Noname manuscript No.
(will be inserted by the editor)

A New Initialization Procedure for the Distributed Estimation of Distribution Algorithms

Santiago Muelas · José-María Peña · Antonio LaTorre · Víctor Robles

the date of receipt and acceptance should be inserted later

Abstract Estimation of Distribution Algorithms (EDAs) are one of the most promising paradigms in today's evolutionary computation. In this field there has been an incipient activity in the so called parallel Estimation of Distribution Algorithms (pEDAs). One of these approaches is the distributed Estimation of Distribution Algorithms (dEDAs). This paper introduces a new initialization mechanism for each of the populations of the islands based on Voronoi cells. In order to analyze the results, a series of different experiments using the benchmark suite for the special session on Real-parameter Optimization of the IEEE CEC 2005 conference has been carried out. The results obtained suggest that the Voronoi initialization method considerably improves the performance obtained from a traditional uniform initialization.

Keywords Estimation of Distribution Algorithms · Distributed Evolutionary Algorithms · Initialization · Continuous Optimization

1 Introduction

There is currently a wide range of optimization tools to deal with many complex problems in very different fields, such as engineering, bioinformatics or scheduling. Evolutionary techniques are receiving more and more attention in these complex optimization scenarios. Their exploratory characteristics play a significant role in problems with difficult fitness landscapes. On the other hand, the use of population-based Evolutionary Algorithms (EAs) has the drawback of the number of evaluations required to guide the search.

Since Evolutionary Algorithms are inherently parallel, the so-called Parallel Evolutionary Algorithms (pEAs) have been studied as an alternative to tackle one of the aspects of this drawback [1]. A successful example of Parallel Evolutionary Algorithms is the model called distributed Evolutionary Algorithms (dEAs) or island model. In this model, independent nodes execute a local EA, exchanging information under given conditions. The information exchanged provides the mechanism to enhance the local population with the improvements already achieved in other nodes (populations). This kind of EAs seems to improve the numerical and runtime behavior of the basic algorithm in many cases [2,3].

Estimation of Distribution Algorithms (EDAs) [4, 5], which have become a fruitful new paradigm for population-based Evolutionary Computation, have not been an exception. Parallel Estimation of Distribution Algorithms (pEDAs) have been proposed with a broad range of possible parallel approaches [6–8].

A relevant aspect in EAs is the initialization of the starting population. This issue is important to provide a good supply of initial individuals to start up the stochastic search. When there is more than one population, the influence of the initial individuals on each of the subpopulations should also be considered.

This paper proposes a new initialization procedure based on a topological tool (Voronoi diagrams) to restrict the initial search space of the different nodes of the island model. Within each island, the initialization method applies the $D2$ Method [9], an heuristic used for the maximum diversity problem, for generating a diverse set of individuals which tries to maximize the coverage of the solutions space. For analyzing the ef-

S. Muelas, J.M. Peña, A. LaTorre and V. Robles
Department of Computer Systems Architecture and Technology
Facultad de Informática
Universidad Politécnica de Madrid, Spain
E-mail: [smuelas,jmpena,atorre,vrobles]@fi.upm.es

fects of the proposed mechanism, an experiment with 108 distributed EDA configurations has been conducted over the benchmark suite of the special session on Real-Parameter Optimization of the IEEE CEC 2005 conference [10]. The results have been validated using a statistical non-parametric test.

The rest of the paper is organized as follows: Section 2 presents an overview of the parallel evolutionary and initialization techniques. Section 3 details the proposed technique and the rationale behind it. In Section 4 the experimental scenario is described in detail. Section 5 presents and comments on the results obtained and lists the most relevant facts extracted from this analysis. Finally, Section 6 contains the concluding remarks obtained from this study.

