Cellular Evolutionary Algorithms: Evaluating the Influence of Ratio

Enrique Alba and José Mª Troya

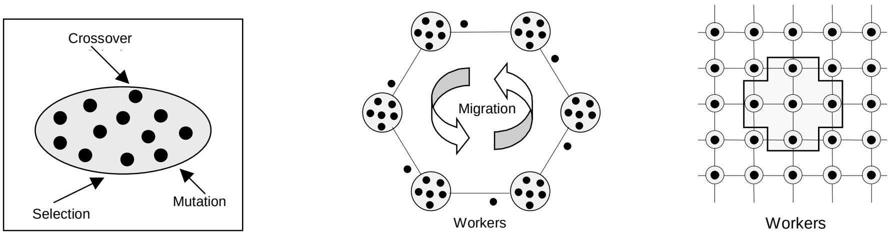
Dpto. Lenguajes y Ciencias de la Computación
E.T.S.I. Informática, Campus de Teatinos (3.2.12)
29071, Málaga, Spain
eat@lcc.uma.es

Abstract. Spatially structured evolutionary algorithms (EAs) have shown to be endowed with useful features for global optimization. Distributed EAs (dEA) and cellular EAs (cEA) are two of the most widely known types of structured algorithms. In this paper we deal with cellular EAs. Two important parameters guiding the search in a cEA are the population topology and the neighborhood defined on it. Here we first review some theoretical results which show that a cEA with a 2D grid can be easily tuned to shift from exploration to exploitation. We initially make a study on the relationship between the topology and the neighborhood by defining a ratio measure between they two. Then, we encompass a set of tests aimed at discovering the performance that different ratio values have on different classes of problems. We find out that, with the same neighborhood, rectangular grids have some advantages in multimodal and epistatic problems, while square ones are more efficient for solving deceptive problems and for simple function optimization. Finally, we propose and study a cEA in which the ratio is dynamically changed.

1 Introduction

Evolutionary algorithms (EAs) are optimization techniques for searching complex spaces for an optimum [4]. Normally, EAs use a single population (*panmixia*) although there exists a long tradition in using structured populations, especially for parallel implementations. Among the most widely known types of structured EAs, *distributed* and *cellular* algorithms are popular optimization tools [1].

Decentralizing a single population can be achieved by partitioning it in several subpopulations where island EAs are run with sparse string exchanges (*distributed EAs*), or in the form of neighborhoods (*cellular EAs*). Let us see these three EA types in Fig. 1. In distributed EAs, additional parameters controlling when migration occurs and how migrants are selected/incorporated in the target islands are needed. In cellular EAs the existence of overlapped small neighborhoods helps in exploring the search space [12]. These two kinds of EAs provide a better sampling of the search space and improve the numerical and runtime behavior of the algorithm in many cases [2].



(a) **(b)** **(c)**

Fig. 1. A panmictic EA has all its strings -black points- in the same population **(a)**. Structuring the population usually leads to distinguish between distributed **(b)** and cellular **(c)** EAs.

The main difference of a cEA with respect a panmictic EA is its decentralized selection, since the reproductive loop is performed inside each of the numerous string pools. In a cEA, one given string has its own pool defined by neighboring strings, and at the same time, one string belongs to many pools. This 2D structure with overlapped neighborhoods is used to provide a smooth diffusion of good solutions along the grid. We directly focus on a grid of strings due to its generality [10].

The importance of cellular EAs is growing for several reasons. First of all, they are endowed of an internal spatial structure that allows fitness and genotype diversity for a larger number of iterations [12]. Also, some works established the advantages of using cEAs for complex optimization tasks (high efficacy and reduced number of steps) in relation with other EA models [5] [9]. Finally, the similitude of cEAs and cellular automata [13] [15], their potential applications, and their suitability for being implemented in SIMD machines make them worth of study. We analyze these algorithms as a kind of 2D spatially structured EAs run sequentially. Using a SIMD machine would have allowed for a much quicker execution, but the search is basically the same.

Pervious works have suggested the higher efficacy obtained when using cEAs in which the grid is not perfectly square i.e., thin grids [3]. See also the results of [9] for large TSP problem instances or [5] [8] for function optimization. They suggest (but not study) that the shape of the grid influences the quality of the search.

