A Study of Multi-Objective Metaheuristics when Solving Parameter Scalable Problems

J.J. Durillo, A.J. Nebro, C.A. Coello Coello, J. García-Nieto, F. Luna, E. Alba

***Abstract*—To evaluate the search capabilities of a multi-objective algorithm the usual approach is to choose a benchmark of known problems, to perform a fixed number of function evaluations, and to apply a set of quality indicators. However, while real problems could have hundreds or even thousands of decision variables, current benchmarks are normally adopted with relatively few decision variables (normally from ten to thirty). Furthermore, performing a constant number of evaluations does not provide information about the effort required by an algorithm to get a satisfactory set of solutions; this information would also be of interest in real scenarios, where evaluating the functions defining the problem can be computationally expensive. In this paper we study the effect of parameter scalability in a number of state-of-the-art multi-objective metaheuristics. We adopt a benchmark of parameter-wise scalable problems (the ZDT test suite) and analyze the behavior of eight multi-objective metaheuristics on the these test problems when using a number of decision variables that ranges from 8 up to 2048. By using the hypervolume indicator as a stopping condition, we also analyze the computational effort required by each algorithm in order to reach the Pareto front. We conclude that the algorithms based on particle swarm optimization and differential evolution yield the best overall results.**

I. Introduction

Many real-world optimization problems require the optimization of more than one objective function at the same time. These problems are called Multi-objective Optimization Problems (MOPs). In constrast to single-objective optimization problems, the solution to MOPs is not a single solution, but a set of *non-dominated* solutions called the *Pareto optimal set*. A solution that belongs to this set is said to be a *Pareto optimum* and, when the solutions of this set are plotted in objective space, they are collectively known as the *Pareto front*. Obtaining the Pareto front is the main goal in multi-objective optimization.

The fact that real-world MOPs tend to be nonlinear and with objective functions that are very expensive to evaluate has led to the use of *metaheuristics* [1], [3], [13] to deal with them. Metaheuristics are a family of techniques comprising *Evolutionary Algorithms* (EAs), *Particle Swarm Optimization* (PSO), *Ant Colony Optimization*, *Tabu Search*, *Differential Evolution*, *Scatter Search*, and many others. The most popular algorithms for multi-objective optimization based on metaheuristics in current use (NSGA-II [8] and SPEA2 [35]) adopt EAs as their search engine [5], [7].

The performance of these algorithms has been typically assessed using benchmark problems, such as the Zitzler-Deb-Thiele (ZDT) test problems [34], the Deb-Thiele-Laumanns-Zitzler (DTLZ) test problems [9], and the Walking-Fish-Group (WFG) test problems [14], [15]. These three problem families are scalable in the number of decision variables, and the last two are also scalable in the number of objectives. The methodology commonly adopted in the specialized literature is to compare several algorithms using a fixed (pre-defined) number of objective function evaluations and then to evaluate the values of different quality indicators (e.g., *generational distance* [32] or *hypervolume* [36], among others).

The main motivation of this work is that many real-world problems have hundreds or even thousands of decision variables, which constrasts with the current practice of validating multi-objective metaheuristics using the aforementioned benchmarks, but with a low number of decision variables (normally, no more than 30). Thus, the studies currently available do not consider the capability of current multi-objective evolutionary algorithms to properly scale when dealing with a very large number of decision variables. Scalability is a key issue in optimization algorithms, but it has been rarely addressed in the multi-objective domain. An example is the study presented in [30], in which the so-called intelligent multi-objective evolutionary algorithm (IMOEA) is compared to several elitist and non-elitist multi-objective evolutionary algorithms (MOEAs) in several of the ZDT test problems, adopting 63 decision variables. This comparative study was undertaken to validate the hypothesis of the authors of IMOEA, who argued the capability of such approach to deal with large parameter MOPs. Another example can be found in [33], where a study using only the ZDT1 problem with up to 100 variables is included. This contrasts with the interest in studying function scalability, which is currently an active research topic, leading to the area known as *many-objective optimization* [2], [16], [10], [26], [25], [28].

Another interesting issue that has been scarcely covered in the specialized literature is the analysis of the behavior of a multi-objective metaheuristic reaching the Pareto front of a problem. Typically, a fixed number of evaluations (and,

J.J Durillo, A.J. Nebro, J.García Nieto, F. Luna and E. Alba are with the Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, Spain (e-mail: {durillo,antonio,jnieto,flv,eat}@lcc.uma.es)

C.A. Coello Coello is with the Department of Computer Science, CINVESTAV-IPN, Mexico (e-mail: {ccoello@cs.cinvestav.mx). He is also affiliated to the UMI-LAFMIA 3175 CNRS.

This work has been partially funded by the "Consejería de Innovación, Ciencia y Empresa", Junta de Andalucía under contract P07-TIC-03044 DIRICOM project, `http://diricom.lcc.uma.es`. Juan J. Durillo is supported by grant AP-2006-03349 from the Spanish goverment.

in consequence, of iterations) is defined by the user, and the performance of the different algorithms studied is compared. However, this sort of comparison only measures the front aspect, and it does not provide any indication regarding the computational effort that a given algorithm requires to reach the true Pareto front of a problem, i.e., the efficiency of the algorithm. We believe that this is an important issue, because if we take into account that evaluating the objective functions of a MOP can be very time-consuming, it becomes of interest to know how expensive for a certain algorithm it is to generate the Pareto front.

We carried a first analysis of these ideas in [11], where six state-of-the-art multi-objective metaheuristics were compared when solving the ZDT benchmark, considering their formulation ranging from 8 up to 2048 variables. The algorithms included in the study reported in [11] were three genetic algorithms (GAs) (NSGA-II [8], SPEA2 [35], and PESA-II [6]), one evolution strategy (PAES [17]), one PSO (OMOPSO [27]), and one cellular GA (MOCell [23]). In that work, the number of evaluations required to provide a satisfactory solution was also analyzed. Given that the Pareto fronts of the ZDT problems are known, an algorithm was considered successful when the hypervolume of its current population (or archive, depending on the algorithm) was higher than the 95% of the hypervolume of the Pareto front.

