The Exploration/Exploitation Tradeoff in Dynamic Cellular Genetic Algorithms

Enrique Alba and Bernabé Dorronsoro

***Abstract*—This paper studies static and dynamic decentralized versions of the search model known as cellular genetic algorithm (cGA), in which individuals are located in a specific topology and interact only with their neighbors. Making changes in the shape of such topology or in the neighborhood may give birth to a high number of algorithmic variants. We perform these changes in a methodological way by tuning the concept of ratio. Since the relationship (ratio) between the topology and the neighborhood shape defines the search selection pressure, we propose to analyze in depth the influence of this ratio on the exploration/exploitation tradeoff. As we will see, it is difficult to decide which ratio is best suited for a given problem. Therefore, we introduce a preprogrammed change of this ratio during the evolution as a possible additional improvement that removes the need of specifying a single ratio. A later refinement will lead us to the first adaptive dynamic kind of cellular models to our knowledge. We conclude that these dynamic cGAs have the most desirable behavior among all the evaluated ones in terms of efficiency and accuracy; we validate our results on a set of seven different problems of considerable complexity in order to better sustain our conclusions.**

***Index Terms*—Cellular genetic algorithm (cGA), evolutionary algorithm (EA), dynamic adaptation, neighborhood-to-population ratio.**

I. Introduction

THE APPLICATION of evolutionary algorithms (EAs) to optimization problems has been very intense during the last decade [1]. It is possible to find this kind of algorithm applied for solving complex problems like constrained optimization tasks, problems with a noisy objective function, or problems having high epistasis and multimodality. These algorithms work over a set (*population*) of potential solutions (*individuals*) by applying some stochastic operators on them in order to search for the best solutions. Most EAs use a single population (panmixia) of individuals and apply operators on them as a whole [see Fig. 1(a)]. In contrast, there exists also some tradition in using structured EAs (where the population is decentralized somehow), especially in relation to their parallel implementation.

Among the many types of structured EAs, *distributed* and *cellular* algorithms are two popular optimization tools [2], [3] (see Fig. 1). In many cases [4], these algorithms (those using decentralized populations) provide a better sampling of the search space and improve the numerical behavior of an equivalent algorithm in panmixia.

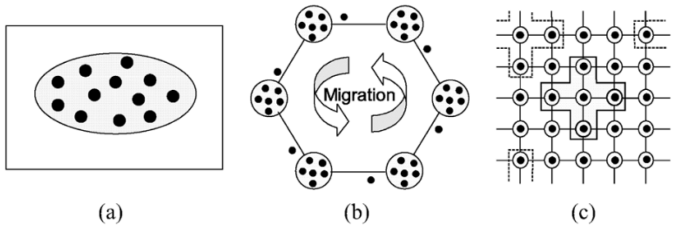

Fig. 1. (a) Panmictic EA has all its individual black points in the same population. Structuring the population usually leads to distinguish between (b) distributed, and (c) cEAs.

On the one hand, in the case of distributed EAs (dEAs), the population is partitioned in a set of islands in which isolated EAs are executed. Sparse exchanges of individuals are performed among these islands with the goal of introducing some diversity into the subpopulations, thus preventing them from getting stuck in local optima.

On the other hand, in a cellular EA (cEA) the concept of (small) *neighborhood* is intensively used; this means that an individual may only interact with its nearby neighbors in the breeding loop [5]. The overlapped small neighborhoods of cEAs help in exploring the search space because the induced slow diffusion of solutions through the population provides a kind of exploration (diversification), while exploitation (intensification) takes place inside each neighborhood by genetic operations.

These cEAs were initially designed for working in massively parallel machines, although the model itself has been adopted recently also for monoprocessor machines, with no relation to parallelism at all. This issue may be stated clearly from the beginning, since many researchers still hold in their minds the relationship between massively parallel EAs and cEAs, what nowadays represents an incorrect link: cEAs are just a different sort of EAs, like memetic algorithms, the cross generational elitist selection heterogeneous recombination cataclysmic mutation algorithm (CHC), or estimation of distribution algorithms (EDAs) [6].

The importance of cEAs is growing for several reasons. First, they are endowed of an internal spatial structure that allows fitness and genotype diversity for a larger number of iterations [7] than panmictic EAs. Also, some works [8], [9] have established the advantages of using cEAs over other EA models for complex optimization tasks, where high efficacy and reduced number of steps are needed (e.g., training neural networks [10] and solving

Manuscript received June 16, 2003; revised October 14, 2004. This work was supported in part by the Ministry of Science and Technology (MCYT) and in part by the Regional Development European Fund (FEDER) under Contract TIC2002-04498-C05-02.

E. Alba is with the Department of Computer Science, University of Málaga, Málaga 29071, Spain (e-mail: eat@lcc.uma.es).

B. Dorronsoro is with the Computing Central Services, University of Málaga, Málaga 29071, Spain (e-mail: dorronsoro@uma.es).

Digital Object Identifier 10.1109/TEVC.2005.843751

SAT problems [11], etc.). Finally, the similarity between cEAs and cellular automata (formally, proven in [12] and [13]), their potential applications, and their suitability for including local search embedded algorithms make them worth of study.

In the literature, there are additional results (like [9] for large TSP problem instances, or [8] and [14] for function optimization) that *suggest* (but do not *analyze*) that the shape of the grid actually influences the quality of the search. Some more recent works [15]–[17] have studied quantitatively the performance improvement got when using cEAs with a non perfectly square grid. Specifically, in [17], a comparative study among different grid shapes for several problems can be found. In all these works, the use of narrow (non square) grids has led to very efficient algorithms.

Our first objective in this paper is to check whether the results in [17] (nonsquare grids are often more efficient than square ones) still hold when using an extended set of complex problems. Our second objective is to analyze comparatively the behavior of cGAs using one static grid during the overall search with other cGAs introducing an *a priori* change in the grid shape (i.e., using two different grid shapes during the evolution). This is made for the group of three problems presented in [17], but we want to analyze this preprogrammed change on other problems. Therefore, we will focus on cGAs in this paper, although the same methodology can be extended to other EAs.

The underlying contribution of this work is to get a deeper knowledge of the high importance of the topology and neighborhood relationship when using cGAs. By using the quantitative notion of ratio, we will study static and preprogrammed algorithms, as well as we will define a new adaptive algorithmic model that will choose the most desirable degree of exploration/exploitation by itself. Consequently, the designer will not need to preset the value of the ratio when solving a problem.

This paper is organized as follows. In the next section, we will discuss the cGA canonical model, and the concept of ratio. Section III summarizes the state of the art on self-adaptation. In Section IV, we present the different cEAs analyzed in this paper, and a simple proposal for relocating the individuals when the topology shape changes. Section V discusses and justifies the selection of the problems used for our analysis, while in Section VI, we explain the results of our experiments. Finally, in Section VII, some conclusions are drawn and future research lines are suggested.

II. Characterizing the Cellular GA (cGA)

We begin this section by including a pseudocode description of the canonical cGA search model and an explanation of the algorithm. After that, we will characterize the diversification capabilities of the cGA by a single parameter (the ratio).

The EA family we are using as a case study here is a cellular genetic algorithm (cGA), which is described in Algorithm 1. The most common population topology used in cEAs is a toroidal grid where all the individuals live in; moreover, it can be proved that using this topology does not restrict our study [15]. Hence, we always use in this work a population structured in a two-dimensional (2-D) toroidal grid, and the neighborhood defined on it (line 5) contains five individuals: the considered one [`position (x,y)`] plus the north, east, west, and south strings (called NEWS, linear5, or Von Neumann). We study algorithms using either binary tournament and fitness proportional selection methods—`Local_Select` (lines 5 and 7). The considered individual itself is always selected for being one of its two parents (line 6).

We use a two point crossover operator (*DPX1*) that yields only one child [the one having the larger portion of the best parent (line 8)], and a traditional binary mutation operator—*bit-flip* (line 9). After applying these operators, the algorithm calculates the fitness value of the new individual (line 10).

Algorithm 1 Pseudocode of a simple cGA
```
 1: proc Steps_Up(cga)      //Algorithm parameters in "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 ← Individual_At(cga, position(x, y));
 7:       parent2 ← Local_Select(n_list);
 8:       DPX1(cga.Pc, n_list[parent1], n_list[parent2],
               aux_ind.chrom);       // Recombination
 9:       Bit-Flip(cga.Pm,aux_ind.chrom);     // Mutation
10:       aux_ind.fit ← cga.Fit(Decode(aux_ind.chrom));
11:       Insert_New_Ind(position(x, y), aux_ind,
               [if_better | always], cga, aux_pop);
12:     end for
13:   end for
14:   cga.pop ← aux_pop;
15: Update_Statistics(cga);
16: end for
17: end_proc Steps_Up.
```

In our monoprocessor implementation of this cGA, the successive populations replace each other completely (line 14), so the new individuals generated by local selection, crossover, and mutation are placed in a temporal population (line 11).

This replacement step (line 11) 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 [4] for this study.

The second alternative is called an "asynchronous" update method in the literature, and lies in placing the offsprings directly in the current population by following some rules [18] instead of updating all the individuals simultaneously (Algorithm 1). This issue is not explored here because of the many implications and numerous asynchronous policies.

Computing basic statistics (line 15) is rarely found in the pseudocodes of other authors. However, it is an essential step for the work and monitoring of the algorithm. In fact, it is possible to use some kind of statistical descriptors to guide an adaptive search, as we will show in this work. Also, we need to compute the list of neighbors for a given individual located at coordinates `(x,y)` (line 5). This operation is needed whenever it is contemplated a future change in the type of neighborhood (although we do not change it here).

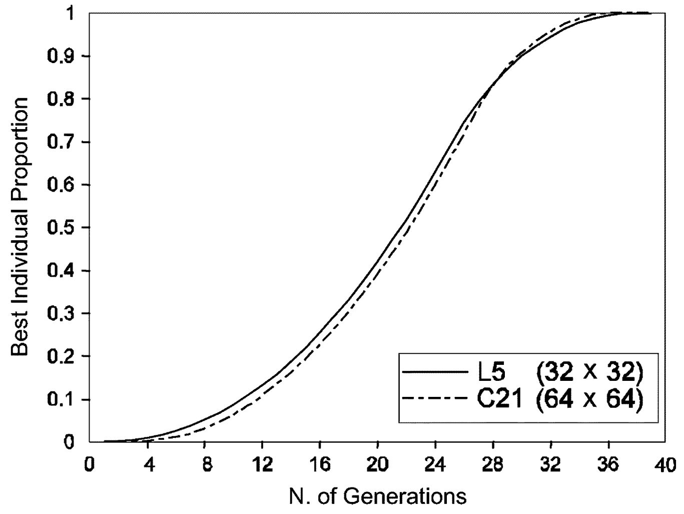

Fig. 2. Growth curves of the best individual for two cGAs (using binary tournament selection) with different neighborhood and population shapes, but similar ratio values.

After explaining our basic algorithm, we now proceed to characterize the population grid. We use the "radius" definition given in [17], which is refined from the seminal one appeared in [15] to account for nonsquare grids. The grid is considered to have a radius equal to the dispersion of n^* points in a circle centered in $(\bar{x}, \bar{y})$ (1). This definition always assigns different numeric values to different grids.