As we will show, the size, shape, and neighborhood defined for the grid have a major impact on the behavior of the cEA. Of course, other factors such as the operators and application rates influence the search, but the same holds for any other EA model. However, decentralized selection is determined by the topology and by the neighborhood; and decentralized selection is by far the main feature of a cEA [11].

The contribution of this work is to get a deeper insight into the importance of the relationship between topology and neighborhood. We use the ratio of the cEA [10] to quantify this relationship, and then study both a static and a dynamic ratio definitions. This study is presented in connection with the kind of problem being addressed, and not generically, in the research spirit inspired from the No Free Lunch theorem [16].

The paper is organized as follows. In the next section we characterize the shape and neighborhood of the cEA. Section 3 describes the problem domains we will use in the study. In Section 4 we present the comparative results of using cEAs with different ratios in a problem-dependent manner, and then extend this study to the case of changing the ratio during the search. Finally, Section 5 provides some conclusions.

2 Characterizing the Cellular EA

In this section we include a pseudo-code description of the canonical cEA and explain the algorithm. The EA we are using as a case study is a cellular GA (cGA) (see Fig. 2). The population is structured in a toroidal 2D grid and the neighborhood defined on it -line 5- always contains 5 strings: the considered one (`position(x,y)`) plus the north, east, west, and south strings -`NEWS`-. Fitness proportional selection is used in the neighborhood (`Local_Select`) -lines 5, 6, 7- along with a two point crossover operator that yields only one child (the one having the larger portion of the best parent: `DPX1` -lines 8, 9-), and a traditional bit-flip mutation -line 10-.

In our monoprocessor implementation of this cGA, the successive populations replace each other completely -line 16-. Meanwhile, the new strings generated by local selection, crossover, and mutation are placed in a temporal population -lines 12, 13-.

This replacement step -lines 12, 13- could be implemented by using the old population (e.g. replacing if the new string is better than the old one) or not (always adding the new string to the next population). The first issue (replacement by binary tournament) is the preferred one [3]. Computing basic statistics -line 17- is rarely found in the pseudo-codes of other authors. However, it is necessary for the work and monitoring of the algorithm. The same holds for the step that computes the list of neighbors of a given string with coordinates `(x,y)`-line 5-. This operation is needed whenever it is planed to change the type of neighborhood (although we do not change it here).

After presenting the algorithm, we now characterize the grid of our cGA. We extend the work in [10] to define a “radius”. The basic difference here is that the grid is not forced to be square. We consider the grid to have a radius equal to the dispersion of $n*$ points in a circle centered in $(\bar{x},\bar{y})$ (Equation 1). This definition also provides a radius value for the neighborhood. As shown in [10], the grid-neighborhood relationship can be quantified by the relative ratio between their radii (Equation 2). Algorithms with similar ratio show a similar selection pressure.

<u>PSEUDOCODE OF A SIMPLE CELLULAR GA</u>

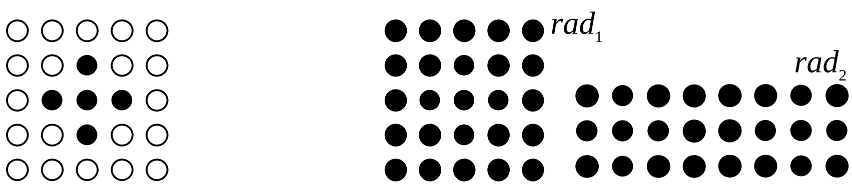
```
1. proc Steps_Up(cga):
2.    for s←1 to MAX_STEPS do
3.         for x←1 to WIDTH do
4.               for y←1 to HEIGHT do
5.                     n_list ←Compute_Neigh(cga,position(x,y));
6.                     parent1←Local_Select(n_list);
7.                     parent2←Local_Select(n_list);
8.                     DPX1(cga.Pc,n_list[parent1],n_list[parent2],
9.                          aux_ind.chrom);
10.                    Mutate(cga.Pm,aux_ind.chrom);
11.                    aux_ind.fit←cga.Fit(Decode(aux_ind.chrom));
12.                    Insert_New_Ind(position(x,y),aux_ind,
13.                                 [if_better|always],cga,aux_pop);
14.              end_for;
15.        end_for;
16.        cga.pop←aux_pop;
17.        Update_Statistics(cga);
18.   end_for;
19. end_proc Steps_Up;
```

