# A Cellular Genetic Algorithm for Multiobjective Optimization

*A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba*

Departamento de Lenguajes y Ciencias de la Computación
E.T.S. Ingeniería Informática
Campus de Teatinos, 29071 Málaga (Spain)
`antonio,durillo,flv,bernabe,eat@lcc.uma.es`

**Abstract.** *This paper introduces a new cellular genetic algorithm for solving multiobjective continuous optimization problems. Our approach is characterized by using an external archive to store non-dominated solutions and a feedback mechanism in which solutions from this archive randomly replaces existing individuals in the population after each iteration. The result is a simple and elitist algorithm called MOCell. Our proposal has been evaluated with both constrained and unconstrained problems and compared against NSGA-II and SPEA2, two state-of-the-art evolutionary multiobjective optimizers. For the used benchmark, preliminary experiments indicate that MOCell obtains competitive results in terms of convergence, and it clearly outperforms the other two compared algorithms concerning the diversity of solutions along the Pareto front.*

## 1 Introduction

Most optimization problems in the real world involve the minimization and/or maximization of more than one function. Generally speaking, multiobjective optimization does not restrict to find a unique single solution of a given multiobjective optimization problem (MOP), but a set of solutions called *non-dominated solutions*. Each solution in this set is said to be a *Pareto optimum*, and when they are plotted in the objective space they are collectively known as the *Pareto front*. Obtaining the Pareto front of a given MOP is the main goal of multiobjective optimization. In general, the search spaces in MOPs use to be very large, and evaluating the functions can require a significant amount of time. These features make difficult to apply deterministic techniques and, therefore, stochastic techniques have been widely proposed within this domain. Among them, evolutionary algorithms (EAs) have been investigated by many researchers, and some of the most well-known algorithms for solving MOPs belong to this class (e.g. NSGA-II [10], PAES [12], and SPEA2 [17]).

EAs are especially well-suited for tackling MOPs because of their ability for finding multiple trade-off solutions in one single run. Well-accepted subclasses of EAs are Genetic Algorithms (GA), Genetic Programming (GP), Evolutionary Programming (EP), and Evolution Strategies (ES). These algorithms work over a set (*population*) of potential solutions (*individuals*) which undergoes stochastic operators in order to search for better solutions. Most EAs use a single population (panmixia) of individuals and apply the operators to them as a whole (see Fig. 1a). Conversely, there exist the so-called structured EAs, in which the population is decentralized somehow. Among the many types of structured EAs, *distributed* and *cellular* models are two popular optimization variants [4,6] (see Fig. 1b and Fig. 1c). In many cases, these decentralized algorithms provide a better sampling of the search space,

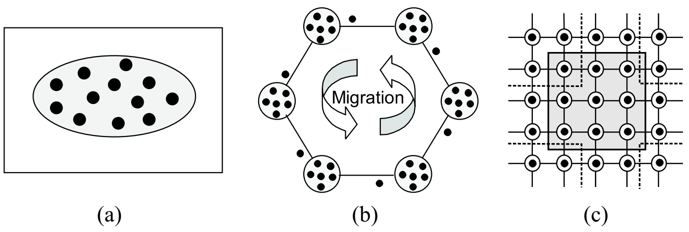

**Fig. 1.** Panmictic (a), distributed (b), and cellular (c) GAs

resulting in an improved numerical behavior with respect to an equivalent algorithm in panmixia.

In this work, we focus on the cellular model of GAs (cGAs). In cGAs, the concept of (small) *neighborhood* is intensively used; this means that an individual may only interact with its nearby neighbors in the breeding loop [14]. The overlapped small neighborhoods of cGAs 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 cGAs were initially designed for working in massively parallel machines, although the model itself has been adopted also for mono-processor machines, with no relation to parallelism at all. Besides, the neighborhood is defined among tentative solutions in the algorithm, with no relation to the geographical neighborhood definition in the problem space.

cGAs have proven to be very effective for solving a diverse set of single objective optimization problems from both classical and real world settings [1,2], but little attention has been paid to its use in the multiobjective optimization field. In [13], a multiobjective evolution strategy following a predator-prey model is presented. This is a model similar to a cGA, because solutions (preys) are placed on the vertices of an undirected connected graph, thus defining neighborhoods, where they are 'caught' by predators. Murata and Gen presented in [15] an algorithm in which, for an $n$-objective MOP, the population is structured in an $n$-dimensional weight space, and the location of individuals (called cells) depends on their weight vector. Thus, the information given by the weight vector of individuals is used for guiding the search. A metapopulation evolutionary algorithm (called MEA) is presented in [11]. This algorithm is a cellular model with the peculiarity that disasters can occasionally happen in the population, thus dying all the individuals located in the disaster area (extinction). Additionally, these empty areas can also be occupied by individuals (colonization). Thus, this model allows a flexible population size, combining the ideas of cellular and spatially distributed populations. Finally, Alba et al. proposed in [3] cMOGA, the unique cellular multiobjective algorithm based on the canonical cGA model before this work, to the best of our known. In that work, cMOGA was used for optimizing a broadcasting strategy specifically designed for mobile ad hoc networks.