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

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

This definition does not only characterize the grid shape but it also can provide a radius value for the neighborhood. As proposed in [15], the grid-to-neighborhood relationship can be quantified by the ratio between their radii (2). Algorithms with similar ratio show a similar selection pressure when using the same selection method, as stated in [16]. In Fig. 2, we plot such a similar behavior for two algorithms with different neighborhood and population radii, but having two very similar ratio values. The algorithms plotted are those using a linear5 -L5- neighborhood with a 32×32 population, and a compact21 -C21- neighborhood with a population of 64×64 individuals

$$\text{ratio}_{\text{cGA}} = \frac{\text{rad}_{\text{Neighborhood}}}{\text{rad}_{\text{Topology}}}. \tag{2}$$

When solving a given problem with a constant number of individuals ($n = n^*$, for making fair comparisons) the topology radius will increase as the grid gets thinner [Fig. 3(b)]. Since the neighborhood is kept constant in size and shape throughout this paper [we always use NEWS, Fig. 3(a)], the ratio will be smaller as the grid gets thinner.

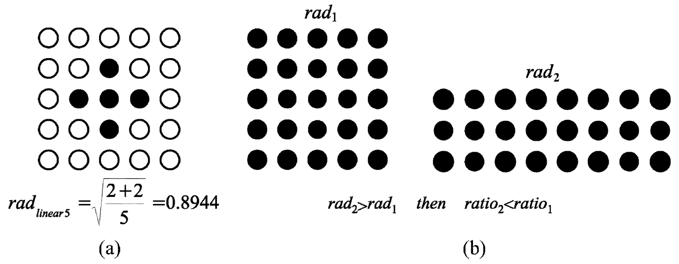
$\text{rad}_{linear5} = \sqrt{\frac{2+2}{5}} = 0.8944$
(a)
rad_1
rad_2
$rad_2 > rad_1$ then $ratio_2 < ratio_1$
(b)

Fig. 3. (a) Radius of neighborhood NEWS. (b) $5 \times 5 = 25$ and $3 \times 8 \approx 25$ grids; equal number of individuals with two different ratios.

Reducing the ratio means reducing the global selection intensity on the population, thus promoting *exploration*. This is expected to allow for a higher diversity that could improve the results in difficult problems (like in multimodal or epistatic tasks). Besides, the search performed inside each neighborhood is guiding the *exploitation* of the algorithm. We study in this paper how the ratio affects the search efficiency over a variety of domains. Changing the ratio during the search is a unique feature of cGAs that can be used to shift from exploration to exploitation at a minimum complexity without introducing just another new algorithm family in the literature.

Many techniques for managing the exploration/exploitation tradeoff are possible. Among them, it is worth making a special mention of heterogeneous EAs [19], [20], in which algorithms with different features run in multiple subpopulations and collaborate in order to avoid premature convergence. A different alternative is using *memetic algorithms* [21], [22], in which *local search* is combined with the genetic operators in order to promote local exploitation.

In the case of cGAs, it is very easy to increase population diversity by simply relocating the individuals that compose it by changing the grid shape, as we will see in Section IV. The reason is that a cellular model provides restrictions on the distance for the mating of solutions due to the use of neighborhoods (one solution may only mate with one of its neighbors). Hence, a cGA can be seen as a mating restriction algorithm based on the Euclidean distance.

III. Background on Self-Adaptation

One important contribution of the present paper is to define cGAs capable of self-adapting their exploration/exploitation ratio. In this section, we will make an introduction to the field of adaptation and self-adaptation in EAs.

Let us first revisit a well-known classification of adaptive EAs [23], [24] (see Fig. 4 for the details). We can make a classification of the *type of adaptation* on the basis of the mechanism used for this purpose; in particular, attention is paid to the issue of whether a feedback from the EA is used or not.

- *Parameter tuning*: It appears whenever the strategy parameters have constant values throughout the run of the EA (there is no adaptation). Consequently, an external agent or mechanism (e.g., a person or program) is needed to tune the desired strategy parameters and to choose the most appropriate values. We study some algorithms with hand tuned parameters in this work.

$$\textbf{Criterion}\begin{cases}\textbf{Parameter Tuning – No ratio change}\\ \textbf{Dynamic}\begin{cases}\textbf{Deterministic – Change point(s) fixed before execution}\\ \textbf{Adaptive – Automatic change (feedback from EA)}\\ \text{Self-Adaptive – Automatic change (coded in individuals)}\end{cases}\end{cases}$$

Fig. 4. Taxonomy of adaptation models for EAs in terms of the ratio.

- *Dynamic*:. Dynamic adaptation appears when there is some mechanism which modifies a strategy search parameter without any external control. The class of EAs that use dynamic adaptation can be further subdivided into three subclasses, where the *mechanism of adaptation* is the criterion.
  — *Deterministic*: Deterministic dynamic adaptation takes place if the value of a strategy parameter is altered by some deterministic rule; this rule modifies the strategy parameter deterministically without using any feedback from the EA. Examples of this kind of adaptation are the preprogrammed criteria we study in this work.
  — *Adaptive*: Adaptive dynamic adaptation takes place if there is some form of feedback from the EA that is used to set the direction and/or magnitude of the change to the strategy parameters (this kind of adaptation is also used in this work).
  — *Self-adaptive*: The idea of the "evolution of evolution" can be used to implement the self-adaptation of parameters. In this case, the parameters to be adapted are encoded into the representation of the solution (e.g., the chromosome) and undergo mutation and recombination [25].

We can distinguish among different families of these dynamic EAs attending to the level at which the adaptive parameters operate: environment (adaptation of the individual as a response to changes in the environment, e.g., penalty terms in the fitness function), population (parameters global to the entire population), individual (parameters held within an individual), and component (strategy parameters local to some component or gene of the individual). These levels of adaptation can be used with each of the types of dynamic adaptation; in addition, a mixture of levels and types of adaptation can be used within an EA, leading to algorithms of difficult classification.

This taxonomy on adaptation will guide the reader in classifying the kind of algorithms we are dealing with in this work. This provides an appropriate context to discuss the ideas included in the forthcoming sections.

IV. Description of the Algorithms

The performance of a cGA may change as a function of several parameters. Among them, we will pay special attention to the ratio, which is the relationship between the neighborhood and the population radii (2), as stated in Section II. Our goal is to study the effects of this ratio on the behavior of the algorithm. Since we always use the same neighborhood (NEWS), the study of such a ratio is reduced to the analysis of the effects of using different population shapes.

We begin by considering three different static population shapes. Then, we propose two methods for changing the value of the ratio during the execution in order to promote the exploration or exploitation in certain steps of the search. The first approach to this idea is to modify the value of the ratio at a fixed (predefined) point of the execution. For this goal, we propose two different criteria: 1) changing from exploration to exploitation and 2) changing from exploitation to exploration. In the second approach, we propose a dynamic self-manipulation of the ratio as a function of the evolution progress.

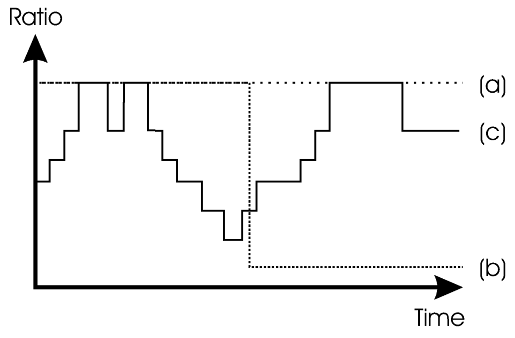

Fig. 5. Idealized evolution of the ratio for (a) static, (b) preprogrammed, and (c) adaptive criteria.

In Fig. 5, we show a theoretical idealization of the evolution of the ratio value in the three different classes of algorithms we study in this paper. We can see how static algorithms [Fig. 5(a)] keep constant the ratio value along the whole evolution, while the other two algorithms change it. For preprogrammed algorithms [Fig. 5(b)], this change in the ratio is made in an *a priori* point of the execution, in contrast to adaptive algorithms [Fig. 5(c)], where the ratio varies automatically along the search as a function of the convergence speed of the algorithm.

All the algorithms studied in this paper (shown in Fig. 6) are obtained from the same canonical cGA by changing only the ratio between the neighborhood and the population topology radii in different manners. We study the influence of this ratio over a representative family of nontrivial problems. This family of problems is extended from the initial benchmark included in [17], where a relationship between low/high ratio and exploration/exploitation of the algorithm is established. In order to help the reader to understand the results, we explain in Section IV-A the algorithms employed in that work. After that, we will introduce an adaptive algorithmic proposal with the objective of avoiding the researcher to make an ad hoc definition of the ratio, that will (hopefully) improve the efficiency or accuracy of the algorithm (Section IV-B).

A. Static and Preprogrammed Algorithms

In this section, we discuss five different algorithms which were initially proposed in [17]. Three of them use static ratios and the other two implement preprogrammed ones (see Fig. 6). Our first aim is to extend this seminal study to a larger and harder set of problems.

First, we tackle a cGA in which three clearly different static ratios have been used (remember, we always use NEWS–linear5–neighborhood).

- *Square*: Ratio $= 0.110$ (20×20 individuals).
- *Rectangular*: Ratio $= 0.075$ (10×40 individuals).
- *Narrow*: Ratio $= 0.031$ (4×100 individuals).

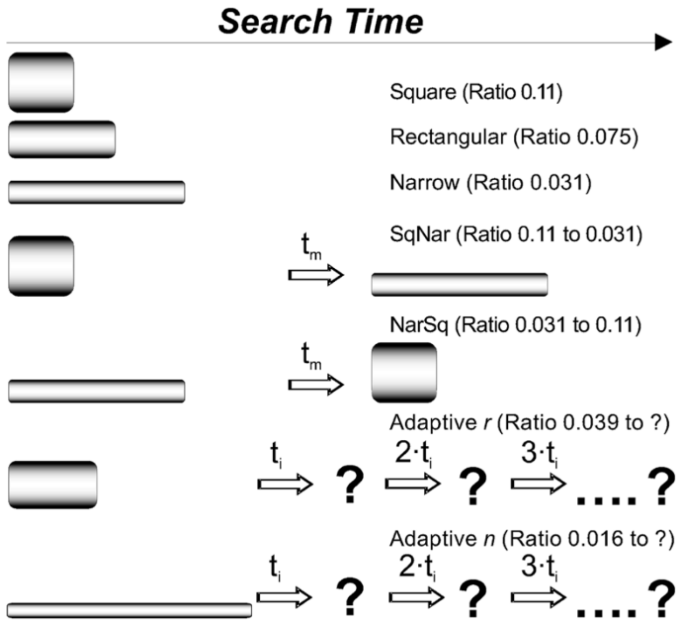
Search Time
Square (Ratio 0.11)
Rectangular (Ratio 0.075)
Narrow (Ratio 0.031)
SqNar (Ratio 0.11 to 0.031)
NarSq (Ratio 0.031 to 0.11)
Adaptive r (Ratio 0.039 to ?)
Adaptive n (Ratio 0.016 to ?)
t_m
t_i
$2 \cdot t_i$
$3 \cdot t_i$

Fig. 6. Algorithms studied in this work.

The total population size is always 400 individuals, structured in three different grid shapes, each one with different exploration/exploitation features. Additionally, we have used two more criteria with a dynamic but *a priori* preprogrammed ratio change. These two criteria change the algorithm ratio in a different manner at a fixed point of the execution. In our case, this change is performed in the "middle" of a "typical" execution (t_m). We define the "middle" of a "typical" run (t_m) as the point wherein the algorithm reaches half the mean number of evaluations needed to solve the problem with the (traditional) square grid. The preprogrammed algorithms studied change the ratio from 0.110 to 0.031 [we call it *square* to *narrow* (*SqNar*)] and from 0.031 to 0.110 [we call it *narrow* to *square* (*NarSq*)].