$$rad = \sqrt{\frac{\sum (x_i - \bar{x})^2 + \sum (y_i - \bar{y})^2}{n*}} \qquad \bar{x} = \frac{\sum_{i=1}^{n*} x_i}{n*} \quad \bar{y} = \frac{\sum_{i=1}^{n*} y_i}{n*} \tag{1}$$

$$ratio_{cGA} = \frac{rad_{neighborhood}}{rad_{topology}} \tag{2}$$

Although it is called a "radius", *rad* measures the dispersion of $n*$ patterns. Other possible measures for symmetrical neighborhoods like the *radius* of a circle surrounding a rectangle containing the neighborhood or an *asymmetry coefficient* would allocate the same numeric value to different neighborhoods (which is undesirable).

When solving a given problem with a constant number of individuals ($n=n*$) the grid radius will increase as the grid gets thinner (Fig. 2b). Since the *neighborhood* is kept constant in size and shape throughout this paper (we always use NEWS, Fig. 2a), then the thinner the grid the smaller the resulting ratio.



$$rad_{linear5} = \sqrt{\frac{2+2}{3+2}} = 0.8944 \qquad rad_2 > rad_1 \quad \& \quad ratio_2 < ratio_1$$

(a) **(b)**

Fig. 2. Neighborhood NEWS -linear 5- and its radius **(a)**, and a square and thin grids of (5×5=25 and 3×8≈25) equal number of individuals with two different radii **(b)**.

Reducing the ratio means reducing the global selection intensity on the population, thus promoting exploration. This is expected to allow for a higher diversity in the algorithm that improves the results in difficult problems (e.g., multimodal and epistatic problems). On the other hand, the search performed inside each neighborhood is guiding the exploitation of the algorithm. We explore in this paper how this ratio effects on the search efficiency over a variety of domains. Changing the ratio during the search is a unique feature of cEAs that can be used to shift from exploration to exploitation at a minimum complexity without introducing just another new algorithm.

3 Test Suite

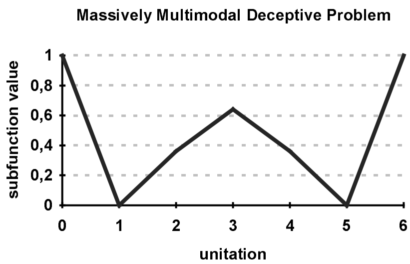
In this section we explain the optimization tasks addressed in the paper. The problems used for this study are MMDP, FMS and P-PEAKS, representing three large classes of difficulty with interest in evolutionary computation namely: deception, multimodality, and epistasis. All of them are sought for a maximum (maximization).

The massively multimodal deceptive problem (MMDP) has been specifically designed to be difficult for an EA [7]. MMDP is made up of k subproblems of 6 bits each one. The optimum has a value of k and it is attained when every subproblem is composed of zero or six ones (*unitation*). Every subproblem contributes to the fitness according to its unitation (Fig. 3). The number of local optima is quite large (22^k), while only 2^k are global solutions. Thus, the degree of multimodality is defined by k. In this paper we are using a considerably large instance of k=40 subproblems.

$$f_{MMDP}(\vec{x}) = \sum_{i=1}^{k} unitation(x_i) \tag{3}$$

6-BIT BIPOLAR DECEPTION	
unitation	**subfunction value**
0	1.000000
1	0.000000
2	0.360384
3	0.640576
4	0.360384
5	0.000000
6	1.000000

(a)



(b)

Fig. 3. Tabulated **(a)** and graphic **(b)** aspects of the basic 6-bit bipolar deception.