2 Related Work

2.1 Estimation of Distribution Algorithms: EDAs

EDAs are non-deterministic, stochastic heuristic search strategies that are part of the Evolutionary Computation paradigm [4,5]. In EDAs, multiple solutions or individuals are created every generation, evolving successively until a satisfactory solution is achieved. In brief, the characteristic that most differentiates EDAs from other evolutionary search strategies such as Genetic Algorithms (GAs) is that the evolution from one generation to the next one is achieved by estimating the joint probability distribution of a set of individuals followed by sampling the induced model. This avoids the use of crossing and mutation operators, thus reducing the number of parameters that are required by EDAs. The general schema of the algorithm is described in Algorithm 1.

Algorithm 1 EDA schema

```
Create initial population (popSize individuals) D_0
i = 0
repeat
    Evaluate population D_i
    D_i^s = Select N ≤ popSize individuals from D_i
    Estimate a new model M from D_i^s
    D' = Sample popSize individuals from M
    Evaluate D'
    D_{i+1} = Select popSize individuals from D_i ∪ D'
    i = i + 1
until stop criterion
```

Graphical models have been commonly used for estimating the joint probability distribution. Some authors have proposed Bayesian networks to represent the probability distribution for discrete domains, whereas Gaussian networks are usually employed for continuous domains. 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 dependences between pairs of variables; and multivariate models, in which there is no limitation on the number of dependences. In this study, we are going to focus on the Univariate Marginal Distribution Algorithm for Gaussian Models ($UMDA_g$) [5] because it has usually been considered as baseline for comparison. Also, as a result of its simplicity, it is easier to identify and analyze the benefits coming from the proposal. $UMDA_g$ uses the normal distribution to model the density of each variable. Therefore, the induction of the model is reduced to the estimation of μ and σ^2 of each variable.

2.2 Distributed Estimation of Distribution Algorithms: dEDAs

In the distributed Evolutionary Algorithm[1], the overall 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 [11]: (i) Migration frequency: How often (in generations) is information sent?, (ii) Migration rate: How many individuals migrate each time?, (iii) Information selection: What information is selected to migrate?, (iv) Acceptance policy: How are the incoming information and the local algorithm state combined? and (v) Migration topology: Which island sends information to which other?

Close scrutiny of migration parameters [12] has verified that, even though EAs with small populations risk being trapped in a local optimum, an appropriate migration strategy can avoid a suboptimal solution from dominating all the populations. This appropriate strategy must be adjusted between the limits of a low interaction (which would practically imply the execution of N independent algorithms) and an excessive interaction (that would lead to the predominance of only one solution). A correct configuration can help to obtain better results with fewer evaluations, but configuring these optimal parameters is not a simple issue [13–15].

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

2.3 Initialization

The initialization of population-based Evolutionary Algorithms is hardly addressed in the literature [16]. Nevertheless, every expert in the field agrees that a bad initialization can make evolution to converge prematurely at suboptimal solutions.

In many cases, the initialization process depends on the application field if an approximate solution to the problem is known. In [17], Ramsey concluded that, initializing the population with members of previously seen states, accelerated the learning in a changing environment. Otherwise, if the individuals of the population can be built through certain heuristic techniques, these could be a good starting point to reach the optimum [18]. However, this strategy has general drawbacks: (i) it completely depends on the field of application and (ii) it may involve biasing the search process towards certain kinds of solutions (possibly others than the optimal ones). Another approach is to use other metaheuristic algorithms (with different fitness functions) for initializing the population [19]. In [20] it was proposed to develop the initial population with a good randomized heuristic. In [21], this recommendation was followed by using the construction phase of a GRASP algorithm as the initial population of a Genetic Algorithm.

Island models are especially sensitive to initialization, not only for the aforementioned reasons, but also because of the possible mutual dependence between the different populations of the islands. There is very little literature on this topic and, therefore, this is one of the aspects to be studied in depth.

3 Contribution

In this paper, a new initialization mechanism for dEDAs is presented. The main idea of this procedure is to use a Voronoi tessellation [22] to define a partition set of the solution space in which each island or node will start its own exploration. The proposal applies several steps as presented in Figure 1.

