On the evaluation of information exchange strategies in dEDAs

S. Muelas
Univ. Politécnica
de Madrid
smuelas@fi.upm.es

A. Mendiburu
Univ. del
País Vasco
alexander.mendiburu@ehu.es

A. LaTorre
Univ. Politécnica
de Madrid
atorre@fi.upm.es

J.M. Peña
Univ. Politécnica
de Madrid
jmpena@fi.upm.es

Abstract

One of the most promising areas in which probabilistic graphical models have shown an incipient activity is the field of heuristic optimization and, in particular, in the Estimation of Distribution Algorithms (EDAs). EDAs constitute a well-known family of Evolutionary Computation techniques, similar to Genetic Algorithms. Due to their inherent parallelism, different research lines have tried to improve EDAs from the point of view of execution time and/or accuracy. Among these proposals, we focus on the so-called island-based models. This approach defines several islands (EDA instances) running independently and exchanging information with a given frequency. The information sent by the islands can be a set of individuals or a probabilistic model. This paper presents a comparative study of both information exchanging techniques for a univariate EDA ($UMDA_g$) over a wide set of parameters and problems –the standard benchmark developed for the IEEE Workshop on *Evolutionary Algorithms and other Metaheuristics for Continuous Optimization Problems* of the ISDA 2009 Conference.

1 Introduction

Estimation of Distribution Algorithms are a set of techniques that belong to the field of Evolutionary Computation. Since they were introduced in the 90s [13], the research community has put a lot of effort in their development, providing powerful algorithms which have been successfully applied to both artificial and real-world problems. In general terms, EDAs are similar to Genetic Algorithms, but their main characteristic is the use of probabilistic models to extract information from the current population (instead of using crossover or mutation operators) in order to create a new and presumably better population.

The main drawback of the most complex EDAs –those that try to consider all the possible (in)dependencies among the variables– is the high computational cost. Due to this reason, and thanks to the modularity of EDAs, several parallel approaches have been proposed. These proposals can be divided into two groups:

- Direct parallelization (pEDAs): Those which behavior is exactly the same of the corresponding sequential version. Their main goal is the reduction of the execution time, and the applicability to larger problems.
- Island-based approach (dEDAs): Those that create different subpopulations and exchange information among them, trying to improve the quality of the solutions of the sequential algorithm.

In this work, we pay attention to the second approach. In this scheme, an EDA instance is executed in each island, and some information is exchanged among the islands during the execution. This information can be made up of individuals (as done in other EAs), or probabilistic models (following the rationale that EDAs use them to extract and gather information about the population). Migration of individuals is a classic approach and has proven to obtain successful results [1, 2, 3]. In addition, migration of models was explicitly developed for the distributed estimation of distribution algorithms (dEDAS) [6, 4, 7, 8].

Until now, most of the previous work in dEDAs has been conducted in the discrete domain, and little research has been done in comparing both migration methods (individuals versus models). In particular, in continuous optimization, as far as the authors are aware, only two studies have been conducted [6, 5]. Although these papers concluded that the migration of models obtains significantly better results than the migration of individuals, the experimental scenario was restricted to a) a limited number of problems with small dimensionality and b) a small number of parameters that were analyzed. In this paper, we study empirically both approaches over the standard benchmark developed for the IEEE workshop on *Evolutionary Algorithms and other Metaheuristics for Continuous Optimization Problems* of the ISDA 2009 Conference. Therefore, our goal is to carry out an extensive study combining a wide set of parameters and using a standard benchmark of problems.

The rest of the paper is organized as follows: Section 2 presents an overview of the previous studies on EDAs and dEDAs. Section 3 describes the proposed experimental scenario. Section 4 presents and comments the results obtained and lists the most relevant facts extracted from this analysis. Finally, Section 5 contains the concluding remarks derived from this study.