The Frequency Modulation Sounds (or FMS) parameter identification problem [14] has been proposed as hard real task consisting in adjusting a general model $y(t)$ to a basic sound function $y_0(t)$. The goal is to minimize the sum of square errors given by

$$f_{FMS}(\vec{x}) = \sum_{t=0}^{100} \left(y(t) - y_0(t)\right)^2 \tag{4}$$

The problem is to evolve 6 parameters $\vec{x}=(a_1, w_1, a_2, w_2, a_3, w_3)$ in order $y(t)$ to fit the target $y_0(t)$. The evolved and target models have the following expressions:

$$y(t) = a_1 sin\left(w_1 t\theta + a_2 sin\left(w_2 t\theta + a_3 sin\left(w_3 t\theta\right)\right)\right) \tag{5}$$

$$y_0(t) = 1.0 sin\left(5.0 t\theta - 1.5 sin\left(4.8 t\theta + 2.0 sin\left(4.9 t\theta\right)\right)\right) \tag{6}$$

In the above equations $\theta=2\pi/100$, and each real parameter is encoded with 32 bits in the range -6.4 to +6.35. The resulting problem is a highly complex multimodal function having strong epistasis with minimum value in f_{FMS}=0.0. We stop the algorithms when the error falls below 10^{-2}.

The last optimization task solved in this paper is a problem generator proposed in [6]. A problem generator is an easily parameterizable task which has a tunable degree of epistasis, thus allowing to derive instances with growing difficulty at will. Also, using a problem generator removes the opportunity to hand-tune algorithms to a particular problem, therefore allowing a larger fairness when comparing algorithms. With a problem generator we evaluate our algorithms on a high number of random problem instances, thus increasing the predictive power of the results for the problem class as a whole. Here we utilize the multimodal generator we call P-PEAKS [6].

The idea is to generate a set of P random N-bit strings that represent the location of the P peaks in the search space. To evaluate an arbitrary bit string, first locate the nearest peak in Hamming space. Then the fitness of the bit string is the number of bits the string has in common with that nearest peak, divided by N (Equation 7). Problems with a small/large number of peaks are weakly/strongly epistatic. Our instance uses P=100 peaks of N=100 bits each, which represents a medium-high epistasis level.

$$f_{P-PEAKS}(\vec{x}) = \frac{1}{N} \max_{i=1}^{P} \left\{ N - \mathrm{HammingD}\left(\vec{x}, Peak_i\right) \right\} \tag{7}$$

4. Analysis of the Performance with Different cEA Ratios

We now proceed to present and analyze the results of our cGA when solving these three different problems with different ratios. In all the problems we have used the same parameterization (summarized in Table 1).

Table 1. Parameterization of the cGA for solving the problems MMDP, FMS, and P-PEAKS.

Popsize	⇨	400 individuals
String Length (L)	⇨	240, 192, 100 bits, for MMDP, FMS, and P-PEAKS
Parent's Selection	⇨	Roulette Wheel + Roulette Wheel
Crossover	⇨	DPX1, p_c=1.0
Bit Mutation	⇨	p_m=1/L=0.0042, p_m=10/L=0.052, p_m=1/L=0.01
Replacement	⇨	Rep_If_Better

The selected techniques and parameter values are very usual in EA's in order to get results widely useful for the research community. We use 400 individuals (of bit length L), proportional selection of two parents within the neighborhood of 5 strings, double point crossover with probability 1.0, and bit-flip mutation with probability 1/L. The exception is for FMS, because it is very difficult to achieve an error below 10^{-2} with a mutation rate of 1/L; thus, we have used the larger value 10/L for it.

For every point in the grid, we replace it only if the new string is better than the existing one. The stopping condition for all the algorithms and problems is to find a solution. We analyze the cost of solving a problem by measuring the number of function evaluations. For every problem, the same cGA is applied in which we only change the ratios between neighborhood and topology. Three clearly different ratios are used corresponding to the grid shapes 20×20, 10×40, and 4×100 individuals. The computed ratio values with NEWS is 0.110, 0.075, and 0.031 respectively (Table 2).

Table 2. Three different grid shapes, their radius, and their ratio with NEWS.

Shape	$rad_{topology}$	rad_{NEWS}	ratio
20×20	8.15	0.8944	0.10968≈0.110
10×40	11.9	0.8944	0.07519≈0.075
4×100	28.9	0.8944	0.03096≈0.031

Besides studying the impact of using a given single ratio during the search, we have also analyzed two cGAs having dynamic ratios, that is, a change in the ratio from its initial value to a different one while the cGA is still running. One cGA varies its ratio to shift from exploitation (0.110) to exploration (0.031), and conversely, the other cGA varies its ratio to shift from exploration (0.031) to exploitation (0.110).