The current paper conducts further research along this line. Compared to our previous work in [11], the main contributions of this paper are the following:

- Two additional modern multi-objective metaheuristics have been included, GDE3 [19] (a Differential Evolution algorithm) and AbYSS [22] (a Scatter Search algorithm), leading to a total of eight multi-objective metaheuristics, representative of the state-of-the-art.
- We analyze the search capabilities of the algorithms when solving the scalable ZDT problems using a stronger stopping condition, by which the algorithms stop either when they find a solution set having a hypervolume higher than the 98% of the hypervolume of the true Pareto front or when they have performed 10,000,000 function evaluations (500,000 in [11]).
- A more complete statistical analysis is performed, including pair-wise comparisons among the techniques in order to determine the significance of the results.
- We study the behavior of the most promising techniques in order to propose mechanisms that enhance their search capabilities.

The remainder of this paper is organized as follows. Section II includes basic background on multi-objective optimization. Sections III and IV describe, respectively, the problems and the metaheuristics we have used. Section V is devoted to the presentation and analysis of the experiments carried out. In Section VI, we include a discussion about the obtained results. Finally, Section VII summarizes the paper and discusses possible lines of future work.

II. Multi-objective optimization

In this section, we include some background on multi-objective optimization. More specifically, 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 multi-objective optimization problem (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* Ω 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 smaller 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 multi-objective 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 true Pareto front, as well as being 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, and a uniform spread of the solutions means that we have made a good exploration of the search space and no regions are left unexplored.

III. Scalable Parameter-wise Multi-objective Optimization Problems

To carry out our study, it would be helpful to use problems which are scalable in terms of the number of decision variables while keeping an invariable Pareto front. The ZDT test function family [34] fulfills this requirement. It offers, furthermore, a group of problems with different properties:

TABLE I
ZDT TEST FUNCTIONS

Problem	Objective functions	Variable bounds	Comments
ZDT1	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x})[1-\sqrt{x_1/g(\vec{x})}]$ $g(\vec{x}) = 1+9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$0 \leq x_i \leq 1$	convex
ZDT2	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x})\left[1-(x_1/g(\vec{x}))^2\right]$ $g(\vec{x}) = 1+9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$0 \leq x_i \leq 1$	non-convex
ZDT3	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x})\left[1-\sqrt{\frac{x_1}{g(\vec{x})}}-\frac{x_1}{g(\vec{x})}\sin(10\pi x_1)\right]$ $g(\vec{x}) = 1+9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$0 \leq x_i \leq 1$	convex disconnected
ZDT4	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x})[1-(x_1/g(\vec{x}))^2]$ $g(\vec{x}) = 1+10(n-1)+$ $\sum_{i=2}^{n}[x_i^2-10\cos(4\pi x_i)]$	$0 \leq x_1 \leq 1$ $-5 \leq x_i \leq 5$ $i=2,\ldots,n$	non-convex multi-frontal
ZDT6	$f_1(\vec{x}) = 1-e^{-4x_1}\sin^6(6\pi x_1)$ $f_2(\vec{x}) = g(\vec{x})[1-(f_1(\vec{x})/g(\vec{x}))^2]$ $g(\vec{x}) = 1+9[(\sum_{i=2}^{n} x_i)/(n-1)]^{0.25}$	$0 \leq x_i \leq 1$	non convex many-to-one non uniformly spaced

convex, non-convex, disconnected, multi-frontal, many-to-one problems (see Table I). These problems have been widely used in many studies in the field since they were first formulated. Table I shows the formulation of the ZDT test problem family. We omitted problem ZDT5 because it is binary encoded. The Pareto front of each problem is plotted in Fig. 1.

Since we are interested in studying the behavior of the algorithms when solving scalable parameter-wise problems, we have evaluated each ZDT problem with 8, 16, 32, 64, 128, 256, 512, 1024, and 2048 variables. This way, we can study not only what techniques behave more efficiently when solving problems having many variables, but also if their search capabilities remain constant or not when the number of decision variables increases.

IV. Multi-objective optimization algorithms

In this section, we briefly describe the eight metaheuristics that we have considered in this study. We have used the implementation of these algorithms provided by jMetal [12], a Java-based framework for developing metaheuristics to solve multi-objective optimization problems.[1]

The NSGA-II algorithm was proposed by Deb *et al.* [8]. It is a genetic algorithm based on obtaining a new population from the original one by applying the typical genetic operators (selection, crossover, and mutation); then, the individuals in the two populations are sorted according to their rank, and the best solutions are chosen to create a new population. In case of having to select some individuals with the same rank, a density estimation based on measuring the crowding distance to the surrounding individuals belonging to the same rank is used to get the most promising solutions.

SPEA2 was proposed by Zitler *et al.* in [35]. In this algorithm, each individual has a fitness value that is the sum of its strength raw fitness plus a density estimation. The algorithm applies the selection, crossover, and mutation operators to fill an archive of individuals; then, the non-dominated individuals of both the original population and the archive are copied into a new population. If the number of non-dominated individuals is greater than the population size, a truncation operator based on calculating the distances to the *k*-th nearest neighbor is used. This way, the individuals having the minimum distance to any other individual are chosen.

PESA-II [6] uses an internal population from which parents are selected to create new solutions, and an external population in which the non-dominated solutions found are stored. This external population uses the same hyper-grid division of phenotype (i.e., objective funcion) space adopted by PAES [18] to maintain diversity in which region-based selection is adopted. In region-based selection, the unit of selection is a hyperbox rather than an individual. The procedure lies in selecting (using any of the traditional selection techniques) a hyperbox and then randomly choosing an individual within that hyperbox.

PAES is a metaheuristic proposed by Knowles and Corne [18]. The algorithm is based on a simple (1+1) evolution strategy. To find diverse solutions in the Pareto optimal set, PAES uses an external archive of nondominated solutions, which is also used to decide about the new candidate solutions. An adaptive grid is used as a density estimator in the archive. We have used a real coded version of PAES, applying a polynomial mutation operator.

