# MOCell: A Cellular Genetic Algorithm for Multiobjective Optimization

Antonio J. Nebro,[*] Juan J. Durillo,[†] Francisco Luna,[‡] Bernabé Dorronsoro,[§] Enrique Alba[¶]
*Departamento de Lenguajes y Ciencias de la Computación, E.T.S. Ingeniería Informática, Campus de Teatinos, 29071 Málaga, Spain*

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 nondominated solutions and a feedback mechanism in which solutions from this archive randomly replace 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 studied benchmark, our experiments indicate that MOCell obtains competitive results in terms of convergence and hypervolume, and it clearly outperforms the other two compared algorithms concerning the diversity of the solutions along the Pareto front. © 2009 Wiley Periodicals, Inc.

## 1. INTRODUCTION

Most optimization problems in the real world involve the minimization and/or maximization of more than one function. Generally speaking, multiobjective optimization is not restricted to find a unique single solution of a given multiobjective optimization problem (MOP), but a set of solutions called *nondominated 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 methods have been widely proposed within this domain. Among them, evolutionary algorithms (EAs) have been

*Author to whom all correspondence should be addressed; e-mail: antonio@lcc.uma.es.
†e-mail: durillo@lcc.uma.es.
‡e-mail: flv@lcc.uma.es.
§e-mail: bernabe@lcc.uma.es.
¶e-mail: eat@lcc.uma.es.

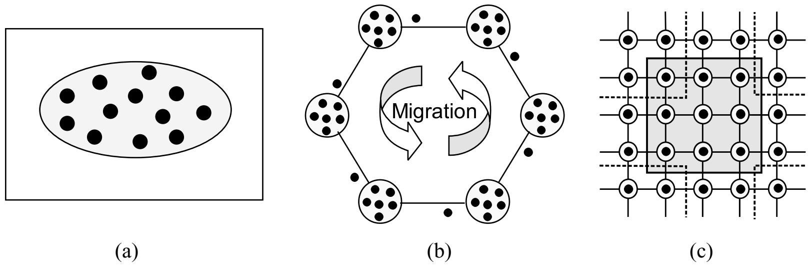

**Figure 1.** Panmictic (a), distributed (b), and cellular (c) GAs.

investigated by many researchers,[1,2] and some of the most well-known algorithms for solving MOPs belong to this class (e.g., NSGA-II,[3] PAES,[4] and SPEA2[5]).

EAs are nature-inspired search strategies based on the natural selection of evolution, and they are especially well suited for tackling MOPs because of their ability to find 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. These operators are the recombination operator, which enables the cooperation among individuals, and the mutation operator, consisting of random alterations of the problem variables. Most EAs use a single population (panmixia) of individuals and apply the operators to them as a whole (see Figure 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[6,7] (see Figures 1b and 1c). In many cases, these decentralized algorithms provide a better sampling of the search space, 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 cooperate with its nearby neighbors in the breeding loop.[8] 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 successfully adopted also for monoprocessor 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,[9,10] but little attention has been paid to its use in the multiobjective optimization field. In Ref. 11, 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[12] presented an algorithm in which, for an $n$-objective MOP, the population is structured into 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 Ref. 13. 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). In addition, 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.[14] proposed cMOGA, the unique cellular multiobjective algorithm based on the canonical cGA model before this work, to the best of our knowledge. In that work, cMOGA was used for optimizing a broadcasting strategy specifically designed for metropolitan mobile ad hoc networks.

Our proposal is called MOCell, and it is, like in the case of Ref. 14, an adaptation of a canonical cGA to the multiobjective field. MOCell uses an external archive to store the nondominated 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 new cGA for solving continuous MOPs. The algorithm uses an external archive and a feedback of solutions from the archive to the population.
- The resulting configurations are validated using a number of unconstrained MOPs usually applied in many studies in the field, including the ZDT family[15] and the recent benchmark constructed using the WFG Toolkit.[16] We include also constrained MOPs in our study.
- 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 MOP can be formally defined as follows:

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

The set of all values satisfying the constraints defines the *feasible region* $\Omega$ and any point $\vec{x} \in \Omega$ 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* $\vec{x}^* \in \Omega$ *is Pareto optimal if for every* $\vec{x} \in \Omega$ *and* $I = \{1, 2, \ldots, k\}$ *either* $\forall_{i \in I}(f_i(\vec{x}) = f_i(\vec{x}^*))$ *or there is at least one* $i \in I$ *such that* $f_i(\vec{x}) > f_i(\vec{x}^*)$.

This definition states that $\vec{x}^*$ is Pareto optimal if no feasible vector $\vec{x}$ 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* $\vec{u} = (u_1, \ldots, u_k)$ *is said to dominate* $\vec{v} = (v_1, \ldots, v_k)$ *(denoted by* $\vec{u} \preccurlyeq \vec{v}$*) if and only if* $\vec{u}$ *is partially less than* $\vec{v}$*, 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* $\vec{f}(\vec{x})$*, the Pareto optimal set is defined as* $\mathcal{P}^* = \{\vec{x} \in \Omega | \neg\exists \vec{x}' \in \Omega, \vec{f}(\vec{x}') \preccurlyeq \vec{f}(\vec{x})\}$.

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

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. Closeness to the Pareto front ensures that we are dealing with optimal solutions, where a uniform spread of the solutions means that we have made a good exploration of the search space and no regions are left unexplored.

## 3. THE ALGORITHM

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

**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;
```

## 3.1. Cellular Genetic Algorithms

A canonical cGA follows the pseudocode 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. Recombination 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 (if known), 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, our proposed multiobjective algorithm based on the cGA model. Its pseudocode is given in Algorithm 2. We can observe that Algorithms 1 and 2 are very similar. One of the main differences between them is the existence of a *Pareto front* (Definition 5) in the multiobjective case. The Pareto front is just an additional population (an external archive) composed of a number of the nondominated solutions found, since it has a maximum size. To manage the insertion of solutions in the Pareto front and with the goal of obtaining a diverse set, a density

**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;
```

estimator based on the crowding distance (proposed for NSGA-II[3]) has been used. This measure is also used to remove solutions from the archive when it becomes full.

MOCell can be considered as the next step toward using the canonical cellular model of GAs in the multiobjective field. cMOGA[14] was the first attempt in this direction: analyzing the search engine of cGAs for solving MOPs. MOCell differs from cMOGA in the inclusion of a feedback mechanism at the end of each iteration which allows the search to be guided toward the nondominated solutions already found. Therefore, this new approach not only fully exploits the cGA search engine but also the search experience stored in the external archive (enhanced intensification capabilities).