In order to create the tesselation, a set of reference points (r_i) need to be created and assigned to each of the n islands. The initial population of island i will be a set of diverse points in the solution space which are closer to the i-th reference point than to any other reference point.

In order to avoid the generation of small partitions, we have applied two methods for selecting a good set of diverse reference points. The first method uses a controlled randomization and frequency memory procedure for generating a set $S = \{s_1, s_2, \ldots, s_N\}, N > n$ of initial diverse individuals. This method is influenced by the work in [23] for a Scatter Search algorithm. The procedure starts with the division of the range of each dimension into sr subranges of equal size. Then, for each generated individual, a subrange for each dimension is selected based on the inverse probability value of the frequency count associated with the sub-range. Finally, a value is uniformly generated within the selected interval and the frequency count associated to the subrange is incremented.

The second method takes the set of N points and extracts the $N-n$ individuals with the minimum distance between any pair of points. This procedure is based on the $D2$ Method presented in [9] for the maximum diversity problem. This method was chosen because it provides a good balance between the diversity of the individuals and the speed of computation. The general schema of the algorithm is described in Algorithm 2.

Algorithm 2 D2 Method

```
Sel = S
while |Sel| > n do
  s_i^* = argmin_{s_i ∈ Sel}{d(s_i, Sel)}
  Sel = Sel − {s_i^*}
end while
```

The distance between an individual s_i and a set $X = \{s_j : j \in I\}$ is defined as follows:

$$d(s_i, X) = min\{d(s_i, s_j)/s_j \in X\}$$

For our experiments with continuous problems, the Euclidean distance has been considered between every pair of individuals.

Once the reference points are created, the next action to execute is the generation of the population of each island. For this task, two steps are applied one after another. First, $k * popSize$ individuals are created and distributed uniformly among the islands, i.e, each island is assigned $k * popSize/n$ individuals. Each new individual is assigned to the island which distance to its reference point is minimum. This procedure is described by the pseudocode of Algorithm 3. Then the $D2$ Method is applied to each set of individuals so that the final island set contains the most diverse $popSize/n$ solutions.

For the purpose of clarifying the effects of the procedure, a simple example is provided. Figure 2 details the results of initializing a 2 dimensional function with the new method and with the traditional uniform approach. The cell lines in the new method diagram delimit the individuals that would be assigned to each island.

Fig. 1: Initialization Procedure

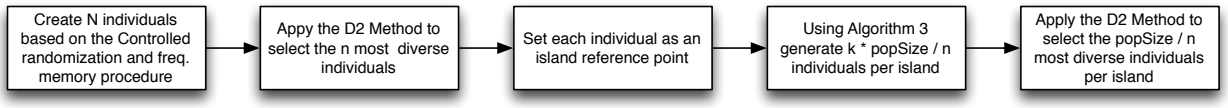
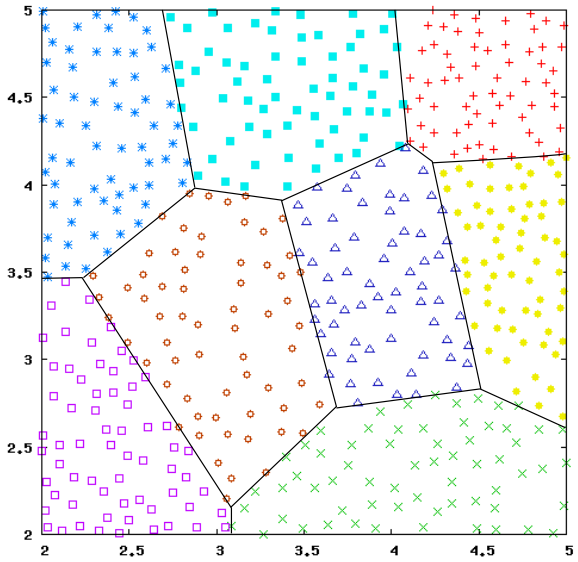