OMOPSO (Optimized MOPSO) is a particle swarm optimization algorithm for solving MOPs [27]. Its main features include the use of the crowding distance from NSGA-II to filter out leader solutions and the use of mutation operators to accelerate the convergence of the swarm. The original OMOPSO algorithm makes use of the concept of ϵ-dominance to limit the number of solutions produced by the algorithm. We consider here a variant discarding the use of ϵ-dominance, and considering the leader population as the result yielded by the algorithm.

GDE3 [19] is an improved version of the Generalized Differential Evolution (GDE) algorithm [20]. It starts with a population of random solutions, which becomes the current population. At each generation, an offspring population is created using the differential evolution operators; then, the

[1] jMetal is freely available for download at the following URL: `http://jmetal.sourceforge.net/`.

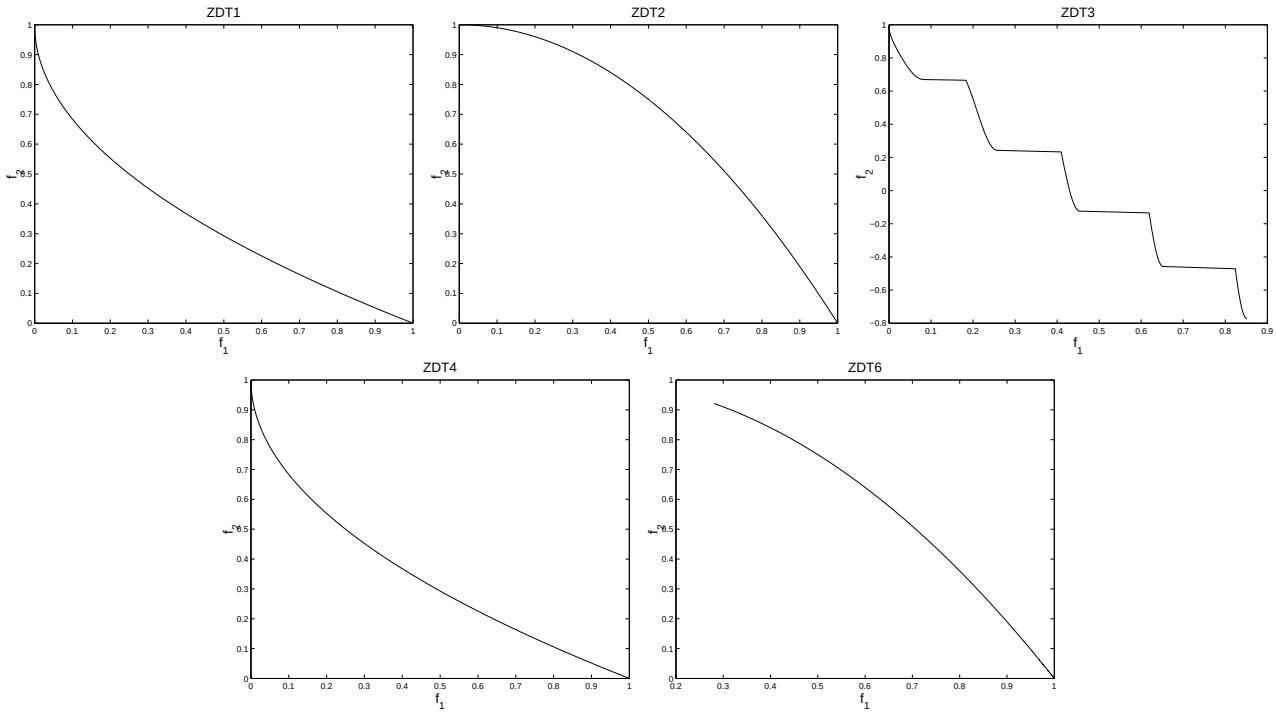

Fig. 1. Pareto fronts of the ZDT test functions.

current population for the next generation is updated using the solutions of both, the offspring and the current population. Before proceeding to the next generation, the size of the population is reduced using non-dominated sorting and a pruning technique aimed at diversity preservation, in a similar way as NSGA-II, although the pruning used in GDE3 modifies the crowding distance of NSGA-II in order to solve some of its drawbacks when dealing with problems having more than two objectives.

MOCell [23] is a cellular genetic algorithm (cGA). Like many multi-objective metaheuristics, it includes an external archive to store the non-dominated solutions found so far. This archive is bounded and uses the crowding distance of NSGA-II to keep diversity in the Pareto Front. We have used here an asynchronous version of MOCell, called aMOCell4 in [24], in which the cells are explored sequentially (asynchronously). The selection is based on taking an individual from the neighborhood of the current solution (called *cell* in cGAs) and another one randomly chosen from the archive. After applying the genetic crossover and mutation operators, the new offspring is compared with the current one, replacing it if better; if both solutions are non-dominated, the worst individual in the neighborhood is replaced by the current one. In these two cases, the new individual is inserted into the archive.

AbYSS is an adaptation of the *scatter search* metaheuristic to the multi-objective domain [22]. It uses an external archive similar to the one employed by MOCell. The algorithm incorporates operators from the evolutionary algorithms domain, including polynomial mutation and simulated binary crossover in the improvement and solution combination methods, respectively.

V. Experimentation

In this section, we describe the parameter settings used in the experiments, as well as the methodology we have followed in the tests, and the results we have obtained.

A. Parameterization

We have chosen a set of parameter values such that we allow a fair comparison among all the algorithms compared. All the GAs (NSGA-II, SPEA2, PESA-II, and MOCell) as well as GDE3, use an internal population size equal to 100; the size of the archive is also 100 in PAES, OMOPSO, GDE3, MOCell, and AbYSS. OMOPSO has been configured with 100 particles. For AbYSS, the population and the reference set have a size of 20 solutions.

