Continuous Optimization by Evolving Probability Density Functions with a Two-Island Model

Alicia D. Benítez and Jorge Casillas

Dept. Computer Science and Artificial Intelligence
University of Granada. Spain, E-18071
casillas@decsai.ugr.es

Abstract. The work presents a new evolutionary algorithm designed for continuous optimization. The algorithm is based on evolution of probability density functions, which focus on the most promising zones of the domain of each variable. Several mechanisms are included to self-adapt the algorithm to the feature of the problem. By means of an experimental study, we have observed that our algorithm obtains good results of precision, mainly in multimodal problems, in comparison with some state-of-the-art evolutionary methods.

1 Introduction

Continuous (real-parameter, global) optimization implies an important problem for engineering, because, it is hard to find a method able to obtain solutions constituting global optimum for the problem we are dealing with. Metaheuristics, such as Genetic Algorithms (GAs), Evolutionary Strategies (ESs), and Memetic Algorithms (GAs hybrided with local search techniques), are the ones being most currently applied to continuous optimization.

There are algorithms specifically designed for this type of optimization, Estimation of Distribution Algorithms (EDAs) [3], which do not use crossover operators, or mutation operators, unlike GAs and ESs. In order to evolve population evolution they use a mechanism consisting of individuals sampling from a given density of probability. In the same way of this algorithm, we find continuous PBIL [5], with a density of probability being normal for each variable, from which a population updating the normal ones is generated. Mechanisms to adapt standard deviation are also included.

In many cases, a number of populations working in a parallel way [1] can give rise to a good behavior of the algorithm. This kind of algorithms are known as Multideme Parallel Evolutionary Algorithms (PEAs) or island models. They consist of a number of subpopulations evolving independently which, occasionally, exchange information among them.

In this work, a new algorithm is proposed, as a result from taking essential ideas of several metaheuristics: GAs, ESs, Memetic Algorithms, EDAs, and

L. Jiao et al. (Eds.): ICNC 2006, Part I, LNCS 4221, pp. 796–805, 2006.

PEAs. The basic algorithm is based on evolving a mixture of normal distributions by keeping the best obtained solutions. It is embedded on an algorithm that evolves several set of them to avoid premature convergence and that considers two islands of different behaviors to self-adapt the algorithm to the problem.

According to that, this contribution is organized as follows: Section 2 describes the algorithm; Section 3 shows an empirical study of the algorithm compared with other proposals; and finally, Section 4 shows some conclusions and future works.

2 Description of the EvolPDF-2 Algorithm

For a better understanding, we have distinguished between what we call the *basic algorithm* and the final proposed algorithm (called EvolPDF-2). The basic algorithm, that can be considered the search process' core, is described in subsections 2.1 and 2.2. This basic algorithm would not work properly as such due to its high risk of falling in local optima and its high dependence with respect to the value parameters set. To face this, we have designed a more complex algorithm based on the basic one that is described in subsections 2.3 and 2.4.

2.1 Basic Components: Representation, Initialization, Sampling, and Replacement

The basic algorithm consists on four main components: *representation*, that describes how solutions are coded; *initialization*, that creates a set of initial solutions; *sampling*, that generates new solutions with the probability density functions (PDFs) estimated from the current best solutions; and *replacement*, that update the set of solutions considered to generate new ones. The mentioned components are designed as follows:

- *Representation*: The algorithm maintains a set of k solutions during the search process. Each solution is represented as follows: $S_i = \left(\mu_i^1, ..., \mu_i^v, ..., \mu_i^n\right)$, with n being the number of continuous variables of the problem.
- *Initialization*: The algorithm begins generating randomly k solutions according to a uniform distribution.
- *Sampling*: In each iteration, a number of solutions is generated from the current set. To do that, we consider that the current set of solutions constitutes a PDF (mixture of normal distributions) for each variable:

$$PDF_v = \sum_{i=1}^{k} \omega_i \cdot N(\mu_i^v, \sigma), \sum_{i=1}^{k} \omega_i = 1 \tag{1}$$

where

$$N(\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

It should be noticed that the standard deviation is common to all the normals.