2 Preliminaries

2.1 Estimation of Distribution Algorithms: EDAs

EDAs are non-deterministic, stochastic heuristic search strategies that are part of the Evolutionary Computation paradigm. In EDAs, multiple solutions or individuals are created at every generation, evolving successively until a satisfactory solution is achieved. In brief, the characteristic that clearly differentiates EDAs from other evolutionary search strategies, such as Genetic Algorithms (GAs), is that the evolution from one generation to the next is achieved by estimating the probability distribution of a set of individuals, sampling later the induced model. This avoids the use of crossing or mutation operators, and the number of parameters required by EDAs is considerably reduced. Based on the probabilistic model considered, three main groups of EDAs can be distinguished: univariate models, which assume that variables are marginally independent; bivariate models, which accept dependencies between pairs of variables; and multivariate models, in which there is no limitation on the number of dependencies. The complexity of the different EDA approaches is usually related to the probabilistic model used, and the ability of that model to identify and represent the (in)dependencies among the variables. Detailed information about the main characteristics of EDAs, as well as the different algorithms that belong to this family, can be found in [10, 11].

In this study, we focus on the Univariate Marginal Distribution Algorithm for Gaussian Models ($UMDA_g$) [9]. This algorithm considers no dependencies between the variables involved in the problem. It is assumed that the joint density function follows a n-dimensional normal distribution, which is factorized by a product of one-dimensional and independent normal densities.

2.2 Distributed Estimation of Distribution Algorithms: dEDAs

In the distributed Evolutionary Algorithms (dEAs)[1], the entire population is distributed over multiple subpopulations and occasionally allows the migration or exchange of some individuals among the different islands. Therefore, each node executes an independent algorithm on an independent population. An important aspect of the performance of dEAs is the migration strategy. This is configured through different parameters: (i) Migration frequency: How often (number of generations) is information sent?, (ii) Migration rate: How many

[1] also known as coarse-grained, multiple-deme or island models

individuals migrate each time?, (iii) Information selection: What kind of information is exchanged?, (iv) Acceptance policy: How are the incoming and the local information combined?, (v) Migration topology: Which island sends information to which other?

Regarding the information exchanged among islands, two possible alternatives are available: (i) the straightforward approach of selecting a pull of individuals that will be later sent to the consignees and (ii) the alternative of using the main characteristic of EDAs: the probabilistic models. These probabilistic models will be (or should be) able to represent the (in)dependencies among the variables, and, therefore, comprise more information than a group of individuals. This second approach opens a new challenge: how should the different probabilistic models be combined? In the simplest case, a convex combination of the resident model with the immigrants one can be formalized by the following rule:

$$M'_R = \beta M_R + (1-\beta) M_I \qquad (1)$$

where β varies in the range $[0,1]$ and represents the influence of the immigrant model M_I on the resident model M_R. An extended version of this formula for n immigrant models would be:

$$M'_R = \beta_R M_R + \beta_{I1} M_{I1} + \beta_{I2} M_{I2} + \ldots + \beta_{In} M_{In} \qquad (2)$$

In order to compute the value of β, two different strategies have been traditionally considered. The simplest one is called *constant value* and it simply assigns to each β a constant value within the interval $[0,1]$. The second one, called *adaptive value*, computes the β value based on the *quality* of the population associated to each model. A conservative policy is followed in thise approach and only the models with a population of better quality than the resident model are considered for the computation:

For n immigrants, the β value is defined as:

$$\beta_R = \frac{F_R}{F_R + \sum_j^n F_{I_j}}, \beta_{I_i} = \frac{F_{I_i}}{F_R + \sum_j^n F_{I_j}} \qquad (3)$$

where F_R represents the mean fitness value of the resident subpopulation and F_{I_i} represents the mean fitness value of the i-th immigrant subpopulation. Due to the conservative policy, this value has to be better than F_R.

A different approach for combining the models is to use a mixture model $M = \sum_i \beta_i M_i$ as a linear combination of simple distributions, where the β_i satisfy that $\sum_i \beta_i = 1$ [5]. In this case, the population from the mixture model would be created by sampling from each of the involved models according to a uniform distribution and their associated β value. This way, the models with higher β values, would have a higher probability of producing more individuals of the new population. This approach has the advantage that can combine any probabilistic model since it does not depend on the details of the model.

For this work we have introduced a new combination model called *uniform combination.* This method does not combine the models, but it selects each model component from a model of the global set of the immigrants and resident models. Each model has a probability β of being selected for each of the components of the new model. The β values are computed using the same formula of the adaptive combination method.