In the GAs we have used simulated binary crossover (SBX) and polynomial mutation [7]. The distribution indices for both operators are $\eta_c = 20$ and $\eta_m = 20$, respectively. The crossover probability is $p_c = 0.9$ and the mutation probability is $p_m = 1/L$, where L is the number of decision variables. In PAES we have also adopted a polynomial mutation operator, with the same distribution index as indicated before. AbYSS uses polynomial mutation in the improvement method and SBX in the solution combination method. GDE3 uses 0.5 for the two parameters: CR and F [19]. OMOPSO applies a combination of uniform and non-uniform mutation to the particle swarm [27]. A detailed description of the parameter values adopted for our experiments is provided in Table II.

TABLE II
PARAMETERIZATION

Parameterization used in NSGA-II [8]	
Population Size	100 individuals
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
	(L = individual length)
Parameterization used in SPEA2 [35]	
Population Size	100 individuals
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
	(L = individual length)
Parameterization used in PESA-II [6]	
Population Size	100 individuals
Selection of Parents	region based selection + region based selection
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
	(L = individual length)
Archive Size	100 individuals
Parameterization used in PAES [17]	
Mutation	polynomial, $p_m = 1.0/L$
	(L = individual length)
Archive Size	100
Parameterization used in OMOPSO [27]	
Particles	100 particles
Mutation	uniform + non-uniform
Leaders Size	100
Parameterization used in GDE3 [19]	
Population Size	100 individuals
Recombination	Differential Evolution, $CR = 0.1$, $F = 0.5$
Parameterization used in MOCell [24]	
Population Size	100 individuals (10×10)
Neighborhood	1-hop neighbors (8 surrounding solutions)
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
	(L = individual length)
Archive Size	100 individuals
Parameterization used in AbYSS [22]	
Population Size	20 individuals
Reference Set Size	10 + 10
Recombination	simulated binary, $p_c = 1.0$
Mutation (local search)	polynomial, $p_m = 1.0/L$
	(L = individual length)
Archive Size	100 individuals

B. Methodology

We are interested in two main goals: analyzing the behavior of the algorithms when solving the scalable ZDT benchmark and their speed (efficiency) in reaching the Pareto front. Given that the Pareto fronts of the ZDT problems are known beforehand, a strategy could be to run the algorithms until they are able to produce them. However, it is possible that some of them never produce the true Pareto front, or simply take too long to do it. Thus, we adopt instead a stopping condition for all the algorithms compared, based on the *high quality* of the Pareto front produced. For that purpose, the hypervolume [36] quality indicator is adopted.

The hypervolume computes the volume (in objective function space) covered by members of a non-dominated set of solutions Q for problems in which all objectives are to be minimized. 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, the union of all hypercubes is found and its hypervolume (HV) is calculated:

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

Higher values of the HV performance measure imply more desirable solutions. A property of this quality indicator is that it measures both convergence to the Pareto front and diversity of the obtained fronts.

Once the quality indicator we are going to use has been described, we need to establish a stopping condition to be used in the execution of the algorithms. The idea is that the metaheuristics stop when they reach a certain percentage of the HV of the Pareto front, which ensures that the obtained front represents an accurate approximation to it. To decide about that percentage, we show different approximations of the Pareto front for the problem ZDT1 with different percentages of HV in Fig. 2. We can observe that a front with a hypervolume of 98.26% represents a reasonable approximation to the true Pareto fronts in terms of convergence and diversity of solutions. This same value has been corroborated using the other test problems from the ZDT suite. So, we have taken 98% of the HV of the Pareto front as a criterion to consider that a MOP has been successfully solved. Furthermore, those algorithms requiring fewer function evaluations to achieve this termination condition can be considered to be more efficient or *faster*. In those situations in which an algorithm is unable to obtain a front fulfilling this condition after the maximum number of function evaluations, we consider that it has failed in solving the problem; this way, we can obtain a hit rate for the algorithms, i.e., their percentage of successful executions. We set the maximum number of evaluations to ten million.

In our experiments, we check the stopping condition every 100 evaluations (that is, each iteration in the population-based metaheuristics), where we measure the hypervolume of the non-dominated solutions found so far. Therefore, in NSGA-II, SPEA2, and GDE3 we have considered the non-dominated solutions at each generation; in PESA-II, PAES, AbYSS, and MOCell, the external population and, in MOPSO, the leaders archive.

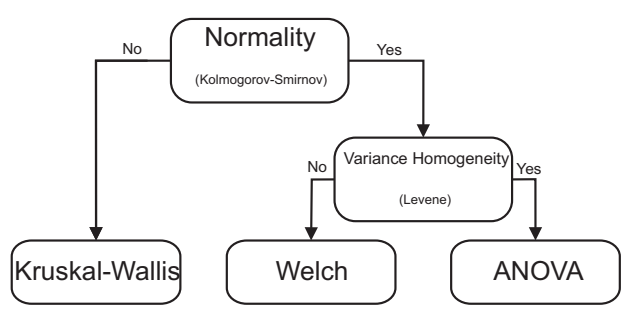

Fig. 3. Statistical analysis performed in this work.

We have executed 100 independent runs for each algo-

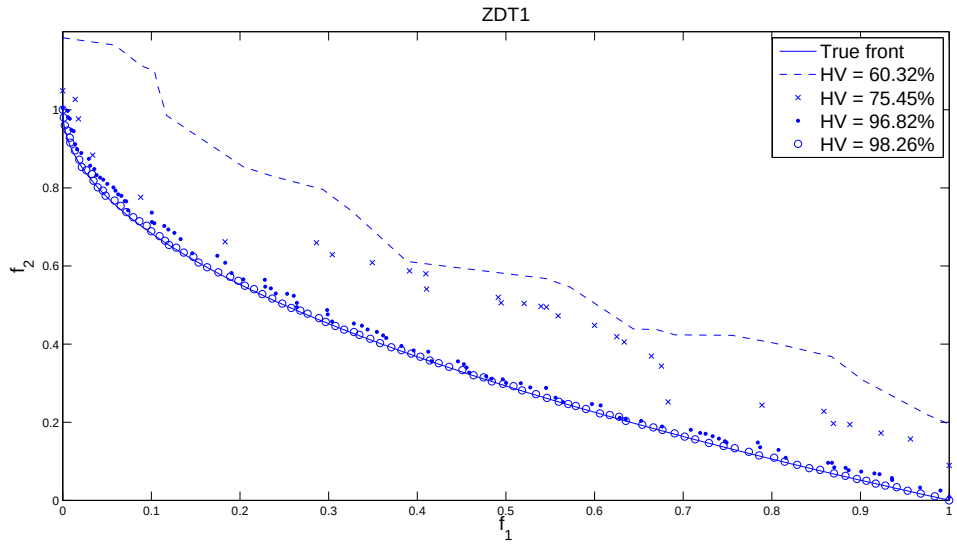

Fig. 2. Pareto fronts with different HV values obtained for problem ZDT1.

