Extending Cellular Evolutionary Algorithms with Message Passing*

Severino F. Galán
Artificial Intelligence Dept. at UNED
C/ Juan del Rosal, 16. 28040 Madrid, Spain
E-mail: seve@dia.uned.es

Abstract

Cellular evolutionary algorithms (cEAs) use structured populations whose evolutionary cycle is governed by local interactions among individuals. This helps to prevent the premature convergence to local optima that usually takes place in panmictic populations. The present work extends cEAs by means of a message passing phase whose main effect is a more effective exploration of the search space. The mutated offspring that potentially replaces the original individual under cEAs is considered under *message passing cellular evolutionary algorithms* (MPcEAs) as a message sent from the original individual to itself. In MPcEAs, unlike in cEAs, a new message is sent from the original individual to each of its neighbors, representing a neighbor's mutated offspring whose second parent is selected from the neighborhood of the original individual. Thus, every individual in the population ultimately receives one additional candidate for replacement from each of its neighbors rather than having a unique candidate. Experimental tests conducted in the domain of real function optimization for continuous search spaces show that, in general, MPcEAs significantly outperform cEAs in terms of effectiveness. Specifically, the best solution obtained through MPcEAs has an importantly improved fitness quality in comparison to that obtained by cEAs.

Keywords: optimization problem, cellular evolutionary algorithm, message passing, effective exploration.

1 Introduction

Evolutionary algorithms (EAs) [6, 7, 15, 11] use stochastic search methods based on natural evolution in order to solve optimization problems in fields such as design, learning, or scheduling, among others. An EA obtains a population of individuals each generation, where an individual represents a potential solution to the optimization problem. The fitness of an individual determines its chance to survive and reproduce. Two processes form the basis of EAs: variation (recombination and mutation) and selection. While the former facilitates diversity and novelty, the latter favors

*This version of the article has been accepted for publication after peer review, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available at:
https://doi.org/10.1007/s00500-021-05612-9
Use of this Accepted Version is subject to the publisher's Accepted Manuscript terms of use:
https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms

quality. Ideally, at the end of running an EA, an individual with optimal or near-optimal fitness is found.

Premature convergence to local optima is one of the most frequent difficulties that arises when applying panmictic EAs to complex problems. (In a panmictic EA, any individual can interact with any other individual in the population.) It occurs when genetic operators can no longer generate offspring that are fitter than their suboptimal parents. Premature convergence is associated with the loss of diversity in the population. However, too much population diversity can lead to a dramatic deterioration of EA efficiency. Therefore, an important issue in the design and application of EAs is the tradeoff between exploitation of the best individuals and exploration of alternative regions of the search space.

Landscape orography is one of the main factors determining the type of interaction among the individuals of a population and has an influence on the distribution of the great diversity of species in our planet. *Cellular evolutionary algorithms* (cEAs) [19, 26, 28, 1, 32] constitute a technique that models how the orographic factor determines the interaction among individuals and maintains the population diversity. In this technique, parallelism arises due to the fact that a processing unit can be dedicated to simulate the interactions between an individual and its neighbors. These local interactions among individuals are the only ones possible in a cEA. The term "cellular" specifies that each processing unit performs easy and fast computations. In fact, the equivalence between cEAs and cellular automata was proved in [33].

Several works [10, 5, 31, 12] report that cEAs perform better than panmictic EAs in many cases. This is due to the fact that cEAs maintain a higher diversity in the population and accomplish a better tradeoff between exploration and exploitation in the search space. Since individuals interact locally in a cEA, the genetic information of the fittest individuals spreads slowly throughout the population, analogously to a diffusion process.

The goal of the present work is to enhance the effectiveness of cEAs to reach good solutions, that is, to obtain solutions whose fitness is closer to that of the global optimum. This is accomplished by carrying out additional local interactions among individuals. In a cEA, given the neighborhood of the original individual, a mutated offspring is generated from the neighborhood in order to potentially replace the original individual. This operation can be extended by generating a new mutated offspring for each individual in the neighborhood. The whole process can be developed in the context of a message passing mechanism, where every individual in the population receives a candidate individual for potential replacement from each of its neighbors. Thus, the resulting approach is named *message passing cellular evolutionary algorithm* (MPcEA). We have conducted thorough experimental tests in order to compare cEAs and MPcEAs in the domain of real function optimization. The results show that MPcEAs significantly outperform cEAs in many cases.

Given the promising results provided by MPcEAs in terms of effectiveness in comparison to those offered by cEAs, MPcEAs constitute a convenient method to be applied to any optimization problem that can be tackled from an evolutionary perspective. A detailed list of these problems can be found in the following references: [6, Chapter 2], [9, Chapter 6], and [8]. Thus, in general, MPcEAs can be used in practice in a broad range of real-world domains of interest.

The rest of this paper is organized as follows. Section 2 reviews previous work on cEAs. Section 3 introduces the novel MPcEA approach. Section 4 includes an extensive empirical evaluation of MPcEAs. Finally, Section 5 contains the main conclusions and discusses future research directions.

2 Related work on cEAs

In a cEA, each node of a network with fixed structure contains one of the individuals in the population. Usually, the network structure is designed as a two-dimensional square lattice. Every individual in the lattice has direct links with a set of neighbors and interacts only with them. Like in cellular automata, the von Neumann and the Moore neighborhoods are the most popular ones. In a two-dimensional square lattice, the von Neumann neighborhood is formed by a cell and its vertical and horizontal neighbors, whereas the Moore neighborhood incorporates the diagonal neighbors. When periodic boundary conditions are defined for a finite two-dimensional square lattice using von Neumann neighborhood, the population is structured as a toroid.

Algorithm 1 contains the pseudocode for a canonical cEA. In the first two steps, each lattice cell is initialized with an individual whose fitness is calculated (see lines 1 and 2). Usually, the initialization is performed in a random way, although specific knowledge-based methods for the problem at hand can be used. Then, the following operations are carried out for each individual at each generation in the typical implementation of a canonical cEA:

1. Two parents are selected from the set formed by the original individual and its neighbors (see lines 5 and 6). The selection can be conducted in a random way or through any fitness-based method. Usually, the original individual is chosen as a parent, and one of its neighbors is selected to act as the other parent.