MOCell starts creating an empty Pareto front (line 2 in Algorithm 2). Individuals are arranged in a two-dimensional toroidal grid, and the genetic operators are successively applied to them (lines 7 and 8) until the termination condition is met (line 3). The breeding loop of MOCell is illustrated in Figure 2. Hence, for each individual, the algorithm lies in selecting two parents from its neighborhood, recombining them to obtain an offspring, mutating it, and evaluating the resulting individual. This individual is then asked for insertion into both the auxiliary population and the external archive. In the first case, the resulting offspring replaces the individual at the current position if the latter is worse 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 nondominated and the current individual has the worst crowding distance (as defined in NSGA-II) in a population composed of the *C9* or *Compact9* neighbors (see Figure 1c).[17]

For inserting the individual in the Pareto front, the solutions in the archive are also ranked according to the crowding distance; therefore, when inserting a

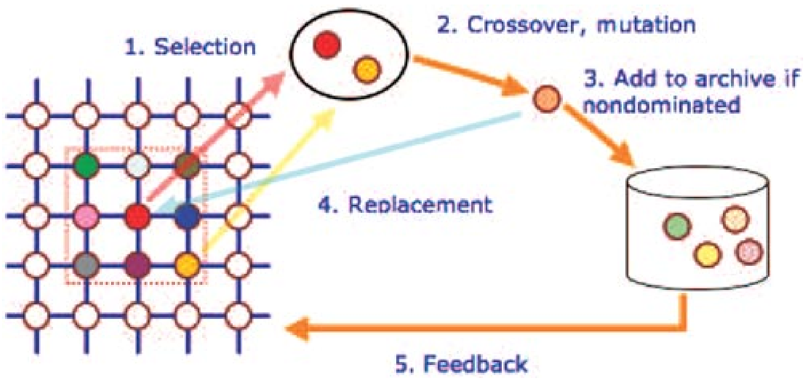

**Figure 2.** Breeding loop in MOCell.

nondominated solution, if the Pareto front is already full, the solution with the worst crowding distance value is removed. Finally, after each generation, the old population is replaced by the auxiliary one, and a feedback procedure is invoked (line 14) to replace a fixed number of randomly chosen individuals of the population by the same number of solutions taken from the archive.

MOCell has been engineered with the same constraint-handling mechanism as NSGA-II[3] for dealing with constrained MOPs. Whenever two individuals are compared, their constraints are evaluated. If both are feasible, a Pareto dominance test (Definition 3) is directly applied. If one is feasible and the other one is unfeasible, the feasible one dominates. Otherwise, if both individuals are infeasible, then the one with the lowest amount of constraint violation dominates the other.

## 4. EXPERIMENTS

This section is devoted to the evaluation of MOCell. We first detail the test problems used in this work. Then, we describe the indicators which have been used to measure the quality of the approximated Pareto fronts computed by the multiobjective algorithms. A comparison against cMOGA, the predecessor of MOCell, is performed next to effectively prove the enhanced search capabilities provided by the newly developed feedback mechanism. The section ends with a extensive comparative analysis between our approach and two algorithms which are representative of the state-of-the-art: NSGA-II and SPEA2.

All the algorithms used in this work (cMOGA, MOCell, NSGA-II, and SPEA2) have been implemented using the jMetal framework.[18] jMetal stands for *Metaheuristic Algorithms in Java*, and it is an object-oriented Java-based framework aimed at facilitating the development of metaheuristics for solving MOPs. jMetal provides a rich set of classes, which can be used as the building blocks of multiobjective metaheuristics; thus, taking advantage of code reusing, the algorithms share the same base components, such as implementations of genetic operators and density estimators, so making the fair comparison of different metaheuristics for MOPs possible. In this sense, jMetal has been validated by comparing the original implementations of NSGA-II and SPEA2 against their jMetal versions in Ref. 18, where it is shown that the latter implementations are fully competitive and thus they can be used as a reference in comparative studies using metaheuristics developed with jMetal.[a]

For each experiment, we have made 100 independent runs, and we have obtained the median, $\tilde{x}$, and interquartile range, $IQR$, as measures of location (or central tendency) and statistical dispersion, respectively. Since we are dealing with stochastic algorithms and we do want to provide the results with confidence, the following statistical analysis has been performed in all this work. First, a Kolmogorov–Smirnov test is performed to check whether the values of the results follow a normal (Gaussian) distribution or not. If so, an ANOVA test is done, otherwise we perform a Kruskal–Wallis test. We always consider in this work a confidence level of 95% (i.e., significance level of 5% or $p$-value under 0.05) in the statistical tests, which means that the differences are unlikely to have occurred by chance with a probability of 95%. Successful tests are marked with "+" symbols in the last column in all the tables; conversely, "−" means that no statistical confidence was found ($p$-value $> 0.05$). The best quality measure that shows statistical confidence for each problem has been highlighted with a gray-colored background in all the result tables.

## 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. Regarding unconstrained problems, we emphasize the use of well-known benchmarking problems, such as Schaffer,[19] Fonseca and Flemming,[20] and Kursawe,[21] as well as the ZDT[15] problem family; they are formulated in Table I. We include also here a set of MOPs obtained using the WFG Toolkit.[16] This toolkit allows the user to define benchmarks of MOPs having different properties (convex, nonconvex, degenerate, disconnected, etc), and it is becoming a standard in performance analyses in the area. In this study, we use the biobjective version of the nine problems, WFG1 to WFG9, defined in Ref. 16. The properties of these problems are detailed in Table II.

In addition, we have chosen a number of constrained problems to make our study more complete. They are Osyczka2,[22] Tanaka,[23] Srinivas,[24] and ConstrEx[3]; their descriptions can be found in Table III.

## 4.2. Quality Indicators

For assessing the performance of the algorithms to the test problems, two different issues are normally taken into account: (i) to minimize the distance of the Pareto front generated by the proposed algorithm to the exact Pareto front (convergence), and (ii) to maximize the spread of solutions found (diversity), so that

[a]jMetal is available at: `http://jmetal.sourceforge.net`.

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

**Table II.** Properties of the MOPs created using the WFG toolkit.

| Problem | Separability | Modality | Bias | Geometry |
|---|---|---|---|---|
| WFG1 | Separable | Uni | Polynomial, flat | Convex, mixed |
| WFG2 | Nonseparable | $f_1$ uni, $f_2$ multi | No bias | Convex, disconnected |
| WFG3 | Nonseparable | Uni | No bias | Linear, Degenerate |
| WFG4 | Nonseparable | Multi | No bias | Concave |
| WFG5 | Separable | Deceptive | No bias | Concave |
| WFG6 | Nonseparable | Uni | No bias | Concave |
| WFG7 | Separable | Uni | Parameter dependent | Concave |
| WFG8 | Nonseparable | Uni | Parameter dependent | Concave |
| WFG9 | Nonseparable | Multi, deceptive | Parameter dependent | Concave |

we can have a distribution as smooth and uniform as possible. To determine these issues, it is usually necessary to know the exact location of the true Pareto front; in this work we have obtained these fronts by using an enumerative search strategy (an exception are the ZDT and WFG problem families, whose fronts can be easily computed because their solutions are known).

A number of quality indicators have been proposed to be used in comparative studies among metaheuristics for solving MOPs. They can be classified into those measuring convergence, diversity, or both. We use in this work one of each type:

**Table III.** Constrained test functions.

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

- *Generational Distance.* This indicator was introduced by Van Veldhuizen and Lamont[25] for measuring how far the elements in the set of nondominated vectors found are from those in the Pareto optimal set. It is defined as

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

where $n$ is the number of vectors in the set of nondominated solutions, and $d_i$ is the Euclidean distance (measured in objective space) between each of the solutions found 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. To get reliable results, nondominated sets are normalized before calculating this distance measure.