Since shifting between exploration and exploitation is made by changing the shape of the population (and, thus, its radius) in this work, we theoretically consider the population as a list of length $m \cdot n$, such that the first row of the $m \times n$ grid is composed by the n first individuals of the list, the second row is made up with the next n individuals, and so on. Therefore, when performing a change from a $m \times n$ grid to a new $m' \times n'$ grid (being $m \cdot n = m' \cdot n'$) the individual placed at position (i, j) will be relocated as follows:

$$(i, j) \rightarrow ([i * n + j] \operatorname{div} n', [i * n + j] \bmod n'). \qquad (3)$$

We call this redistribution method *contiguous*, because the new grid is filled up by following the order of appearance of the individuals in the list. It is shown in Fig. 7 how the considered individual plus its neighbors are relocated when the grid shape changes from 8×8 to 4×16, as described in (3). The reader can notice how (in general) a part of a cell neighborhood is kept, while the other part may change. In Fig. 7, we have drawn the relocation of the individual in position (2, 4) and its neighbors. When the grid shape changes, that individual is relocated at position (1, 4), changing its neighbors placed at its north and south positions, and keeping close those placed at its east and west positions. This change in the grid shape can be seen as an actual migration of individuals among neighborhoods, which will introduce additional diversity into the population for the forthcoming generations.

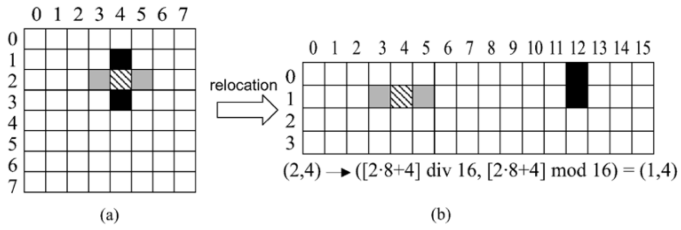

Fig. 7. Relocation of an individual and its neighbors when the grid changes from (a) 8×8 to other with (b) shape 4×16.

B. Adaptive Algorithms

Let us focus in this section on our last proposed class of cGAs. The core idea, like in most other adaptive algorithms, is to use some feedback mechanism that monitors the evolution and that dynamically rewards or punishes parameters according to their impact on the quality of the solutions. An example of these mechanisms is the method of Davis [26] to adapt operator probabilities in genetic algorithms based on their observed success or failure to yield a fitness improvement. Other examples are the approaches of [27] and [28] to adapt population sizes either by assigning lifetimes to individuals based on their fitness or by having a competition between subpopulations based on the fitness of the best population members.

A main difference here is that the monitoring feedback and the actions undertaken are computationally inexpensive in our case. We really believe that this is the primary feature of any adaptive algorithm in order to be useful for other researchers. The literature contains numerous examples of interesting adaptive algorithms whose feedback control or adaptive actions are so expensive that many other algorithms or hand-tuning operations could result in easier or more efficient algorithms.

Our adaptive mechanisms are shown in Fig. 6 (we analyze several mechanisms, not only one). They modify the grid shape during the search as a function of the convergence speed. An important advantage for these adaptive criteria lies in that it is not necessary to set the population shape to any one ad hoc value, because a dynamical recalculation of which is the most appropriate one is achieved with a given frequency (t_i) during the search. This also helps in reducing the overhead to a minimum. Algorithm 2 describes the basic adaptive pattern; C_1 and C_2 represent the measures of the convergence speed to use.

Algorithm 2 Pattern for our dynamic adaptive criteria.

```
1: if $C_1$ then
2:     ChangeTo(square)        //exploit
3: else if $C_2$ then
4:     ChangeTo(narrow)        //explore
5: else
6:     Do_not_Change
7: end if
```

The adaptive criteria try to increase the local exploitation by changing to the next more square grid shape whenever the

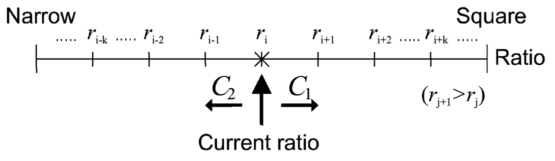
Narrow
Square
..... r_{i-k} ..... r_{i-2} r_{i-1} r_i r_{i+1} r_{i+2} ..... r_{i+k} .....
Ratio
C_2
C_1
$(r_{j+1}>r_j)$
Current ratio

Fig. 8. Change in the ratio performed when conditions C_1 and C_2 of Algorithm 2 hold.

algorithm evolves slowly, i.e., when the convergence speed decays below a given threshold value ($\varepsilon \in [0,1]$). This is represented by the condition of line 1 in Algorithm 2. If, on the other side, the search is proceeding too fast, diversity could be lost quickly and there exists a risk of getting stuck in a local optimum. In this second case, the population shape will be changed to a narrower grid and, thus, promote exploration and diversity in the forthcoming generations. The condition for detecting this situation is expressed by line 3 in Algorithm 2. If neither C_1 nor C_2 are true, the shape is not changed (line 6).

As it can be seen, this is a general search pattern, and multiple criteria (C_i) could be used to control whether the algorithm should explore or exploit the individuals for the next t_i generations.

The objective of these adaptive criteria is to maintain the diversity and to pursue the global optimum during the execution. Notice that a change in the grid shape implies the relocation of individuals, which is a kind of migration since individuals in different areas become neighbors. Therefore, this change in the topology is also an additional source for increasing the diversity, intrinsic to the adaptive criterion.

Whenever the criterion is fulfilled, the adaptive search pattern performs a change in the grid shape. The function **ChangeTo**(square/narrow) in Algorithm 2 changes the grid to the immediately next more square/narrow allowed shape. The bounding cases are the square grid shape and the completely linear (ring-like) shape, as shown in Fig. 8. When the current grid is square and the algorithm needs to change to the next "more square" one, or when the present grid shape is the narrowest one and the algorithm needs to change to the next "narrower" one, the algorithm does not make any change in the population shape at all. In the rest of cases, when the change is possible, it is accomplished by computing a new position for every individual whose general location is (i,j), as shown in (3). Of course, other changes could have been used, but, in our quest for efficiency, the proposed one is considered to introduce a negligible overhead.

Once we have fixed the basic adaptive pattern and grid shape modification rules, we now proceed to explain the criteria used to determine when the population is going "too" fast or "too" slow. We propose in this paper three different criteria for measuring the search speed. The measures are based on the *average fitness* (criterion AF), the *population entropy* (criterion PH), or on a combination of the two (criterion $\mathrm{AF}+\mathrm{PH}$). Since these criteria check simple conditions over the average fitness and the population entropy (calculated in every run in the population statistics computation step), we can say that they are inexpensive to measure, and they are indicative of the search states at the same time too. The complexity for calculating the average fitness is $O(\mu)$, while in the case of the population entropy it is $O(\mu \cdot L)$, μ being the size of the population and L the length of the chromosomes. The details on their internals are as follows.

- **AF**: This criterion is based on the *average fitness* of the population. Hence, our measure of the convergence speed is based in terms of the diversity of phenotypes. We define $\Delta\bar{f}_t$ as the difference between the average fitness values in generation t and $t-1$: $\Delta\bar{f}_t = \bar{f}_t - \bar{f}_{t-1}$. The algorithm will change the ratio value to the immediately larger one (keeping constant the population size) if the difference $\Delta\bar{f}_t$ between two contiguous generations (t and $t-1$) decreases at least in a factor of ε : $\Delta\bar{f}_t - \Delta\bar{f}_{t-1} < \varepsilon \cdot \Delta\bar{f}_{t-1}$ (condition C_1 of Algorithm 2). On the contrary, the ratio will be changed to its closer smaller value if that difference increases in a factor greater than $(1-\varepsilon)$: $\Delta\bar{f}_t - \Delta\bar{f}_{t-1} > (1-\varepsilon) \cdot \Delta\bar{f}_{t-1}$ (condition C_2). Formally, we define C_1 and C_2 as follows:
$$C_1 ::= \Delta\bar{f}_t < (1+\epsilon) \cdot \Delta\bar{f}_{t-1}$$
$$C_2 ::= \Delta\bar{f}_t > (2-\epsilon) \cdot \Delta\bar{f}_{t-1}.$$
- **PH**: We now propose to measure the convergence speed in terms of the genotypic diversity. The *population entropy* is the metric used for this purpose. We calculate this entropy (H_t) as the average value of the entropy of each gene in the population. Hence, this criterion is similar to AF except for that it uses the change in the population entropy between two generations $(\Delta H_t = H_t - H_{t-1})$ instead of the average fitness variation. Consequently, conditions C_1 and C_2 are expressed as
$$C_1 ::= \Delta H_t < (1+\epsilon) \cdot \Delta H_{t-1}$$
$$C_2 ::= \Delta H_t > (2-\epsilon) \cdot \Delta H_{t-1}.$$
- $\mathbf{AF}+\mathbf{PH}$: This is the third and last proposed criterion to create an adaptive cGA. It considers both the population entropy and the average fitness acceleration by combining the two previous criteria (AF and PH). Thus, it relays on the phenotype and genotype diversity in order to obtain the best exploration/exploitation tradeoff. This is a more restrictive criterion with respect to the two previous ones, since although genetic diversity usually implies phenotypic diversity, the reciprocal is not always true. Condition C_1 in this case is the result of the logic and operation of conditions C_1 of the two preceding criteria. In the same way, C_2 will be the and operation of conditions C_2 of AF and PH
$$C_1 ::= (\Delta\bar{f}_t < (1+\epsilon) \cdot \Delta\bar{f}_{t-1}) \quad \textbf{and} \quad (\Delta H_t < (1+\epsilon) \cdot \Delta H_{t-1})$$
$$C_2 ::= (\Delta\bar{f}_t > (2-\epsilon) \cdot \Delta\bar{f}_{t-1}) \quad \textbf{and} \quad (\Delta H_t > (2-\epsilon) \cdot \Delta H_{t-1}).$$

Moreover, for each adaptive criterion (AF, PH, and AF+PH), we can begin the execution by using: 1) the square grid shape; 2) the narrowest one; or 3) the one having a medium ratio value (rectangular). In our terminology, we will specify the initial grid shape by adding at the beginning of the criterion name a s, n, or

Unitation	Subfunction value
0	1.000000
1	0.000000
2	0.360384
3	0.640576
4	0.360384
5	0.000000
6	1.000000

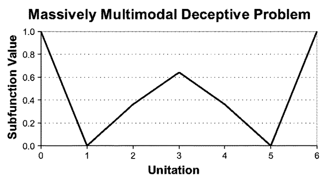

Fig. 9. Basic deceptive bipolar function (s_i) for MMDP.

r for executions that begins at the square, narrowest, or rectangular ratio grid shapes, respectively. For example, the algorithm which uses criterion AF beginning at the narrowest shape will be denoted as nAF.

In short, we have proposed in this section three different adaptive criteria. The first one, AF, is based on the phenotypic diversity, while the second one, PH, is based on genotypic diversity. Finally, $\mathrm{AF}+\mathrm{PH}$ is a combined criterion accounting for diversity at these two levels.

V. Test Problems

In this section, we present the set of problems chosen for this study. The benchmark is representative because it contains many different interesting features in optimization, such as epistasis, multimodality, deceptiveness, use of constraints, parameter identification, and problem generators. These are important ingredients in any work trying to evaluate algorithmic approaches with the objective of getting reliable results, as stated by Whitley *et al.* in [29].

Initially, we will experiment with the reduced set of problems studied in [17], which includes the massively multimodal deceptive problem (MMDP), the frequency modulation sounds (FMS), and the multimodal problem generator P-PEAKS; next, we will extend this basic three-problem benchmark with COUNTSAT (an instance of MAXSAT), error correcting code design (ECC), maximum cut of a graph (MAXCUT), and the minimum tardy task problem (MTTP). The election of this set of problems is justified by both their difficulty and their application domains (combinatorial optimization, continuous optimization, telecommunications, scheduling, etc.). This guarantees a high level of confidence in the results, although the evaluation of conclusions will result more laborious than with a small test suite.