2. A child is generated by means of recombination, also known as crossover (see line 7).

3. The generated child is mutated, and its fitness is evaluated (see lines 8 and 9).

4. A decision is taken as to whether or not the original individual is replaced by the mutated child (see line 10). Usually, the replacement takes place when the fitness of the mutated child improves that of the original individual.

In summary, both parent selection and individual replacement (also known as survivor selection) are allowed only in the neighborhood of each individual. It is interesting to note that, as discussed in [29], the operations repeated during a generation over all the cells in the lattice can be performed either sequentially (asynchronously) or in parallel (synchronously). Actually, Algorithm 1 considers synchronous updating of individuals, implemented in a sequential manner. To that end, an auxiliary population is created that stores the partial results obtained throughout a generation.

2.1 Advanced techniques in cEAs

The canonical cEA has been enhanced with advanced techniques in order to increase both its effectiveness and its efficiency. An in-depth review of these advanced techniques can be found in [1, 13].

The structured population makes cEAs appropriate for parallel implementation. Fast parallel cEAs can be designed from two perspectives: (1) By assigning a single individual to each processor of massively parallel computers [26], or (2) By evolving several cEAs in parallel and connecting them in an arbitrary manner (for example, as a ring) in order to allow the occasional migration of individuals [24, 25]. A detailed survey of parallel cEAs can be found in [1, Chapter 8].

Hierarchical cEAs [22] create a hierarchy in the population such that better individuals are moved towards the same region of the lattice. In this way, a faster and more effective convergence of the best individuals is achieved in this region, while the diversity is maintained in other distant

Algorithm 1 Canonical cEA

Input:
An optimization problem
The usual parameters of a canonical EA
A lattice where each cell will be occupied by an individual at each generation

Output:
The population of individuals in the lattice at the final generation

```
 1: Initialize(population);
 2: Evaluate(population);
 3: while not stop-condition
 4:    for i ← 1 to population.size
 5:       neighborhood ← population.individual[i] ∪ CalculateNeighbors(population.individual[i]);
 6:       parents ← Select(neighborhood);
 7:       child ← Recombine(parents);
 8:       Mutate(child);
 9:       Evaluate(child);
10:       auxiliary-population.individual[i] ← Replace?(population.individual[i], child);
11:    end-for
12:    population ← auxiliary-population; (synchronous updating)
13: end-while
```

locations. A hierarchical cEA is similar to a canonical cEA except that the population is rearranged after every generation with a fitness-based swap operation between contiguous cells in the lattice.

An important issue in EAs is how to achieve a proper tradeoff between exploration and exploitation during the search process, which depends on the optimization problem at hand and the elapsed number of generations. The shape of the lattice has a great influence on the exploratory capabilities of a cEA. For example, in the case of $N \times M$ lattices where N and M vary and $N \cdot M$ (the population size) remains constant, the linear $(N \cdot M) \times 1$ lattice produces the highest degree of exploration, whereas the square $\sqrt{N \cdot M} \times \sqrt{N \cdot M}$ lattice yields the highest degree of exploitation. In [4, 1], the degree of exploration of a cEA is controlled by modifying the shape of the lattice or, in other words, by changing N and M such that $N \cdot M$ is kept constant. This control can be performed: (1) dynamically at specific generations [4], or (2) adaptively according to the evolution of population diversity measures like entropy or population convergence measures like average fitness [1, Section 6.2.2].

Parameter control is important for EAs in general and for cEAs in particular. The specific values assigned to the parameters determine the degree of exploration applied during the search process. Usually, finding the optimal set of parameter values is a costly and difficult problem for the user, since it depends on the complexity of the problem at hand. The terrain-based genetic algorithm [18] is a cEA characterized by spreading a range of parameter values evenly along the axes of the lattice. Cells in different locations are subjected to different parameter values, although the parameter values in neighboring cells are similar. The experimental results reported in [18] indicate that the terrain-based genetic algorithm performs better than the canonical cEA, even when the latter is assigned parameter values thought to be good in prior studies. Interestingly, the terrain-based genetic algorithm can be used to automatically determine better parameter settings for the canonical cEA. The resulting canonical cEA produces even better results than those achieved by the terrain-based genetic algorithm that found those parameter settings.

A drawback of cEAs is that they slowly converge to near-optimal solutions. Both the efficiency and the effectiveness of cEAs can be increased by hybridizing them with local search. Cellular memetic algorithms [17, 2, 21] apply an additional phase of local search improvement to the

mutated offspring in a cEA. Since an exhaustive local search on all individuals in the population can be computationally expensive, some methods exist that restrict the local search to a subset of the individuals [21].

In multiobjective optimization through evolutionary methods, the goal is to estimate the set of non-dominated solutions (or Pareto set) as accurately as possible. To that end, it is important to maintain the population diversity throughout the generations, since the shape of the Pareto front is unpredictable in advance. Due to the fact that structured populations promote diversity maintenance, cEAs can be effectively applied to multiobjective optimization problems. The first multiobjective cEA, called cMOGA [3], was used to optimize the broadcasting strategy in ad hoc mobile networks. A review of relevant multiobjective cEAs, including an improved version of cMOGA called MOCell, can be found in [1, Chapter 9].

The present work extends cEAs with message passing capabilities. The main goal is to improve their effectiveness by performing a more elaborate sampling of the search space, as explained in the next section.

3 Message passing cellular evolutionary algorithms

MPcEAs extend cEAs by introducing message passing into the evolutionary cycle. As described in Section 2, cEAs perform selection and variation locally in a lattice where each cell is occupied by an individual. Algorithm 1 in Section 2 specifies the typical local operations that are carried out in the lattice at each generation. In the rest of this paper, a version of Algorithm 1 will be assumed that applies the following four operations to each individual:

1. A parent is selected from the neighbors of the original individual.
2. The selected parent is recombined with the original individual in order to generate a child.
3. The child is mutated and evaluated.
4. The mutated child replaces the original individual in the lattice if the fitness of the mutated child improves that of the original individual.