The change in the ratio (shape) of the grid is performed in the "middle" of a "typical" execution (t_m). We choose the middle of a typical run t_m as located in half the mean number of evaluations needed by the square shape (ratio 0.110) to solve the problem in hands (Fig. 4). Thus, t_m is problem-dependent. In short, we have five different algorithms for solving each problem, each one performing the same cEA. Three cEAs have different static ratios, and two more algorithms use dynamic ratios.

All five algorithms then utilize a different kind of ratio. In Fig. 4 we plot the shape used in every case: a constant shape (first three cases above) and two kinds of shape change. Notice in Fig. 4 that the shape *does not relate* to the number of generations that it is used, although someone could erroneously think that the narrow shape seems to span across more generations than a square one.

The results are shown in Table 3. We summarize the average results of 30 independent runs for solving the three problems. If we first analyze the case of using a static ratio (the first three columns) we can see that reducing the ratio is more efficient in FMS and P-PEAKS. For irregular (FMS) and, especially, for highly epistatic problems (P-PEAKS) the reduced selection pressure of a smaller ratio improves exploration. However, for the MMDP problem the higher pressure of the square grid (0.110) is more efficient. We note in bold the more efficient results in Table 3.

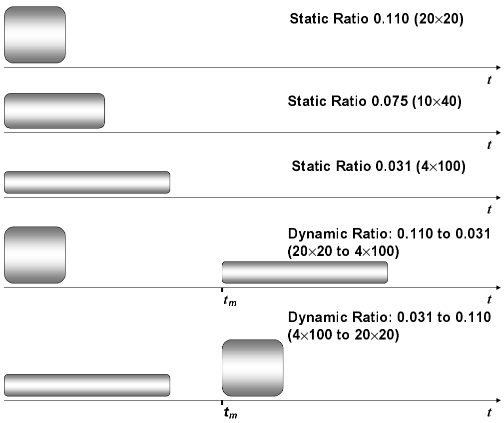


Figure 4. Static and dynamic ratio cEAs yielding 5 different algorithms. From top to bottom: a square grid, a rectangular grid, and a narrow grid, kept constant throughout the search. The two last cases show two cEAs that change from high (low) to low (high) ratio at generation t_m.

Table 3. Mean Number of Evaluations.

	0.110	0.075	0.031	0.110 / 0.031	0.031 / 0.110
MMDP	**170637.9**	237003.4	331719.6	1002900.0	1002900.0
FMS	612112.1	594093.9	593519.1	692205.2	**575353.8**
P-PEAKS	50458.1	50364.6	**48653.6**	57769.7	**48012.1**

These results with a static ratio confirm our previous work [3] in which a high ratio is more efficient in the absence of epistatis, and also for a medium degree of multimodality. A small ratio is desirable when epistasis or multimodality are high (P-PEAKS and FMS), because it speeds up the search and quickly gets out of local optima, since the grid is mapped to different search regions sought at the same time.

As to performing a change in the ratio during the search, we can see that changing from a large to a small ratio is undesirable (0.110 to 0.031) in all the studied problems. If we assume exploitation to be related to a high ratio (square grids) and exploration to be related to a small ratio (narrow grids) we can explain this result. On the contrary, we might thought that the complementary change in the ratio (from 0.031 to 0.110) will lead to a better or equal performance than using a static ratio in the domains where using a small ratio is already worth.

In effect, changing the ratio in this last way improves the efficiency when solving P-PEAKS and also it gives a new better efficiency for FMS (see Table 3). We cannot expect dynamic ratios to be desirable for all problems (NFL theorem) [16], because in fact the dynamic ratio e.g., in MMDP performs poorly: we had to stop the algorithm after a million evaluations without finding a solution. For non-epistatic problems a static square grid (high ratio) seems to be the better (initial) choice.

Since we want to be sure that these results are statistically significant, Tables 4, 5 and 6 contain the results of performing *t*-tests on the average results of Table 3. We consider a 0.05 level of significance, although slightly worse values like 0.06 and 0.08 are considered to indicate a significant trend in the analyzed result.