rithm and each problem instance. Since we are dealing with stochastic algorithms, we need to perform a statistical analysis of the obtained results to compare them with a certain level of confidence. Next, we describe the statistical test that we have carried out for assuring this [29]. First, a Kolmogorov-Smirnov test is performed in order to check whether the values of the results follow a normal (Gaussian) distribution or not. If so, the Levene test checks for the homogeneity of the variances. If samples have equal variance (positive Levene test), an ANOVA test is done; otherwise we perform a Welch test. For non-Gaussian distributions, the non-parametric Kruskal-Wallis test is used to compare the medians of the algorithms.

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 row in the tables containing the results (see Tables III to VII); conversely, "$-$" means that no statistical confidence was found (*p*-value > 0.05). For the sake of homogeneity in the presentation of the results, all the tables include the median, $\tilde{x}$, and the interquartile range, IQR, as measures of location (or central tendency) and statistical dispersion, respectively.

We have performed a post-hoc testing phase using the `multcompare` function provided by Matlab©, which allows for a multiple comparison of samples. This way, we can make pairwise comparison between algorithms to know about the significance of their results.

C. Analysis of results

Tables III, IV, V, VI, VII show the median and the interquartile range of the number of evaluations needed by the different optimizers for ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, respectively. This indicates that all the 100 independent runs have been successful, which means a hit rate of 1.0. When an optimizer is not able to reach an acceptable front upon performing 10,000,000 function evaluations in all the 100 independent runs, its cell in the tables includes the '$-$' symbol, and it is not taken into account in the statistical tests. In other words, the '$-$' symbol means that, in order to solve successfully the problem in all the independent runs, the optimizer may need more than 10,000,000 of function evaluations. In these cases, the hit rate would be less than 1.0. To ease the analysis of the results in these tables, the cells containing the lowest number of function evaluations have a grey colored background. There are two grey levels: the darker grey indicates the best (lowest) value, while lighter grey is used to point out the second best value. We can observe that the results in these tables are significant, as can be seen in the last row of each of them, where each cell contains a "+" symbol, except for ZDT4 with 1024 and 2048 variables (where there are no results to compare).

Next, we analyze the results obtained for each of the problems. To make the results clearer, we include a figure summarizing the values, using a logarithmic scale, in addition to the corresponding table. The discussion is organized in the following order: first, we analyze the success of the algorithms when solving the different instances of the problem; second, we analyze the speed of the techniques to obtain the Pareto front when solving the problems; and, finally, we make a pairwise comparison of the algorithms, considering those problems in which there are no statistical differences between each pair of techniques (if we included the problems with statistical significance, this would lead to larger tables).

- *ZDT1:* This problem presents a uniform density of solutions in the search space and a convex Pareto front. Table III and Fig. 4 show the number of evaluations needed to obtain a Pareto front with 98% of the HV of

TABLE III
EVALUATIONS FOR ZDT1

Algorithm	8 variables	16 variables	32 variables	64 variables	128 variables	256 variables	512 variables	1024 variables	2048 variables
NSGA-II	4.40e+3 $_{4.0e+2}$	8.10e+3 $_{5.0e+2}$	1.52e+4 $_{9.0e+2}$	2.88e+4 $_{1.4e+3}$	5.72e+4 $_{3.0e+3}$	1.21e+5 $_{5.2e+3}$	2.71e+5 $_{9.3e+3}$	6.31e+5 $_{1.6e+4}$	1.45e+6 $_{3.3e+4}$
SPEA2	5.00e+3 $_{3.0e+2}$	9.30e+3 $_{8.0e+2}$	1.69e+4 $_{1.1e+3}$	3.14e+4 $_{1.4e+3}$	6.03e+4 $_{2.7e+3}$	1.25e+5 $_{5.0e+3}$	2.69e+5 $_{9.4e+3}$	6.01e+5 $_{1.6e+4}$	1.34e+6 $_{2.8e+4}$
PESA-II	4.15e+3 $_{1.2e+3}$	8.40e+3 $_{9.5e+2}$	1.73e+4 $_{1.6e+3}$	3.74e+4 $_{2.8e+3}$	8.36e+4 $_{5.0e+3}$	1.96e+5 $_{9.3e+3}$	4.78e+5 $_{1.8e+4}$	1.18e+6 $_{4.0e+4}$	2.93e+6 $_{8.9e+4}$
PAES	3.40e+3 $_{2.5e+3}$	7.25e+3 $_{4.4e+3}$	1.34e+4 $_{8.0e+3}$	2.57e+4 $_{1.6e+4}$	4.70e+4 $_{3.5e+4}$	9.07e+4 $_{3.9e+4}$	1.73e+5 $_{1.3e+5}$	3.68e+5 $_{4.7e+5}$	-
OMOPSO	1.40e+3 $_{4.0e+2}$	3.40e+3 $_{9.0e+2}$	7.40e+3 $_{1.8e+3}$	1.38e+4 $_{2.8e+3}$	2.80e+4 $_{4.4e+3}$	6.31e+4 $_{7.4e+3}$	1.58e+5 $_{1.8e+4}$	4.55e+5 $_{3.4e+4}$	1.41e+6 $_{1.0e+5}$
GDE3	2.80e+3 $_{2.0e+2}$	5.30e+3 $_{3.0e+2}$	1.00e+4 $_{4.0e+2}$	1.81e+4 $_{4.5e+2}$	3.30e+4 $_{9.0e+2}$	6.10e+4 $_{1.2e+3}$	1.16e+5 $_{1.8e+3}$	2.40e+5 $_{3.0e+3}$	5.64e+5 $_{7.0e+3}$
MOCell	1.80e+3 $_{3.0e+2}$	3.80e+3 $_{6.0e+2}$	9.20e+3 $_{8.0e+2}$	2.18e+4 $_{1.8e+3}$	4.92e+4 $_{4.0e+3}$	1.13e+5 $_{6.4e+3}$	2.56e+5 $_{8.6e+3}$	5.62e+5 $_{1.6e+4}$	1.20e+6 $_{2.7e+4}$
AbYSS	3.40e+3 $_{7.0e+2}$	6.80e+3 $_{9.0e+2}$	1.49e+4 $_{1.8e+3}$	3.30e+4 $_{2.7e+3}$	7.29e+4 $_{6.1e+3}$	1.58e+5 $_{7.8e+3}$	3.50e+5 $_{1.2e+4}$	7.06e+5 $_{2.1e+4}$	1.39e+6 $_{2.6e+4}$
	+	+	+	+	+	+	+	+	-