Our proposal is called MOCell, and it is, like in the case of [3], an adaptation of a canonical cGA to the multiobjective field. MOCell uses an external archive to store the non-dominated solutions found during the execution of the algorithm, like many other multiobjective evolutionary algorithms do (e.g., PAES, SPEA2, or cMOGA). However, the main feature characterizing MOCell with respect to these algorithms is that a number of solutions are moved back into the population from the archive after each iteration, replacing randomly selected existing individuals. The contributions of our work can be summarized as follows:

- We propose a cGA for solving continuous MOPs. The algorithm uses an external archive and a feedback of solutions from the archive to the population.
- The algorithm is evaluated using a benchmark of constrained and unconstrained MOPs.
- MOCell is compared against NSGA-II and SPEA2, two state-of-the-art GAs for solving MOPs.

The rest of the paper is organized as follows. In Section 2, we present several basic concepts on multiobjective optimization. In Section 3, we describe MOCell, our proposal for facing MOPs. Our results are presented and discussed in Section 4. Finally, in Section 5 we give our main conclusions and suggest some future research lines.

## 2 Multiobjective Optimization Fundamentals

In this section, we include some background on multiobjective optimization. In concrete, we define the concepts of MOP, Pareto optimality, Pareto dominance, Pareto optimal set, and Pareto front. In these definitions we are assuming, without loss of generality, the minimization of all the objectives. A general multiobjective optimization problem (MOP) can be formally defined as follows:

**Definition 1 (MOP).** Find a vector $\boldsymbol{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]$ which satisfies the $m$ inequality constraints $g_i(\boldsymbol{x}) \geq 0, i = 1, 2, \ldots, m$, the $p$ equality constraints $h_i(\boldsymbol{x}) = 0, i = 1, 2, \ldots, p$, and minimizes the vector function $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})]^T$, where $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables.

The set of all values satisfying the constraints defines the *feasible region* $\Omega$ and any point $\boldsymbol{x} \in \Omega$ is a *feasible solution.* As mentioned before, we seek for the *Pareto optima.* Its formal definition is provided next:

**Definition 2 (Pareto Optimality).** A point $\boldsymbol{x}^* \in \Omega$ is Pareto Optimal if for every $\boldsymbol{x} \in \Omega$ and $I = \{1, 2, \ldots, k\}$ either $\forall_{i \in I} (f_i(\boldsymbol{x}) = f_i(\boldsymbol{x}^*))$ or there is at least one $i \in I$ such that $f_i(\boldsymbol{x}) > f_i(\boldsymbol{x}^*)$.

This definition states that $\boldsymbol{x}^*$ is Pareto optimal if no feasible vector $\boldsymbol{x}$ exists which would improve some criterion without causing a simultaneous worsening in at least one other criterion. Other important definitions associated with Pareto optimality are the following:

**Definition 3 (Pareto Dominance).** A vector $\boldsymbol{u} = (u_1, \ldots, u_k)$ is said to dominate $\boldsymbol{v} = (v_1, \ldots, v_k)$ (denoted by $\boldsymbol{u} \preccurlyeq \boldsymbol{v}$) if and only if $\boldsymbol{u}$ is partially less than $\boldsymbol{v}$, i.e., $\forall i \in \{1, \ldots, k\}, \ u_i \leq v_i \ \wedge \ \exists i \in \{1, \ldots, k\} : \ u_i < v_i$.

**Definition 4 (Pareto Optimal Set).** For a given MOP $\boldsymbol{f}(\boldsymbol{x})$, the Pareto optimal set is defined as $\mathcal{P}^* = \{\boldsymbol{x} \in \Omega | \neg \exists \boldsymbol{x'} \in \Omega, \boldsymbol{f}(\boldsymbol{x'}) \preccurlyeq \boldsymbol{f}(\boldsymbol{x})\}$.

**Definition 5 (Pareto Front).** For a given MOP $\boldsymbol{f}(\boldsymbol{x})$ and its Pareto optimal set $\mathcal{P}^*$, the Pareto front is defined as $\mathcal{PF}^* = \{\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{x} \in \mathcal{P}^*\}$.

Obtaining the Pareto front of a MOP is the main goal of multiobjective optimization. However, given that a Pareto front can contain a large number of points, a good solution must contain a limited number of them, which should be as close as possible to the exact Pareto front, as well as they should be uniformly spread. Otherwise, they would not be very useful to the decision maker.

**Algorithm 1** Pseudocode for a Canonical cGA

```
1: proc Steps_Up(cga)        //Algorithm parameters in 'cga'
2: while not Termination_Condition() do
3:   for individual ← 1 to cga.popSize do
4:      n_list←Get_Neighborhood(cga,position(individual));
5:      parents←Selection(n_list);
6:      offspring←Recombination(cga.Pc,parents);
7:      offspring←Mutation(cga.Pm,offspring);
8:     Evaluate_Fitness(offspring);
9:     Insert(position(individual),offspring,cga,aux_pop);
10:   end for
11:   cga.pop←aux_pop;
12: end while
13: end_proc Steps_Up;
```

## 3 The Algorithm

In this section we detail first a description of a canonical cGA; then, we describe the algorithm MOCell.