3 Experimentation

For the experimentation, the benchmark from the workshop on *Evolutionary Algorithms and other Metaheuristics for Continuous Optimization Problems - A Scalability Test* to be held at the ISDA 2009 Conference has been considered. These functions have different degrees of difficulty and can scale to any dimension.

Table 1 shows the different parameters used throughout the experiments. In order to analyze the effects of the migration strategies, several island configurations of $UMDA_g$ instances were compared against each other. Some of the parameters have been used in previous studies with dEDAs [4, 8], and additional parameters have been included to obtain a wider view. For each combination, 25 independent executions were carried out. The stopping criterion, as defined in the benchmark, was a fixed number of fitness evaluations (5,000 times the dimension of the problem). The performance criterion is the distance (error) between the best individual found and the global optimum in terms of fitness value. A sequential version of the $UMDA_g$ algorithm was also executed with different population sizes (64, 128, 256, 512, 1024 and 2048) in order to have a baseline comparison.

Table 1: Parameters Values

Common Parameter Values	
Problem Size	50, 100 and 200
Population Size	512, 1,024 and 2,048
Learning Model	$UMDA_g$
Selected Inds.	best 50%
#Islands	8 and 16
Topology	ring and all-to-all (a2a)
Migration Period	10, 20 and 40 iterations
Accept. Policy	best individuals from
	resident and immigrant
	populations
Particular Parameter Values	
Migration Rate	10% and 20%
Emigrants Selec.	best or random inds.
Models Comb.	convex, mixture and uniform
β Strategy	adaptive

4 Analysis of the Results

In order to compare all the configurations across all the functions, the average rank according to the Friedman test was computed for each function and for all the functions. The nWins procedure [12] was also applied to the average ranks per function to perform a global comparative analysis. This procedure carries out a pair-wise statistical comparison over the distribution values of all the available configurations by means of the Wilcoxon signed-rank test with a confidence level of 0.05. With these results, the following analysis is carried out: if one algorithm is significantly better than other ($p - value < 0.05$), the winning algorithm is granted +1 wins and the losing algorithm is penalized with -1 wins. The sum of all the "wins" constitutes the nWins value.

Tables 2, 3 and 4 display a ranking of the configurations on both 50-D, 100-D and 200-D according to the global average rank, together with the nWins score. Due to the size of these tables (258 rows), only the best and worst 5 configurations as well as the sequential configurations are displayed. From these results, it can be seen that the best results for all dimensions are obtained by configurations based on sending individuals whereas the worst results are obtained by the configurations based on sending models. It can also be seen that the ring topology and the selection of the best emigrants are the values than can be found in all the best configurations whereas the worst configurations tend to have a higher number of islands and migration periods. Therefore, it seems that the best configurations are those which are based on sending individuals, have a reasonable population size per island but still small enough to have a considerable number of iterations (due to the restriction of the benchmark), use the ring topology and send their best individuals. On the other hand, the worst configurations have in common the smallest population size per island and generally have values that quickly decrease the diversity of the populations, i.e highest topology degree and highest migration rate. The sequential versions of the algorithms are mostly placed around positions near the middle of the table, neither too good nor too bad. The best results are obtained with the configuration with 512 individuals in 50-D and 100-D and with 1,024-D in 200-D which has a good balance between the population size and the number of iterations.

The next study consisted in analyzing the performance *on each function* of the configurations based on sending individuals against the equivalent ones based on sending models. For this task, all the configurations were grouped in each of the 7 possible groups (36 configurations per group) for sending individuals or models based on the values of the specific parameters of Table 1. Then, the average rank and the number of wins was obtained for each group (36 values per group) following a procedure similar to the one described before.