The mixture of normals defining the PDF of each variable are weighted in such a way that the integral of the PDF equals to 1, with the weights being directly related to the objective of the solution. They are computed as follows:

$$\omega_i = \frac{f(S_i)}{\sum_{h=1}^{k} f(S_h)} \tag{3}$$

where $f(S_i)$ is the objective function of the solution from which the ith normal comes from. We assume the function is maximized.

To generate, i.e. sample, new solutions, the following process is performed. For each variable, firstly, we choose with proportional probability a normal of the corresponding PDF (the higher ω_i, the greater the probability of being chosen). Secondly, a random number with normal density is generated centered on the mean of the normal chosen in the previous step. The obtained value will be the one assigned to the corresponding variable of the generated solution. In case of generating a value out of the allowed range for each variable, the value is set with the corresponding extreme of the interval. Consequently, a saturation takes place in the extremes to compensate the lack of accumulation of density in these regions.

– *Replacement*: Among the solutions generated in the current iteration, the best one according to the fitness $f(S_i)$ is chosen and replaces to the worst of the k current best solutions kept by the algorithm in case that the former outperforms it. This replacement can be considered as a kind of estimation.

2.2 Standard Deviation Convergence

In every metaheuristic, it is necessary the existence of exploration and exploitation. In the beginning of the algorithm, we are more interested on a higher exploration, so diversification can be favored in the search space. As long as we get close to the final point of the algorithm, it is more convenient a higher exploitation, in order to reach a local or global optimum, from the area which has turned out to be more promising. According to that, our algorithm regulates the standard deviation of the distributions, trying to make it higher at the beginning of the algorithm and lower at the end. In order to achieve that, we have defined a schema of convergence of the standard deviation, according to the following formula:

$$C(t) = \begin{cases} \sigma_0 & \text{if} \quad t \leq \alpha \\ (\sigma_0 - \sigma_f)(1 - 2(\frac{t-\alpha}{\beta-\alpha})^2) & \text{if} \quad \alpha < t \leq \frac{\alpha+\beta}{2} \\ (\sigma_0 - \sigma_f)(2(\frac{t-\beta}{\beta-\alpha})^2) + \sigma_f & \text{if} \quad \frac{\alpha+\beta}{2} < t \leq \beta \\ \sigma_f & \text{if} \quad t > \beta \end{cases} \tag{4}$$

with t being the evaluation number, σ_0 and σ_f the initial and final desired values of the standard deviation respectively, and α and β two fixed constants depending on the number of evaluations: $\alpha = evaluations \cdot 0.1$, $\beta = evaluations \cdot 0.8$. With this formula, we regulate the decrease of the standard deviation to distinguish three stages during the evolution of PDFs:

1. *Initialization*: At first (10% of evaluations), the standard deviation remains constant and the algorithm is used for the configuration of a number of initial PDFs.
2. *Consolidation*: Then, we slowly reduce the standard deviation in order to gradually favor the exploitation to the detriment of exploration.
3. *Tuning*: Finally, during the last 20% of evaluations, the final adjustment of the PDFs is carried out, keeping the standard deviation low and constant.

The following values are used for σ_0 and σ_f: σ_0 = 10e-01 and σ_f = 10e-04. These values have been proven to work properly in different problems.

2.3 Island-Based Model

Once defined the basic algorithm as described in the three previous sections, we have studied its behavior in different situations. It is clear that two of the more important parameters of the basic algorithm is the number of samples and normals. Obtained results with different combinations of them are shown in Table 1, for Rosenbrock (unimodal) and Rastrigin (multimodal) problems, both with 25 variables. The shown results are the mean and standard deviation of 10 runs of the basic algorithm. In this table, it can be observed how in the Rosenbrock unimodal problem, the less samples per iteration are carried out and the lower the number of normals we have, the better the result is. With low values for number of samples and number of normals, a higher intensification appears, which is good for the behavior of the algorithm to face unimodal problems. If we pay attention to Rastrigin (multimodal problem), we can see that what happens is just the opposite than in Rosenbrock's case. That is, the solutions improve as long as the number of samples and normals higher. This is due to the fact that, with the increase of the value of these parameters, diversity is encouraged, which involves appropriate functioning in multimodal problems.