Under MPcEAs, every time these four operations are applied to an individual, they are also applied to each of its neighbors. The additional operations are applied to each neighbor by considering the neighborhood of the original individual, as detailed in Algorithm 2 for a canonical MPcEA. Thus, as shown in lines 9-16 of Algorithm 2, a new candidate is generated for each of the cells in the neighborhood of the original individual. In this way, MPcEAs extend the generation of candidates for replacement as follows:

- In Algorithm 1, only a candidate for cell i is obtained.
- In Algorithm 2, an additional candidate for each neighbor j of cell i is obtained (see line 15). In this case, when a candidate for cell j is generated, one of the two parents used in recombination is always the individual occupying cell j (see line 10). Throughout this paper, a recombination probability equal to unity will be used; however, if such a probability was less than unity and the recombination did not actually take place, the candidate for cell j would not be generated and the corresponding message would be empty.

As a result, at the end of a generation, every cell in the lattice is assigned a set of candidate individuals for replacement. Overall, each cell is provided with a candidate by each of its neighboring

cells plus its own candidate. Thus, the number of candidates to replace the individual located in a cell is constant across the lattice. For example, if the von Neumann neighborhood is considered, each cell is assigned five candidates: one candidate sent from the cell itself and four candidates sent from its North, East, South, and West neighboring cells respectively. As indicated in line 19 of Algorithm 2, the decision function `Replace?` is applied to each cell of the lattice in order to select which individual survives among the original one and its candidates. The selected individual will occupy the cell at the next generation of the evolutionary cycle.

Algorithm 2 Canonical MPcEA

Input:
An optimization problem
The usual parameters of a canonical EA
A lattice where each cell will be occupied by an individual at each generation
Output:
The population of individuals in the lattice at the final generation

```
 1: Initialize(population);
 2: Evaluate(population);
 3: while not stop-condition
 4:     for i ← 1 to population.size
 5:         population.individual[i].candidates ← ∅;
 6:     end-for
 7:     for i ← 1 to population.size
 8:         neighborhood ← population.individual[i] ∪ CalculateNeighbors(population.individual[i]);
 9:         for current-individual ← neighborhood
10:             parent1 ← current-individual;
11:             parent2 ← Select(neighborhood − {current-individual});
12:             child ← Recombine(parent1, parent2);
13:             Mutate(child);
14:             Evaluate(child);
15:             current-individual.candidates ← current-individual.candidates ∪ {child};
16:         end-for
17:     end-for
18:     for i ← 1 to population.size
19:         population.individual[i] ← Replace?(population.individual[i], population.individual[i].candidates);
20:     end-for
21: end-while
```

An important characteristic of MPcEAs is that every cell sends a message to itself and to each of its neighbors at each generation. The message from cell i to cell j represents a candidate individual to occupy cell j in the next generation. This candidate individual is generated from selection and variation operations applied to individuals located in the neighborhood of cell i, that is, cell i and its neighbors. Furthermore, one of the two parents of the candidate individual (previously to mutation) is always cell j. Figure 1 illustrates how the message from cell i to cell j is generated. Note that the mate for cell j (previously to recombination) is selected from the cells in the neighborhood of cell i excluding cell j. In Figure 1, such a mate is arbitrarily depicted in the cell denoted as k.

In MPcEAs, there are two ways of organizing message passing at each generation:

1. *Synchronously*: The cells of the lattice are visited in an arbitrary order, which can be fixed from generation to generation. Each visited cell sends its outgoing messages and, once the last message has been sent in the lattice, every cell updates its individual. Algorithm 2 employs this updating scheme, which will be assumed in the rest of the paper.

2. *Asynchronously*: The cells of the lattice are visited in a random order, which varies from

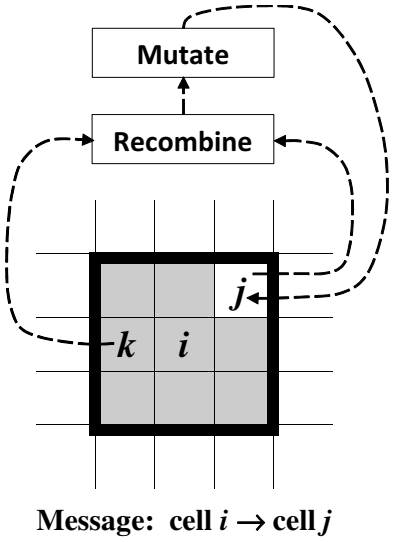

Figure 1: Message sent from cell i to cell j, which correspond to *population.individual*[i] and *current-individual* in Algorithm 2 respectively. Cell j is recombined with cell k, which is selected from the Moore neighborhood of cell i excluding cell j. The potential mates for cell j are depicted in gray.

generation to generation. Each visited cell sends its outgoing messages, which are immediately processed by its neighbors as they arrive: The new individual for each neighboring cell is the fittest between the new candidate and the present individual in the cell.

A cEA can be considered as an MPcEA where each cell sends a unique message (to itself). In terms of computational complexity, under MPcEAs each cell executes the same operations as under cEAs and repeats those operations for each of its neighbors. The expression of the time complexity for cEAs depends on the particular selection and variation methods used during the evolutionary cycle [30], although in the general case it has been shown to vary linearly with the number of cells [5, Section 3.1]. If the Moore neighborhood is considered, an MPcEA executes nine times the operations carried out by a cEA. These operations include both message passing and candidate selection for replacement. Therefore, assuming that the genotype length is bounded, cEAs and MPcEAs have the same order of complexity in terms of the number of cells in the lattice. Section 4 comparatively evaluates both the efficiency and the effectiveness of cEAs and MPcEAs through a series of experimental tests.

4 Experimental evaluation

This section compares cEAs and MPcEAs in terms of effectiveness (optimization quality) and efficiency (running time). A number of tests are designed that deal with the optimization of real functions defined over continuous domains. Given an n-dimensional function, f: $\mathbb{R}^n \longrightarrow \mathbb{R}$, the problem of global optimization consists in determining $\mathbf{x}^* \in \mathbb{R}^n$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \neq \mathbf{x}^*$ with $\mathbf{x} \in \mathbb{R}^n$. This definition corresponds to a minimization problem, which will be assumed throughout the present section unless otherwise specified.