Table 2: Average Ranks and nWins on 50-D functions

Size	**#islands**	**period**	**topology**	**rate**	**emm. selec**	**model**	**avg. ranking**	**nWins**
Best Configurations								
512	8	10	ring	0.2	best	-	32.45	240
512	8	10	ring	0.1	best	-	33.22	245
512	16	10	ring	0.2	best	-	35.59	235
512	16	10	ring	0.1	best	-	37.90	232
512	8	20	ring	0.2	best	-	38.13	235
Sequential Configurations								
512							93.40	68
256							110.13	41
1024							126.04	-23
128							129.22	-11
64							187.18	-140
2048							190.0	-148
Worst Configurations								
2048	16	20	ring	-	-	mixture	209.91	-201
512	16	40	ring	-	-	convex	214.45	-180
512	16	20	a2a	-	-	convex	215.82	-183
512	16	40	ring	-	-	uniform	219.82	-184
512	16	10	a2a	-	-	convex	220.00	-188

Tables 5, 6 and 7 present the results in 50, 100 and 200 dimensions. For each function, the best average rank is highlighted on both tables. From these results, it can be seen that, in 9 out of 11 functions, the groups based on sending individuals obtained a superior average ranking and number of wins than the ones based on sending models. Only on two functions, f4 and f8, the configurations based on combining the models with the convex procedure obtained a better average rank and number of wins than the sending individuals configurations. Within the individuals configurations, sending the best 10% of individuals, achieves the best performance in most of the functions. It seems that sending a small number of the best individuals is the best overall strategy for most of the common configurations.

With the average ranks per function for each of the eight groups, a global analysis was also conducted. These results are shown in the last row of Tables 5, 6 and 7. It can be seen that the conclusions from the previous analysis are also confirmed in the global one: all the sending individuals groups obtain better values in both average rank and number of wins than the sending models groups. The best results are also achieved by the group which sends the best 10% of individuals.

5 Conclusions

This paper presents an extensive comparison of several configurations of dEDAs over a standard benchmark of continuous functions in 50, 100 and 200 dimensions. Several analyses from different points of view have been carried out and non-parametrical tests have been applied. The attention has been put to which method for exchanging information between dEDAs, the migration of individuals or the migration of probabilistic models, is the best approach for a researcher who would like to apply the $UMDA_g$ dEDAs in a continuous domain. From this perspective, the results from this study clearly express that, for most of the functions, the exchange of individuals obtains significantly better results than the alternative approach of sending models. Furthermore, the question of whether the dEDAs configurations obtain better results than their equivalent sequential versions has also been addressed: the study shows that the best dEDAs configurations outperform the best results of the sequential counterparts. However, it is neccesary to carry out a correct selection of the distributed parameter values in order to achieve these results.

6 Acknowledgment

This work was supported by the Madrid Regional Education Ministry and the European Social Fund, financed by the Spanish Ministry of Science TIN2007- 67148 and supported by the Cajal Blue Brain Project. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Centro de Supercomputación y Visualización de Madrid (CeSViMa) and the Spanish Supercomputing Network.

References

[1] E. Alba. Parallel evolutionary algorithms can achieve super-linear performance. *IPL*, 82(1):7–13, 2002.

[2] E. Alba and J. M. Troya. Improving flexibility and efficiency by adding parallelism to genetic algorithms. *Statistics and Computing*, 12(2):91–114, 2002.

[3] J. P. Cohoon, S. U. Hegde, W. N. Martin, and D. S. Richards. Punctuated equilibria: A parallel genetic algorithm. In *Proc. of the 2nd ICGA*, pages 148–154, July 1987.

[4] L. Delaossa, J. A. Gámez, and J. M. Puerta. Migration of probability models instead of individuals: An alternative when applying the island model to edas. In *PPSN VIII*, pages 242–252, 2004.

[5] L. Delaossa, J. A. Gámez, and J. M. Puerta. Initial approaches to the application of islands-based parallel EDAs in continuous domains. In *Int. Conf. Workshops on Parallel Processing*, pages 580–587, 2005.

[6] L. Delaossa, J. A. Gámez, and J. M. Puerta. Initial approaches to the application of islands-based parallel edas in continuous domains. *Journal of Parallel and Distributed Computing*, 66(8):991–1001, August 2006.

[7] J. Jaros and J. Schwarz. Parallel BMDA with probability model migration. In *Proc. of the IEEE CEC*, pages 1059–1066, 2007.

[8] J. Jaros and J. Schwarz. Parallel BMDA with an aggregation of probability models. In *Proc. of the IEEE CEC*, pages 1683–1690, 2009.

[9] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Optimization in continuous domains by learning and simulation of Gaussian networks. In *Proc. of GECCO*, pages 201–204, 2000.