The problems selected for this benchmark are explained in Sections V-A–V-G. We include the explanations in this paper to make it self-contained and to avoid the typical small lacks that could preclude other researchers from reproducing the results.

A. Massively Multimodal Deceptive Problem (MMDP)

The MMDP is a problem that has been specifically designed to be difficult for an EA [30]. It is made up of k deceptive subproblems (s_i) of 6 bits each one, whose value depends on the number of ones (*unitation*) a binary string has (see Fig. 9). It is easy to see (graphic of Fig. 9) that these subfunctions have two global maxima and a deceptive attractor in the middle point.

In MMDP, each subproblem s_i contributes to the fitness value according to its *unitation* (Fig. 9). The global optimum has a value of k and it is attained when every subproblem is composed of zero or six ones. The number of local optima is quite large (22^k), while there are only 2^k global solutions. Therefore, the degree of multimodality is regulated by the k parameter. We use here a considerably large instance of $k=40$ subproblems. The instance we try to maximize for solving the problem is shown in (4), and its maximum value is 40

$$f_{\mathrm{MMDP}}(\vec{s}) = \sum_{i=1}^{k} \mathrm{fitness}_{s_i}. \tag{4}$$

B. Frequency Modulation Sounds (FMS)

The FMS problem [31] is defined as determining the six real-parameters $\vec{x}=(a_1,w_1,a_2,w_2,a_3,w_3)$ of the frequency modulated sound model given in (5) for approximating it to the sound wave given in (6) (where $\theta = 2\cdot\pi/100$). The parameters are defined in the range $[-6.4,+6.35]$, and we encode each parameter into a 32 bit substring in the individual

$$\begin{aligned} y(t) &= a_1\cdot\sin(\omega_1\cdot t\cdot\theta + a_2 \\ &\qquad \cdot \sin(\omega_2\cdot t\cdot\theta + a_3\cdot\sin(\omega_3\cdot t\cdot\theta))) \qquad (5)\\ y_0(t) &= 1.0\cdot\sin(5.0\cdot t\cdot\theta - 1.5 \\ &\qquad \cdot\sin(4.8\cdot t\cdot\theta + 2.0\cdot\sin(4.9\cdot t\cdot\theta))). \qquad (6)\end{aligned}$$

The goal is to minimize the sum of square errors given by (7). This problem is a highly complex multimodal function having strong epistasis, with optimum value 0.0. Due to the high difficulty of solving this problem with high accuracy without applying local search or specific operators for continuous optimization (like gradual GAs [19]), we stop the algorithm when the error falls below 10^{-2}. Hence, our objective for this problem will be to minimize (7)

$$f_{\mathrm{FMS}}(\vec{x}) = \sum_{t=0}^{100}(y(t)-y_0(t))^2. \tag{7}$$

C. Multimodal Problem Generator (P-PEAKS)

The P-PEAKS problem [32] is a multimodal problem generator. A problem generator is an easily parameterizable task which has a tunable degree of difficulty. 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, since a different instance is solved each time the algorithm runs, then the predictive power of the results for the problem class as a whole is increased.

The idea of P-PEAKS is to generate P random N-bit strings that represent the location of P peaks in the search space. The fitness value of a string is the number of bits the string has in common with the nearest peak in that space, divided by N [as shown in (8)]. As stated in [32], "... Problems with a small/large number of peaks are weakly/strongly epistatic ...." In this paper, we have used an instance of $P = 100$ peaks of length $N =$

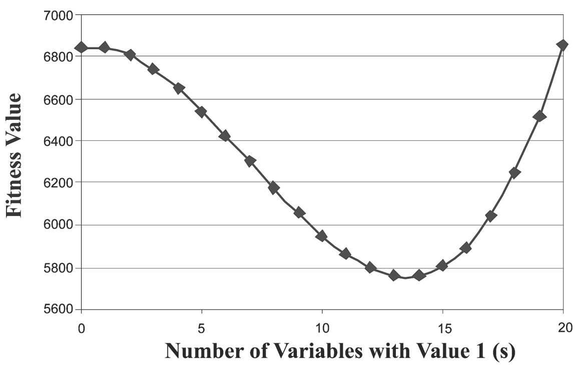

Fig. 10. COUNTSAT function with $n = 20$ variables.

100 bits each, which represents a medium/high difficulty level [17]. The maximum fitness value for this problem is 1.0

$$f_{P-\text{PEAKS}}(\vec{x}) = \frac{1}{N} \max_{1 \le i \le p} \{N - \text{Hamming } D(\vec{x}, \text{Peak}_i)\}. \tag{8}$$

D. COUNTSAT Problem

The COUNTSAT problem [33] is an instance of MAXSAT. Let us consider the following clauses [34]:

- $x_i, 1 \le i \le n$;
- $x_i \vee \bar{x}_j \vee \bar{x}_k, (i,j,k) \in \{1,\ldots,n\}^3, i \ne j \ne k \ne i$.

In COUNTSAT, the solution value is the number of clauses (among those specified above) that are satisfied by a n-bit input string. It is easy to check that the optimum value is that having all the variables set to 1 $(s = n)$. In this work, an instance of $n = 20$ variables has been used, with optimum value of 6860 (9)

$$\begin{aligned} f_{\text{COUNTSAT}}(s) &= s + n \cdot (n-1) \cdot (n-2) \\ &\quad - 2 \cdot (n-2) \cdot \binom{s}{2} + 6 \cdot \binom{s}{3} \\ &= s + 6840 - 18 \cdot s \cdot (s-1) \\ &\quad + s \cdot (s-1) \cdot (s-2). \end{aligned} \tag{9}$$

The COUNTSAT function is extracted from MAXSAT with the objective of being very difficult to solve with GAs [33]. Every uniformly generated random input will have approximately $n/2$ ones. Then, local changes decreasing the number of ones will lead to better inputs, while local changes increasing the number of ones decrease the fitness (Fig. 10). Hence, we might expect that EAs quickly find the all-zero string and have difficulties to find the all-one string.

E. Error Correcting Code Design Problem (ECC)

The ECC problem was presented in [35]. We will consider a three-tuple (n, M, d), where n is the length of each codeword (number of bits), M is the number of codewords, and d is the minimum Hamming distance between any pair of codewords. Our objective will be to find a code which has a value for d as large as possible (reflecting greater tolerance to noise and errors), given previously fixed values for n and M. The problem we have studied is basically the same presented in [35]. The fitness function to be maximized is

$$f_{\text{ECC}}(C) = \frac{1}{\sum_{i=1}^{M} \sum_{\substack{j=1 \\ i \ne j}}^{M} \frac{1}{d_{ij}^2}} \tag{10}$$

where d_{ij} represents the Hamming distance between codewords i and j in the code C (made up of M codewords, each of length n). We consider in the present paper an instance where $M = 24$ and $n = 12$. The search space is of size $\binom{4096}{24}$, which is approximately 10^{87}. The optimum solution for $M = 24$ and $n = 12$ has a fitness value of 0.0674 [36].

F. Maximum Cut of a Graph (MAXCUT)

The MAXCUT problem looks for a partition of the set of vertices (V) of a weighted graph $G = (V, E)$ into two disjoint subsets V_0 and V_1 so that the sum of the weights of the edges with one endpoint in V_0 and the other one in V_1 is maximized. For encoding the problem, we use a binary string $(x_1, x_2, \ldots, x_n)$ of length n, where each digit corresponds to a vertex. If a digit is 1, then the corresponding vertex is in set V_1; if it is 0, then the corresponding vertex is in set V_0. The function to be maximized [37] is

$$f_{\text{MAXCUT}}(\vec{x}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} \cdot [x_i \cdot (1 - x_j) + x_j \cdot (1 - x_i)]. \tag{11}$$

Note that w_{ij} contributes to the sum only if nodes i and j are in different partitions. We have considered three different graph examples in this study. Two of them are randomly generated graphs of moderate sizes: a sparse one “cut20.01” and a dense one “cut20.09,” both of them are made up of 20 vertices. The other instance is a scalable weighted graph of 100 vertices. The globally optimal solutions for these instances are 10.119 812 for “cut20.01,” 56.740 064 in the case of “cut20.09,” and 1077 for “cut100” [37].

G. Minimum Tardy Task Problem (MTTP)

MTTP [38] is a task-scheduling problem, wherein each task i from the set of tasks $T = \{1, 2, \ldots, n\}$ has a length l_i (the time it takes for its execution), a deadline d_i (before which a task must be scheduled, and its execution completed), and a weight w_i. The weight is a penalty that has to be added to the objective function in the event that the task remains unscheduled. The lengths, weights and deadlines of tasks are all positive integers. Scheduling the tasks of a subset S of T is to find the starting time of each task in S, such as at most one task at time is performed and such that each task finishes before its deadline.

We characterize a one-to-one scheduling function g defined on a subset of tasks $S \subseteq T : S \mapsto \mathbb{Z}^+ \cup \{0\}$, so that for all tasks $i, j \in S$ has the following properties.

1) A task cannot be scheduled before any previous one has finished: $g(i) < g(j) \Rightarrow g(i) + l_i \le g(j)$.
2) Every task finishes before its deadline: $g(i) + l_i \le d_i$.

The objective function for this problem is to minimize the sum of the weights of the unscheduled tasks. Therefore, the optimum scheduling minimizes (12)

$$f_{\mathrm{MTTP}}(\vec{x}) = \sum_{i \in T-S} w_i. \tag{12}$$

The schedule of tasks S can be represented by a vector $\vec{x} = (x_1, x_2, \ldots, x_n)$ containing all the tasks ordered by its deadline. Each $x_i \in \{0,1\}$, where if $x_i = 1$, then task i is scheduled in S, while if $x_i = 0$ means that task i is not included in S. The fitness function is the inverse of (12), as described in [37]. We have used in this study three different instances [37] for analyzing the behavior of our algorithms with this function: "mttp20," "mttp100," and "mttp200," with sizes 20, 100, and 200, and known maximum fitness values of 0.024 39, 0.005, and 0.0025, respectively.

VI. Experiments

Although a full-length study of the problems presented in the previous section is beyond the scope of this work, in this section, we present and analyze the results obtained when solving all the problems with all the discussed cGA variants, always with a constant neighborhood shape (L5). Note that it is not our aim here to compare cGA's performance with state-of-the-art algorithms and heuristics for combinatorial and numerical optimization. To this end, we should at least tune the parameters or include local search capabilities in the algorithm, which is not the case. Thus, the results only pertain to the relative performance of the different cGA update ratios among themselves.

The resulting work is a direct continuation of a past work [17], extended here by including more problems in the test suite and some new algorithms (adaptive cGAs). Besides, we perform the tests with two different selection schemes: roulette wheel and binary tournament. We have chosen these two selection methods in order to analyze both, fitness proportionate and ranking-like techniques. On the one hand, roulette wheel will allow us to tests to which extent selection errors are important for the proposed algorithms. On the other hand, tournament selection should throw similar results with respect to ranking methods, but we have selected tournament and not ranking because of its lower computational cost. In addition, tournament selection is being used to check whether it shows any improvement on the search with respect to roulette wheel. In summary, by including these two selection methods (and not only one of them), we aim at offering a more complete analysis.

We proceed in three stages, namely, analysis of static ratios, preprogrammed dynamic ratios, and dynamic adaptive ratios. In the two first steps, we use square and rectangular grids in *a priori* well-known points of the evolution, i.e., either using always the same grid shape (static), or setting an *off-line* fixed change in the grid shape before the optimization (preprogrammed).