Table 1 contains the thirteen functions used in the experimental evaluation. These functions, which are widely used in the literature on EAs (see for example [23]), provide a diverse range of multimodal problems. All of them have a global minimum whose fitness, $f(\mathbf{x}^*)$, is equal to

Function	$f(x_1, \ldots, x_n)$	Domain
Ackley	$20 + \exp(1) - 20\exp\left(-0.2\left(\frac{1}{n}\sum_{i=1}^{n} x_i^2\right)^{1/2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$	$[-32.768, 32.768]^n$
Alpine1	$\sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	$[0, 10]^n$
Griewank	$1 + \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	$[-600, 600]^n$
HappyCat	$\left\lvert \sum_{i=1}^{n} {x_i}^2 - n \right\rvert^{\frac{1}{4}} + \frac{1}{n}\left(0.5\sum_{i=1}^{n} {x_i}^2 + \sum_{i=1}^{n} x_i\right) + 0.5$	$[-2, 2]^n$
High Conditioned Elliptic	$\sum_{i=1}^{n} (10^6)^{\frac{i-1}{n-1}} {x_i}^2$	$[-100, 100]^n$
Rastrigin	$10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$	$[-5.12, 5.12]^n$
Rosenbrock	$\sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$	$[-5, 10]^n$
Salomon	$1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$	$[-100, 100]^n$
Schwefel1.2	$\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	$[-100, 100]^n$
Schwefel2.26	$418.9829n - \sum_{i=1}^{n}\left(x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right)$	$[-500, 500]^n$
Sphere	$\sum_{i=1}^{n} x_i^2$	$[-5.12, 5.12]^n$
Weierstrass	$\sum_{i=1}^{n}\left\{\sum_{k=0}^{20}\left[0.5^k \cos(2\pi 3^k(x_i + 0.5))\right]\right\} - n\sum_{k=0}^{20}\left[0.5^k\cos(\pi 3^k)\right]$	$[-0.5, 0.5]^n$
Zakharov	$\sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^4$	$[-5, 10]^n$

Table 1: Definition of the functions used in the experimental evaluation. All of them have a minimum value $f(\mathbf{x}^*) = 0$.

zero. The dimension $n = 50$ is chosen for the tests, since it gives rise to thirteen optimization problems of different complexity (from easy to hard) over the functions in Table 1. A real interval is considered for each dimension of the function domain (see Table 1).

4.1 Implementation

Every point (or individual) in the search space is naturally encoded as a string of n real-coded variables. Thus, the variation operators must be designed for chromosomes that use real representation. A cEA and an MPcEA are implemented with identical evolutionary operators in the following way:

- The *population* is distributed in a 10x10 toroid where each of the one hundred individuals is connected to other eight individuals by means of the Moore neighborhood. Initially, every cell is assigned an individual whose allele l, with $l \in \{1, \ldots, n\}$, is randomly generated from a uniform distribution in the domain of variable x_l.

- In *parent selection*, the first parent is always deterministically obtained as explained in Section 2 for the cEA and in Section 3 for the MPcEA. The second parent is chosen through tournament selection (without replacement) with tournament size equal to five.

- *Parent recombination* is performed, with probability equal to one, by employing the blend recombination operator (BLX-α) [16] with $\alpha = 0.5$. Under this operator, the child $\mathbf{z} \equiv (z_1, \ldots, z_n)$ is generated from parents $\mathbf{x} \equiv (x_1, \ldots, x_n)$ and $\mathbf{y} \equiv (y_1, \ldots, y_n)$ such that, for all $l \in \{1, \ldots, n\}$, z_l is randomly generated from a uniform distribution in the real interval $[\min(x_l, y_l) - I \cdot \alpha, \max(x_l, y_l) + I \cdot \alpha]$, where $I = \max(x_l, y_l) - \min(x_l, y_l)$. BLX-0.5 was successfully applied in [14] in the context of a simple cellular genetic algorithm for continuous optimization.

- *Child mutation* is developed, with probability for each gene equal to $1/n$, by means of the non-uniform mutation operator [27, 20] with $b = 1$. This operator uses a non-uniform distribution whose step size is decreased as generations pass, which favors exploration in the initial stage of the search process and promotes exploitation in the final stage. The degree of decrease for the step size throughout the generations can be manually adjusted by the user through parameter b. Like the recombination operator BLX-α, this mutation operator was successfully applied in [14].

- The *replacement policy* for each individual in the population at generation t substitutes the individual only when a better candidate to occupy its cell is found at generation $t + 1$.

- The cells in the population are updated synchronously (in a sequential way) at each generation.

- The algorithms are executed for a fixed number of generations which is specified in advance by the user.

The evaluated algorithms are implemented within NetLogo [34], an agent-based programming environment well suited for modeling and inspecting complex systems developing over time. The tests are carried out on an Intel Xeon processor (2.67 GHz) with 6 Gb of memory.

4.2 Results

When applied to the functions in Table 1, the effectiveness of a cEA and that of an MPcEA are compared by measuring the best fitness at the final generation. Twenty-five independent runs are repeated for each particular test. Different seeds for the random number generator are employed for each run of the algorithms. Tables 2 and 3 show the mean best fitness and its standard deviation over the twenty-five independent runs. Two different stopping conditions are considered: one thousand generations (see Table 2) and ten thousand generations (see Table 3).

The comparison of the cEA and the MPcEA in terms of efficiency is developed by measuring running times in the same tests already described when comparing effectiveness. Table 4, for the case of one thousand generations, and Table 5, for ten thousand generations, contain the mean running time and its standard deviation when the cEA and the MPcEA are applied to the functions in Table 1.

It is interesting to compare MPcEAs and cEAs with a panmictic EA (pEA). Basically, in this work a pEA differs from MPcEAs and cEAs in the way the second parent is selected for each individual in the population previously to recombination. In the case of pEAs, the second parent is obtained through tournament selection (without replacement and with tournament size equal to five) from the whole population (excluding the first parent). Tables 2-5 have been extended to include the experimental results for pEAs. Additionally, the progress curves for the three algorithms are included in Figures 2 and 3. These curves plot, generation by generation, the mean fitness for the best individual in the population over the twenty-five independent runs.