Fig. 2: Initialization Comparison

(a) Voronoi Initialization



(b) Random Uniform Initialization

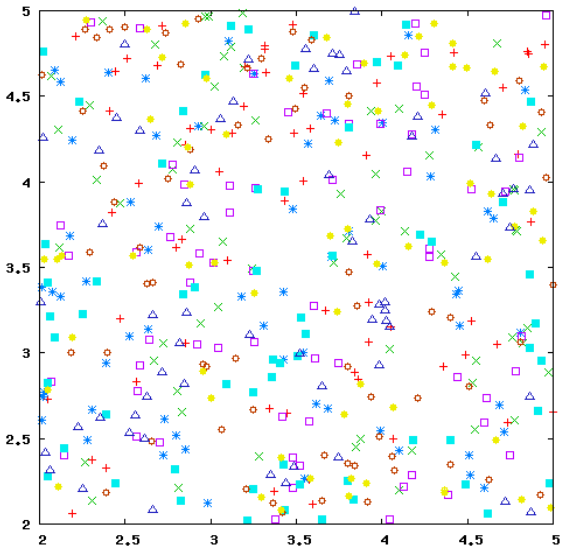


Table 1: Parameters chosen for the experiments

Population Size:	64, 100 and 200 individuals per island (i.e 512, 800 and 1600 of global population). For the proposed method, k has been fixed to 10.
Offspring size:	Equal to the population size
Selected individuals for learning:	100% of the population size
Learning Model:	$UMDA_g$
N# islands:	8
Migration ratio:	1, 4 and 8 individuals
Migration period:	Migrate every 10, 20 or 40 generations
Acceptance Criterion:	Select the best individuals between the immigrants and the resident population
Topology:	Ring and Hypercube
Selected emigrants:	best and random policies
Full Elitism:	Best individuals from the parent and the offspring populations

Algorithm 3 Population initialization

for $i = 0$ to n **do**
  $Population_i = \emptyset$
end for
while $|\bigcup_{i=0}^{i=n} Population_i| < k * PopSize$ **do**
  $newindividual = GenerateARandomIndividual$
  Let i^* / $d(newindividual, s_{i^*}) = d(newindividual, S)$
  if $|Population_{i^*}| < k * PopSize/n$ **then**
    $Population_{i^*} = Population_{i^*} \cup newindividual$
  end if
end while

Rationale: Our approach carries out a systematic initialization procedure following two criteria: (i) homogeneous coverage of the whole solution space, (ii) no overlap of the solution space explored by each island.

4 Experimental Scenario

In order to carry out the experimental validation, we have selected the benchmark suite of the special session on Real-Parameter Optimization of the IEEE CEC

Table 2: #N of configurations in 10D with significant differences

	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	
Voronoi ≻ Uniform	4	11	3	2	0	9	0	0	0	1	
Uniform ≻ Voronoi	0	0	0	6	2	1	30	0	6	3	
	f16	f17	f18	f19	f20	f21	f22	f23	f24	f25	Sum
Voronoi ≻ Uniform	1	1	63	59	57	59	18	62	83	31	464
Uniform ≻ Voronoi	15	12	0	0	0	1	0	0	0	0	76

Note: A≻B represent that the results of A were statistically better than those of B.

Table 3: #N of configurations in 30D with significant differences

	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	
Voronoi ≻ Uniform	2	14	1	2	0	0	0	2	1	0	
Uniform ≻ Voronoi	1	0	1	4	2	0	0	0	0	11	
	f16	f17	f18	f19	f20	f21	f22	f23	f24	f25	Sum
Voronoi ≻ Uniform	1	0	13	21	20	0	5	1	0	29	112
Uniform ≻ Voronoi	3	3	0	0	0	0	0	3	0	0	28

Note: A≻B represent that the results of A were statistically better than those of B.