- *Spread.* The *Spread* measurement[3] is a diversity indicator that measures the extent of spread achieved among the obtained solutions. It is defined as

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

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 Ref. 3 for the details). This measure takes a zero value for an ideal distribution, pointing out a perfect spread out of the solutions in the Pareto front. We also apply this indicator after a normalization of the objective function values.

- *Hypervolume.* This quality indicator calculates the volume (in the objective space) covered by members of a nondominated set of solutions $Q$ (the region enclosed into the discontinuous line in Figure 3, $Q = \{A, B, C\}$) for problems where all objectives are to be minimized.[26] Mathematically, for each solution $i \in Q$, a hypercube $v_i$ is constructed with a reference point $W$ and the solution $i$ as the diagonal corners of the hypercube. The reference point can simply be found by constructing a vector of worst objective function values. Thereafter, a union of all hypercubes is found and its hypervolume ($HV$)

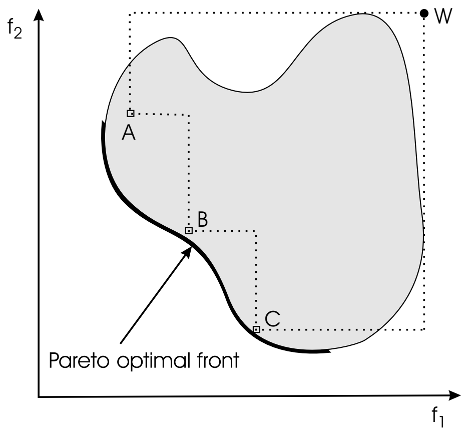

**Figure 3.** The hypervolume enclosed by the nondominated solutions.

is calculated:

$$HV = \text{volume}\left(\bigcup_{i=1}^{|Q|} v_i\right). \tag{3}$$

Algorithms with larger values of $HV$ are desirable. Since this indicator is not free from arbitrary scaling of objectives, we have evaluated it by using normalized objective function values.

### 4.3. MOCell vs. cMOGA

The first step in the evaluation of MOCell is to compare it against cMOGA, its previous version, in order to confirm that the new feedback mechanism is able to improve the search capabilities of the algorithm. cMOGA was proposed by Alba et al.[14] It is characterized by adding a *Pareto front* to the canonical cellular model for its adaptation to deal with multiples objectives. The Pareto front is just an additional population composed of the nondominated solutions found so far, which has a maximum size. To maintain a set of diverse solutions, this population uses a crowding procedure based on the adaptive grid mechanism used in PAES.[4]

For this comparison we have used the problems ZDT1 (convex), ZDT2 (nonconvex), ZDT3 (nonconvex, disconnected), ZDT4 (convex, multimodal), and ZDT6 (nonconvex, nonuniformly spaced).[15] We consider that the features of these MOPs make them meaningful enough to support our claims and they will allow us to draw clear conclusions about the accuracy of the two multiobjective optimizers without using the entire benchmark.

Table IV includes the parameters used in cMOGA and MOCell. First of all, the same number of function evaluations (25,000) has been used as stopping condition to perform a fair comparison between the algorithms. They also share many other

**Table IV.** Parameterization used in cMOGA and MOCell.

| | cMOGA | MOCell |
|---|---|---|
| *Stopping Condition* | 25000 function evaluations | |
| *Population Size* | 100 individuals (10 × 10) | |
| *Neighborhood* | NEWS | MOORE (C9 or Compact9) |
| *Selection of Parents* | current + binary tournament | binary tournament |
| *Recombination* | simulated binary, $p_c = 1.0$ | |
| *Mutation* | polynomial, $p_m = 1.0/L$ ($L$ = individual length) | |
| *Archive Size* | 100 individuals | |
| *Replacement* | rep_if_non_dominated | rep_if_better_ranking_and_crowding |
| *Density Estimator* | Adaptive grid | crowding distance |
| *Feedback* | — | 20 individuals |

**Table V.** Performance comparison using the three quality indicators between MOCell and cMOGA.

| | Generational Distance ($GD$) | | |
|---|---|---|---|
| | MOCell | cMOGA | |
| Problem | $\tilde{x}_{IQR}$ | $\tilde{x}_{IQR}$ | |
| ZDT1 | 6.288e−4 $_{1.5e-4}$ | 5.037e−1 $_{5.9e-2}$ | + |
| ZDT2 | 5.651e−4 $_{2.0e-4}$ | 9.678e−1 $_{1.6e-1}$ | + |
| ZDT3 | 3.326e−4 $_{8.5e-5}$ | 2.652e−1 $_{3.5e-2}$ | + |
| ZDT4 | 9.668e−4 $_{6.4e-4}$ | 2.912e+1 $_{6.3e+0}$ | + |
| ZDT6 | 3.963e−3 $_{1.3e-3}$ | 2.063e+0 $_{3.5e-1}$ | + |
| | Spread ($\Delta$) | | |
| | MOCell | cMOGA | |
| Problem | $\tilde{x}_{IQR}$ | $\tilde{x}_{IQR}$ | |
| ZDT1 | 1.541e−1 $_{2.1e-2}$ | 8.676e−1 $_{5.2e-2}$ | + |
| ZDT2 | 1.753e−1 $_{3.8e-2}$ | 9.283e−1 $_{4.4e-2}$ | + |
| ZDT3 | 7.106e−1 $_{7.5e-3}$ | 8.527e−1 $_{4.4e-2}$ | + |
| ZDT4 | 1.964e−1 $_{9.1e-2}$ | 9.142e−1 $_{7.8e-2}$ | + |
| ZDT6 | 3.806e−1 $_{1.1e-1}$ | 9.412e−1 $_{2.8e-2}$ | + |
| | Hypervolume ($HV$) | | |
| | MOCell | cMOGA | |
| Problem | $\tilde{x}_{IQR}$ | $\tilde{x}_{IQR}$ | |
| ZDT1 | 6.543e−1 $_{2.0e-3}$ | 0.000e+0 $_{0.0e+0}$ | + |
| ZDT2 | 3.216e−1 $_{2.8e-3}$ | 0.000e+0 $_{0.0e+0}$ | + |
| ZDT3 | 5.111e−1 $_{2.2e-3}$ | 0.000e+0 $_{0.0e+0}$ | + |
| ZDT4 | 6.487e−1 $_{9.6e-3}$ | 0.000e+0 $_{0.0e+0}$ | + |
| ZDT6 | 3.487e−1 $_{1.7e-2}$ | 0.000e+0 $_{0.0e+0}$ | + |

parameter settings: a square toroidal grid of 100 individuals; as recombination and mutation operators, SBX and polynomial mutation[27] with distribution indexes $\eta_c = 20$ and $\eta_m = 20$, respectively; the recombination and mutation rates are $p_c = 1.0$ and $p_m = 1.0/L$; and the maximum archive size is 100. cMOGA and MOCell differ in the neighborhood structure used (NEWS and C9, respectively[17]), in the selection strategy (cMOGA chooses the current individual as the first parent and the second one is selected with a binary tournament, whereas MOCell uses binary tournament to choose the two parents), in the replacement strategy (see the description of the algorithms for the details on this issue), and in the density estimator used to spread out the nondominated solutions along the Pareto front (PAES adaptive grid *vs* NSGA-II ranking and crowding). Finally, only MOCell has been endowed with a feedback mechanism so that $N$ randomly chosen individuals in the population are replaced by the $N$ best solutions from the external archive according to the crowding distance. After some preliminary experimentation, we chose a value of $N$ equal to 20.