### 3.1 Cellular Genetic Algorithms

A canonical cGA follows the pseudo-code included in Algorithm 1. In this basic cGA, the population is usually structured in a regular grid of $d$ dimensions ($d = 1, 2, 3$), and a neighborhood is defined on it. The algorithm iteratively considers as current each individual in the grid (line 3). An individual may only interact with individuals belonging to its neighborhood (line 4), so its parents are chosen among its neighbors (line 5) with a given criterion. Crossover and mutation operators are applied to the individuals in lines 6 and 7, with probabilities $P_c$ and $P_m$, respectively. Afterwards, the algorithm computes the fitness value of the new offspring individual (or individuals) (line 8), and inserts it (or one of them) into the equivalent place of the current individual in the new (auxiliary) population (line 9) following a given replacement policy.

After applying this reproductive cycle to all the individuals in the population, the newly generated auxiliary population is assumed to be the new population for the next generation (line 11). This loop is repeated until a termination condition is met (line 2). The most usual termination conditions are to reach the optimal value, to perform a maximum number of fitness function evaluations, or a combination of both of them.

### 3.2 A Multiobjective cGA: MOCell

In this section we present MOCell, a multiobjective algorithm based on a cGA model. Its pseudo-code is given in Algorithm 2. We can observe that Algorithms 1 and 2 are very similar. One of the main differences between the two algorithms is the existence of a *Pareto front* (Definition 5) in the multiobjective case. The Pareto front is just an additional population (the external archive) composed of a number of the non-dominated solutions found, since it has a maximum size. In order to manage the insertion of solutions in the Pareto front with the goal of obtaining a diverse set, a density estimator based on the crowding distance (proposed for NSGA-II [10]) has been used. This measure is also used to remove solutions from the archive when this becomes full.

**Algorithm 2** Pseudocode of MOCell

```
1: proc Steps_Up(mocell)      //Algorithm parameters in ‘mocell’
2: Pareto_front = Create_Front() //Creates an empty Pareto front
3: while !TerminationCondition() do
4:   for individual ← 1 to mocell.popSize do
5:     n_list←Get_Neighborhood(mocell,position(individual));
6:     parents←Selection(n_list);
7:     offspring←Recombination(mocell.Pc,parents);
8:     offspring←Mutation(mocell.Pm,offspring);
9:     Evaluate_Fitness(offspring);
10:    Insert(position(individual),offspring,mocell,aux_pop);
11:    Insert_Pareto_Front(individual);
12:  end for
13:  mocell.pop←aux_pop;
14:  mocell.pop←Feedback(mocell,ParetoFront);
15: end while
16: end_proc Steps_Up;
```

MOCell starts by creating an empty Pareto front (line 2 in Algorithm 2). Individuals are arranged in a 2-dimensional toroidal grid, and the genetic operators are successively applied to them (lines 7 and 8) until the termination condition is met (line 3). Hence, for each individual, the algorithm consists of selecting two parents from its neighborhood, recombining them in order to obtain an offspring, mutating it, evaluating the resulting individual, and inserting it in both the auxiliary population (if it is not dominated by the current individual) and the Pareto front. Finally, after each generation, the old population is replaced by the auxiliary one, and a feedback procedure is invoked to replace a fixed number of randomly chosen individuals of the population by solutions from the archive.

We have incorporated a constrain handling mechanism in MOCell to deal with constrained problems. The mechanism is the same used by NSGA-II [10].

## 4 Computational Results

This section is devoted to the evaluation of MOCell. For that, we have chosen several test problems taken from the specialized literature, and, in order to assess how competitive MOCell is, we decided to compare it against two algorithms that are representative of the state-of-the-art, NSGA-II and SPEA2. Next, we briefly comment the main features of these algorithms, including the parameter settings used in the subsequent experiments.

The NSGA-II algorithm was proposed by Deb *et al.* [10]. It is characterized by a Pareto ranking of the individuals and the use of a crowding distance as density estimator. We have used Deb's NSGA-II implementation[1]. Specifically, we used the real-coded version of the algorithm and the parameter settings proposed in [10]. A crossover probability of $p_c = 0.9$ and a mutation probability $p_m = 1/n$ (where $n$ is the number of decision variables) are used. The operators for crossover and mutation are SBX and polynomial mutation [9], with distribution indexes of $\eta_c = 20$ and $\eta_m = 20$, respectively. The population and archive sizes are 100 individuals. The algorithm stops after 25000 function evaluations.