In the third stage, we apply three mechanisms of grid adaptation (dynamic) based on convergence acceleration, as explained in Section IV-B. We study these mechanisms with two different initial values for the ratio: the smallest one, and that with the middle value (i.e., nearest to the middle ratio value).

TABLE I
Parameterization Used in Our Algorithms

Population Size	400 individuals
Selection of Parents	itself + (BT or RW)
Recombination	DPX1, $p_c = 1.0$
Bit Mutation	Bit-flip, $p_m = 1/L$
Replacement	Replace if Better

All the algorithms have been compared in terms of the efficiency and efficacy, so the figures in Tables III–VI stand for the average number of evaluations needed to solve each problem with all the algorithms, and the percentage of success runs after making 100 independent runs for each problem. We have performed ANOVA tests in order to obtain statistically significant results in our comparisons (see Tables III–VI).

We have organized this section into five subsections. In the first one, we reproduce the parameters used for the execution of the algorithms. In the next subsection, we will compare the behavior of a cGA and a GA with a panmictic population. In the two next subsections, we present and analyze the results obtained with the different algorithms for all the problems with the two proposed selection methods. Finally, we offer an additional global graphic interpretation of the results in the last subsection.

A. Parameterization

In order to make a meaningful comparison among all the algorithms, we have used a common parameterization. The details are described in Table I, where L is the length of the string representing the chromosome of the individuals. Two different selection methods have been used in this work. One parent is always the individual itself, while the other one is obtained by using *binary tournament* (BT) (Section VI-C) or *roulette whee*l (RW) selection (Section VI-D). The two parents are forced to be different.

In the recombination operator, we obtain just one offspring from two parents: the one having the largest portion of the best parent. The *DPX1* recombination is applied always (probability $p_c = 1.0$). The bit mutation probability is set to $p_m = 1/L$. The exceptions are COUNTSAT, where we use $p_m = 1/(2 \cdot L)$, and the FMS problem, for which a value of $p_m = 10/L$ is used. These two values are needed because the algorithms had a negligible solution rate with the standard $p_m = 1/L$ probability.

We will replace the considered individual on each generation only if its offspring has a better fitness value, called *replace if better* [39]. The cost of solving a problem is analyzed by measuring the number of evaluations of the objective function made during the search. The stop condition for all the algorithms is to find a solution or to achieve a maximum of one million function evaluations.

We have computed p-values by performing ANOVA tests on the average results to assess the statistical significance of the results. We will consider a 0.05 level of significance. Statistical significant differences among the algorithms are shown with symbol "+" in Tables III–VI, while nonsignificance is shown with "•." In the experiments, we perform 100 independent runs for every problem and algorithm.

In short, we have eleven different algorithms using RW selection plus 11 algorithms more with BT selection. The other

TABLE II
RESULTS WITH NONSTRUCTURED (PANMICTIC) AND STRUCTURED (CELLULAR) SQUARE (20 × 20) POPULATIONS

	Panmictic Population		Cellular Population	
	Binary Tournament	Roulette Wheel	Binary Tournament	Roulette Wheel
MMDP	555505.4	●	160090.30	157167.05
	93%	0%	93%	89%
FMS	●	●	435697.94	520537.23
	0%	0%	59%	87%
P-PEAKS	20904.0	703250.0	54907.93	51499.43
	100%	8%	100%	100%
COUNTSAT	●	●	6845.56	6846.32
	0%	0%	24%	28%
ECC	47887.0	754133.3	167701.21	153497.79
	46%	6%	100%	100%
"cut20.01"	16052.0	11548.0	6054.10	5103.73
	100%	100%	100%	100%
"cut20.09"	14700.0	85940.0	9282.15	9542.80
	100%	100%	100%	100%
"cut100"	319520.0	●	263616.77	176591.80
	5%	0%	63%	58%
"mttp20"	4584.0	4860.0	6058.11	5496.71
	100%	100%	100%	100%
"mttp100"	467901.8	48412.0	194656.43	177950.77
	57%	100%	100%	100%
"mttp200"	962400.0	201200.0	565364.89	514706.56
	1%	99%	100%	100%

variation operators used are the same for the 22 algorithms (see Table I). For each selection method, three algorithms use static ratios, two others employ dynamic *preprogrammed* ratios, and the last six ones make use of the proposed *dynamic adaptive* criteria.

B. Studying the Effects of Structuring the Population

In order to justify the use of a decentralized population, we present in this section the results obtained by two generational GAs working on a panmictic square population, and two cellular GAs with square populations. All these algorithms work with the parameters presented in the previous section.

The results are shown in Table II. As we can see, the GAs with structured populations outperform those with a panmictic population (nonstructured) in almost all the problems. The success rate of the algorithms is always higher (or equal in some cases) for the cellular GAs than in the case of the corresponding nonstructured algorithms. Moreover, among all the algorithms studied in this work, these two algorithms with a panmictic population work out the very undesirable 0% hit rate for some problems. The average hit rate for all the problems is considerably higher in the case of the cellular algorithms for the two selection methods, with values of 85.36% for BT and 87.46% for RW, with respect to the panmictic algorithms (values of 54.72% for BT and 46.64% for RW).

In terms of the average number of evaluations, Table II shows that, in general, the cellular algorithms also perform faster than those using a panmictic population. In addition, we want to emphasize the important differences in the behavior of the panmictic algorithms depending on the selection method: it seems to be quite important in most cases.

C. Experimental Results With Binary Tournament (BT)

In this section, we present the results obtained when solving the problems of our benchmark with all the proposed algorithms using the BT selection method. This selection is expected to reduce the sampling error in small neighborhoods with respect to fitness proportionate selection. The *globally* best values throughout the paper with this selection method (for each problem) are **bolded**; for example, the best tested algorithm—using BT—that solves the FMS problem with the lowest effort (355 209.39 evaluations) is that called *SqNar* (preprogrammed criterion), as shown in Table III.

This section is actually organized into two parts. In the first part, a study of the results obtained with the different static and preprogrammed algorithms is developed for all the problems. The second part contains this same study, but in this case for the adaptive algorithms.

1) Static and Preprogrammed Ratios: In this section, we tackle the optimization of all the problems employing static and preprogrammed ratios and the BT selection operator. Our results confirm those of [17] in which narrow grids were quite efficient for multimodal and/or epistatic problems, while the square and rectangular grids performed better in the case of simple and nonepistatic problems. It must be taken into account that here we are using the BT selection method, while in [17], RW was used.

Looking at Table III, we can see that narrow grids are suitable for multimodal and epistatic problems (*narrow* obtains better results than the others for P-PEAKS, and *SqNar* obtains the best results for FMS). Conversely, these narrow grids perform worse than the others in the case of MMDP (deceptive), and some instances of MTTP (combinatorial optimization). Finally, the two preprogrammed criteria have the worst performances in the case of ECC. In the other problems, there are no statistically significant differences.

Although it is not a globally valid strategy for any unseen problem, our conclusions up to now do explain the common belief that a "good" optimization algorithm must initially seek promising regions, and then gradually search in the

TABLE III
RESULTS WITH STATIC AND PREPROGRAMMED RATIOS USING BINARY TOURNAMENT SELECTION

Problem/Criterion	Static			Pre-programmed		ANOVA
	square	*rectangular*	*narrow*	*SqNar*	*NarSq*	
MMDP	160090.30	182305.67	247841.13	148505.48	172509.28	+
	93%	**96%**	93%	86%	92%	
FMS	435697.94	494400.25	499031.79	**355209.39**	462288.25	+
	59%	67%	68%	42%	59%	
P-PEAKS	54907.93	54984.12	50853.82	63545.47	60674.31	+
	100%	**100%**	**100%**	**100%**	**100%**	
COUNTSAT	6845.56	6851.26	6856.77	6845.18	6849.17	·
	24%	54%	83%	22%	43%	
ECC	167701.21	169469.62	156541.38	190875.00	187658.98	+
	100%	**100%**	**100%**	**100%**	**100%**	
"cut20.01"	6054.10	5769.39	5713.25	5929.79	5785.43	·
	100%	**100%**	**100%**	**100%**	**100%**	
"cut20.09"	**9282.15**	9606.96	9899.69	9775.38	10260.59	·
	100%	**100%**	**100%**	**100%**	**100%**	
"cut100"	263616.77	217509.87	197977.18	223909.83	240346.84	·
	63%	55%	53%	55%	53%	
"mttp20"	6058.11	5753.35	5681.17	6134.30	**5584.93**	+
	100%	**100%**	**100%**	**100%**	**100%**	
"mttp100"	194656.43	201409.27	206177.16	190566.23	194371.72	+
	100%	**100%**	**100%**	**100%**	**100%**	
"mttp200"	565364.89	566279.17	606146.59	565465.14	570016.49	+
	100%	**100%**	**100%**	**100%**	**100%**	

neighborhood of the best so far points. In this process, a clear need of avoiding full convergence to a local optimum is needed, which exactly describes the behavior of the algorithm *SqNar*, since it shifts to a narrow grid to avoid such premature convergence in the second half of the search. In effect, this algorithm has a high accuracy and efficiency on this benchmark.

In summary, these results also conform to the No Free Lunch Theorem (NFL) [40], which predicts that it is not possible to find "a globally best" algorithm for all possible problems. Thus, we included in the test suite a large number of different problems, since only analyzing two or three problems can lead to an undesirable bias in the conclusions. Each algorithm has then been pointed out as the most efficient for a different subclass of problems. However, by inspecting Table III, we can observe just three **bolded** results (statistically, significant differences only for FMS, in general), what means that the overall most efficient algorithm has still to be discovered in this work. As we will see, the different adaptive criteria evaluated in the next subsection will be able of providing the optimum solution at a lower cost with respect to all the nonadaptive ratios for every tested problem (with the exception of FMS).

2) Adaptive Criteria: In this section, we continue our work by addressing the study of the adaptive algorithms proposed in Section IV-B. For each adaptive criterion, we have made two different tests: one beginning at the grid shape with a middle ratio value (r), and the other starting at the lowest one (n).

Also, since the adaptive criteria rely on a small ε value to define what is meant by "too fast convergence," we have developed a first set of the tests for all the criteria with different ε values: 0.05, 0.15, 0.25, 0.3. From all this plethora of tests, we have finally selected $\varepsilon = 0.05$ as the best one. In order to get this conclusion two criteria have been applied, and in both of them $\varepsilon = 0.05$ was the winner. Of course, values other than $\varepsilon = 0.05$ could reduce the effort for a given problem, but we need to select a given value to focus our discussion. The used selection criteria look for an ε value which allows us to obtain: 1) the larger accuracy and 2) the best efficiency. We wanted to make such a formal and statistical study to avoid an ad hoc definition of ε, but we have moved the details to a final Appendix since including the results, tables, and explanations could have distracted the attention of the reader.

Next, we will discuss the results we have obtained with our adaptive cGAs using BT selection (see Table IV), and we will also compare our best adaptive algorithms with all the studied static and preprogrammed ones for each problem. There are statistically significant differences among the adaptive criteria of Table IV in just a few cases. We will emphasize just the highly desirable behavior of nPH and rPH, which are (with statistical significance) the best adaptive criteria for P-PEAKS and ECC (slightly worse than rAF and nAF + PH for the FMS problem).

Comparing the adaptive criteria with the cGAs using static and preprogrammed ratios, we can see (Tables III and IV) that the adaptive algorithms are more efficient in general than the static and preprogrammed ones, standing out the nPH algorithm over the others.