The results of the three quality indicators are include in Table V. As it can be seen in this table, MOCell clearly outperforms cMOGA in all the measures for all the MOPs considered. Indeed, the generational distance values show that the Pareto fronts computed by MOCell are between 3 and 5 orders of magnitude closer to the optimal Pareto front than those computed by cMOGA. Although differences in the spread indicator are not that large in terms of absolute values, small variations in this measure have a major impact on the distribution of nondominated solutions along the resulting Pareto fronts. Finally, it is remarkable the zero values for cMOGA in the $HV$ indicator. This means that all the solutions are outside the limits of the true Pareto front (used at the normalization stage); when computing this quality indicator, these solutions are no longer considered, because the obtained value would be unreliable otherwise. Consequently, $HV$ shows that cMOGA is not able to converge toward the optimal front, whereas MOCell is (reaching very accurate results, as it is shown later in the next section).

### 4.4. Comparing MOCell against NSGA-II and SPEA2

To know how competitive MOCell is, we decided to compare it against two algorithms that are representative of the state-of-the-art algorithms. These algorithms are NSGA-II and SPEA2. Next, we briefly describe these algorithms, including the parameter settings used in the subsequent experiments.

The NSGA-II algorithm was proposed by Deb et al.[3] It is characterized by a Pareto ranking of the individuals and the use of a crowding distance as density estimator. Specifically, we used the real-coded version of the algorithm and the parameter settings proposed in Ref. 3. A recombination 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 recombination and mutation are SBX and polynomial mutation, with distribution indexes of $\eta_c = 20$ and $\eta_m = 20$, respectively. The population and archive sizes are 100 individuals. The algorithm stops after 25,000 function evaluations.

SPEA2 was proposed by Zitler et al.[5] In this algorithm, each individual has a fitness value assigned that is the sum of its strength raw fitness and a density estimation based on the distance to the $k$th nearest neighbor. The implementation of SPEA2 (provided by jMetal) is a modification of the original one, which is capable of solving constrained problems. It was made by including in the original algorithm the same constraint mechanism used in NSGA-II and MOCell. We have adopted the

**Table VI.** Median and interquartile range of generational distance ($GD$).

| Problem | MOCell $\tilde{x}_{IQR}$ | NSGA-II $\tilde{x}_{IQR}$ | SPEA2 $\tilde{x}_{IQR}$ | |
|---|---|---|---|---|
| Schaffer | 2.380e-4 $_{1.8e-5}$ | 2.372e-4 $_{1.9e-5}$ | 2.374e-4 $_{1.8e-5}$ | – |
| Fonseca | 2.093e-4 $_{2.0e-5}$ | 3.544e-4 $_{5.6e-5}$ | 2.181e-4 $_{2.9e-5}$ | + |
| Kursawe | 1.466e-4 $_{1.6e-5}$ | 2.167e-4 $_{3.1e-5}$ | 1.586e-4 $_{2.0e-5}$ | + |
| ZDT1 | 6.288e-4 $_{1.5e-4}$ | 2.198e-4 $_{4.8e-5}$ | 2.211e-4 $_{2.8e-5}$ | + |
| ZDT2 | 5.651e-4 $_{2.0e-4}$ | 1.674e-4 $_{4.3e-5}$ | 1.770e-4 $_{4.8e-5}$ | + |
| ZDT3 | 3.326e-4 $_{8.5e-5}$ | 2.126e-4 $_{2.1e-5}$ | 2.320e-4 $_{2.0e-5}$ | + |
| ZDT4 | 9.668e-4 $_{6.4e-4}$ | 4.353e-4 $_{3.2e-4}$ | 5.753e-4 $_{4.4e-4}$ | + |
| ZDT6 | 3.963e-3 $_{1.3e-3}$ | 1.010e-3 $_{1.3e-4}$ | 1.750e-3 $_{2.9e-4}$ | + |
| ConstrEx | 2.194e-4 $_{3.2e-5}$ | 2.947e-4 $_{4.5e-5}$ | 2.027e-4 $_{2.2e-5}$ | + |
| Srinivas | 4.892e-5 $_{1.3e-5}$ | 2.012e-4 $_{4.8e-5}$ | 1.115e-4 $_{3.6e-5}$ | + |
| Osyczka2 | 1.051e-3 $_{1.2e-4}$ | 1.040e-3 $_{9.2e-5}$ | 1.472e-3 $_{2.2e-4}$ | + |
| Tanaka | 6.984e-4 $_{1.5e-4}$ | 7.588e-4 $_{1.2e-4}$ | 6.417e-4 $_{1.2e-4}$ | + |
| WFG1 | 1.962e-4 $_{8.0e-3}$ | 1.967e-4 $_{8.3e-3}$ | 6.438e-4 $_{1.0e-2}$ | + |
| WFG2 | 4.408e-4 $_{1.4e-4}$ | 5.196e-4 $_{1.7e-4}$ | 4.474e-4 $_{1.2e-4}$ | + |
| WFG3 | 1.372e-4 $_{1.4e-5}$ | 1.553e-4 $_{1.9e-5}$ | 1.448e-4 $_{1.2e-5}$ | + |
| WFG4 | 6.423e-4 $_{2.2e-5}$ | 6.870e-4 $_{1.4e-4}$ | 6.377e-4 $_{3.0e-5}$ | + |
| WFG5 | 2.634e-3 $_{2.6e-5}$ | 2.655e-3 $_{3.2e-5}$ | 2.718e-3 $_{1.7e-5}$ | + |
| WFG6 | 4.984e-4 $_{4.3e-4}$ | 5.539e-4 $_{6.5e-4}$ | 4.654e-4 $_{6.7e-4}$ | – |
| WFG7 | 3.069e-4 $_{2.2e-5}$ | 3.444e-4 $_{4.7e-5}$ | 3.020e-4 $_{4.5e-5}$ | + |
| WFG8 | 1.009e-2 $_{6.6e-3}$ | 1.446e-2 $_{5.3e-3}$ | 1.569e-2 $_{6.0e-3}$ | + |
| WFG9 | 1.072e-3 $_{6.1e-5}$ | 1.223e-3 $_{2.1e-4}$ | 9.313e-4 $_{9.1e-5}$ | + |

**Table VII.** Problems in which the `multcompare` function finds statistical differences for $GD$.

| | | Problems |
|---|---|---|
| MOCell | NSGA-II | Fonseca, Kursawe, ZDT1, ZDT2, ZDT3, ZDT4, ZDT6 |
| | | ConstrEx, Srinivas, Tanaka |
| | | WFG2, WFG3, WFG4, WFG5, WFG7, WFG8, WFG9 |
| MOCell | SPEA2 | Kursawe, ZDT1, ZDT2, ZDT3, ZDT4, ZDT6 |
| | | ConstrEx, Srinivas, Osyczka2, Tanaka |
| | | WFG1, WFG3, WFG5, WFG8, WFG9 |
| NSGA-II | SPEA2 | Fonseca, Kursawe, ZDT3, ZDT4, ZDT6 |
| | | ConstrEx, Srinivas, Osyczka2, Tanaka |
| | | WFG1, WFG3, WFG4, WFG5, WFG7, WFG8, WFG9 |

following values for the parameters. Both the population and the archive have a size of 100 individuals, and the recombination and mutation operators are the same as those of NSGA-II, with the same values concerning their application probabilities and distribution indexes. As in NSGA-II and MOCell, the stopping condition is to compute 25,000 function evaluations.