Given these premises, we have decided to specialize the algorithm in both behaviors with the aim of fixing these parameters *a priori* regardless the problem we are facing. To do that, we consider a two island-based parallel model as shown in Figure 1.

Therefore, each island is assigned to a different behavior. Basically, each island implements the above described basic algorithm with different parameters. When several iterations have passed, the current best solutions of each island are migrated to the other one. In each island, their sets of PDFs evolve independently and in parallel. The exchange of information between them favors the cooperation between islands to attain the global optimum. An advantage of the island-based model is that it is not compulsory the use of the same kind of algorithm in each island, which indicates that we can use each island for one kind of approach, so we use an island for problems requiring intensification rather than diversification, and another one for problems requiring more diversification than intensification. It makes the algorithm being capable of properly adapting itself to the most promising approach for the problem to be solved. Each island has a number of characteristics: number of set of PDFs, number of normals constitut-

Table 1. Experimental results of the basic algorithm with Rosenbrock and Rastrigin functions (10 runs)

Rosenbrock				Rastrigin			
# Samples	*# Normals*	*Mean*	*Std*	*# Samples*	*# Normals*	*Mean*	*Std*
5	1	1.43e+1	4.97e+0	5	1	6.62e+1	6.42e+1
	5	1.73e+1	2.99e+0		5	6.20e+0	7.53e+0
	10	1.76e+1	2.02e+0		10	2.48e+0	7.41e+0
	20	1.80e+1	1.72e+0		20	3.64e-1	3.58e+0
	30	1.79e+1	1.71e+0		30	9.94e-2	1.63e+0
10	1	1.69e+1	7.25e+0	10	1	6.41e+1	8.43e+1
	5	1.72e+1	2.81e+0		5	5.53e+0	9.71e+0
	10	1.98e+1	5.88e+1		10	1.98e+0	5.26e+0
	20	1.79e+1	2.46e+0		20	4.97e-1	3.37e+0
	30	1.82e+1	1.85e+0		30	6.63e-2	1.35e+0
15	1	1.71e+1	1.49e+1	15	1	6.62e+1	7.68e+1
	5	1.74e+1	2.79e+0		5	5.37e+0	1.03e+1
	10	1.78e+1	3.02e+0		10	1.72e+0	4.86e+0
	20	1.81e+1	1.99e+0		20	1.65e-1	2.47e+0
	30	1.80e+1	2.20e+0		30	3.31e-2	9.78e-1
20	1	1.78e+1	6.30e+0	20	1	6.17e+1	8.64e+1
	5	1.80e+1	2.59e+0		5	4.94e+0	1.26e+1
	10	1.81e+1	6.84e+0		10	1.62e+0	5.88e+0
	20	2.02e+1	5.88e+1		20	1.32e-1	1.85e+0
	30	2.02e+1	5.74e+1		30	6.63e-2	1.35e+0

ing each PDF to a fixed value of the k, number of samples, and implementation of Local Search (LS).

- *Island 1 (I1)*: The aim of this island is to intensify the search and keeping the most promising solutions. To attain that, we employ the following parameters of the basic algorithm:
  1. Number of normals $k = 1$.
  2. Number of samples per iteration = 5.
  3. LS is applied to the best solution sampled, as soon as it has been updated.
- *Island 2 (I2)*: The aim of this latter island is to provide with the necessary diversity to the algorithm, so it does not get stuck in local optima. The following parameters and components are used:
  1. Number of normals $k = 30$.
  2. Number of samples per iteration = 20.
  3. LS is applied to the solution migrated to I2.