In terms of the efficacy of the algorithms (success rate), we can see that, in general, the adaptive algorithms find the optimal value more frequently out of the 100 tested than the static and preprogrammed ones. Moreover, these adaptive algorithms obtain the best hit rates for almost every problem.

D. Experimental Results With Roulette Wheel (RW)

In this section, we will extend the previous study by using the RW selection method (this is the method used by the authors in [17]). Let us begin by explaining the values worked out by the static, preprogrammed, and dynamic ratio algorithms, summarized in Tables V and VI. Like in Section VI-C, the figures in

TABLE IV
RESULTS WITH ADAPTIVE CRITERIA USING BINARY TOURNAMENT SELECTION

Problem/Criterion	*n*AF	*r*AF	*n*PH	*r*PH	*n*AF+PH	*r*AF+PH	ANOVA
MMDP	155342.46	**137674.50**	180601.45	160747.96	144483.41	162039.13	+
	93%	83%	93%	91%	90%	**96%**	
FMS	493449.91	491922.13	528850.10	555155.67	465396.00	439310.97	+
	64%	61%	73%	**76%**	59%	57%	
P-PEAKS	56475.84	58031.72	**49253.83**	49855.33	53127.49	54350.54	+
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	
COUNTSAT	5023.71	**4678.77**	5296.32	5188.08	4999.59	5252.22	·
	82%	**90%**	89%	86%	88%	88%	
ECC	178921.19	179081.59	**151256.20**	156008.05	193373.23	184058.00	+
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	
"cut20.01"	5416.51	5641.07	5701.22	6082.17	**5340.32**	5657.11	·
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	
"cut20.09"	9554.83	9659.09	9430.52	10015.98	9935.78	9598.94	·
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	
"cut100"	131359.13	142470.83	151048.22	**111662.07**	132574.18	141945.57	·
	45%	46%	46%	39%	43%	41%	
"mttp20"	5729.29	5941.82	6206.48	5941.82	6038.06	5853.60	·
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	
"mttp100"	190425.88	192179.23	**190081.02**	196958.17	192871.98	192579.25	·
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	
"mttp200"	529158.60	523913.52	**523372.17**	525349.10	531019.24	541397.12	·
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	

TABLE V
RESULTS WITH STATIC AND PREPROGRAMMED RATIOS USING ROULETTE WHEEL SELECTION

Problem/Criterion	Static			Pre-programmed		ANOVA
	square	*rectangular*	*narrow*	*SqNar*	*NarSq*	
MMDP	157167.05	183717.22	248984.00	173279.26	162893.30	+
	89%	93%	91%	86%	92%	
FMS	520537.23	545591.67	481716.45	525032.49	552079.84	·
	87%	91%	84%	82%	92%	
P-PEAKS	51499.43	50837.78	**47325.02**	57594.63	58176.08	+
	100%	**100%**	**100%**	**100%**	**100%**	
COUNTSAT	6846.32	6850.50	6857.91	6846.89	6851.07	·
	28%	50%	89%	31%	53%	
ECC	153497.79	151444.67	**148393.06**	168551.33	164100.23	+
	100%	**100%**	**100%**	**100%**	**100%**	
"cut20.01"	5103.73	5224.03	**5075.66**	5869.64	5424.53	+
	100%	**100%**	**100%**	**100%**	**100%**	
"cut20.09"	9542.80	9294.18	9406.46	9923.75	9522.75	·
	100%	**100%**	**100%**	**100%**	**100%**	
"cut100"	176591.80	154500.80	181969.32	184756.24	171190.46	·
	58%	49%	47%	50%	45%	
"mttp20"	5496.71	5653.10	**5236.06**	5260.12	5532.80	·
	100%	**100%**	**100%**	**100%**	**100%**	
"mttp100"	177950.77	169690.17	203446.35	168334.79	177746.26	+
	100%	**100%**	**100%**	**100%**	**100%**	
"mttp200"	514706.56	533818.22	588835.42	525176.67	531195.68	+
	100%	**100%**	**100%**	**100%**	**100%**	

these tables correspond to the average number of function evaluations needed to find an optimal solution, the success rate, and the statistical significance of the comparison of the algorithms. The best values among all the criteria are **bolded**.

This section follows the same structure as that of the previous one. Hence, in the first part, we study the static and preprogrammed algorithms, while in the second one the adaptive criteria are analyzed.

1) Static and Preprogrammed Ratios: Let us focus first on the static criteria. Our first conclusion is that our results mainly confirm those drawn in [17], as in the case of the static algorithms using BT. According to this previous work, we conclude that a static *narrow* ratio is more efficient in the case of highly epistatic and multimodal problems (FMS, P-PEAKS), where its reduced selective pressure helps in exploration; although the difference is only significant for P-PEAKS. Conversely, the high pressure of the static *square* ratio seems best suited for the case of a deceptive multimodal problem like MMDP. Moreover, we additionally conclude that the *square* ratio performs better than the two other static ones for combinatorial problems like MAXCUT or MTTP (in the last one, *Narrow* is the worst criterion with statistical significance).

With respect to the preprogrammed criteria, the reader can verify (by consulting Table V) that the ratio *NarSq*

TABLE VI
RESULTS WITH ADAPTIVE CRITERIA USING ROULETTE WHEEL SELECTION

Problem/Criterion	*n*AF	*r*AF	*n*PH	*r*PH	*n*AF+PH	*r*AF+PH	ANOVA
MMDP	161934.96	153830.66	190365.78	168471.30	**142911.49**	172794.01	+
	87%	**96%**	95%	83%	90%	90%	
FMS	503863.63	529495.51	512789.91	**478175.61**	515448.56	538109.07	+
	89%	**93%**	87%	85%	85%	85%	
P-PEAKS	54390.64	54222.22	49518.49	48985.16	51423.24	51327.00	+
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	
COUNTSAT	6858.86	5284.26	5292.34	**4947.50**	5284.33	5725.35	·
	94%	92%	86%	84%	85%	93%	
ECC	157239.12	160840.10	151344.42	148425.14	166794.95	168535.29	+
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	
"cut20.01"	5452.60	5705.23	5937.81	5436.56	5484.68	5436.56	·
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	
"cut20.09"	**8953.33**	9322.25	9803.45	9117.74	9326.26	9338.29	·
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	
"cut100"	**130035.85**	178652.97	140662.35	193774.65	157464.19	143617.70	·
	43%	55%	43%	**58%**	49%	45%	
"mttp20"	5520.77	5665.13	5781.42	5584.93	5572.90	5889.69	·
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	
"mttp100"	170861.09	**166325.78**	192338.65	197258.92	180244.49	180757.77	+
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	
"mttp200"	**494022.98**	495542.77	527261.87	524799.73	508446.95	499215.93	+
	100%	**100%**	**100%**	**100%**	**100%**	**100%**	

TABLE VII
COMPARING THE BEST ADAPTIVE CRITERIA VERSUS NONADAPTIVE RATIOS USING BINARY TOURNAMENT AND ROULETTE WHEEL (BT, RW)

Problem	Best Adaptive Criterion (BT,RW)	*square*	*rectangular*	*narrow*	*SqNar*	*NarSq*
MMDP	*r*AF , *n*AF+PH	·, ·	·, +	+, +	·, +	·, ·
FMS	*r*AF+PH , *r*PH	·, ·	·, ·	·, ·	·, +	·, ·
P-PEAKS	*n*PH , *r*PH	+, +	+, ·	·, ·	+, +	+, +
COUNTSAT	*r*AF , *r*PH	+, ·	·, ·	·, ·	·, ·	·, ·
ECC	*nPH* , *r*PH	+, ·	+, ·	·, ·	+, +	+, +
"cut20.01"	*n*AF+PH , *r*PH	·, ·	·, ·	·, ·	·, ·	·, ·
"cut20.09"	*n*PH , *n*AF	·, ·	·, ·	·, ·	·, ·	·, ·
"cut100"	*r*PH , *n*AF	·, ·	·, ·	·, ·	·, ·	+, ·
"mttp20"	*n*AF , *n*AF	·, ·	·, ·	·, ·	·, ·	·, ·
"mttp100"	*n*PH , *r*AF	·, ·	·, ·	+, +	·, ·	·, ·
"mttp200"	*n*PH , *n*AF	+, ·	+, +	+, +	+, +	+, +

(exploration-to-exploitation) does not improve the best results obtained with respect to the rest of static and preprogrammed studied criteria in any case. Besides, the other proposed preprogrammed ratio *SqNar* (which represents a change for larger diversity in the second phase of the algorithm) outperforms the efficiency of the other criteria of Table V only for the instance "mttp100," although there is no statistical significance. The two preprogrammed criteria are worse than all the static ones with statistical significance in the cases of ECC and P-PEAKS problems.

2) Adaptive Criteria: As we did in Section VI-C2, we will study the behavior of our adaptive algorithms with the RW selection method, and compare them with the static and preprogrammed ones. In Table VI, we show the results obtained with the six adaptive criteria. The first conclusion we can draw from that table is that, as in the case of Section VI-C2, the behavior of the algorithms is globally improved by making them adaptive (notice the large number of boldfaced results, i.e., most efficient algorithms).

Moreover, the most effective algorithm for every problem is always an adaptive one. The reader can verify it in Tables V and VI, since the highest hit rates for every problem are obtained by the adaptive algorithms. Hence, we can conclude that, in general, the adaptive algorithms (either using binary tournament or roulette wheel) outperform the other studied ones in the two checked features: efficiency and efficacy.

In Table VII, we compare, for every problem, our best adaptive algorithm with the static and preprogrammed ones. This table accounts for algorithms using both BT and RW selection methods. Symbol "+" means that the adaptive algorithm is better with statistical significance than the compared nonadaptive one, while "·" stands for nonsignificant differences. We can see that, for the two selection methods, there is not any static or preprogrammed algorithm better than the best adaptive one for every problem. Hence, the **bolded** values in Tables III and V corresponding to a best nonadaptive algorithm are undistinguishable of the adaptive ones. For all the problems, the existing statistically significant differences favors the adaptive criteria against the static and preprogrammed ones.

E. Additional Discussion

If we look among all the adaptive criteria, we conclude (as expected from the theory) that there is no adaptive criterion that significatively outperforms the rest for all the test suite,

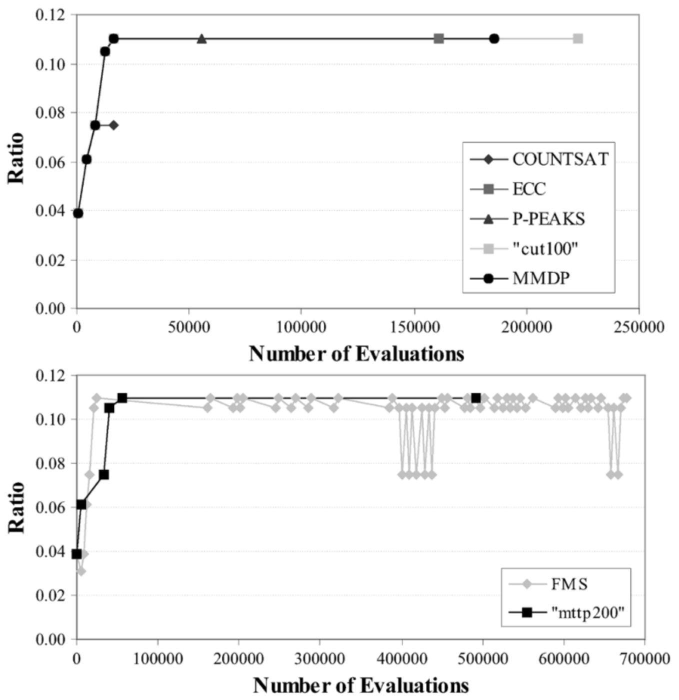

Fig. 11. Dynamics of the ratio observed when using rAF.