TABLE IV
EVALUATIONS FOR ZDT2

Algorithm	8 variables	16 variables	32 variables	64 variables	128 variables	256 variables	512 variables	1024 variables	2048 variables
NSGA-II	7.50e+3 $_{6.0e+2}$	1.37e+4 $_{8.0e+2}$	2.60e+4 $_{1.6e+3}$	4.98e+4 $_{3.2e+3}$	1.01e+5 $_{4.8e+3}$	2.08e+5 $_{1.0e+4}$	4.54e+5 $_{1.8e+4}$	1.01e+6 $_{2.6e+4}$	2.28e+6 $_{4.5e+4}$
SPEA2	7.80e+3 $_{1.0e+3}$	1.46e+4 $_{1.3e+3}$	2.60e+4 $_{1.7e+3}$	4.86e+4 $_{3.2e+3}$	9.50e+4 $_{4.4e+3}$	1.90e+5 $_{6.0e+3}$	3.95e+5 $_{9.0e+3}$	8.51e+5 $_{1.7e+4}$	1.89e+6 $_{2.6e+4}$
PESA-II	1.50e+4 $_{7.8e+3}$	3.51e+4 $_{2.5e+4}$	9.38e+4 $_{6.0e+4}$	3.98e+5 $_{4.2e+5}$	-	-	-	-	-
PAES	3.14e+4 $_{4.5e+4}$	6.94e+4 $_{7.1e+4}$	1.29e+5 $_{1.2e+5}$	2.93e+5 $_{4.2e+5}$	-	-	-	-	-
OMOPSO	1.75e+3 $_{4.0e+2}$	3.90e+3 $_{1.6e+3}$	9.70e+3 $_{3.8e+3}$	1.68e+4 $_{5.0e+3}$	3.06e+4 $_{5.3e+3}$	5.44e+4 $_{8.4e+3}$	1.11e+5 $_{1.1e+4}$	2.83e+5 $_{2.7e+4}$	7.71e+5 $_{7.9e+4}$
GDE3	3.20e+3 $_{2.0e+2}$	6.10e+3 $_{4.5e+2}$	1.18e+4 $_{6.5e+2}$	2.26e+4 $_{1.2e+3}$	4.33e+4 $_{1.8e+3}$	8.18e+4 $_{1.3e+3}$	1.58e+5 $_{2.1e+3}$	3.27e+5 $_{4.2e+3}$	7.66e+5 $_{9.8e+3}$
MOCell	2.90e+3 $_{1.0e+3}$	4.90e+3 $_{1.4e+3}$	8.25e+3 $_{2.2e+3}$	1.74e+4 $_{1.1e+4}$	4.42e+4 $_{2.5e+4}$	1.26e+5 $_{4.3e+4}$	2.91e+5 $_{1.1e+4}$	6.60e+5 $_{1.1e+4}$	1.46e+6 $_{2.0e+4}$
AbYSS	4.50e+3 $_{7.5e+2}$	9.10e+3 $_{1.7e+3}$	1.86e+4 $_{2.5e+3}$	3.96e+4 $_{3.5e+3}$	8.26e+4 $_{7.2e+3}$	1.75e+5 $_{8.6e+3}$	3.68e+5 $_{9.6e+3}$	7.74e+5 $_{1.9e+4}$	1.60e+6 $_{2.4e+4}$
	+	+	+	+	+	+	+	+	+