If we take a problem-dependent interpretation of the results we can see that a dynamic ratio is statistically worse than a static ratio for MMDP (see Table 4). It is clear for this deceptive problem that a high pressure (ratio) works out a better efficiency.

When solving a real problem with a large number of irregularities and local optima (FMS), a reduced ratio seems better than a large one (Table 3), although none of the static ratios can be statistically said to have advantages over the two others (see Table 5). A dynamic ratio cGA is worse when the pressure is high at the beginning and low at the end (0.110 to 0.031). This is common sense in evolutionary algorithms: exploitation plus exploration are not the normal phases of an EA. Conversely, the change from low to high ratio (0.031 to 0.110) has improved the results in FMS. This improvement in efficiency is clear with respect the other dynamic cGA, although we cannot ensure a statistical significance over the static ratios (see in Table 5 the associated *p*-values). In general, the FMS problem only allows us to show the superiority of enlarging (and not of diminishing) the ratio during search.

Table 4. MMDP: *t*-test indicating the significance of the results (*p*-values).

p-values	0.110	0.075	0.031	0.110 / 0.031	0.031 / 0.110
0.110	-	**0.0177**	**5.126e-06**	**2.2e-16**	**2.2e-16**
0.075	**0.0177**	-	**0.0181**	**2.2e-16**	**2.2e-16**
0.031	**5.126e-06**	**0.0181**	-	**2.2e-16**	**2.2e-16**
0.110 / 0.031	**2.2e-16**	**2.2e-16**	**2.2e-16**	-	1.0
0.031 / 0.110	**2.2e-16**	**2.2e-16**	**2.2e-16**	1.0	-

Table 5. FMS: *t*-test indicating the significance of the results (*p*-values).

p-values	0.110	0.075	0.031	0.110 / 0.031	0.031 / 0.110
0.110	-	0.6818	0.7128	0.1089	0.4115
0.075	0.6818	-	0.9904	**0.0397**	0.6536
0.031	0.7128	0.9904	-	**0.0665**	0.7087
0.110 / 0.031	0.1089	**0.0397**	**0.0665**	-	**0.0164**
0.031 / 0.110	0.4115	0.6536	0.7087	**0.0164**	-

Table 6. P-PEAKS: *t*-test indicating the significance of the results (*p*-values).

p-values	0.110	0.075	0.031	0.110 / 0.031	0.031 / 0.110
0.110	-	0.9409	**0.0829**	**2.852e-08**	**0.0212**
0.075	0.9409	-	0.1531	**3.514e-07**	**0.0531**
0.031	**0.0829**	0.1531	-	**8.03e-12**	0.4966
0.110 / 0.031	**2.852e-08**	**3.514e-07**	**8.03e-12**	**-**	**1.061e-12**
0.031 / 0.110	**0.0212**	**0.0531**	0.4966	**1.061e-12**	-

The results with the P-PEAKS problem confirm the mentioned trends because the domain is highly epistatic. A dynamic ratio is more efficient than a static ratio only when enlarging the pressure during the search. Also, from the three tested static ratio cGAs, the algorithm with a narrow grid (0.031) is the more efficient one, especially when compared with the square grid (0.110). See in Table 6 the calculated *p*-values.

5. Conclusions and Further Work

In this paper we have analyzed the performance of cellular EAs with a grid of strings folded to a torus. The used model is embedded in many other similar algorithms, and hence we hope these results to be broadly helpful. Our initial motivation has been the idea that the ratio influences the search in structured populations, assuming a constant neighborhood [3]. We wanted to get deeper in researching on *how* this occurs.

The main conclusion of this work is that a cEA with a thin grid -low ratio- is more efficient for multimodal and/or epistatic problems while it compares unfavorably with higher-ratio cEAs in non-epistatic and simple problems. A thin grid provides a twofold advantage: (1) it reduces the selection pressure (more exploration) and (2) it allows local exploitation in every neighborhood by using common variation operators or other problem-specific techniques the user might want to include (e.g. *memetic* EAs).