In Table 1 we show the parameters used by MOCell. A square toroidal grid of 100 individuals has been chosen for structuring the population. The neighborhood used is composed of nine individuals: the considered individuals plus those located at its North, East, West,

[1] The implementation of NSGA-II is available for downloading at: `http://www.iitk.ac.in/kangal/soft.htm`

**Table 1.** Parameterization used in MOCell

| | |
|---|---|
| *Population Size* | 100 individuals (10 × 10) |
| *Stopping Condition* | 25000 function evaluations |
| *Neighborhood* | 1-hop neighbours (8 surrounding solutions) |
| *Selection of Parents* | binary tournament + binary tournament |
| *Recombination* | simulated binary, $p_c = 1.0$ |
| *Mutation* | polynomial, $p_m = 1.0/L$ ($L$ = individual length) |
| *Replacement* | rep_if_better_individual (NSGA-II crowding) |
| *Archive Size* | 100 individuals |
| *Density Estimator* | crowding distance |
| *Feedback* | 20 individuals |

South, NorthWest, SouthWest, NorthEast, and SouthEast (see Fig. 1c). We have also used SBX and polynomial mutation with the same distribution indexes as NSGA-II and SPEA2. Crossover and mutation rates are $p_c = 1.0$ and $p_m = 1/L$, respectively.

The resulting offspring replaces the individual at the current position if the latter is better than the former, but, as it is usual in multiobjective optimization, we need to define the concept of "best individual". Our approach is to replace the current individual if it is dominated by the offspring or both are non-dominated and the current individual has the worst crowding distance (as defined in NSGA-II) in a population composed of the neighborhood plus the offspring. For inserting the individuals in the Pareto front, the solutions in the archive are also ordered according to the crowding distance; therefore, when inserting a non-dominated solution, if the Pareto front is already full, the solution with a worst crowding distance value is removed. Finally, after each iteration, 20 randomly chosen individuals in the population are replaced by the 20 best solutions from the external archive according to the crowding distance (feedback mechanism).

## 4.1 Test Problems

We have selected for our tests both constrained and unconstrained problems that have been used in most studies in this area. Given that they are widely known, we do not include full details of them here for space constraints. They can be found in the cited references and also in books such as [7] and [8].

SPEA2 was proposed by Zitler *et al.* in [17]. In this algorithm, each individual has assigned a fitness value that is the sum of its strength raw fitness and a density estimation based on the distance to the $k$-th nearest neighbor. Like in the case of NSGA-II, we have used the authors' implementation of SPEA2[2]. The algorithm is implemented within the framework PISA [5]. However, the implementation of SPEA2 does not contain a constraint-handling management, so we were forced to modify the original implementation for including the same constraint mechanism used in NSGA-II and MOCell. We have used the following values for the parameters. Both the population and the archive have a size of 100 individuals, and the crossover and mutation operators are the same used in NSGA-II, using the same values concerning their application probabilities and distribution indexes. As in NSGA-II, the stopping condition is to compute 25000 function evaluations.

[2] The implementation of SPEA2 is available at: `http://www.tik.ee.ethz.ch/pisa/selectors/spea2/spea2.html`

**Table 2.** Unconstrained test functions

| Problem | Objective functions | Variable bounds | n |
|---|---|---|---|
| Schaffer | $f_1(x) = x^2$<br>$f_2(x) = (x-2)^2$ | $-10^5 \leq x \leq 10^5$ | 1 |
| Fonseca | $f_1(\boldsymbol{x}) = 1 - e^{-\sum_{i=1}^{n} x_i - \frac{1}{\sqrt{n}} {}^{2}}$<br>$f_2(\boldsymbol{x}) = 1 - e^{-\sum_{i=1}^{n} x_i + \frac{1}{\sqrt{n}} {}^{2}}$ | $-4 \leq x_i \leq 4$ | 3 |
| Kursawe | $f_1(\boldsymbol{x}) = \sum_{i=1}^{n-1} \; -10e^{\;-0.2*\sqrt{x_i^2 + x_{i+1}^2}}$<br>$f_2(\boldsymbol{x}) = \sum_{i=1}^{n}(\lvert x_i\rvert^a + 5\sin x_i^b); a = 0.8; b = 3$ | $-5 \leq x_i \leq 5$ | 3 |
| ZDT1 | $f_1(\boldsymbol{x}) = x_1$<br>$f_2(\boldsymbol{x}) = g(\boldsymbol{x})[1 - \sqrt{x_1/g(\boldsymbol{x})}]$<br>$g(\boldsymbol{x}) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$ | $0 \leq x_i \leq 1$ | 30 |
| ZDT2 | $f_1(\boldsymbol{x}) = x_1$<br>$f_2(\boldsymbol{x}) = g(\boldsymbol{x})[1 - (x_1/g(\boldsymbol{x}))^2]$<br>$g(\boldsymbol{x}) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$ | $0 \leq x_i \leq 1$ | 30 |
| ZDT3 | $f_1(\boldsymbol{x}) = x_1$<br>$f_2(\boldsymbol{x}) = g(\boldsymbol{x})\left[1 - \sqrt{\frac{x_1}{g(\boldsymbol{x})}} - \frac{x_1}{g(\boldsymbol{x})}\sin(10\pi x_1)\right]$<br>$g(\boldsymbol{x}) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$ | $0 \leq x_i \leq 1$ | 30 |
| ZDT4 | $f_1(\boldsymbol{x}) = x_1$<br>$f_2(\boldsymbol{x}) = g(\boldsymbol{x})[1 - (x_1/g(\boldsymbol{x}))^2]$<br>$g(\boldsymbol{x}) = 1 + 10(n-1) + \sum_{i=2}^{n}[x_i^2 - 10\cos(4\pi x_i)]$ | $0 \leq x_1 \leq 1$<br>$-5 \leq x_i \leq 5$<br>$i = 2, ..., n$ | 10 |
| ZDT6 | $f_1(\boldsymbol{x}) = 1 - e^{-4x_1}\sin^6(6\pi x_1)$<br>$f_2(\boldsymbol{x}) = g(\boldsymbol{x})[1 - (f_1(\boldsymbol{x})/g(\boldsymbol{x}))^2]$<br>$g(\boldsymbol{x}) = 1 + 9[(\sum_{i=2}^{n} x_i)/(n-1)]^{0.25}$ | $0 \leq x_i \leq 1$ | 10 |