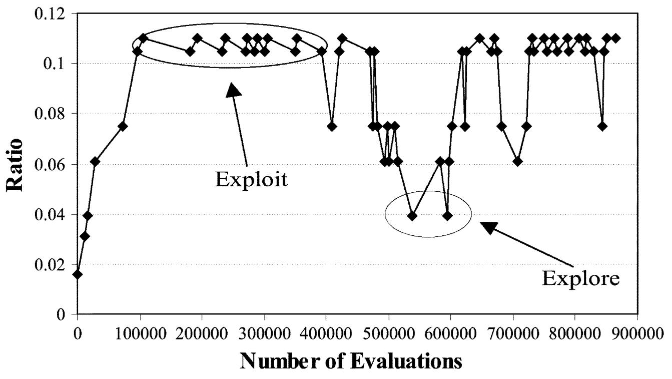

Fig. 12. Typical evolution of the ratio for the FMS problem with algorithm $n\mathrm{AF} + \mathrm{PH}$.

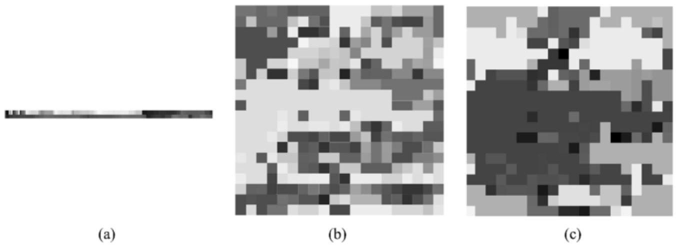

Fig. 13. (a) Population at the beginning, (b) middle, and (c) end of a sample execution of MAXCUT ("cut100") with $n\mathrm{AF} + \mathrm{PH}$.

according to the calculated p-values. However, it is clear after these results that adaptive ratios are the first kind of algorithm one must select in order to have a really good baseline for comparisons. For the problems evaluated and for other problems sharing their same features, we make a contribution by defining new competitive and robust algorithms that can improve the existing performance at a minimum implementation requirement (changing the grid shape).

In order to provide an intuitive explanation of what is happening during the search of the algorithms, we first track the evolution of the ratio value along the search (RW selection used in this section). Second, we have taken some snapshots of the population at different points of the evolution in order to study the diffusion of solutions through the population. Finally, we have studied a representative convergence scenario of an adaptive algorithm through the evolution.

In Fig. 11, we have drawn an example of the ratio fluctuation automatically produced by the rAF adaptive criterion during a typical execution on all the problems (we just plot the hardest instances in the case of MAXCUT and MTTP: "cut100" and "mttp200"). We have plotted separately in the bottom part of the figure the hardest problems: MTTP with 200 variables "mttp200" and FMS. Criterion rAF has been selected to describe the typical scenarios found with all the adaptive algorithms.

In Fig. 11, it can be noticed a clear trend toward promoting exploitation (ratios over 0.1). These general trend to evolve toward the square shape is expected, because the search focuses toward population exploitation after an initial phase of exploration.

This shift from small to high ratios is detected also in all the problems (top of Fig. 11). However, the adaptive algorithms seem to need a more complex search of the optimum ratio when faced with the most complex cases: FMS and MTTP. In these two cases (Fig. 11 down), there are brief and periodic visits to smaller ratios (below 0.08) with the goal of improving the exploration of the algorithm. This automatic alternate shifting between exploitation and exploration in some problems is a noticeable feature of the adaptive mechanism we propose, because it shows how the algorithm decides by itself when to explore and when to exploit the population.

But, we want to show that this search for the appropriate exploration/exploitation tradeoff can be achieved in a gradual and progressive way. In order to prove this claim, we zoom into the behavior of the nAF + PH algorithm for the FMS problem (Fig. 12). We notice that when the algorithm finds difficulties during an exploiting phase (high ratio), it decreases the ratio (even repeatedly) to improve the exploration. This provides, in general, a higher hit rate for the adaptive cGAs with respect to the other ones. In Fig. 12, we can see that the ratio slope is developed by the adaptive algorithm to appropriately regulate by itself the speedup of the search for the optimum.

Now, let us turn to analyze the evolution for MAXCUT and FMS from a phenotypic diversity point of view. The pictures in Figs. 13 and 14 have been taken at the beginning, middle, and end of a typical execution with an adaptive criteria which begins using the narrowest population shape (specifically nAF + PH). These figures show three snapshot representations of the fitness distribution in the population at different stages of the search. Different gray tones mean different fitness values, and darker gray tones correspond to the best individuals. It can be observed in the two figures that diversity (variety of gray tones) decreases during the run, as expected.

The particular case of MAXCUT with the instance "cut100" (Fig. 13) is a sample scenario of fast evolution toward the square population shape during the execution of the problem; once the

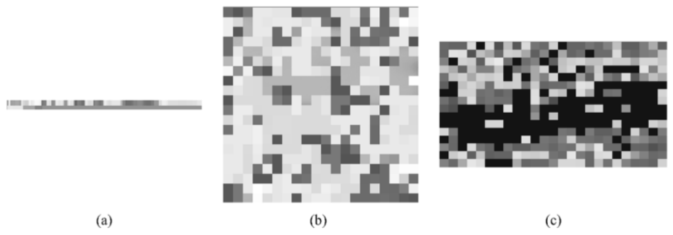

Fig. 14. (a) Population at the beginning, (b) middle, and (c) end of a sample execution of FMS with $n\mathrm{AF} + \mathrm{PH}$.

algorithm has reached the square shape, it is maintained until the solution is found (see also Fig. 11). The reason for this behavior is that the exploration/exploitation tradeoff that the algorithm finds is adequate to maintain a permanent and not too fast convergence of the individuals toward the solution. This convergence can be observed in Fig. 13, where we can notice that the number of grey tones in the population decreases along the execution. The convergence is noticeable at the middle of the execution [Fig. 13(b)] and continues growing until the optimal solution is found [Fig. 13(c)]. Also, since the same ratio value is kept during all the evolution (except at the beginning), no relocation of individuals (which introduces diversity) is made, and the algorithm inherently creates islands of similar fitness values [Fig. 13(c)], corresponding to those areas with the same gray tone. These different areas represent different convergence paths motivated by the smooth diffusion of good solutions, a distinctive feature of cGAs. In particular, the exploration of the search space is specially enforced in the bounds of different colored regions.

In the case of FMS, a similar scenario appears; in this case, we plot an example situation in which the algorithm is near to convergence to a local optimum. As soon as the adaptive criterion detects such a condition, it tries to get out of this local optimum by promoting exploration (as it also occurs in Fig. 12). We can see in Fig. 14(b) that the diversity is smaller than in the case of MAXCUT in the middle of the execution. Since most individuals have similar fitness values, when a better solution appears in the grid it will be spread through all the population reasonably fast. This effect produces an acceleration in the mean fitness of the individuals and, therefore, a change to the next more rectangular grid shape (if possible) is performed, thus introducing some more diversity at the neighborhoods for the next generations. Contrary to this reasoning, a high level of diversity is maintained during all the evolution because, due to the epistatic nature of the problem, small changes in the individuals lead to big differences in the fitness values. That is the reason, as well as the high number of ratio changes made for this problem, for the absence of homogeneous colored areas. It also justifies the important levels of diversity present at the end of the evolution for FMS (in relation to the MAXCUT problem).

Finally, we want to exemplify the differences in the convergence speed of the three main classes of the studied algorithms (static, preprogrammed, and adaptive). Hence, we plot in Fig. 15 the maximum fitness value in every generation in typical runs for three different algorithms (*square, SqNar*, and rPH) when solving the instance of 100 vertices of the MAXCUT problem ("cut100"). As it can be seen, the evolution of the best so far solution during the run is quite similar in the three algorithms. The difference in the behavior of the algorithms can be seen at the end of the evolution, where *square* and *SqNar* get stuck during many generations before finding the optimal solution, while the adaptive algorithm, due to the diversity of the population, finds quickly the optimal solution.

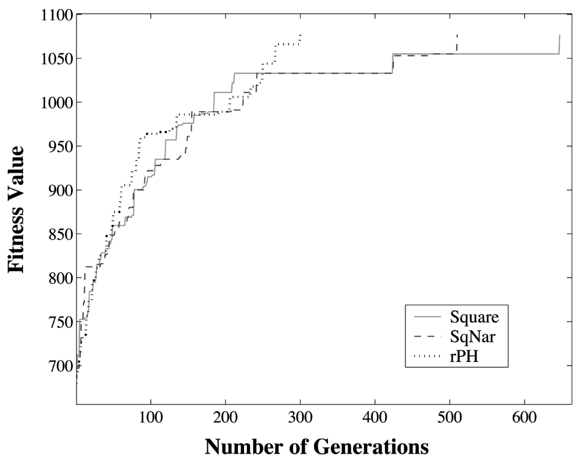

Fig. 15. Evolution of the maximum fitness found with "cut100."

VII. Conclusion and Further Work

In this paper, we have analyzed the behavior of many cEAs over an assorted set of problems. The ratio, defined as the relationship between the neighborhood and the population radii, represents a unique parameter rich in possibilities for developing efficient algorithms based on the canonical cGA. As the population is distributed in a 2-D toroidal grid, the studied models are all embedded in many other similar algorithms; thus, we hope these results to be broadly helpful for other researchers.

Our main motivation for this work has been to advance the knowledge of the appropriate exploration/exploitation tradeoff for an algorithm. Specifically, we have studied the influence of the ratio on the research with structured dynamical populations. This feature is, to our knowledge, used in this paper for the first time.

Technically, we can confirm the results in [17], which concluded that a narrow grid (small ratio) is quite efficient for multimodal and/or epistatic problems, while it performs slowly with respect to the square and rectangular grids (high ratios) in the case of simple and nonepistatic problems. Additionally, we have demonstrated the importance of using an intermediate grid (*rectangular*) for combinatorial optimization problems in this work.

We must emphasize that any of the developed adaptive criteria is just a new light and easy way to obtain competent search techniques. An important feeling after this work is that the search can be easily guided by using structured populations. These kind of models are very efficient from a numerical point of view, while they still admit any advanced technique developed in the EA field to be included in their basic execution. These adaptive algorithms outperform the other studied ones for all the proposed problems. The exceptions do not have

TABLE VIII
VALUES OF ε ACHIEVING THE BEST ACCURACY

$Problem/Cr.$	nAF	rAF	nPH	rPH	nAF+PH	rAF+PH
MMDP	**0.30**	0.25	0.30	0.15	0.05	0.15
FMS	0.30	0.05	**0.25**	0.30	0.05	0.25
P-PEAKS	0.05	0.15	**0.15**	0.05	0.25	0.05
COUNTSAT	0.15	0.30	0.15	0.15	**0.05**	0.30
ECC	0.15	0.15	**0.30**	0.30	0.15	0.15
"cut20.01"	0.05	0.15	**0.05**	0.05	0.05	0.30
"cut20.09"	0.05	0.25	0.05	**0.05**	0.25	0.05
"cut100"	0.25	**0.05**	0.25	0.25	0.25	0.15
"mttp20"	0.15	0.15	0.05	**0.15**	0.05	0.30
"mttp100"	0.30	**0.05**	0.05	0.05	0.30	0.30
"mttp200"	0.30	0.05	0.05	0.05	0.05	**0.30**

TABLE IX
VALUES OF ε ACHIEVING THE BEST EFFICIENCY

$Problem/\ \varepsilon$	**0.05**	**0.15**	**0.25**	**0.3**
MMDP	**183146.70**	189793.80	210315.40	187191.80
FMS	699641.20	703694.90	**646679.20**	692630.60
P-PEAKS	**51710.17**	52267.12	52137.91	52362.91
COUNTSAT	19000.63	**17388.93**	20064.32	18178.54
ECC	160913.60	**157456.10**	161247.80	158979.90
"cut20.01"	**14492.92**	17108.33	19212.00	16179.35
"cut20.09"	**29153.93**	35291.46	34946.15	35362.75
"cut100"	319606.90	336478.80	**283714.30**	303075.70
"mttp20"	9545.02	**9326.71**	11641.36	9605.18
"mttp100"	**234022.60**	273287.20	271051.60	241260.70
"mttp200"	**504780.00**	528445.70	552269.60	542810.40

TABLE X
FINAL RANKING OF ε VALUES. THE BEST VALUE ACCORDING TO OUR TWO CRITERIA IS $\varepsilon = 0.05$

Accuracy	Efficiency	ϵ	sum of positions	rank
0.05	0.05	0.05	2	1
0.15	0.15	0.15	4	2
0.30	0.25	0.25	7	3
0.25	0.30	0.30	7	3

statistical differences. However, we did not validate our conclusions by using binary tournament and fitness proportionate selection methods.

