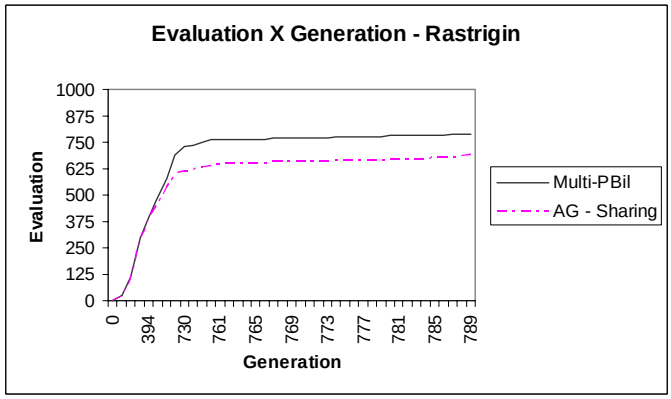

Figure 5: Performance of Multi-PBil method and GA-sharing.

Figure 5 shows that the results achieved by Multi-PBil are more accurate and are attained faster than a standard genetic algorithm using sharing. For example, the average final result obtained by the Multi-PBil method was attained in generation 789 (after this generation, no improvement was made). The Multi-PBil method was able to find equivalent solutions to the GA-sharing, measured over the entire run, in fewer generations. Figure 5 also shows in which generation the Multi-PBil was able to first achieve its highest evaluation, and in which generation was first able to surpass it.

5. Conclusions

This paper presents a method called Multi-PBil with the goal to supply a set of probability models capable to find solutions in multimodal problems. The Multi-PBil presents as advantage the possibility to initiate the probability models with values next to a 0 or 1, what have a strong influence on

migration of the population to the optimum global point. The method allows the creation of many probability models as necessary.

The Multi-PBil allows a simultaneous search for more then one solution, where this possibility to work with more than one probability model on the search allows a parallelization of the probability models. Extensively empirical studies on non linear multimodal functions were carry out to evaluate the performance of the proposed method.

The Multi-PBil performs better than a standard genetic algorithm using sharing in the problems empirically compared in this paper. The results achieved by Multi-PBil are more accurate and are attained faster than a standard GA-sharing, both in terms of number of evaluations performed and the success of the best local search procedures. A relative gain in clock speed was achieved because of the simplicity of the method. The method does not require all the mechanisms of a GA; rather the few steps in the method are small and simple.

One explanation for the good behavior of Multi-PBil is the ability to focus search effort in many regions of the space very quickly. When one region is explored extensively, the GA-sharing population loses diversity, and thereby also loses its ability to explore other regions. The empirical results show that Multi-PBil does enough exploration before the commitment is made to outperform a standard GA-sharing. That is because of the possibility to create how many probability models were necessary. This characteristic is a benefit over a traditional GA, highlighting its easily parallelizable.

But the method presents some limitation related to the amount of probability model created. In the most part of the test it was created more probability models then the necessary. Another limitation was the convergence of many probability models to the same objectives.

The good results obtained in this paper serve as stimulus for the application of the method in new domains of problems. Amongst the possible improvements to be carrying through, there are the use of a metric measure with the purpose to analyze with precision the distances between probability models and objectives searched and the consequent impact on the elimination on real time of the probability models with unsatisfactory performance. A deep study about the number of probability models created to be used in the search space. And as third future work there is the study of the method Multi-PBil in a parallel architecture.

6 References

1. Darwin, C.: On The Origin of Species by Means of Natural Selection of the Preservation of Favored Races in the Struggles for Life., 1859. Murray, London, UK.
2. Hopper, N. J., McPhee, N. F.: Analisys of Genetic Diversity Through Population History. In: GECCO99: Proceedings of the Genetic and Evolutionary Computation Conference, July 1999.

3. Keller, R. E., Francone, F. D., Banzhaf, W., Nodin, P.: Genetic Programming: An Introduction. Morgan Kaufmann, 1998.

4. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)

5. Cook, M. L.: Genetical and Ecological Diversity: The Sport of Nature. London:Chapman & Hall, 1991.

6. Shaw, M. J., Sikora, R.: A Double-layered Learning Approach to Acquiring Rules for Classification: Integrating Genetic Algorithms with Similarity-based Learning. The RSA Journal of Computing, 1994.

7. Langton, C. G.: Artificial Life. Addison-Wesley, 1989. Redwood City: CA.

8. Parmee, I. C., Beck, M. A.: Extending the Bounds of the Search Space: A Multipopulation Approach. GECCO99: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann Publishers, 1999.

9. Eshelman, L. J., Schaffer, J. D., Caruana, R. A.: A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, 1989.

10. Lozano, J. A., Larranãga, P.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publisher, 2001.

11. Baluja, S.: Population Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Technical Report, Carnegie Mellon University, 1994.

12. Stern, D., Servet, I., Massuyes, L. T.: Telephone Network Traffic Overloading Diagnosis and Evolutionary Computation techniques. In Artificial Evolution 07, J. K. Hao et al. Eds., 1997. 137-144.

13. Ducoulombier, A., Sebag, M.: Extending Population-Based Incremental Learning to Continuous Search Spaces. In Parallel Problem Solving From Nature – PPSN V, A. E. Aiben e al. Eds., 1998. 418-427.

14. Lobo, F., Pelikan, M., Goldberg, D. E.: A Survey of Optimization by Building and using Probabilistic Models. Technical Report, University of Illinois at Urbana Champaign, Illinois Genetic Algorithms Laboratory, 1999.

15. X. Yao, Y Liu: Fast Evolution Strategies. Sixth Annual Conference on Evolutionary Programming, Indianapolis, USA, 13-16, April 1997.

16. Goldberg, D. E., and Richardson, J., "Genetic Algorithms with Sharing for Multimodal Function Optimization," Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms, 1987, pp. 41-49.