Just with a quick glance, we can see the different configuration of each island. The number of normals is so much higher in I2 than in I1. Migrations is carried out every given number of iterations in the following way: the best solution of each island is chosen to migrate to the other island. The best individual of I2 becomes a part of I1 if it outperforms its current best solution. The solution

Inputs:

- Parameters: σ_0, σ_f
- Let n be the number of variables and $T^l = \{S_1^l, \ldots, S_{k_l}^l\}$ the lth island, $l \in \{1, 2\}$.

Algorithm:

1. Initialization($S_i^l \in T^l$, for $l \in \{1, 2\}$, $i \in \{1, \ldots, k_l\}$
2. Repeat during 90% of evaluations
   (a) For T^1:
   $R_t^1 \leftarrow \mathsf{Sampling}(T^1)$
   if ($\mathsf{better}(\mathsf{best}(R_t^1), \mathsf{worst}(T^1))$)
   $r \leftarrow \mathsf{Simplex}(\mathsf{best}(R_t^1))$
   $T^1 \leftarrow (T^1 - \{\mathsf{worst}(T^1)\}) \cup \{r\}$
   (b) For T^2:
   $R_t^2 \leftarrow \mathsf{Sampling}(T^2)$
   if ($\mathsf{better}(\mathsf{best}(R_t^2), \mathsf{worst}(T^2))$)
   $T^2 \leftarrow (T^2 - \{\mathsf{worst}(T^2)\}) \cup \mathsf{best}(R_t^2)$
   (c) if (*condition of migration*)
   $r \leftarrow \mathsf{Simplex}(\mathsf{best}(T^1))$
   $\mathsf{Migrate}(r, T^2)$
   $\mathsf{Migrate}(\mathsf{best}(T^2), T^1)$
3. During the rest 10% of evaluations, to apply Simplex algorithm to the best found solution

Fig. 1. EvolPDF-2 algorithm

coming from I1 becomes a part of I2, taking the place of the worst of each of the existing solutions in I2. We have fixed the migration time to every 100 iterations.

We want to emphasize that this migration allows the algorithm to self-adapt to the faced problem. For multimodal problems, the I2 works better due to their characteristics; if a stagnation is produced in the I1, the migration of a solution of I2 to I1 takes place, and this make that the I1 leaves an area with local optimum, because it will jump to another promising area of the space of search given by I2. In the same way, for unimodal problems, the solution of I1 migrates to I2 with the aim of parallelize the search with the aim of speed up the process.

2.4 Local Search

LS is a metaheuristic consisting of the search of good solutions in a neighborhood. Occasionally, this technique could lead us to local optimum, but it can turn out to be an efficient tool in the polishing of solutions. In our algorithm, we use LS as a way of polishing solutions. In I1, we apply LS to the best sampled solution, as long as it is better than the best elitist in its island. This reinforces even more the intensification of the search space. In I2, we apply LS to the solution migrating from I1 to I2, as long as it is better than the worst of each set of PDFs. LS is also applied at the last 10% of evaluations in order to polish the

best found solution. The LS used in this work is based on the implementation of the Simplex algorithm available in [4].

3 Empirical Study

This section presents the empirical results of the experiment carried out, where our algorithm is compared with the results obtained by some state-of-the-art evolutionary algorithms for global optimization.

3.1 Experiment Setup

We have considered the 25 functions proposed in the Special Session on "Real-parameter optimization" of the IEEE CEC 2005 [2], with the parameters established in this session: experimentations have been made on 30 variables for each function, with a number of 25 runs for each one; and with a maximum of 300,000 evaluations. There are three kinds of functions: unimodal (from function 1 to 6), multimodal (from 7 to 14), and hybridations of multimodal functions (from function 15 to 25). It is remarkable the fact that all the functions are displaced to avoid symmetric and centered location of the optimum. The maximum error allowed in the function is 10e-8. We have considered the eleven algorithms included in this special session for comparison.