**Table 3.** Constrained test functions

| Problem | Objective functions | Constraints | Variable bounds | n |
|---|---|---|---|---|
| Osyczka2 | $f_1(\boldsymbol{x}) = -(25(x_1-2)^2 +$<br>$(x_2-2)^2 +$<br>$(x_3-1)^2(x_4-4)^2 +$<br>$(x_5-1)^2)$<br>$f_2(\boldsymbol{x}) = x_1^2 + x_2^2 +$<br>$x_3^2 + x_4^2 + x_5^2 + x_6^2$ | $g_1(\boldsymbol{x}) = 0 \leq x_1 + x_2 - 2$<br>$g_2(\boldsymbol{x}) = 0 \leq 6 - x_1 - x_2$<br>$g_3(\boldsymbol{x}) = 0 \leq 2 - x_2 + x_1$<br>$g_4(\boldsymbol{x}) = 0 \leq 2 - x_1 + 3x_2$<br>$g_5(\boldsymbol{x}) = 0 \leq 4 - (x_3-3)^2 - x_4$<br>$g_6(\boldsymbol{x}) = 0 \leq (x_5-3)^3 + x_6 - 4$ | $0 \leq x_1, x_2 \leq 10$<br>$1 \leq x_3, x_5 \leq 5$<br>$0 \leq x_4 \leq 6$<br>$0 \leq x_6 \leq 10$ | 6 |
| Tanaka | $f_1(\boldsymbol{x}) = x_1$<br>$f_2(\boldsymbol{x}) = x_2$ | $g_1(\boldsymbol{x}) = -x_1^2 - x_2^2 + 1 +$<br>$0.1\cos(16\arctan(x_1/x_2)) \leq 0$<br>$g_2(\boldsymbol{x}) = (x_1-0.5)^2 +$<br>$(x_2-0.5)^2 \leq 0.5$ | $-\pi \leq x_i \leq \pi$ | 2 |
| ConstrEx | $f_1(\boldsymbol{x}) = x_1$<br>$f_2(\boldsymbol{x}) = (1+x_2)/x_1$ | $g_1(\boldsymbol{x}) = x_2 + 9x_1 \geq 6$<br>$g_2(\boldsymbol{x}) = -x_2 + 9x_1 \geq 1$ | $0.1 \leq x_1 \leq 1.0$<br>$0 \leq x_2 \leq 5$ | 2 |
| Srinivas | $f_1(\boldsymbol{x}) = (x_1-2)^2 +$<br>$(x_2-1)^2 + 2$<br>$f_2(\boldsymbol{x}) = 9x_1 - (x_2-1)^2$ | $g_1(\boldsymbol{x}) = x_1^2 + x_2^2 \leq 225$<br>$g_2(\boldsymbol{x}) = x_1 - 3x_2 \leq -10$ | $-20 \leq x_i \leq 20$ | 2 |

The selected unconstrained problems include the studies of Schaffer, Fonseca, and Kursawe, as well as the problems ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. Their formulation is provided in Table 2. The constrained problems are Osyczka2, Tanaka, Srinivas, and ConstrEx. They are described in Table 3.

## 4.2 Performance Metrics

For assessing the performance of the algorithms on the test problems, two different issues are normally taken into account: (i) minimize the distance of the Pareto front generated by the proposed algorithm to the exact Pareto front, and (ii) to maximize the spread of solutions found, so that we can have a distribution of vectors as smooth and uniform as possible. To determine the first issue it is usually necessary to know the exact location of the true Pareto front; in this work we have obtained these fronts using an enumerative search strategy, and they are publicly available at `http://neo.lcc.uma.es/software/esam` (an exception are the ZDTx problem family, whose fronts can be easily computed because their solutions are known).