Function	cEA	MPcEA	pEA
Ackley	1.221E–5 (3.781E–6)	**5.977E–15 (1.176E–15)**	4.667E–7 (1.6126E–7)
Alpine1	3.798E–7 (5.373E–8)	**9.842E–30 (2.094E–30)**	3.533E–8 (6.047E–9)
Griewank	1.069E–3 (3.256E–3)	**0 (0)**	1.182E–3 (4.069E–3)
HappyCat	4.242E–1 (6.054E–2)	**2.029E–1 (3.046E–2)**	3.409E–1 (4.204E–2)
High Conditioned Elliptic	2.01E–6 (1.046E–6)	**8.211E–63 (8.259E–63)**	3.058E–9 (1.585–9)
Rastrigin	46.905 (6.887)	**2.693 (2.588)**	39.176 (14.294)
Rosenbrock	106.609 (40.271)	**56.171 (23.695)**	77.493 (35.725)
Salomon	4.349E–1 (5.408E–2)	**2.199E–1 (4E–2)**	3.12E–1 (3.246E–2)
Schwefel1.2	7186.355 (1881.995)	**51.93 (36.233)**	4864.633 (1201.887)
Schwefel2.26	1585.176 (201.377)	**137.601 (166.63)**	2458.142 (761.406)
Sphere	1.613E–11 (9.035E–12)	**5.596E–68 (6.524E–68)**	2.434E–14 (1.023E–14)
Weierstrass	2.095E–3 (3.521E–3)	**0 (0)**	6.756E–5 (2.198E–5)
Zakharov	53.64 (7.661)	**1.807E–4 (9.166E–5)**	50.324 (10.177)

Table 2: Mean **best fitness** (and its standard deviation in parentheses) for the thirteen functions in Table 1 when the stopping condition is **one thousand generations**. The best result in each row is highlighted in bold.

Function	cEA	MPcEA	pEA
Ackley	6.217E–15 (3.155E–30)	**5.791E–15 (1.154E–15)**	6.075E–15 (6.962E–16)
Alpine1	9.208E–76 (3.921E–76)	**2.016E–302 (1.194E–302)**	1.152E–85 (6.389E–86)
Griewank	5.91E–4 (2.895E–3)	**0 (0)**	2.958E–4 (1.449E–3)
HappyCat	2.178E–1 (2.84E–2)	**1.2E–1 (1.721E–2)**	1.785E–1 (2.365E–2)
High Conditioned Elliptic	4.151E–116 (9.142E–116)	**0 (0)**	1.076E–144 (1.603E-144)
Rastrigin	3.186E–1 (5.524E–1)	**2.979E–13 (6.012E–13)**	4.625 (5.225)
Rosenbrock	63.501 (27.819)	**13.296 (9.211)**	55.397 (24.122)
Salomon	2.919E–1 (3.919E–2)	**2.119E–1 (3.25E–2)**	2.199E–1 (4E–2)
Schwefel1.2	26.204 (26.498)	**2.816E–17 (5.065E–17)**	8.637E–1 (4.09E–1)
Schwefel2.26	3.493E–3 (5.751E–3)	**6.364E–4 (3.254E–12)**	279.8 (336.971)
Sphere	1.604E–121 (3.162E–121)	**0 (0)**	4.008E–149 (1.375E–148)
Weierstrass	4.011E–4 (8.462E–4)	**0 (0)**	**0 (0)**
Zakharov	3.709E–5 (2.031E–5)	**1.607E–64 (1.827E–64)**	7.161E–6 (3.518E–6)

Table 3: Mean **best fitness** (and its standard deviation in parentheses) for the thirteen functions in Table 1 when the stopping condition is **ten thousand generations**. The best result in each row is highlighted in bold.

Function	cEA	MPcEA	pEA
Ackley	12.776 (2.255E–2)	102.742 (9.217E–2)	**12.351 (1.909E–2)**
Alpine1	13.045 (1.592E–2)	104.656 (8.653E–2)	**12.539 (1.982E–2)**
Griewank	12.965 (2.219E–2)	103.89 (9.325E–2)	**12.461 (2.015E–2)**
HappyCat	**10.794 (1.918E–2)**	90.39 (1.707E–1)	11.459 (2.684E–2)
High Conditioned Elliptic	**11.916 (3.615E–2)**	97.658 (1.581E–1)	12.506 (3.076E–2)
Rastrigin	12.17 (2.147E–2)	101.529 (1.504E–1)	**11.708 (2.929E–2)**
Rosenbrock	12.963 (4.725E–2)	107.259 (2.289E–1)	**12.635 (2.138E–2)**
Salomon	11.184 (2.107E–2)	92.405 (9.481E–1)	**10.765 (1.583E–2)**
Schwefel1.2	33.868 (2.65E–2)	295.668 (1.451E–1)	**33.265 (2.632E–2)**
Schwefel2.26	12.161 (1.929E–2)	102.232 (1.175E–1)	**11.886 (5.001E–2)**
Sphere	11.468 (1.672E–2)	94.64 (1.325E–1)	**11.114 (1.997E–2)**
Weierstrass	107.006 (1.087E–1)	978.498 (5.66E–1)	**106.991 (2.241E–1)**
Zakharov	11.8 (1.594E–2)	99.842 (1.689E–1)	**11.615 (2.552E–2)**

Table 4: Mean **running time** in seconds (and its standard deviation in parentheses) for the thirteen functions in Table 1 when the stopping condition is **one thousand generations**. The best result in each row is highlighted in bold.