The experimentation have been conducted here in the same way as in the previous section, but it now involves the three algorithms and the whole benchmark presented in Section 4.1 (see Tables VI, VIII, and X). To further analyze the results statistically, we have then included a posthoc testing phase which allows for a multiple comparison of samples. We have used the `multcompare` function provided

**Table VIII.** Median and interquartile range of Δ.

| Problem | MOCell $\tilde{x}_{IQR}$ | NSGA-II $\tilde{x}_{IQR}$ | SPEA2 $\tilde{x}_{IQR}$ | |
|---|---|---|---|---|
| Schaffer | 2.511e-1 $_{3.3e-2}$ | 2.864e-1 $_{3.9e-2}$ | 2.572e-1 $_{4.0e-2}$ | + |
| Fonseca | 1.035e-1 $_{1.4e-2}$ | 4.019e-1 $_{4.6e-2}$ | 1.381e-1 $_{1.8e-2}$ | + |
| Kursawe | 4.136e-1 $_{5.1e-3}$ | 5.603e-1 $_{3.1e-2}$ | 4.396e-1 $_{1.1e-2}$ | + |
| ZDT1 | 1.541e-1 $_{2.1e-2}$ | 3.753e-1 $_{4.2e-2}$ | 1.486e-1 $_{1.8e-2}$ | + |
| ZDT2 | 1.753e-1 $_{3.8e-2}$ | 3.814e-1 $_{3.9e-2}$ | 1.558e-1 $_{2.8e-2}$ | + |
| ZDT3 | 7.106e-1 $_{7.5e-3}$ | 7.458e-1 $_{2.0e-2}$ | 7.099e-1 $_{7.7e-3}$ | + |
| ZDT4 | 1.964e-1 $_{9.1e-2}$ | 3.849e-1 $_{5.3e-2}$ | 2.612e-1 $_{1.7e-1}$ | + |
| ZDT6 | 3.806e-1 $_{1.1e-1}$ | 3.591e-1 $_{4.6e-2}$ | 2.268e-1 $_{3.0e-2}$ | + |
| ConstrEx | 1.417e-1 $_{1.5e-2}$ | 4.651e-1 $_{5.4e-2}$ | 5.141e-1 $_{2.5e-2}$ | + |
| Srinivas | 7.541e-2 $_{1.3e-2}$ | 4.086e-1 $_{5.2e-2}$ | 1.733e-1 $_{1.9e-2}$ | + |
| Osyczka2 | 3.211e-1 $_{1.3e-1}$ | 5.928e-1 $_{1.2e-1}$ | 7.553e-1 $_{1.5e-1}$ | + |
| Tanaka | 7.168e-1 $_{4.6e-2}$ | 8.034e-1 $_{3.5e-2}$ | 7.535e-1 $_{4.4e-2}$ | + |
| WFG1 | 5.469e-1 $_{9.3e-2}$ | 7.170e-1 $_{4.5e-2}$ | 6.578e-1 $_{7.0e-2}$ | + |
| WFG2 | 7.490e-1 $_{1.1e-2}$ | 7.968e-1 $_{1.5e-2}$ | 7.519e-1 $_{1.1e-2}$ | + |
| WFG3 | 3.698e-1 $_{9.8e-3}$ | 6.101e-1 $_{3.8e-2}$ | 4.379e-1 $_{1.3e-2}$ | + |
| WFG4 | 1.349e-1 $_{1.9e-2}$ | 3.835e-1 $_{4.3e-2}$ | 2.681e-1 $_{3.1e-2}$ | + |
| WFG5 | 1.311e-1 $_{2.5e-2}$ | 4.077e-1 $_{4.0e-2}$ | 2.805e-1 $_{2.7e-2}$ | + |
| WFG6 | 1.178e-1 $_{2.1e-2}$ | 3.807e-1 $_{4.2e-2}$ | 2.506e-1 $_{2.4e-2}$ | + |
| WFG7 | 1.059e-1 $_{1.8e-2}$ | 3.836e-1 $_{4.4e-2}$ | 2.453e-1 $_{2.7e-2}$ | + |
| WFG8 | 5.596e-1 $_{6.3e-2}$ | 6.472e-1 $_{5.1e-2}$ | 6.108e-1 $_{5.7e-2}$ | + |
| WFG9 | 1.597e-1 $_{1.8e-2}$ | 3.994e-1 $_{3.9e-2}$ | 2.945e-1 $_{2.4e-2}$ | + |

**Table IX.** Problems in which the `multcompare` function finds statistical differences for Δ.

| | | Problems |
|---|---|---|
| MOCell | NSGA-II | Schaffer, Fonseca, Kursawe, ZDT1, ZDT2, ZDT3, ZDT4, ZDT6 |
| | | ConstrEx, Srinivas, Osyczka2, Tanaka |
| | | WFG1, WFG2, WFG3, WFG4, WFG5, WFG6, WFG7, WFG8, WFG9 |
| MOCell | SPEA2 | Fonseca, Kursawe, ZDT1, ZDT4, ZDT6 |
| | | ConstrEx, Srinivas, Osyczka2, Tanaka |
| | | WFG1, WFG2, WFG3, WFG4, WFG5, WFG6, WFG7, WFG8, WFG9 |
| NSGA-II | SPEA2 | Schaffer, Fonseca, Kursawe, ZDT1, ZDT2, ZDT3, ZDT4, ZDT6 |
| | | ConstrEx, Srinivas, Osyczka2, Tanaka |
| | | WFG1, WFG2, WFG3, WFG4, WFG5, WFG6, WFG7, WFG8, WFG9 |

by Matlab. This function chooses the most appropriate type of critical value to be used in the multiple comparison, which ranges from the more conservative *HSD* or *Tukey–Kramer* method to less conservative *Scheffe's S* procedure.[28] The same confidence level has been kept for this testing phase ($\alpha = 0.05$). Tables VII, IX and XI include the results of those pairwise tests in which differences are significant for $GD$, Δ, and $HV$, respectively. We have used an special notation on these tables. We have underlined, on the one hand, one of the algorithms involved in the pairwise test and, on the other hand, those MOPs for which this underlined algorithm obtained the best value in the considered quality indicator.