- **Generational Distance** This metric was introduced by Van Veldhuizen and Lamont [16] for measuring how far the elements are in the set of non-dominated vectors found so far from those in the Pareto optimal set, and it is defined as:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \quad , \tag{1}$$

  where $n$ is is the number of vectors in the set of non-dominated solutions, and $d_i$ is the Euclidean distance (measured in objective space) between each of these solutions and the nearest member of the Pareto optimal set. It is clear that a value of $GD = 0$ means that all the generated elements are in the Pareto optimal set. In order to get reliable results, non-dominated sets are normalized before calculating this distance measure.
- **Spread** The *Spread* metric [10] is a diversity metric that measures the extent of spread achieved among the obtained solutions. This metric is defined as:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1)\bar{d}} \quad , \tag{2}$$

  where $d_i$ is the Euclidean distance between consecutive solutions, $\bar{d}$ is the mean of these distances, and $d_f$ and $d_l$ are the Euclidean distances to the *extreme* (bounding) solutions of the exact Pareto front in the objective space (see [10] for the details). This metric takes a zero value for an ideal distribution, pointing out a perfect spread out of the solutions in the Pareto front. We apply this metric after a normalization of the objective function values.

## 4.3 Discussion of the Results

The results are summarized in Tables 4 ($GD$) and 5 ($\Delta$), and the best result for each problem has a grey colored background. For each problem, we carried out 100 independent runs, and the tables include the mean, $\bar{x}$, and the standard deviation, $\sigma_n$, of the results. Since we deal with stochastic algorithms, an statistical analysis of the results has been made. It consists of the following steps. First a Kolmogorov-Smirnov test is performed in order to check whether the values of the results follow a normal distribution or not. If so, an ANOVA I test is done, otherwise we perform a Kruskal-Wallis test. We always consider in this work a 95% confidence level in the statistical tests. Symbol '+' in tables 4 and 5 means that the differences among the values of the three algorithms for a given problem have statistical confidence ($p$-value under 0.05).

We consider first the metric $GD$ (Table 4). We can observe that the three compared algorithms provide the best results for four out of the 12 studied problems. According to these results we cannot decide a winner algorithm considering convergence, although they allow us to conclude that MOCell is a competitive algorithm compared to NSGA-II and SPEA2. Indeed, if we consider only the constrained studied problems, the reader can see that MOCell behaves better than the other compared algorithms, since it reports the best

**Table 4.** Mean and standard deviation of the convergence metric *GD*

| Problem | MOCell $\bar{x}_{\sigma_n}$ | NSGA-II $\bar{x}_{\sigma_n}$ | SPEA2 $\bar{x}_{\sigma_n}$ | |
|---|---|---|---|---|
| Schaffer | 2.408e-4$_{\pm 1.79e-5}$ | 2.328e-4$_{\pm 1.19e-5}$ | 2.365e-4$_{\pm 1.06e-5}$ | + |
| Fonseca | 1.983e-4$_{\pm 1.60e-5}$ | 4.683e-4$_{\pm 3.95e-5}$ | 2.251e-4$_{\pm 2.37e-5}$ | + |
| Kursawe | 1.435e-4$_{\pm 1.01e-5}$ | 2.073e-4$_{\pm 2.22e-5}$ | 1.623e-4$_{\pm 1.51e-5}$ | + |
| Zdt1 | 4.057e-4$_{\pm 6.57e-5}$ | 2.168e-4$_{\pm 3.57e-5}$ | 1.992e-4$_{\pm 1.34e-5}$ | + |
| Zdt2 | 2.432e-4$_{\pm 9.29e-5}$ | 1.714e-4$_{\pm 3.80e-5}$ | 1.095e-4$_{\pm 5.37e-5}$ | + |
| Zdt3 | 2.540e-4$_{\pm 2.78e-5}$ | 2.199e-4$_{\pm 3.72e-5}$ | 2.336e-4$_{\pm 1.34e-5}$ | + |
| Zdt4 | 8.273e-4$_{\pm 1.85e-3}$ | 4.888e-4$_{\pm 2.59e-4}$ | 6.203e-2$_{\pm 3.94e-2}$ | + |
| Zdt6 | 2.106e-3$_{\pm 3.33e-4}$ | 1.001e-3$_{\pm 8.66e-5}$ | 8.252e-4$_{\pm 5.15e-5}$ | + |
| ConstrEx | 1.968e-4$_{\pm 2.29e-5}$ | 2.903e-4$_{\pm 3.19e-5}$ | 2.069e-4$_{\pm 1.78e-5}$ | + |
| Srinivas | 5.147e-5$_{\pm 1.51e-5}$ | 1.892e-4$_{\pm 3.01e-5}$ | 1.139e-4$_{\pm 1.98e-5}$ | + |
| Osyczka2 | 2.678e-3$_{\pm 5.30e-3}$ | 1.071e-3$_{\pm 1.33e-4}$ | 6.149e-3$_{\pm 1.14e-2}$ | + |
| Tanaka | 7.494e-4$_{\pm 7.09e-5}$ | 1.214e-3$_{\pm 7.95e-5}$ | 7.163e-4$_{\pm 7.13e-5}$ | + |

results for ConstrEx and Srinivas, while it is the second best approach in the other two problems.