2005 conference. This is a standard test that has been extensively used in recent years for analyzing algorithms on a continuous domain. The benchmark contains 25 generic test problems. Most of these functions are variations of well-known test functions through rotation, shifting and hybridization. Since all the participants of the special session solved the first 5 functions, functions f1-f5 have not been considered for the experiments. Furthermore, the 50 dimensional functions have not been considered since several algorithms of the session did not execute them and were also not considered for the comparison analysis of the original session. For each function, 25 independent executions were carried out for both 10 and 30 dimensions having a fixed number of fitness evaluations of 10^5 and $3 \cdot 10^5$ respectively. The performance criterion is the distance (error) between the best individual found and the global optimum in terms of fitness value.

With the purpose of analyzing the influence of the proposed strategy, 108 dEDA configurations from the values displayed on Table 1 were executed with both Voronoi and uniform initializations. A set of sequential EDA UMDA$_g$ algorithms with the same global and island populations (64, 100, 200, 512,800 and 1600) were also executed.

The number of islands was fixed to 8. The proposed procedure obtains better results with a high number of islands but, because of the maximum number of evaluations of the benchmark, a higher number would imply a reduced number of individuals per island (not appropriate for EDAs) or a considerably reduced number of iterations. Therefore, the selected number offers a good trade-off between the number of islands and the number of iterations.

5 Analyzing the Results

As mentioned in the previous section, 108 uniformly initialized configurations were compared against their equivalent Voronoi configurations. For each problem, each pair of uniform and Voronoi configurations were compared with a non-parametric Wilcoxon rank-sum test with a significance level of $\alpha = 0.01$. Tables 2 and 3 present the number of comparisons per function in which the results were statistically significant as well as the aggregated number for all the functions. It can be seen that Voronoi configurations clearly obtain a higher number of significant results than their uniform counterparts, this number being higher with the 10 dimensional functions and with the hardest 8 functions of the benchmark (18-25). Since the number of partitions is the same for both the 10 and the 30 dimensional functions, each island is less affected by the local optima in the 10 dimensional scenario and therefore obtains better results.

A global comparison of the best configurations of both types of initializations per population size and the sequential EDAs was also carried out. The criterion used for selecting the best configurations was: for each population size, select the configuration with the minimum sum of average errors over all the functions in both 10 and 30 dimensions. Similarly to the analysis used in the CEC'05 special session, each algorithm was ranked at each function according to its average

error. Although the objective of this study was not to tune an EDA algorithm to obtain the best results in the benchmark, the best algorithm from the special session, G-CMA-ES, has also been included in the analyses. Table 4 presents the average ranking of all the algorithms. Here, Voronoi configurations also obtain better results than their equivalent uniform ones being the Voronoi configuration of 512 individuals the best EDA algorithm. For the purpose of determining the significance of these results, a non-parametric paired Wilcoxon rank-sum test was applied. Table 5 shows the $p-values$ of the comparisons of the best EDA configuration (Voronoi with 512 individuals) against the rest of the algorithms. It can be observed that the best Voronoi configuration is $\alpha = 0.01$ significantly better in each comparison to the rest of the EDAs algorithms in both 10 and 30 dimensions except against the uniform initialized configuration of 512 individuals in 30 dimensions and the Voronoi configuration of 800 individuals.

Table 4: Average ranking of the best algorithms

algorithm	10D	algorithm	30D
G-CMA-ES	1.75	vorodeda512	3.075
vorodeda512	2.8	unifdeda512	3.15
unifdeda512	4.72	G-CMA-ES	3.85
vorodeda800	5.3	vorodeda800	4.57
unifdeda800	6.72	unifdeda800	5.82
vorodeda1600	7.1	vorodeda1600	7.22
eda64	7.1	unifdeda1600	7.52
eda512	7.57	eda128	7.87
eda128	7.8	eda800	7.87
unifdeda1600	8.32	eda512	8.02
eda800	8.42	eda1600	9.3
eda1600	10.3	eda64	9.7