We will use two measures to analyze the obtained results:

1. *Ranking*: It gathers the number of times that an algorithm has been in first, second, third, fourth, and fifth position, according to the obtained results. This measure is calculated as follows. For each function, the different algorithms are ordered by their mean error. Then, we assign a position to each algorithm (1st, 2nd, 3rd, 4th, or 5th). If several algorithms have the same value of mean (or they reach the optimal value), they will have the same value of position. Therefore, if two algorithms reach the optimal value and the third does not, we will assign position 1 to the two first algorithms, and position 3 to the third. For each algorithm, we add the number of times that appears in each position for all the considered functions.
2. *Total mean error*: This method consists of calculating the mean error obtained by each algorithm normalized by the best obtained result for each function. It is computed as follows:

$$TME_{alg} = \sum_{i \in \text{Functions}} \frac{f_i(S_{alg}) - \theta_i}{\max\limits_{j \in \text{Algorithms}} (f_i(S_j)) - \theta_i} \tag{5}$$

with θ_i being the error threshold of the ith function (10e-8 in our case). Besides of the ranking of the algorithms, we consider the total mean error to have a better idea about the accuracy of the algorithm. This measure is very useful when the algorithms do not reach the optimal value, or when all the algorithms find very similar but different values.

3.2 Obtained Results

Tables 2, 3 and 4 summarize the position of each algorithm (according to the ranking described in Sect. 3.1) as well as the total mean error (eq. 5). In these tables, the algorithms are sorted according to the ranking except our algorithm, that is shown at the last row. (Some results of algorithms EDA, L-SaDE and DMS-L-PSO, are not available for dimension 30). Our algorithm is named as EvolPDF-2. We would like to highlight that we have fixed the parameter values of our algorithm as mentioned before for all the experimentation, being the same for the 25 functions. The best results for 1st ranking and total mean error are shown in boldface.

Table 2. Ranking and total mean errors obtained for unimodals functions (functions 1 to 6) for dimension 30

	Ranking					*Total*
Algorithm	*1st*	*2nd*	*3rd*	*4th*	*5th*	*mean error*
G-CMA-ES	**5**	0	0	0	0	**0.0001**
L-CMA-ES	5	0	0	0	0	1
EDA	4	1	0	0	0	0.009
BLX-GL50	2	1	0	1	2	0.04
K-PCX	2	0	1	1	0	0.24
DMS-L-PSO	1	1	2	0	0	0.26
L-SaDE	1	1	1	0	1	0.14
SPC-PNX	1	1	0	1	1	1.51
DE	1	0	1	1	0	0.73
BLX-MA	1	0	0	0	1	1.07
CoEVO	0	0	1	1	0	3.52
EvolPDF-2	0	1	0	0	0	1.17

As we can observe for unimodal problems (Table 2), G-CMA-ES and L-CMA-ES are the algorithms most appeared at the first position with 5 times both. From a global view of the ranking obtained by the algorithms from 1st to 5th positions, we can say that the three algorithms that have the best values are: BLX-GL50 with 6 appearances and G-CMA-ES and L-CMA-ES with 5. We only obtain 1 second position. With respect to the total mean error, G-CMA-ES obtains the lowest error and our algorithm is the second worse in error.

For multimodals problems (Table 3), DMS-L-PSO is the most appeared at the first position with 3 times and it is followed of G-CMA-ES and L-SaDE with 2 first position. We get 1 third, fourth and fifth position. From a global view of the ranking obtained by the algorithms from 1st to 5th positions, we can say that the three algorithms that have the best values are: G-CMA-ES and L-CMA-ES with 7 times and BLX-GL50 with 5. Our algorithm appears 3 times. With respect to the total mean error, G-CMA-ES obtains the lowest error and our algorithm is the ninth better in error.