Function	cEA	MPcEA	pEA
Ackley	120.967 (1.701E–1)	1010.377 (4.096E–1)	**117.222 (1.458E–1)**
Alpine1	129.478 (9.83E–2)	1062.073 (4.943E–1)	**124.398 (9.9295E–2)**
Griewank	122.824 (1.985E–1)	1029.409 (6.673E–1)	**117.886 (1.1323E–1)**
HappyCat	**106.937 (1.384E–1)**	901.171 (8.147E–1)	112.56 (2.044E–1)
High Conditioned Elliptic	**118.371 (1.235E–1)**	1002.246 (1.668)	123.58 (1.452E–1)
Rastrigin	120.053 (1.696E–1)	1008.852 (1.801)	**116.755 (2.681E–1)**
Rosenbrock	129.7 (3.877E–1)	1072.378 (1.687)	**124.452 (1.794E–1)**
Salomon	109.844 (1.424E–1)	924.811 (3.244E–1)	**105.386 (1.41E–1)**
Schwefel1.2	332.893 (2.026E–1)	2954.541 (4.918E–1)	**322.786 (3.989E–1)**
Schwefel2.26	120.741 (2.69E–1)	1010.413 (1.39)	**116.456 (3.346E–1)**
Sphere	114.638 (1.216E–1)	968.51 (7.133E–1)	**110.645 (1.841E–1)**
Weierstrass	**1102.688 (1.949)**	9949.105 (42.284)	1110.117 (9.289E–1)
Zakharov	118.224 (1.344E–1)	995.046 (1.647)	**114.606 (1.629E–1)**

Table 5: Mean **running time** in seconds (and its standard deviation in parentheses) for the thirteen functions in Table 1 when the stopping condition is **ten thousand generations**. The best result in each row is highlighted in bold.

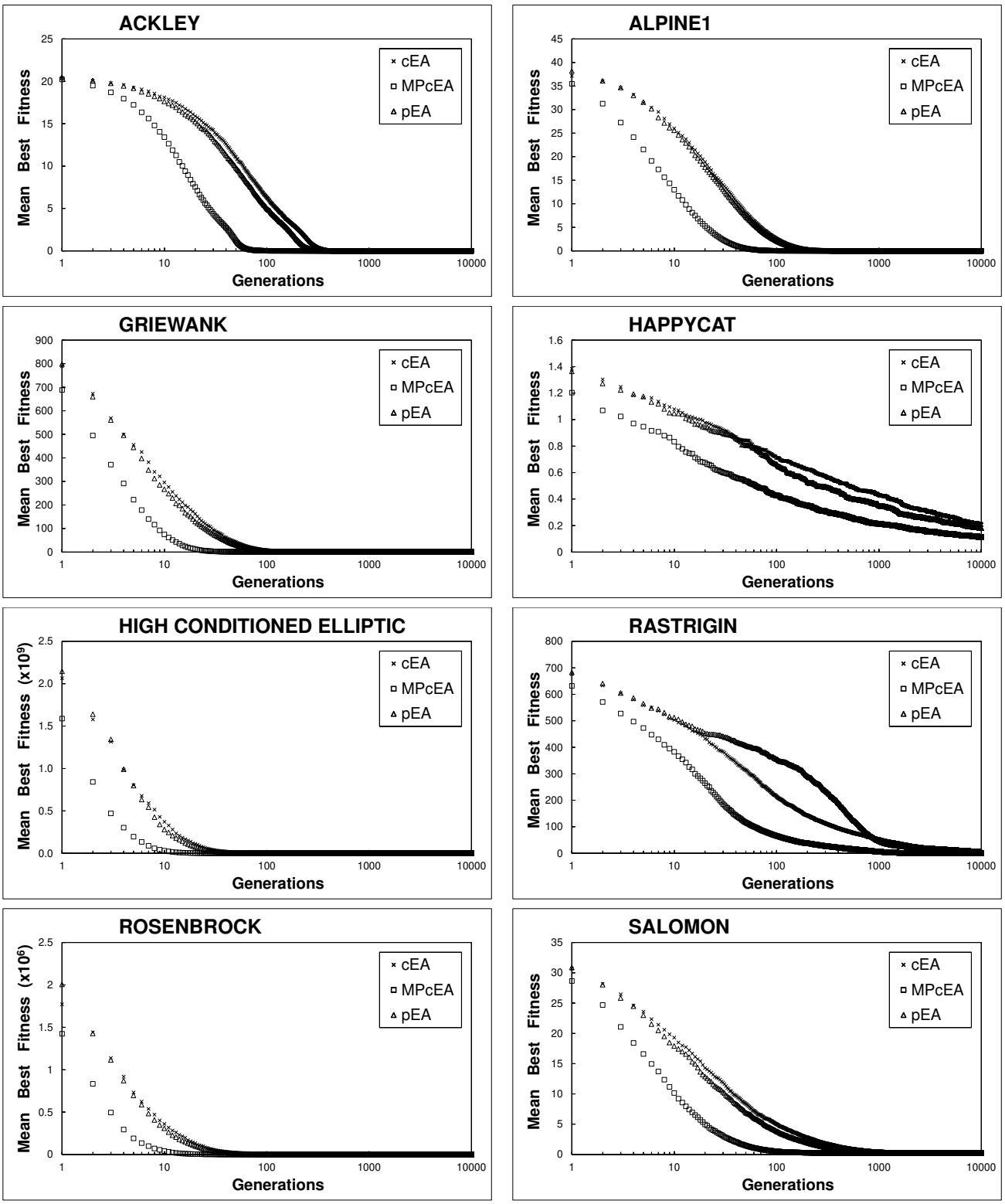

Figure 2: Progress curves (1 of 2) when cEAs, MPcEAs, and pEAs are applied to the functions in Table 1. A logarithmic scale is used. Generation 1 represents the first generation after population initialization.