Table 5: $p-values$ of the comparisons of the best configurations

vorodeda512 VS	10D		30D	
unifdeda512	1.3065E-04	$\surd$	2.1046E-01	$\times$
vorodeda800	2.6585E-02	$\times$	4.1275E-02	$\times$
unifdeda800	1.8120E-05	$\surd$	5.0831E-04	$\surd$
vorodeda1600	2.4300E-03	$\surd$	3.1243E-04	$\surd$
unifdeda1600	9.5367E-07	$\surd$	1.4628E-04	$\surd$
eda64	6.0396E-03	$\surd$	1.9073E-06	$\surd$
eda128	6.6757E-05	$\surd$	9.5367E-06	$\surd$
eda512	4.7684E-06	$\surd$	1.3351E-05	$\surd$
eda800	9.5367E-07	$\surd$	1.3351E-05	$\surd$
eda1600	9.5367E-07	$\surd$	2.8610E-06	$\surd$
G-CMA-ES	9.9953E-01	$\times$	5.7959E-01	$\times$

$\surd$ represents that the *p-value* is $\alpha = 0.01$ significant
$\times$ represents that the *p-value* is not significant

It can also be seen that this configuration is clearly superior to any of the sequential EDAs executed. Also, it can be observed that the dEDA configurations with the fewer number of individuals tend to obtain better results. This is partially due to the constraint in the maximum number of fitness evaluations imposed by the benchmark, which allows them to execute for more iterations. Finally, when analyzing the comparison against a specially tuned algorithm for the session, the G-CMA-ES algorithm, it can be seen that although the G-CMA-ES is significantly better than the best dEDA configuration in 10 dimensions ($p-value = 0.005$), with the 30 dimensional functions, the G-CMA-ES has worse rank and its results are not significantly better.

Figure 3 shows the evolution of the average score for the 10 dimensional $f20$ function[2]. The evolution of the score of the algorithms on the next generations is almost the same for both methods but the influence of the first generations clearly increases the performance of the Voronoi configurations.

Finally, an analysis of the influence of the initialization in the migration schemes was also carried out. For this task, the total number of individuals that are exchanged throughout the evolution for each distributed configuration was measured, i.e.:

$$\frac{\#iterations}{Migration_{period}} \times Migration_{rate} \times Topology_{degree} \quad (1)$$

Table 6 presents the average of this number for the ten distributed configurations with the minimum average error for all the functions. This number represents the total number of individuals that are exchanged along the execution. For obtaining the number of individuals that were actually accepted on each island, the acceptance criterion needs to be taken into account and averaged through all the executions. In general, Voronoi configurations need to exchange fewer individuals than the uniform configurations. This effect is much clearer for the 10 dimensional functions and with bigger population sizes (which, because of the benchmark constraints, have also a smaller number of iterations). It seems that, in order to obtain the best results, the Voronoi initialized islands need less interaction with their neighboring islands so they can intensify the exploration on their isolated region (in particular, in their earlier iterations).

6 Conclusions

This paper presents a new initialization method for the Distributed Estimation of Distribution Algorithms.

[2] The evolution of the average scores in most of the other functions follows a similar pattern