Table 3. Ranking and total mean errors obtained for multimodals functions (functions 7 to 14) for dimension 30

	Ranking					*Total*
Algorithm	*1st*	*2nd*	*3rd*	*4th*	*5th*	*mean error*
DMS-L-PSO	**3**	0	2	0	1	2.83
G-CMA-ES	2	2	0	2	1	**2.30**
L-SaDE	2	1	2	2	0	2.72
K-PCX	2	1	1	0	0	3.58
L-CMA-ES	2	1	0	1	0	4.61
BLX-GL50	1	1	1	0	2	3.03
EDA	1	0	0	0	0	4.95
DE	0	1	0	1	1	3.31
SPC-PNX	0	1	0	0	0	2.87
BLX-MA	0	0	1	1	2	3.23
CoEVO	0	0	0	0	0	6.30
EvolPDF-2	0	0	1	1	1	4.38

Table 4. Ranking and total mean errors obtained for highly multimodals functions (functions 15 to 25) for dimension 30

	Ranking					*Total*
Algorithm	*1st*	*2nd*	*3rd*	*4th*	*5th*	*mean error*
SPC-PNX	1	1	2	0	3	5.34
DE	1	1	1	0	0	6.34
BLX-MA	1	0	5	0	1	6.13
BLX-GL50	1	0	2	3	2	5.87
G-CMA-ES	0	5	1	0	1	5.69
K-PCX	0	4	0	2	2	6.99
L-CMA-ES	0	1	3	4	0	7.31
DMS-L-PSO	0	0	0	1	1	n/a
CoEVO	0	0	0	1	0	9.11
L-SaDE	0	0	0	0	1	n/a
EDA	0	0	0	0	0	n/a
EvolPDF-2	**9**	0	1	0	0	**1.79**

For hybridations of multimodals problems (Table 4), our algorithm is the most appeared at the first position with 9 times and it is followed of SPC-PNX, DE, BLX-MA and BLX-GL50 with 1 first position. From a global view of the ranking obtained by the algorithms according to positions, we can say that the three algorithms that have the best values are: our algorithm with 10 times and BLX-GL50, K-PCX and L-CMA-ES with 8. With respect to the total mean error, our algorithm obtains the lowest error.

To sum up, we can say that our algorithm obtains very good results for problems not solved by any other analyzed algorithm. We can also observe that where we obtain the lowest error are between functions number 15 and 25 (all of them being highly multimodal problems). These results suggest that our al-

gorithm, thanks to the mechanisms included to avoid local optima, works better in multimodal problems.

4 Conclusions and Future Works

We have proposed a simple algorithm for continuous optimization. It involves using several PDF to model the interest region of each variable and evolving them to focus the search region. Some mechanisms to avoid premature convergence and local optima are included. The algorithm does not belong to any specific metaheuristic paradigm but it takes ideas from some of them such as GA and EDA. The obtained empirical results lead us to think that the algorithm has a high performance, obtaining the best results in multimodal problems. As further works, we suggest to consider self-adaptation of more parameters (e.g., σ_0 and σ_f) as well as to do a deeper characterization of the problems where the algorithm has a competitive advantage.

Acknowledgment

The authors would like to thank to Carlos García and Daniel Molina (both from the University of Granada, Spain) for providing us with the tools used in the empirical comparison. This work was supported in part by the Spanish Ministry of Science and Technology under grant no. TIC2003-00877 and by ERDF.

References

1. E. Alba, J.M. Troya. A survey of parallel distributed genetic algorithms. Complexity, 4:303-346, 1999.
2. K. Deb, D. Corne, Z. Michalewicz, Special Session on Real-parameter Optimization, IEEE Congress on Evolutionary Computation 2005, Edimburgh, UK, September 2005.
3. P. Larrañaga, J.A. Lozano (Eds.) (2001). Estimation of distribution algorithms. A new tool for evolutionary computation. Kluwer Academic Publishers.
4. Numerical Recipes in C. The Art of Scientific Computing. Second Edition. William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery. Cambridge University Press.
5. M. Sebag, A. Ducoulombier (1998). Extending population-based incremental learning to continuous search spaces. Lecture Notes in Computer Science 1498: 418-427.