Regarding the spread metric (Table 5), the results indicate that MOCell clearly outperforms the other two algorithms concerning the diversity of the obtained Pareto fronts, since it yields the best values in 9 out of the 12 problems. Additionally, MOCell reports the best results for all the constrained studied problems. It stands out that NSGA-II can not obtain the best value for the spread metric in any problem.

In order to graphically show our results, we plot in Fig. 2 three fronts obtained by MOCell, NSGA-II, and SPEA2, together with the optimal Pareto set obtained by the enumerative algorithm, for problem ConstrEx. The selected fronts are those having the best diversity (lowest value of $\Delta$) among the 100 produced ones by each technique for that problem. We can observe that the nondominated set of solutions generated by MOCell achieves an almost perfect spread out and convergence. Notice that SPEA2 obtains a very good diversity for values of $f_1(\boldsymbol{x})$ lower than 0.66 (similar to the solution of MOCell), but it finds only 10 solutions when $f_1(\boldsymbol{x}) \in [0.66, 1.0]$. Regarding NSGA-II, its front does not have that problem, but its worst diversity with respect to the case of MOCell stands out at a simple glance.

**Table 5.** Mean and standard deviation of the diversity metric $\Delta$

| Problem | MOCell $\bar{x}_{\sigma_n}$ | NSGA-II $\bar{x}_{\sigma_n}$ | SPEA2 $\bar{x}_{\sigma_n}$ | |
|---|---|---|---|---|
| Schaffer | 2.473e-1$_{\pm 3.11e-2}$ | 4.448e-1$_{\pm 3.62e-2}$ | 1.469e-1$_{\pm 1.14e-2}$ | + |
| Fonseca | 9.695e-2$_{\pm 1.08e-2}$ | 3.596e-1$_{\pm 2.83e-2}$ | 1.445e-1$_{\pm 1.28e-2}$ | + |
| Kursawe | 4.121e-1$_{\pm 4.32e-3}$ | 5.460e-1$_{\pm 2.41e-2}$ | 4.390e-1$_{\pm 8.94e-3}$ | + |
| Zdt1 | 1.152e-1$_{\pm 1.40e-2}$ | 3.645e-1$_{\pm 2.91e-2}$ | 1.684e-1$_{\pm 1.29e-2}$ | + |
| Zdt2 | 1.120e-1$_{\pm 1.61e-2}$ | 3.644e-1$_{\pm 3.03e-2}$ | 1.403e-1$_{\pm 6.71e-2}$ | + |
| Zdt3 | 6.998e-1$_{\pm 3.25e-2}$ | 7.416e-1$_{\pm 2.25e-2}$ | 7.040e-1$_{\pm 1.78e-2}$ | + |
| Zdt4 | 1.581e-1$_{\pm 6.14e-2}$ | 3.651e-1$_{\pm 3.32e-2}$ | 1.049e-1$_{\pm 1.71e-1}$ | + |
| Zdt6 | 1.859e-1$_{\pm 2.33e-2}$ | 2.988e-1$_{\pm 2.48e-2}$ | 1.728e-1$_{\pm 1.16e-2}$ | + |
| ConstrEx | 1.323e-1$_{\pm 1.32e-2}$ | 4.212e-1$_{\pm 3.52e-2}$ | 5.204e-1$_{\pm 1.58e-2}$ | + |
| Srinivas | 6.191e-2$_{\pm 8.63e-3}$ | 3.680e-1$_{\pm 3.02e-2}$ | 1.628e-1$_{\pm 1.25e-2}$ | + |
| Osyczka2 | 2.237e-1$_{\pm 3.50e-2}$ | 4.603e-1$_{\pm 5.58e-2}$ | 3.145e-1$_{\pm 1.35e-1}$ | + |
| Tanaka | 6.629e-1$_{\pm 2.76e-2}$ | 7.154e-1$_{\pm 2.35e-2}$ | 6.655e-1$_{\pm 2.74e-2}$ | + |