**Table X.** Median and interquartile range of $HV$.

| Problem | MOCell $\tilde{x}_{IQR}$ | NSGA-II $\tilde{x}_{IQR}$ | SPEA2 $\tilde{x}_{IQR}$ | |
|---|---|---|---|---|
| Schaffer | 8.296e-1 $_{1.8e-4}$ | 8.294e-1 $_{1.7e-4}$ | 8.294e-1 $_{1.7e-4}$ | + |
| Fonseca | 3.107e-1 $_{2.3e-4}$ | 3.080e-1 $_{5.8e-4}$ | 3.107e-1 $_{3.1e-4}$ | + |
| Kursawe | 4.011e-1 $_{1.6e-4}$ | 3.996e-1 $_{3.1e-4}$ | 4.009e-1 $_{2.1e-4}$ | + |
| ZDT1 | 6.543e-1 $_{2.0e-3}$ | 6.594e-1 $_{4.0e-4}$ | 6.600e-1 $_{3.5e-4}$ | + |
| ZDT2 | 3.216e-1 $_{2.8e-3}$ | 3.261e-1 $_{4.8e-4}$ | 3.263e-1 $_{7.4e-4}$ | + |
| ZDT3 | 5.111e-1 $_{2.2e-3}$ | 5.148e-1 $_{2.7e-4}$ | 5.141e-1 $_{3.4e-4}$ | + |
| ZDT4 | 6.487e-1 $_{9.6e-3}$ | 6.552e-1 $_{4.7e-3}$ | 6.518e-1 $_{1.0e-2}$ | + |
| ZDT6 | 3.487e-1 $_{1.7e-2}$ | 3.887e-1 $_{2.2e-3}$ | 3.785e-1 $_{4.3e-3}$ | + |
| ConstrEx | 7.760e-1 $_{2.7e-4}$ | 7.743e-1 $_{3.5e-4}$ | 7.757e-1 $_{2.9e-4}$ | + |
| Srinivas | 5.407e-1 $_{7.9e-5}$ | 5.379e-1 $_{4.0e-4}$ | 5.400e-1 $_{2.0e-4}$ | + |
| Osyczka2 | 7.507e-1 $_{7.0e-3}$ | 7.451e-1 $_{7.9e-3}$ | 7.308e-1 $_{3.4e-2}$ | + |
| Tanaka | 3.082e-1 $_{2.8e-4}$ | 3.077e-1 $_{4.1e-4}$ | 3.085e-1 $_{7.5e-4}$ | + |
| WFG1 | 5.491e-1 $_{1.1e-1}$ | 5.140e-1 $_{1.5e-1}$ | 4.337e-1 $_{1.4e-1}$ | + |
| WFG2 | 5.616e-1 $_{2.9e-3}$ | 5.631e-1 $_{2.9e-3}$ | 5.615e-1 $_{2.9e-3}$ | + |
| WFG3 | 4.420e-1 $_{2.0e-4}$ | 4.411e-1 $_{3.2e-4}$ | 4.418e-1 $_{2.2e-4}$ | + |
| WFG4 | 2.187e-1 $_{3.1e-4}$ | 2.173e-1 $_{5.3e-4}$ | 2.181e-1 $_{3.4e-4}$ | + |
| WFG5 | 1.961e-1 $_{7.5e-5}$ | 1.948e-1 $_{4.8e-4}$ | 1.956e-1 $_{1.5e-4}$ | + |
| WFG6 | 2.051e-1 $_{7.0e-3}$ | 2.033e-1 $_{9.9e-3}$ | 2.056e-1 $_{1.1e-2}$ | − |
| WFG7 | 2.104e-1 $_{1.7e-4}$ | 2.088e-1 $_{4.3e-4}$ | 2.098e-1 $_{2.7e-4}$ | + |
| WFG8 | 1.456e-1 $_{2.1e-2}$ | 1.470e-1 $_{2.3e-3}$ | 1.469e-1 $_{1.7e-3}$ | + |
| WFG9 | 2.380e-1 $_{2.2e-3}$ | 2.372e-1 $_{2.2e-3}$ | 2.386e-1 $_{2.2e-3}$ | + |

**Table XI.** Problems in which the `multcompare` function finds statistical differences for $HV$.

| | | |
|---|---|---|
| MOCell | NSGA-II | Schaffer, Fonseca, Kursawe, ZDT1, ZDT2, ZDT3, ZDT4, ZDT6 |
| | | ConstrEx, Srinivas, Tanaka |
| | | WFG1, WFG2, WFG3, WFG4, WFG5, WFG7, WFG8, WFG9 |
| MOCell | SPEA2 | Kursawe, ZDT1, ZDT2, ZDT3, ZDT6 |
| | | ConstrEx, Srinivas, Osyczka2, Tanaka |
| | | WFG1, WFG2, WFG3, WFG4, WFG5, WFG7 |
| NSGA-II | SPEA2 | Fonseca, Kursawe, ZDT1, ZDT3, ZDT4, ZDT6 |
| | | ConstrEx, Srinivas, Osyczka2, Tanaka |
| | | WFG1, WFG3, WFG4, WFG5, WFG7, WFG9 |

Let us analyze first the $GD$ indicator (Table VI). We can observe that, for the benchmark used, MOCell obtains the best values in 8 of the 21 studied MOPs, whereas NSGA-II and SPEA2 yield the best results in 6 and 5 problems, respectively. According to these results, there is not a clear winner algorithm. It is interesting to note that NSGA-II is the best technique in the first group of MOPs, specially in the ZDT family, whereas it is the less accurate metaheuristic for the WFG problems, since it was not able to reach the best (lowest) indicator value in any instance.

The results included in Table VII show that statistical confidence exists in most pairwise comparisons among the three algorithms (17, 15, and 16 of 21 MOPs, respectively). If we consider MOCell vs. NSGA-II, the underlined problems (i.e., those in which MOCell has a lower $GD$ value) point out that our approach outperforms

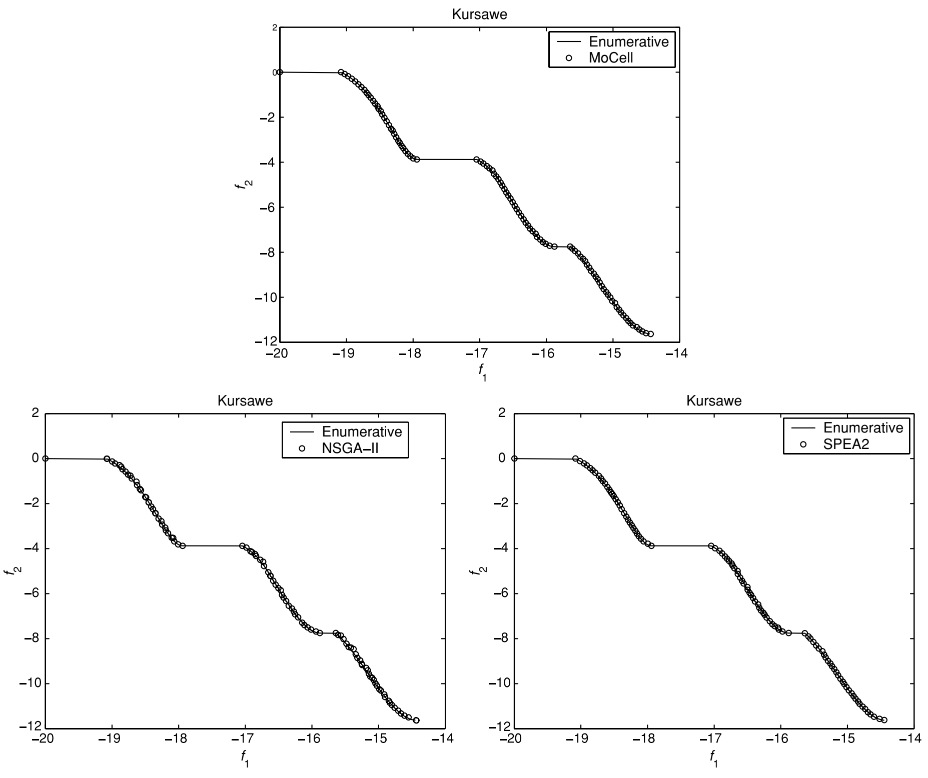

**Figure 4.** Pareto fronts obtained with the three algorithms for Kursawe.

NSGA-II in all MOPs but the ZDT family. Almost the same holds for MOCell and SPEA2 (however, SPEA2 is not only the best in the ZDT family, but also in ConstrEx, Tanaka, and WFG9). Finally, the last pairwise comparison shows that NSGA-II is a more accurate algorithm in the ZDT family, SPEA2 converges better toward the exact Pareto fronts of constrained MOPs, and no clear conclusion can be drawn on the WFG family.