Fig. 3: Evolution of the average score for the $f20$ function

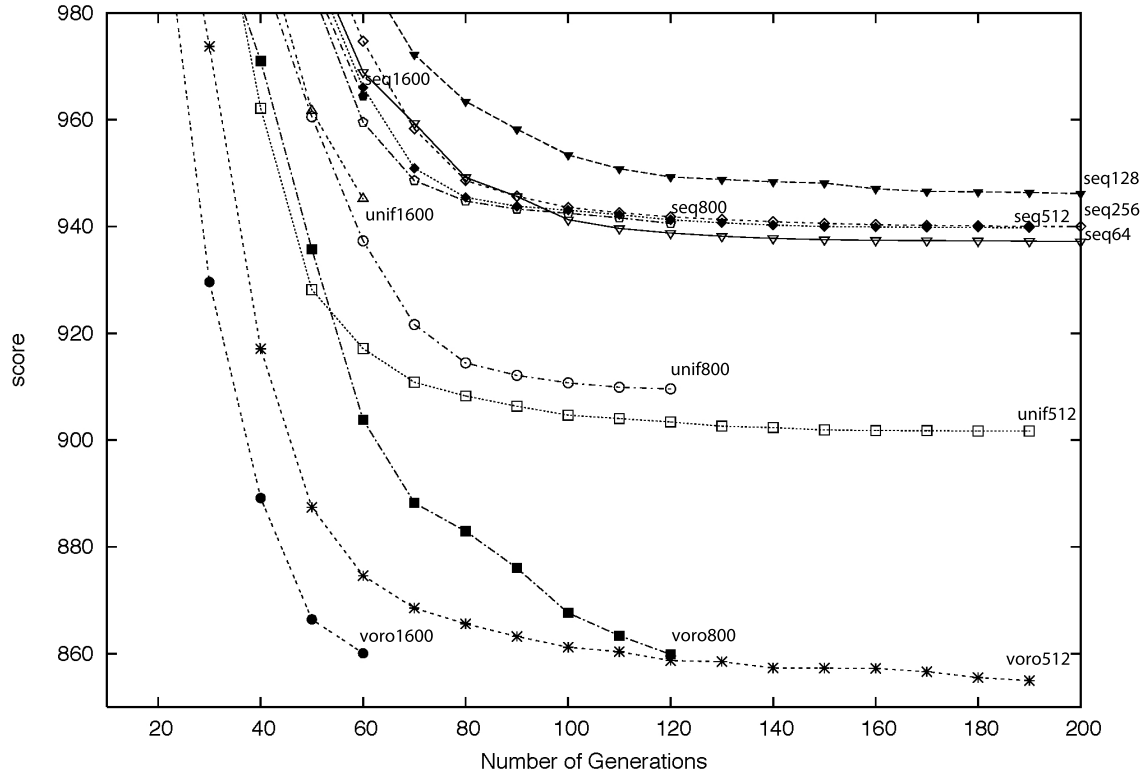


Table 6: Number of individuals exchanged between the best configurations

	10D		30D	
Popsize	Voronoi	Uniform	Voronoi	Uniform
512	331.05	480.47	244.14	295.44
800	288.75	405	180	213.75
1600	60.94	168.75	64.49	63.48

The proposed initialization is based on Voronoi cells which isolate the initial search space of each island and uses a heuristic method for uniformly covering each region of the search space. Several parameter values have been tested on the standard CEC'05 continuous benchmark suite. In order to analyze the results, non-parametrical tests were applied. The obtained results show that the best overall performance is obtained with Voronoi configurations and that, in general, the Voronoi configurations tend to improve the results of the traditional initialization method. The partition of the search space reduces the modality of the constrained regions and offers the possibility, at least in the earliest generations, to emphasize the search in the starting regions. Thus, the search for the optimal solutions is more effective. The analysis also discovered that the best Voronoi configurations need less interaction between the islands than the best uniform configuration. With a greater exchange of individuals, the beneficial properties of the proposed initialization get diluted. The Voronoi configurations also outperformed their equivalent population-sized (both global and island population sizes) sequential EDAs. This approach has been proposed for EDAs since they are more influentiated by the initial population than other evolutionary algorithms with more explorative mechanisms. However, this procedure could also improve the performance of other evolutionary algorithms by reaching a higher score or helping them to improve their convergence speed.

7 Acknowledgments

This work was supported by the Madrid Regional Education Ministry and the European Social Fund and

financed by the Spanish Ministry of Science TIN2007-67148. 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. We would also like to thank the reviewers for their suggestions.