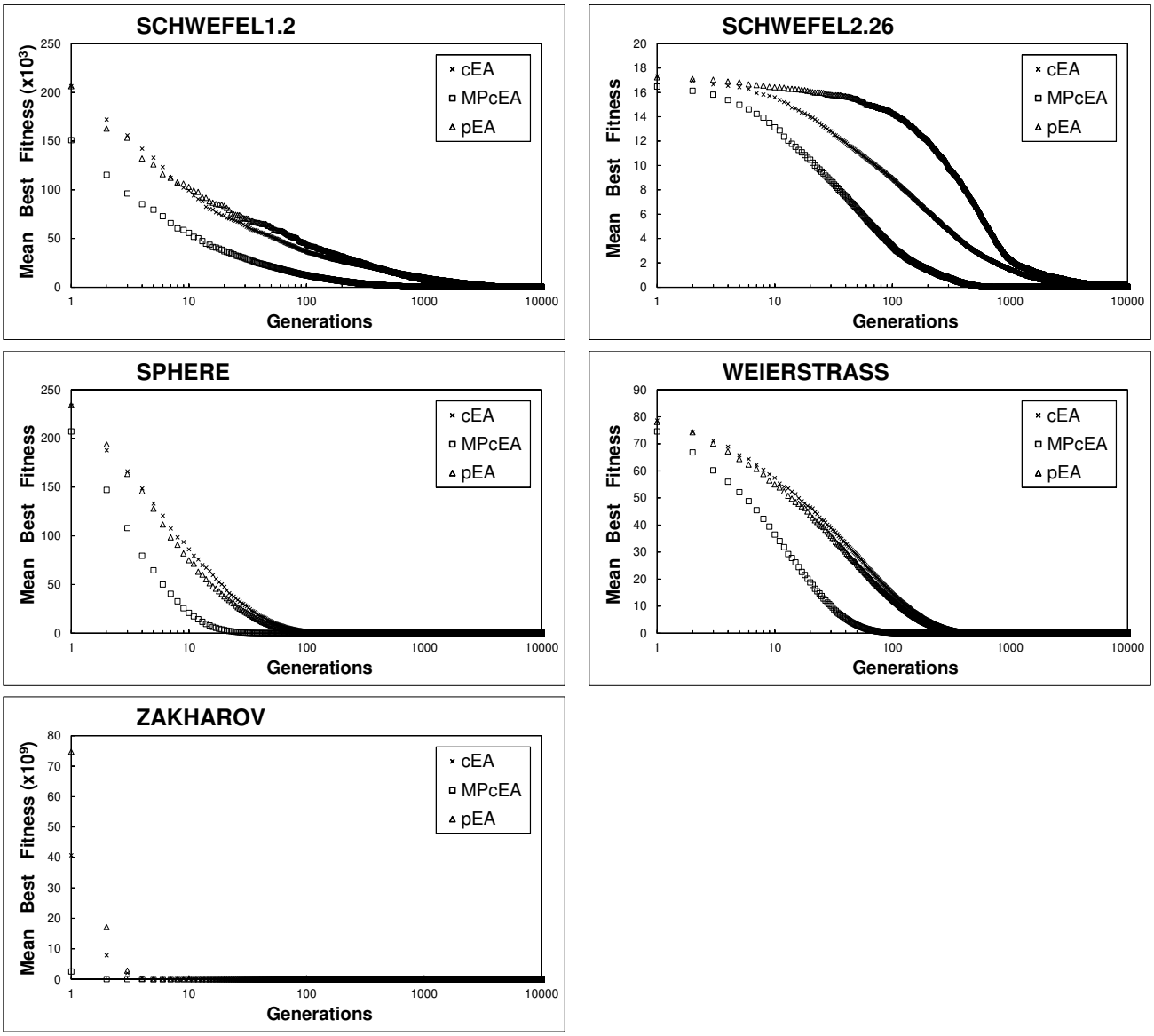

Figure 3: Progress curves (2 of 2) when cEAs, MPcEAs, and pEAs are applied to the functions in Table 1. A logarithmic scale is used. Generation 1 represents the first generation after population initialization.

Table	p
2	0.00148
3	0.03318
4	0.00148
5	0.00148

Table 6: The four p-values obtained by applying Wilcoxon tests (with significance level $\alpha = 0.05$) on columns "cEA" and "MPcEA" in Tables 2-5.

4.3 Analysis

Before discussing the experimental results obtained for cEAs and MPcEAs in Tables 2-5, their statistical significance needs to be established. Thus, for each of the four tables, it needs to be determined whether or not the data (grouped by columns "cEA" and "MPcEA") come from the same distribution and, therefore, the differences are due to random effects (null hypothesis). Wilcoxon tests are applied to each of the four tables, and the probability p of the corresponding result assuming the null hypothesis is shown in Table 6. When probability p is less than a small threshold (typically, $\alpha = 0.05$ is used in the literature), it is justified to reject the null hypothesis. The values of p in Table 6 are small enough to discard the null hypothesis.

The results in Tables 2 and 3 show that the MPcEA clearly outperforms the cEA in terms of effectiveness. Furthermore, for six of the thirteen functions (Ackley, Griewank, HappyCat, Rosenbrock, Salomon, and Weierstrass), the results reported for the MPcEA after one thousand generations (see Table 2) are even better than those reported for the cEA after ten thousand generations (see Table 3). Note that a diverse range of behaviors regarding convergence is obtained in Tables 2 and 3: While some functions have reached convergence to value zero after one thousand generations of the MPcEA, other functions still have room for improvement after ten thousand generations of the MPcEA. Comparing cEAs and MPcEAs for problems of different complexity and for different stages of the search process is precisely one of the main goals of our experimental evaluation. Finally, regarding the pEA, it exhibits a performance which is closer to that of the cEA. Which method between the cEA and the pEA is better depends on the particular function being optimized.

The progress curves in Figures 2 and 3 clearly illustrate how MPcEAs achieve a convenient tradeoff between exploration and exploitation for the considered functions. Thus, with respect to cEAs and pEAs, MPcEAs converge to solutions of higher quality in fewer generations. The cellular characteristic of MPcEAs allows diversity to be properly maintained in the population, while the message passing phase contributes to an effective search of the solution space.

Tables 4 and 5 show that the cEA is more efficient than the MPcEA. In these two tables, the magnitude of runtime improvement for the cEA with respect to the MPcEA is in accordance with the number of (fitness) function evaluations carried out by each of these EAs. In general, given a population whose size is denoted by s, the following number of function evaluations have been performed after generation t by the two algorithms:

$$\begin{aligned} \mathcal{F}_{\text{cEA}}(t) &= s + s * t \\ \mathcal{F}_{\text{MPcEA}}(t) &= s + 9 * s * t \end{aligned} \text{ ,}$$

where it is assumed that the population is initialized at generation $t = 0$. Thus, for a fixed number of generations, the number of function evaluations is $\mathcal{O}(s)$ under both EAs.