[10] P. Larrañaga and J. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation.* Kluwer Academic Publisher, 2002.

[11] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms.* Springer-Verlag New York, Inc., 2006.

[12] S. Muelas, J. M. Peña, V. Robles, A. LaTorre, and P. de Miguel. Machine learning to analyze migration parameters in parallel genetic algorithms. In *Innovations in Hybrid Intelligent Systems*, volume 44 of *Advances in Soft Computing*, pages 199–206, November 2007.

[13] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions i. binary parameters. In *PPSN IV*, pages 178–187, 1996.

Table 3: Average Ranks and nWins on 100-D functions

Size	**#islands**	**period**	**topology**	**rate**	**emm. selec**	**model**	**avg. ranking**	**nWins**
Best Configurations								
512	8	20	ring	0.1	best	-	50.00	227
512	8	20	ring	0.2	best	-	50.23	223
512	8	10	ring	0.2	best	-	50.86	224
512	8	10	ring	0.1	best	-	52.09	219
512	8	10	ring	0.2	random	-	54.68	216
Sequential Configurations								
512							103.86	48
1024							117.41	-18
256							135.32	-23
128							147.41	-31
2048							157.95	-112
64							206.55	-166
Worst Configurations								
1024	16	10	a2a	-	-	convex	218.36	-190
512	16	40	ring	-	-	convex	219.09	-196
512	16	20	a2a	-	-	convex	219.45	-196
512	16	40	ring	-	-	uniform	223.55	-214
512	16	10	a2a	-	-	convex	224.82	-212

Table 4: Average Ranks and nWins on 200-D functions

Size	**#islands**	**period**	**topology**	**rate**	**emm. selec**	**model**	**avg. ranking**	**nWins**
Best Configurations								
1024	8	20	ring	0.1	best	-	55.18	198
1024	8	20	ring	0.2	best	-	55.77	196
1024	8	10	ring	0.2	best	-	57.50	192
1024	16	10	ring	0.2	best	-	57.95	192
1024	16	10	ring	0.1	best	-	58.41	195
Sequential Configurations								
1024							111.00	34
512							119.27	11
256							143.23	-46
2048							151.77	-100
128							184.64	-115
64							213.09	-186
Worst Configurations								
512	16	40	ring	-	-	convex	224.73	-247
512	16	20	ring	-	-	uniform	225.09	-238
512	16	20	a2a	-	-	convex	225.82	-246
512	16	10	a2a	-	-	convex	228.82	-253
512	16	40	ring	-	-	uniform	229.45	-247

Table 5: Average ranking and nWins per function on 50-D functions

Func.	**0.10-best**	**0.10-random**	**0.20-best**	**0.20-random**	**convex**	**mixture**	**uniform**
f1	**84.28**/ 6	89.53/ 3	93.32/ 3	97.04/ 0	219.86/-6	135.03/-2	166.44/-4
f2	91.64/ 3	**90.42**/ 3	96.11/ 3	94.86/ 3	209.44/-6	127.28/-2	175.75/-4
f3	76.69/ 5	**68.92**/ 5	94.17/ 1	93.53/ 1	221.75/-6	148.92/-2	181.53/-4
f4	129.78/-1	144.94/-2	124.36/ 1	141.19/-2	**91.00**/ 5	128.67/ 0	125.56/-1
f5	**82.04**/ 6	92.74/ 3	91.06/ 3	103.40/ 0	220.69/-6	133.08/-2	162.49/-4
f6	**88.71**/ 4	91.67/ 4	93.93/ 3	96.68/ 1	204.65/-6	141.40/-2	168.46/-4
f7	107.94/ 5	120.94/-1	**106.25**/ 5	117.17/-1	136.00/-2	147.51/-2	149.68/-4
f8	139.53/-1	146.17/-1	141.11/-1	142.33/-1	**35.78**/ 6	144.14/-1	136.44/-1
f9	117.28/ 1	120.79/ 0	**110.04**/ 3	119.10/-1	133.50/-2	150.11/-1	134.68/ 0
f10	**109.81**/ 4	115.40/ 1	108.64/ 5	112.68/ 2	153.19/-4	138.10/-4	147.68/-4
f11	115.19/ 2	119.86/-1	**109.69**/ 4	117.61/ 0	132.97/-2	153.86/-3	136.31/ 0
All	**103.89**/5	109.21/2	106.24/4	112.32/1	159.89/-4	140.73/-4	153.18/-4