References

1. E. Alba, Parallel evolutionary algorithms can achieve superlinear performance, Information Processing Letters 82 (1) (2002) 7–13.
2. E. Alba, J. M. Troya, Improving flexibility and efficiency by adding parallelism to genetic algorithms, Statistics and Computing 12 (2) (2002) 91–114.
3. J. L. Risco-Martín, D. Atienza, J. Hidalgo, J. Lanchares, A parallel evolutionary algorithm to optimize dynamic data types in embedded systems, Soft Computing - A Fusion of Foundations, Methodologies and Applications 12 (12) (2008) 1157–1167.
4. H. Mühlenbein, G. Paass, From recombination of genes to the estimation of distributions i. binary parameters, in: PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, 1996, pp. 178–187.
5. P. Larrañaga, J. Lozano, Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation, Kluwer Academic Publisher, 2002.
6. J. Ocenasek, Parallel estimation of distribution algorithms, Ph.D. thesis, Brno University of Technology (2001).
7. E. Bengoetxea, Inexact graph matching using estimation of distribution algorithms, Ph.D. thesis, École Nationale Supérieure des Télécommunications, Paris, France (2002).
8. L. delaOssa, J. Gamez, J. Puerta, Migration of probability models instead of individuals: An alternative when applying the island model to EDAs, Lecture Notes in Computer Science 3242 (2004) 242–252.
9. F. Glover, C. C. Kuo, K. S. Dhir, Heuristic algorithm for the maximum diversity problem, Journal of Information and Optimization Sciences 1 (19) (1998) 109–132.
10. P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the special session on real parameter optimization, Tech. rep., Nanyang Technological University (2005).
11. E. Cantú-Paz, Efficient and accurate parallel genetic algorithms, Kluwer Academic Publishers, 2001.
12. C. Petty, M. Leuze, A theoretical investigation of a parallel genetic algorithm, in: Proceedings of the 3rd Int. Conf. on Genetic algorithms, 1989, pp. 398–405.
13. D. Whitley, S. Rana, R. Heckendorn, The island model genetic algorithm: On separability, population size and convergence, Journal of Computing and Information Technology 7 (1999) 33–47.
14. E. Alba, J. M. Troya, Influence of the migration policy in parallel distributed GAs with structured and panmictic populations, Applied Intelligence 12 (3) (2000) 163–181.
15. S. Muelas, J. M. Peña, V. Robles, A. LaTorre, P. de Miguel, Machine learning to analyze migration parameters in parallel genetic algorithms, in: Innovations in Hybrid Intelligent Systems, Vol. 44 of Advances in Soft Computing, 2007, pp. 199–206.
16. L. Kallel, M. Schoenauer, Alternative random initialization in genetic algorithms, in: Proceedings of the 7th International Conference on Genetic Algorithms, 1997, pp. 268–275.
17. C. Ramsey, J. Grefenstette, Case-based initialization of genetic algorithms, in: Proceedings of the 5th International Conference on Genetic Algorithms, 1993, pp. 84–91.
18. J. Schwarz, J. Očenášek, A problem knowledge-based evolutionary algorithm KBOA for hypergraph bisectioning, in: Proceedings of the 4th Joint Conference on Knowledge-Based Software Engineering, IOS Press, 2000, pp. 51–58.
19. H. de Garis, Genetic programming: Artificial nervous systems, artificial embryos and embryological electronics, in: PPSN I: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature, 1991, pp. 117–123.
20. R. K. Ahuja, J. B. Orlin, Developing fitter genetic algorithms, INFORMS Journal on Computing 9 (3) (1997) 251–253.
21. R. K. Ahuja, J. B. Orlin, A. Tiwari, A greedy genetic algorithm for the quadratic assignment problem, Computers and Operations Research 27 (10) (2000) 917–934.
22. F. Aurenhammer, Voronoi diagrams, a survey of a geometric data structure, ACM Comput. Surv. 23 (1991) 345–405.
23. A. Duarte, R. Martí, F. Glover, Adaptive memory programming for global optimization, in: VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'09), 2009, pp. 473–481.