5 Conclusion and future research

With respect to panmictic EAs, cEAs maintain a higher diversity in the population and accomplish a better balance between exploration and exploitation during the search process. As a consequence, cEAs perform better than panmictic EAs in many optimization problems.

This work extends cEAs by designing more elaborate local operations for each individual in the population. These new local operations are carried out in MPcEAs through message passing techniques.

As shown experimentally in this work, MPcEAs are much more effective than cEAs at the expense of reducing efficiency. As a result, MPcEAs produce a convenient tradeoff between effectiveness and efficiency. Furthermore, any advanced evolutionary technique designed for cEAs (see Section 2.1) can be easily applied to MPcEAs as well.

The present work opens up the following research directions:

- Additional experimental tests in new domains can be conducted in order to thoroughly evaluate MPcEAs. In this way, other different evolutionary operators from those employed in this paper could be tested and the influence of other neighborhoods (different from the Moore neighborhood) could be investigated.

- An interesting topic for future research is to apply the advanced evolutionary techniques explained in Section 2.1 to MPcEAs. For example, the use of parallel computing in the context of MPcEAs could be investigated in order to improve the computation time for larger populations.

References

[1] E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms*. Springer, 2008.

[2] E. Alba, B. Dorronsoro, and H. Alfonso. Cellular memetic algorithms. *Journal of Computer Science & Technology*, 5(4):257–263, 2005.

[3] E. Alba, B. Dorronsoro, F. Luna, A. J. Nebro, P. Bouvry, and L. Hogie. A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan MANETs. *Computer Communications*, 30(4):685–697, 2007.

[4] E. Alba and J. M. Troya. Cellular evolutionary algorithms: Evaluating the influence of ratio. In *Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN VI)*, pages 29–38, 2000.

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

[6] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing, 2000.

[7] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Evolutionary Computation 2: Advanced Algorithms and Operators*. Institute of Physics Publishing, 2000.

[8] R. Chiong, T. Weise, and Z. Michalewicz. *Variants of Evolutionary Algorithms for Real-World Applications*. Springer, Berlin, 2012.

[9] C. A. Coello, D. A. van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[10] R. J. Collins and D. R. Jefferson. Selection in massively parallel genetic algorithms. In *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA-91)*, pages 249–256, 1991.

[11] K. A. de Jong. *Evolutionary Computation: A Unified Approach*. The MIT Press, 2006.

[12] T. Dittrich and W. Elmenreich. Comparison of a spatially-structured cellular evolutionary algorithm to an evolutionary algorithm with panmictic population. In *Proceedings of the 12th International Workshop on Intelligent Solutions in Embedded Systems (WISES 2015)*, pages 145–149, 2015.

[13] B. Dorronsoro. *Diseño e Implementación de Algoritmos Genéticos Celulares para Problemas Complejos*. PhD thesis, Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Málaga, 2006.

[14] B. Dorronsoro and E. Alba. A simple cellular genetic algorithm for continuous optimization. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation (IEEE CEC 2006)*, pages 16–21, 2006.

[15] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[16] L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithms and interval schemata. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 187–202. Morgan Kaufmann, San Mateo, CA, 1993.

[17] G. Folino, C. Pizzuti, and G. Spezzano. Combining cellular genetic algorithms and local search for solving satisfiability problems. In *Proceedings of the 10th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 1998)*, pages 192–198, 1998.

[18] V. S. Gordon, R. Pirie, A. Wachter, and S. Sharp. Terrain-based genetic algorithm (TBGA): Modeling parameter space as terrain. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999)*, pages 229–235, 1999.

[19] M. Gorges-Schleuter. ASPARAGOS: An asynchronous parallel genetic optimization strategy. In *Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA-89)*, pages 422–427, 1989.

[20] F. Herrera, M. Lozano, and J. L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, 12(4):265–319, 1998.

[21] N. Q. Huy, O. Y. Soon, L. M. Hiot, and N. Krasnogor. Adaptive cellular memetic algorithms. *Evolutionary Computation*, 17(2):231–256, 2009.

[22] S. Janson, E. Alba, B. Dorronsoro, and M. Middendorf. Hierarchical cellular genetic algorithm. In *Proceedings of the 6th European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP 2006)*, pages 111–122, 2006.

[23] J. J. Liang, B. Y. Qu, P. N. Suganthan, and Q. Chen. Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization. Technical Report 201411A, Computational Intelligence Laboratory, Zhengzhou University, China and Technical Report, Nanyang Technological University, Singapore, 2014.

[24] G. Luque, E. Alba, and B. Dorronsoro. Analyzing parallel cellular genetic algorithms. In E. Alba, C. Blum, P. Asasi, L. Coromoto, and J. A. Gómez, editors, *Optimization Techniques for Solving Complex Problems*, pages 49–62. Wiley, 2009.

[25] G. Luque, E. Alba, and B. Dorronsoro. An asynchronous parallel implementation of a cellular genetic algorithm for combinatorial optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2009)*, pages 1395–1402, 2009.

[26] B. Manderick and P. Spiessens. Fine-grained parallel genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA-89)*, pages 428–433, 1989.

[27] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York, 1992.

[28] H. Mühlenbein. Parallel genetic algorithms, population genetics, and combinatorial optimization. In *Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA-89)*, pages 416–421, 1989.

[29] C. C. Pettey. Diffusion (cellular) models. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Evolutionary Computation 2: Advanced Algorithms and Operators*, pages 125–133. Institute of Physics Publishing, 2000.

[30] P. Spiessens and B. Manderick. A massively parallel genetic algorithm: Implementation and first analysis. In *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA-91)*, pages 279–286, 1991.

[31] M. Tomassini. *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time*. Springer, 2005.

[32] M. Tomassini. Cellular evolutionary algorithms. In A. G. Hoekstra, J. Kroc, and P. M. A. Sloot, editors, *Simulating Complex Systems by Cellular Automata*, pages 167–191. Springer, 2010.

[33] D. Whitley. Cellular genetic algorihtms. In *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA-93)*, page 658, 1993.

[34] U. Wilensky. NetLogo. `http://ccl.northwestern.edu/netlogo/`. Center for Connected Learning and Computer Science, Northwestern University, Evanston, IL, 1999.