Table 6: Average ranking and nWins per function on 100-D functions

Func.	**0.10-best**	**0.10-random**	**0.20-best**	**0.20-random**	**convex**	**mixture**	**uniform**
f1	85.46/ 3	90.33/ 2	**81.07**/ 5	88.53/ 2	222.12/-6	142.44/-2	175.54/-4
f2	97.58/ 3	**92.00**/ 3	95.39/ 3	93.67/ 3	209.17/-6	126.47/-2	171.22/-4
f3	**74.19**/ 5	69.61/ 4	93.64/ 2	93.92/ 1	225.06/-6	146.25/-2	182.83/-4
f4	127.33/-1	141.58/-1	123.00/-1	139.17/-1	**98.75**/ 4	129.53/ 0	126.14/ 0
f5	**84.22**/ 3	94.65/ 3	90.57/ 3	95.58/ 3	223.08/-6	133.67/-2	163.72/-4
f6	84.22/ 4	**83.32**/ 5	98.46/ 1	92.96/ 2	219.31/-6	141.46/-2	165.78/-4
f7	103.56/ 3	109.17/ 3	**102.28**/ 3	107.03/ 3	182.42/-6	134.51/-3	146.54/-3
f8	124.86/ 2	149.72/-2	129.72/ 0	141.25/-1	**30.11**/ 6	155.67/-2	154.17/-3
f9	**105.53**/ 3	110.86/ 2	108.33/ 2	110.76/ 1	160.92/-5	160.33/-4	128.76/ 1
f10	100.64/ 3	**96.86**/ 3	107.51/ 3	107.15/ 3	176.50/-6	151.56/-3	145.28/-3
f11	**106.57**/ 2	109.83/ 2	107.89/ 2	109.49/ 2	162.64/-5	160.61/-5	128.47/ 2
All	**99.46**/4	104.35/3	103.44/3	107.22/2	173.64/-5	143.86/-4	153.49/-3

Table 7: Average ranking and nWins per function on 200-D functions

Func.	**0.10-best**	**0.10-random**	**0.20-best**	**0.20-random**	**convex**	**mixture**	**uniform**
f1	90.08/ 3	**85.00**/ 3	96.08/ 3	87.92/ 3	227.25/-6	129.08/-2	170.08/-4
f2	98.36/ 3	**97.61**/ 3	100.19/ 3	101.00/ 3	201.78/-6	125.47/-2	161.08/-4
f3	72.44/ 5	**69.11**/ 5	98.78/ 1	94.53/ 1	226.92/-6	136.36/-2	187.36/-4
f4	128.92/ 0	128.28/ 0	126.31/ 0	121.83/ 0	**115.36**/ 0	131.94/ 0	132.86/ 0
f5	88.83/ 3	**83.50**/ 4	96.92/ 2	94.07/ 3	227.22/-6	125.22/-2	169.74/-4
f6	82.79/ 2	**69.44**/ 6	97.54/ 2	94.57/ 2	225.31/-6	142.44/-2	173.40/-4
f7	**93.76**/ 4	99.60/ 0	96.35/ 6	97.39/ 2	184.68/-6	153.67/-3	160.06/-3
f8	113.58/ 2	144.86/-2	117.06/ 2	127.14/ 2	**26.33**/ 6	179.28/-5	177.25/-5
f9	**100.81**/ 3	104.89/ 3	106.89/ 2	106.92/ 3	181.00/-5	159.56/-5	125.44/-1
f10	**92.31**/ 4	96.92/ 4	103.14/ 3	106.25/ 1	189.94/-6	152.53/-3	144.42/-3
f11	**101.14**/ 3	104.58/ 4	104.17/ 3	108.69/ 2	180.97/-5	159.86/-5	126.08/-2
All	**96.63**/5	98.53/3	103.94/2	103.66/2	180.61/-6	145.03/-3	157.07/-3