As a future work, we propose the study of new, more sophisticated criteria for the adaptation. Examples of this kind of criteria may be the use of other techniques based on the fitness value, or other indicators that could meaningfully describe the evolution. In any case, it is desirable to utilize simple measuring criteria in order to avoid translating the adaptation advantage into a disadvantage due to the induced overhead in terms of actual execution time.

As extensions to this work, we are analyzing the conclusions over other complex problems, and the influence of other individual relocation mechanisms in the new population when a change in the grid shape is to be performed.

APPENDIX

As stated in Section VI-D2, we focus on the importance of the threshold for considering that the search is loosing diversity too quickly. For this goal, the behavior of every adaptive criteria has been studied under four different ε values. We need to define an appropriate single ε value to be used in the experiments of the paper body to focus the discussion on the criteria themselves. Clearly, we do not need the "best" ε threshold value, since it is problem dependent.

We have studied four small ε values (RW selection case): 0.05, 0.15, 0.25, and 0.3, which stand for highly to slightly restrictive conditions (respectively) to change the ratio.

We have followed two complementary criteria for defining a single ε. The first one is to get the ε value that, for every problem and criterion, gets the maximum accuracy (Table VIII). We have emphasized in that table with **bold** fonts the ε values that report the most accurate result for each problem and criterion.

The second criterion selects the ε provoking the higher efficiency (Table IX). We get it by computing for each ε value the mean number of the evaluations needed to solve each problem with any criterion. The more efficient ε for each problem (**bolded** values in Table IX) will be that provoking the smaller average number of evaluations.

We then compute a final ranking that sorts the best ε values for each criterion and assigns a weight to each position (smaller weight means better performance). The best ε value will be that with the smaller resulting weight. In fact, this is a multiobjective optimization problem that we have solved by a linear combination (sum) of the rankings achieved in the two considered criteria. This ranking is shown in Table X.

After ranking all the ε values by the two criteria a clear winner springs out: $\varepsilon = 0.05$ is the value that globally provides more accurate and efficient results for all the criteria and problems in our benchmark.

ACKNOWLEDGMENT

The authors gratefully acknowledge the comments of the anonymous reviewers that really helped to improve this work.

REFERENCES

[1] T. Bäck, D. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. London, U.K.: Oxford Univ. Press, 1997.
[2] E. Cantú-Paz, *Efficient and Accurate Parallel Genetic Algorithms*, 2nd ed, ser. Book Series on Genetic Algorithms and Evolutionary Computation. Norwell, MA: Kluwer, 2000, vol. I.
[3] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 443–462, Oct. 2002.
[4] E. Alba and J. Troya, "Improving flexibility and efficiency by adding parallelism to genetic algorithms," *Statist. Comput.*, vol. 12, no. 2, pp. 91–114, 2002.
[5] B. Manderick and P. Spiessens, "Fine-grained parallel genetic algorithm," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. Schaffer, Ed., 1989, pp. 428–133.
[6] Q. Zhang and H. Muhlenbein, "On the convergence of a class of estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 127–136, Apr. 2004.
[7] P. Spiessens and B. Manderick, "A massively parallel genetic algorithm," in *Proc. 4th Int. Conf. Genetic Algorithms*, R. Bclew and L. Booker, Eds., 1991, pp. 279–286.
[8] S. Baluja, "Structure and performance of fine-grain parallelism in genetic search," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed., 1993, pp. 155–162.
[9] H. Mühlenbein, M. Schomish, and J. Born, "The parallel genetic algorithm as a function optimizer," *Parallel Comput.*, vol. 17, pp. 619–632, 1991.
[10] K. Ku, M. Mak, and W. Siu, "Adding learning to cellular genetic algorithms for training recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 2, pp. 239–252, Mar. 1999.
[11] G. Folino, C. Pizzuti, and G. Spezzano, "Parallel hybrid method for SAT that couples genetic algorithms and local search," *IEEE Trans. Evol. Comput.*, vol. 5, no. 4, pp. 323–334, Aug. 2001.

[12] M. Tomassini, "The parallel genetic cellular automata: Application to global function optimization," in *Proc. Int. Conf. Artif. Neural Netw. Genetic Algorithms*, R. Albrecht, C. Reeves, and N. Steele, Eds, 1993, pp. 385–391.

[13] D. Whitley, "Cellular genetic algorithms," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed., 1993, p. 658.

[14] V. Gordon and D. Whitley, "Serial and parallel genetic algorithms as function optimizers," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed., 1993, pp. 177–183.

[15] J. Sarma and K. De Jong, "An analysis of the effect of the neighborhood size and shape on local selection algorithms," in *Lecture Notes in Computer Science*, H. Voigt, W. Ebeling, I. Rechenberg, and H. Schwefel, Eds. Berlin, Germany: Springer-Verlag, 1996, vol. 1141, Proc. Int. Conf. Parallel Prob. Solving from Nature IV, pp. 236–244.

[16] ——, "An analysis of local selection algorithms in a spatially structured evolutionary algorithm," in *Proc. 7th Int. Conf. Genetic Algorithms*, T. Bäck, Ed., 1997, pp. 181–186.

[17] E. Alba and J. Troya, "Cellular evolutionary algorithms: Evaluating the influence of ratio," in *Lecture Notes in Computer Science*, vol. 1917, Proc. Int. Conf. Parallel Prob. Solving from Nature VI, M. Schoenauer, Ed.. Berlin, Germany, 2000, pp. 29–38.

[18] M. Giacobini, E. Alba, and M. Tomassini, "Selection intensity in asynchronous cellular evolutionary algorithms," in *Proc. Genetic Evol. Comput. Conf.*, 2003, pp. 955–966.

[19] F. Herrera and M. Lozano, "Gradual distributed real-coded genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 4, no. 1, pp. 43–62, Apr. 2000.

[20] P. Adamidis and V. Petridis, "Co-operating populations with different evolution behaviors," in *Proc. 3rd IEEE Conf. Evol. Comput.*, 1996, pp. 188–191.

[21] *Handbook of Applied Optimization, Memetic Algorithms*, P. Pardalos and M. Resende, Eds., Oxford Univ. Press, London, U.K., 2000. P. Moscato.

[22] Y. Ong and A. Keane, "Meta-Lamarckian teaming in memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, Apr. 2004.

[23] R. Hinterding, Z. Michalewicz, and A. Eiben, "Adaptation in evolutionary computation: A survey," in *Proc. IEEE Conf. Evol. Comput., IEEE World Congr. Comput. Intell.*, 1997, pp. 65–69.

[24] T. Bäck, "Introduction to the special issue: Self-adaptation," *Evol. Comput.*, vol. 9, no. 2, pp. iii–iv, 2001.

[25] P. Angeline, "Adaptive and self-adaptive evolutionary computations," in *Computational Intelligence: A Dynamic Systems Perspective*, M. Palaniswami and Y. Attikiouzel, Eds. Piscataway, NJ: IEEE Press, 1995, pp. 152–163.

[26] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. Schaffer, Ed., 1989, pp. 61–69.

[27] J. Arabas, Z. Michalewicz, and J. Mulawka, "GAVaPS—A genetic algorithm with varying population size," in *Proc. IEEE Conf. Evol. Comput.*, vol. 1, 1994, pp. 73–78.

[28] D. Schlierkamp-Voosen and H. Mühlenbein, "Adaption of population sizes by competing subpopulations," in *Proc. IEEE Conf. Evol. Comput.*, 1996, pp. 330–335.

[29] D. Whitley, S. Rana, J. Dzubera, and K. Mathias, "Evaluating evolutionary algorithms," *Artif. Intell.*, vol. 85, pp. 245–276, 1997.

[30] D. Goldberg, K. Deb, and J. Horn, "Massively multimodaliry, deception, and genetic algorithms," in *Proc. Int. Conf. Parallel Prob. Solving from Nature II*, R. Männer and B. Manderick, Eds., 1992, pp. 37–46.

[31] S. Tsutsui and Y. Fujimoto, "Forking genetic algorithm with blocking and shrinking modes," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed., 1993, pp. 206–213.

[32] K. De Jong, M. Potter, and W. Spears, "Using problem generators to explore the effects of epistasis," in *Proc. 7th Int. Conf. Genetic Algorithms*, T. Bäck, Ed., 1997, pp. 338–345.

[33] S. Droste, T. Jansen, and I. Wegener, "A natural and simple function which is hard for all evolutionary algorithms," in *Proc. 3rd Asia-Pacific Conf. Simulated Evol. Learning*, 2000, pp. 2704–2709.

[34] C. Papadimitriou, *Computational Complexity*. Reading, MA: Addison-Wesley, 1994.

[35] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.

[36] H. Chen, N. Flann, and D. Watson, "Parallel genetic simulated annealing: A massively parallel SIMD algorithm," *IEEE Trans. Parallel Distrib. Syst.*, vol. 9, no. 2, pp. 126–136, 1998.

[37] S. Khuri, T. Bäck, and J. Heitkötter, "An evolutionary approach to combinatorial optimization problems," in *Proc. 22nd ACM Comput. Sci. Conf.*, 1994, pp. 66–73.

[38] D. Stinson, *An Introduction to the Design and Analysis of Algorithms*, 2nd ed. Winnipeg, MB, Canada: The Charles Babbage Research Center, 1985, 1987.

[39] J. Smith and F. Vavak, "Replacement strategies in steady state genetic algorithms: Static environments," in *Foundations of Genetic Algorithms V*, W. Banzhaf and C. Reeves, Eds: Morgan Kaufmann, 1998, pp. 219–234.

[40] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

Enrique Alba received the Ph.D. degree in designing and analyzing parallel and distributed genetic algorithms from the University of Málaga, Málaga, Spain, in 1999.

He is a Professor of Computer Science at the University of Málaga. He has published many scientific papers in international conferences and journals, as well as he holds national and international awards to his research results. His current research interests involve the design and application of evolutionary algorithms, neural networks, and other bioinspired systems to real problems including telecommunications, combinatorial optimization, and bioinformatics. The main focus of all his work is on parallelism.

Bernabé Dorronsoro received the Engineering degree in computer science from the University of Málaga, Málaga, Spain, in 2002. He is currently working towards the Ph.D. degree in the design of new evolutionary algorithms, specifically on structured algorithms, and their application to complex problems in the domains of logistics, telecommunications, and combinatorial optimization.