With regard to the Spread ($\Delta$) quality measure (Table VIII), the results indicate that MOCell clearly outperforms the other two algorithms concerning the diversity of nondominated solutions along the obtained Pareto fronts, since it yields the best values in 17 of the 21 MOPs (with statistical confidence). Furthermore, MOCell reports the best (lowest) values for $\Delta$ in all the studied constrained problems and the entire WFG family as well. This quality indicator also stands out that NSGA-II is the worst approach, since it does not outperform neither MOCell nor SPEA2 for any MOP.

All pairwise comparisons are presented in Table IX, allowing a stronger support of the previous claims. On the one hand, NSGA-II only outperforms MOCell in one single MOP, ZDT6, and SPEA2 does in ZDT1 and ZDT6. As expected, SPEA2 also

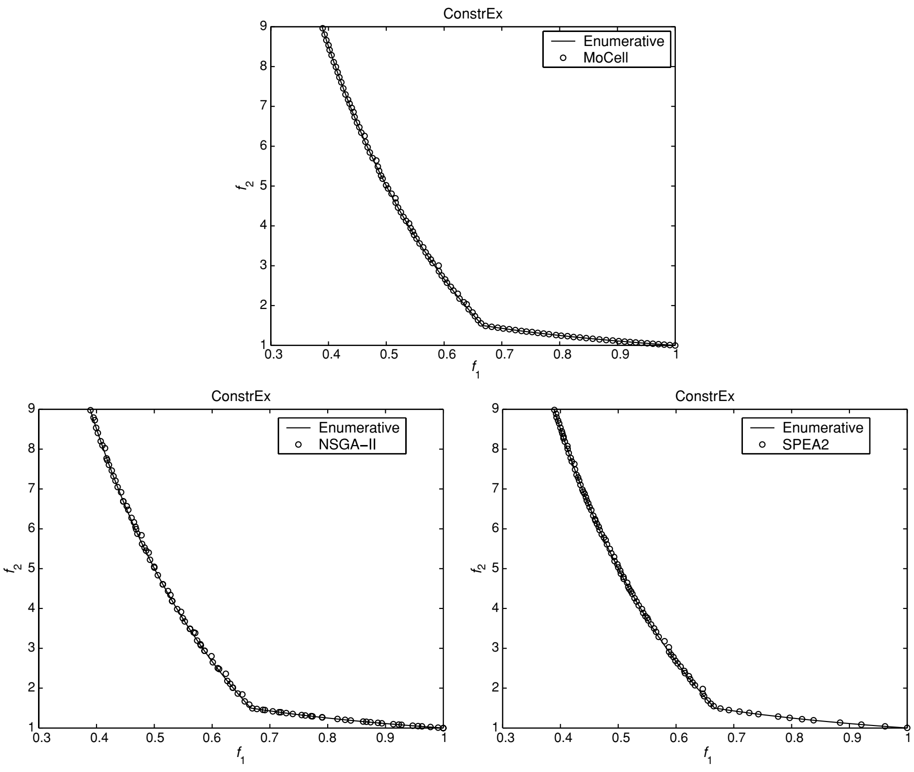

**Figure 5.** MOCell reaches a better convergence and spread of solutions than NSGA-II and SPEA2 for ConstrEx.

better distributes nondominated solutions along the Pareto fronts than NSGA-II, which is able to reach the best $\Delta$ values only in ConstrEx and Osyczka2.

Finally, in terms of the Hypervolume ($HV$) indicator (Table VIII) MOCell obtains better results than NSGA-II and SPEA2. For this quality measure, MOCell yields the best (largest) values for $HV$ in 11 out of the 21 MOPs (with statistical confidence in all the cases), whereas NSGA-II and SPEA2 obtain the best ones in 5 and 3 problems, respectively. MOCell performs specially well on constrained MOPs and the WFG family, as it is shown in the pairwise comparisons of Table XI (first and second rows). The ZDT family of MOPs is better addressed by NSGA-II and SPEA2.

To graphically show our results, we plot in Figures 4–6 fronts obtained by MOCell, NSGA-II, and SPEA2, together with the true Pareto set, for problems Kursawe, ConstrEx, and WFG4. The selected fronts are those having the best diversity (lowest value of $\Delta$) among the 100 produced ones by each technique for each problem. We can observe in Figure 4 that MOCell obtains a better distribution than SPEA2 for Kursawe, and still more noticeable is the difference with respect to NSGA-II, where MOCell again obtains better results. Figure 5 points out that