TABLE V
EVALUATIONS FOR ZDT3

Algorithm	8 variables	16 variables	32 variables	64 variables	128 variables	256 variables	512 variables	1024 variables	2048 variables
NSGA-II	4.20e+3 $_{5.0e+2}$	7.40e+3 $_{6.0e+2}$	1.36e+4 $_{9.5e+2}$	2.53e+4 $_{1.4e+3}$	5.06e+4 $_{3.1e+3}$	1.08e+5 $_{4.8e+3}$	2.38e+5 $_{9.0e+3}$	5.37e+5 $_{1.6e+4}$	1.20e+6 $_{2.5e+4}$
SPEA2	4.90e+3 $_{6.0e+2}$	9.10e+3 $_{7.0e+2}$	1.62e+4 $_{1.0e+3}$	3.00e+4 $_{2.0e+3}$	5.89e+4 $_{4.0e+3}$	1.21e+5 $_{5.8e+3}$	2.56e+5 $_{9.4e+3}$	5.58e+5 $_{1.6e+4}$	1.22e+6 $_{2.5e+4}$
PESA-II	3.75e+3 $_{9.0e+2}$	7.60e+3 $_{1.0e+3}$	1.59e+4 $_{2.0e+3}$	3.43e+4 $_{3.8e+3}$	7.76e+4 $_{7.8e+3}$	1.77e+5 $_{8.0e+3}$	4.14e+5 $_{1.5e+4}$	9.83e+5 $_{3.3e+4}$	2.28e+6 $_{4.5e+4}$
PAES	6.90e+3 $_{8.8e+3}$	1.21e+4 $_{1.2e+4}$	2.56e+4 $_{3.2e+4}$	5.68e+4 $_{6.1e+4}$	9.92e+4 $_{1.2e+5}$	2.03e+5 $_{2.0e+5}$	3.65e+5 $_{6.3e+5}$	8.17e+5 $_{8.4e+5}$	-
OMOPSO	2.60e+3 $_{1.0e+3}$	5.40e+3 $_{2.4e+3}$	1.01e+4 $_{3.3e+3}$	2.20e+4 $_{4.0e+3}$	5.06e+4 $_{8.1e+3}$	1.26e+5 $_{2.3e+4}$	3.53e+5 $_{3.9e+4}$	1.03e+6 $_{8.3e+4}$	3.17e+6 $_{1.9e+5}$
GDE3	2.90e+3 $_{2.5e+2}$	5.60e+3 $_{3.0e+2}$	1.08e+4 $_{4.0e+2}$	1.96e+4 $_{8.0e+2}$	3.46e+4 $_{9.5e+2}$	6.29e+4 $_{1.3e+3}$	1.20e+5 $_{1.8e+3}$	2.50e+5 $_{2.8e+3}$	6.07e+5 $_{7.2e+3}$
MOCell	1.90e+3 $_{6.0e+2}$	4.20e+3 $_{8.0e+2}$	9.90e+3 $_{1.4e+3}$	2.30e+4 $_{2.2e+3}$	5.24e+4 $_{3.4e+3}$	1.16e+5 $_{9.3e+3}$	2.57e+5 $_{1.7e+4}$	5.44e+5 $_{1.6e+4}$	1.15e+6 $_{3.6e+4}$
AbYSS	3.35e+3 $_{1.6e+3}$	6.75e+3 $_{1.8e+3}$	1.40e+4 $_{2.1e+3}$	2.87e+4 $_{3.8e+3}$	6.02e+4 $_{6.1e+3}$	1.23e+5 $_{1.6e+4}$	2.57e+5 $_{2.0e+4}$	5.47e+5 $_{4.4e+4}$	1.12e+6 $_{4.5e+4}$
	+	+	+	+	+	+	+	+	+

TABLE VI
EVALUATIONS FOR ZDT4

Algorithm	8 variables	16 variables	32 variables	64 variables	128 variables	256 variables	512 variables	1024 variables	2048 variables
NSGA-II	1.62e+4 $_{3.4e+3}$	4.38e+4 $_{1.2e+4}$	1.37e+5 $_{3.3e+4}$	4.25e+5 $_{9.1e+4}$	1.20e+6 $_{1.8e+5}$	3.29e+6 $_{3.5e+5}$	8.85e+6 $_{6.7e+5}$	-	-
SPEA2	2.11e+4 $_{4.2e+3}$	4.54e+4 $_{8.7e+3}$	1.34e+5 $_{3.1e+4}$	3.69e+5 $_{6.4e+4}$	1.07e+6 $_{1.3e+5}$	2.98e+6 $_{2.4e+5}$	8.24e+6 $_{5.9e+5}$	-	-
PESA-II	1.84e+4 $_{5.2e+3}$	5.00e+4 $_{1.4e+4}$	1.51e+5 $_{3.2e+4}$	4.12e+5 $_{7.3e+4}$	1.06e+6 $_{1.3e+5}$	2.72e+6 $_{2.5e+5}$	6.69e+6 $_{3.8e+5}$	-	-
PAES	3.08e+4 $_{1.1e+4}$	8.53e+4 $_{4.1e+4}$	2.17e+5 $_{6.4e+4}$	5.41e+5 $_{1.9e+5}$	1.28e+6 $_{3.0e+5}$	3.18e+6 $_{7.2e+5}$	-	-	-
OMOPSO	-	-	-	-	-	-	-	-	-
GDE3	1.18e+4 $_{8.0e+2}$	-	-	-	-	-	-	-	-
MOCell	8.80e+3 $_{2.2e+3}$	2.04e+4 $_{5.4e+3}$	5.86e+4 $_{1.0e+4}$	1.65e+5 $_{2.7e+4}$	4.67e+5 $_{5.8e+4}$	1.23e+6 $_{1.0e+5}$	3.25e+6 $_{1.6e+5}$	8.14e+6 $_{2.8e+5}$	-
AbYSS	1.46e+4 $_{5.4e+3}$	5.46e+4 $_{2.1e+4}$	1.61e+5 $_{4.3e+4}$	4.82e+5 $_{1.0e+5}$	1.39e+6 $_{1.5e+5}$	3.81e+6 $_{4.2e+5}$	-	-	-
	+	+	+	-	+	+	+	-	-