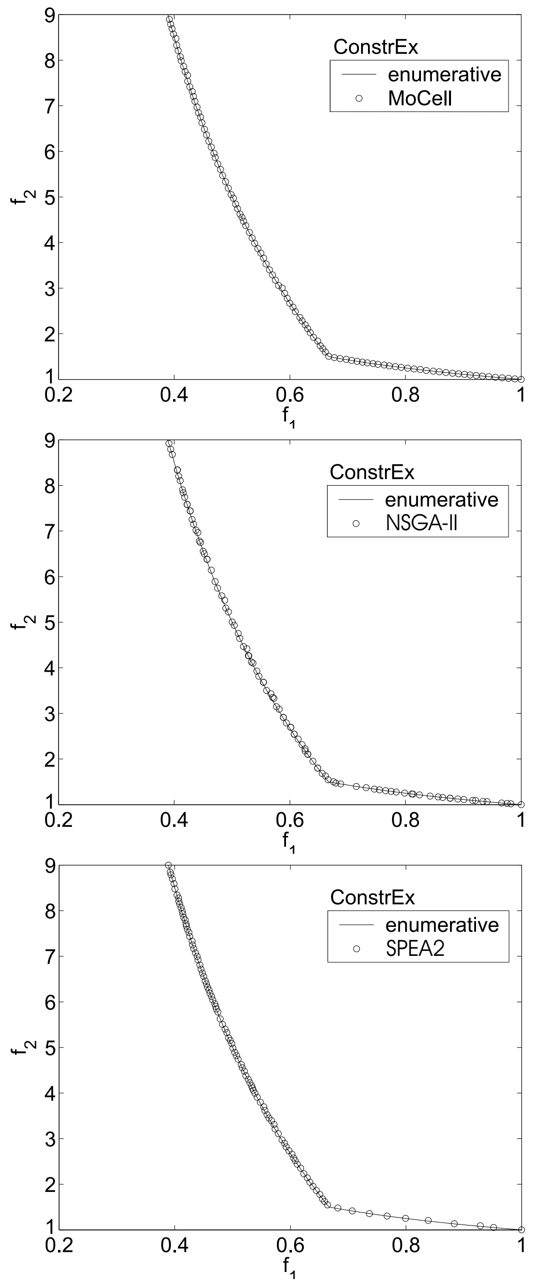

**Fig. 2.** MOCell finds a better convergence and spread of solutions than NSGA-II and SPEA2 on problem ConstrEx

We also want to remark that, concerning diversity, MOCell is not only the best of the three analyzed algorithms, but the differences in the spread values are in general noticeable compared to the rest of algorithms.

## 5 Conclusions and Future Work

We have proposed MOCell, a cellular genetic algorithm to solve multiobjective optimization problems. The algorithm uses an external archive to store the nondominated individuals found during the search. The most salient feature of MOCell with respect to the other cellular approaches for multiobjective optimization is the feedback of individuals from the archive to the population. MOCell was validated using a standard methodology which is currently used within the evolutionary multiobjective optimization community. The algorithm was compared against two state-of-the-art multiobjective optimization algorithms, NSGA-II and SPEA2; for that purpose, twelve test problems, including unconstrained and constrained ones, were chosen and two metrics were used to assess the performance of the algorithms. The results of the metrics reveal that MOCell is competitive considering the convergence metric, and it clearly outperforms all the proposals on the considered test problems according to the spread metric.

Finally, the evaluation of MOCell with other benchmarks and its application to solve real-world problems are matter of future work.

## Acknowledgments

This work has been funded by FEDER and the Spanish MCYT under contract TIN2005-08818-C04-01 (the OPLINK project, `http://oplink.lcc.uma.es`).

## References

1. E. Alba and B. Dorronsoro. The exploration/exploitation tradeoff in dynamic cellular evolutionary algorithms. *IEEE TEC*, 9(2):126–142, April 2005.
2. E. Alba, B. Dorronsoro, M. Giacobini, and M. Tomasini. *Handbook of Bioinspired Algorithms and Applications, Chapter 7*, chapter Decentralized Cellular Evolutionary Algorithms, pages 103–120. CRC Press, 2006.
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*, page To appear, 2006.
4. E. Alba and M. Tomassini. Parallelism and Evolutionary Algorithms. *IEEE Trans. on Evolutionary Computation*, 6(5):443–462, October 2002.
5. S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA - A Platform and Programming Language Independent Interface for Search Algorithms. In *EMO 2003*, pages 494–508, 2003.
6. E. Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms.* Kluwer Academic Publishers, 2000.
7. C.A. Coello, D.A. Van Veldhuizen, and G.B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems.* Genetic Algorithms and Evolutionary Computation. Kluwer Academic Publishers, 2002.
8. K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, 2001.
9. K. Deb and R.B. Agrawal. Simulated Binary Crossover for Continuous Search Space. *Complex Systems*, 9:115–148, 1995.
10. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elist Multiobjective Genetic Algorithm: NSGA-II. *IEEE TEC*, 6(2):182–197, 2002.

11. Michael Kirley. MEA: A metapopulation evolutionary algorithm for multi-objective optimisation problems. In *CEC 2001*, pages 949–956. IEEE Press, 2001.
12. J. Knowles and D. Corne. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 9–105, Piscataway, NJ, 1999. IEEE Press.
13. M. Laumanns, G. Rudolph, and H. P. Schwefel. A Spatial Predator-Prey Approach to Multi-Objective Optimization: A Preliminary Study. In *PPSN V*, pages 241–249, 1998.
14. B. Manderick and P. Spiessens. Fine-grained parallel genetic algorithm. In *Proc. of the Third Int. Conf. on Genetic Algorithms (ICGA)*, pages 428–433, 1989.
15. T. Murata and M. Gen. Cellular Genetic Algorithm for Multi-Objective Optimization. In *Proc. of the 4th Asian Fuzzy System Symposium*, pages 538–542, 2002.
16. D. A. Van Veldhuizen and G. B. Lamont. Multiobjective Evolutionary Algorithm Research: A History and Analysis. Technical Report TR-98-03, Dept. Elec. Comput. Eng., Air Force Inst. Technol., Wright-Patterson, AFB, OH, 1998.
17. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), 2001.