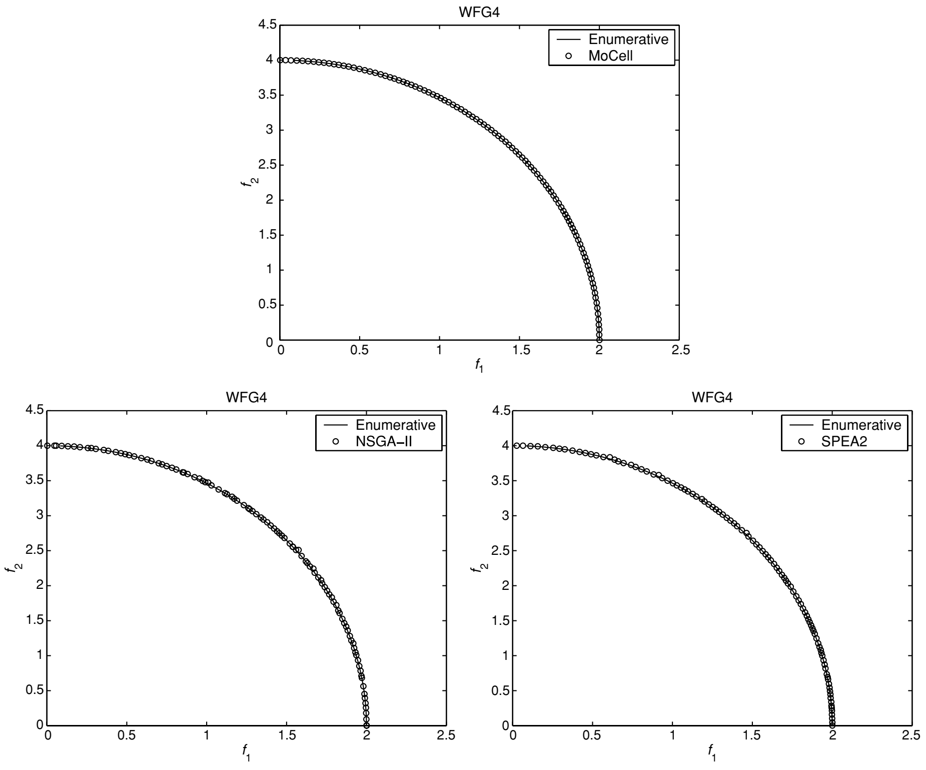

**Figure 6.** Problem WFG4 solved with the three algorithms.

the nondominated set of solutions generated by MOCell achieves an almost perfect spread out and convergence. Note that SPEA2 obtains a very good diversity for values of $f_1(\vec{x})$ lower than 0.66 (similar to the solution of MOCell), but it finds only around 12 solutions when $f_1(\vec{x}) \in [0.66, 1.0]$. Regarding NSGA-II, its front does not show this undesirable effect, but its worst diversity with respect to MOCell is very evident. The same conclusions are applicable to WFG4 problem (Figure 6), where MOCell achieves the best distribution again, followed by SPEA2, and finally NSGA-II.

We also want to remark that, concerning diversity, MOCell is the best out of the three analyzed algorithms by far: The differences in the spread values are in general noticeable with respect to the other compared algorithms.

## 5. CONCLUSIONS AND FUTURE WORK

In this 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 existing 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 to two state-of-the-art multiobjective optimization algorithms, NSGA-II and SPEA2; for that purpose, a wide benchmark of 21 test problems, including unconstrained and constrained ones, was chosen and three quality indicators were used to assess the performance of the algorithms. The results of the quality indicators reveal that MOCell is a highly competitive algorithm considering the convergence and hypervolume measures, and it clearly outperforms all the proposals on the considered test problems according to the diversity measure.

The evaluation of MOCell with problems with more than two objectives and its application to solve real-world problems are matter of future work. In addition it should be interesting to investigate some new innovative feedback techniques of individuals from the archive. Finally, it is possible to profit from other works in single-objective cellular GAs and applying to MOCell some existing algorithmic improvements so as to perform a better exploration/exploitation trade-off of the search space into the population.

## 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. Coello CA, Van Veldhuizen DA, Lamont GB. Evolutionary algorithms for solving multi-objective problems. Genetic algorithms and evolutionary computation. Boston, MA: Kluwer Academic Publishers; 2002.
2. Deb K. Multi-objective optimization using evolutionary algorithms. Hoboken, NJ: Wiley; 2001.
3. Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 2002;6(2):182–197.
4. Knowles J, Corne D. The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization. In: Congress on Evolutionary Computation (CEC). Piscataway, NJ: IEEE Press; 1999. pp 9–105.
5. Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH); 2001.
6. Alba E, Tomassini M. Parallelism and evolutionary algorithms. IEEE Trans Evol Comput 2002;6(5):443–462.
7. Cantú-Paz E. Efficient and accurate parallel genetic algorithms. Boston, MA: Kluwer Academic Publishers; 2000.
8. Manderick B, Spiessens P. Fine-grained parallel genetic algorithm. In: Proc. of the Third Int. Conf. on Genetic Algorithms (ICGA); 1989. pp 428–433.
9. Alba E, Dorronsoro B. The exploration/exploitation tradeoff in dynamic cellular evolutionary algorithms. IEEE Trans Evol Comput 2005;9(2):126–142.
10. Alba E, Dorronsoro B, Giacobini M, Tomasini M. Decentralized cellular evolutionary algorithms. In: Handbook of bioinspired algorithms and applications. Boca Raton, FL: CRC Press; 2006. pp 103–120.

11. Laumanns M, Rudolph G, Schwefel HP. A spatial predator-prey approach to multi-objective optimization: a preliminary study. In: PPSN V; 1998. pp 241–249.
12. Murata T, Gen M. Cellular genetic algorithm for multi-objective optimization. In: Proc. of the 4th Asian Fuzzy System Symposium; 2002. pp 538–542.
13. Kirley M. MEA: A metapopulation evolutionary algorithm for multiobjective optimisation problems. In: Congress on Evolutionary Computation (CEC). Piscataway, NJ: IEEE Press; 2001. pp 949–956.
14. Alba E, Dorronsoro B, Luna F, Nebro AJ, Bouvry P, Hogie L. A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan MANETs. Comput Commun 2007;30(4):685–697.
15. Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: empirical results. IEEE Trans Evol Comput 2000;8(2):173–195.
16. Huband S, Barone L, While RL, and Hingston P. A scalable multiobjective test problem toolkit. In: Coello CA, Hernández A, Zitler E, editors. In: Third International Conference on Evolutionary MultiCriterion Optimization (EMO), volume 3410 of Lecture Notes in Computer Science. Berlin: Springer; 2005. pp 280–295.
17. Sarma J, De Jong KA. An analysis of the effect of the neighborhood size and shape on local selection algorithms. In: Voigt HM, Ebeling W, Rechenberg I, Schwefel HP, editors. Parallel problem solving from nature, volume 1141 of LNCS. Berlin: Springer-Verlag; 1996. pp 236–244.
18. Durillo JJ, Nebro AJ, Luna F, Dorronsoro B, Alba E. jMetal: A java framework for developing multi-objective optimization metaheuristics. Technical Report ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos; 2006.
19. Schaffer JD. Multiple objective optimization with vector evaluated genetic algorithms. In: Grefensttete JJ, editor. In: Proc. of the First International Conference on Genetic Algorithms (ICGA), Hillsdale, NJ; 1987. pp 93–100.
20. Fonseca CM, Flemming PJ. Multiobjective optimization and multiple constraint handling with evolutionary algorithms, Part II: Application example. IEEE Trans Syst Man Cyberne 1998;28:38–47.
21. Kursawe F. A variant of evolution strategies for vector optimization. In: Schwefel HP, Männer R, editors. Parallel problem solving for nature, volume 496 of Lecture Notes in Computer Science. Berlin: Springer-Verlag; 1990. pp 193–197.
22. Osyczka A, Kundo S. A new method to solve generalized multicriteria optimization problems using a simple genetic algorithm. Struct Optim 1995;10:94–99.
23. Tanaka M, Watanabe H, Furukawa Y, Tanino T. GA-based decision support system for multicriteria optimization. In: Proc IEEE Int Conf on Systems, Man, and Cybernetics; 1995. Vol 2, pp 1556–1561.
24. Srinivas N, Deb K. Multiobjective function optimization using nondominatged sorting genetic algorithms. Evol Comput 1995;2(3):221–248.
25. Van Veldhuizen DA, Lamont GB. Multiobjective evolutionary algorithm research: a history and analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson, AFB, OH; 1998.
26. Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans Evol Comput 1999;3(4):257–271.
27. Deb K, Agrawal RB. Simulated binary crossover for continuous search space. Complex Syst 1995;9:115–148.
28. Hochberg Y, Tamhane AC. Multiple comparison procedures. Hoboken, NJ: Wiley; 1987.