Whenever a small ratio shows efficient, changing the ratio from low to high during a run is also efficient, since we explicitly *run* the idea that an EA undergoes two optimization phases, namely exploration and then exploitation. Enlarging the ratio in the middle of the execution has yielded satisfactory results only in epistatic problems, and marginal benefits in multimodal problems with moderate epistasis.

As a future work it could be interesting to extend these results to new kinds of problems. Besides that, the natural next step is to make the algorithm to self-adjust the ratio depending on some genotypic or phenotypic measure such as diversity, average improvement along several generations, or other indicators of the progress of the cEA.

References

1. Alba E. and Troya J.M.: A Survey of Parallel Distributed Genetic Algorithms. Complexity, John Wiley & Sons (1999) 4(4):31-52
2. Alba E. and Troya J.M.: Influence of the Migration Policy in Parallel Distributed GAs with Structured and Panmictic Populations. Applied Intelligence 12(3) (2000) 163-181
3. Alba E., Cotta C., and Troya J.M.: On the Importance of the Grid Shape in 2D Spatially Structured GAs. Journal of Evolutionary Optimization (2000) to appear
4. Bäck T., Fogel D., and Michalewicz Z. (eds.) Handbook of Evolutionary Computation. Oxford University Press (1997)
5. Baluja S.: Structure and Performance of Fine-Grain Parallelism in Genetic Search. In: Proceedings of the 5th International Conference on Genetic Algorithms, Forrest S. (ed.). Morgan Kaufmann (1993) 155-162
6. De Jong K.A., Potter M.A., and Spears W.M.: Using Problem Generators to Explore the Effects of Epistasis. In: Proceedings of the 7th International Conference of Genetic Algorithms, Bäck T. (ed.). Morgan Kaufmann (1997) 338-345
7. Goldberg D.E., Deb K., and Horn J. Massively Multimodality, Deception and Genetic Algorithms. In: Proceedings of the *PPSN II*, Männer R., Manderick B. (eds.). North-Holland (1992) 37-46
8. Gordon V.S. and Whitley D.: Serial and Parallel Genetic Algorithms as Function Optimizers. In: Proceedings of the 5th International Conference on Genetic Algorithms, Forrest S. (ed.). Morgan Kaufmann (1993) 177-183
9. Mühlenbein H., Schomisch M., and Born J.: The Parallel Genetic Algorithm as Function Optimizer. In: Proceedings of the 4th International Conference on Genetic Algorithms, Belew R.K., Booker L.B. (eds.). Morgan Kaufmann (1991) 271-278
10. Sarma J. and De Jong K.A.: An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms. In: Proceedings of the PPSN IV, Voigt H.M., Ebeling W., Rechenberg I., and Schwefel H.P. (eds.). Springer-Verlag (1996) 236-244
11. Sarma J. and De Jong K.A.: An Analysis of Local Selection Algorithms in a Spatially Structured Evolutionary Algorithm. In: Proceedings of the 7th International Conference on Genetic Algorithms, Bäck T. (ed.). Morgan Kaufmann (1997) 181-186
12. Spiessens P. and Manderick B.: A Massively Parallel Genetic Algorithm. In: Proceedings of the 4th International Conference on Genetic Algorithms, Belew R.K., Booker L.B. (eds.). Morgan Kaufmann (1991) 279-286
13. Tomassini M.: The Parallel Genetic Cellular Automata: Application to Global Function Optimization. In: Proceedings of the International Conference on Artificial Neural Nets and GAs, Albretch R.F., Reeves C.R., Steele N.C. (eds.). Springer-Verlag (1993) 385-391
14. Tsutsui S., Ghosh A., Corne D., Fujimoto Y.: A Real Coded Genetic Algorithm with an Explorer and an Exploiter Populations. In: Proceedings of the Seventh ICGA, Bäck T. (ed.). Morgan Kaufmann (1997) 238-245
15. Whitley D.: Cellular Genetic Algorithms. In: Proceedings of the 5th International Conference on Genetic Algorithms, Forrest S. (ed.). Morgan Kaufmann (1993) 658
16. Wolpert D.H., W.G. Macready: No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation 1(1) (1997) 67-82

Acknowledgment

This work is partially supported by the CICYT project TIC1999-0754-C03-03.