TABLE VII
EVALUATIONS FOR ZDT6

Algorithm	8 variables	16 variables	32 variables	64 variables	128 variables	256 variables	512 variables	1024 variables	2048 variables
NSGA-II	2.31e+4 $_{1.3e+3}$	4.60e+4 $_{1.4e+3}$	8.84e+4 $_{2.8e+3}$	1.68e+5 $_{3.5e+3}$	3.24e+5 $_{6.4e+3}$	6.47e+5 $_{1.1e+4}$	1.33e+6 $_{1.6e+4}$	2.77e+6 $_{2.7e+4}$	5.80e+6 $_{4.6e+4}$
SPEA2	2.64e+4 $_{1.2e+3}$	5.26e+4 $_{2.0e+3}$	9.94e+4 $_{2.2e+3}$	1.87e+5 $_{4.2e+3}$	3.53e+5 $_{7.0e+3}$	6.84e+5 $_{1.1e+4}$	1.37e+6 $_{1.4e+4}$	2.81e+6 $_{2.3e+4}$	5.82e+6 $_{3.8e+4}$
PESA-II	2.16e+4 $_{1.8e+3}$	4.78e+4 $_{2.4e+3}$	9.85e+4 $_{4.7e+3}$	2.01e+5 $_{7.2e+3}$	4.10e+5 $_{1.1e+4}$	8.57e+5 $_{2.1e+4}$	1.83e+6 $_{2.9e+4}$	3.89e+6 $_{4.4e+4}$	8.32e+6 $_{8.9e+4}$
PAES	6.80e+3 $_{6.8e+3}$	1.62e+4 $_{1.6e+4}$	3.21e+4 $_{2.8e+4}$	-	-	-	-	-	-
OMOPSO	2.90e+3 $_{1.6e+3}$	4.20e+3 $_{1.9e+3}$	7.70e+3 $_{2.7e+3}$	1.38e+4 $_{3.5e+3}$	2.88e+4 $_{5.2e+3}$	6.10e+4 $_{1.0e+4}$	1.26e+5 $_{1.6e+4}$	2.76e+5 $_{3.0e+4}$	6.12e+5 $_{6.6e+4}$
GDE3	3.70e+3 $_{5.0e+2}$	6.60e+3 $_{5.0e+2}$	1.32e+4 $_{1.1e+3}$	3.14e+4 $_{2.6e+3}$	1.53e+5 $_{6.7e+3}$	3.21e+5 $_{3.8e+3}$	6.36e+5 $_{4.0e+3}$	1.35e+6 $_{7.5e+3}$	3.27e+6 $_{1.8e+4}$
MOCell	1.07e+4 $_{1.0e+3}$	2.58e+4 $_{1.4e+3}$	5.65e+4 $_{2.6e+3}$	1.19e+5 $_{3.0e+3}$	2.47e+5 $_{4.7e+3}$	5.18e+5 $_{7.8e+3}$	1.10e+6 $_{9.4e+3}$	2.35e+6 $_{1.9e+4}$	5.00e+6 $_{3.4e+4}$
AbYSS	1.23e+4 $_{1.1e+3}$	2.57e+4 $_{1.8e+3}$	5.26e+4 $_{3.0e+3}$	1.08e+5 $_{3.8e+3}$	2.26e+5 $_{7.2e+3}$	4.67e+5 $_{9.2e+3}$	9.60e+5 $_{1.4e+4}$	1.96e+6 $_{2.1e+4}$	4.00e+6 $_{4.1e+4}$
	+	+	+	+	+	+	+	+	+

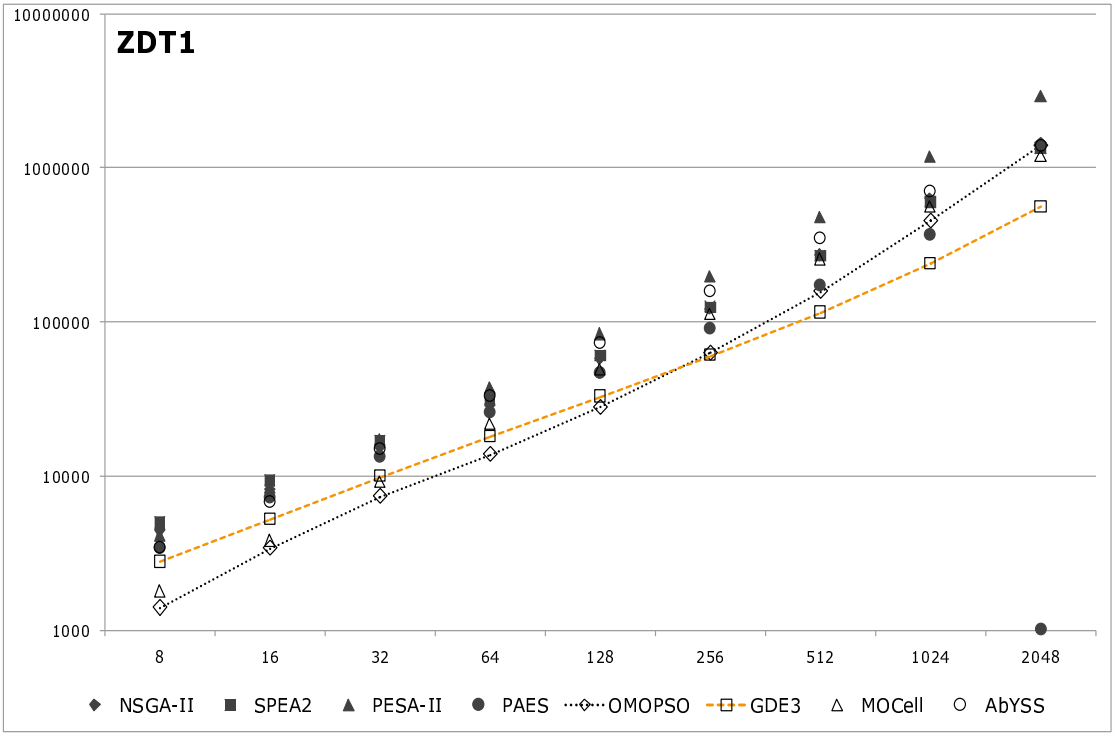

Fig. 4. Number of evaluations when solving ZDT1.

TABLE VIII
HIT RATE FOR ZDT1

Algorithm/Variables.	8	16	32	64	128	256	512	1024	2048
NSGA-II	√	√	√	√	√	√	√	√	√
SPEA2	√	√	√	√	√	√	√	√	√
PESA-II	√	√	√	√	√	√	√	√	√
PAES	√	√	√	√	√	√	√	√	0.95
OMOPSO	√	√	√	√	√	√	√	√	√
GDE3	√	√	√	√	√	√	√	√	√
MOCell	√	√	√	√	√	√	√	√	√
AbYSS	√	√	√	√	√	√	√	√	√

the Pareto front for this problem. Table VIII presents the hit rate indicator, where we have used the symbol $\surd$ to indicate a value of 1.0 (a 100% success rate). Table IX contains the problems for which no statistical differences have been found between each pair of algorithms.

We begin by analyzing which algorithms are able to solve the problems in all the independent runs carried out. The results in Table III show that all the algorithms have success in all the instances, except for PAES in the instance with 2048 variables. This is corroborated considering the hit rate (see Table VIII), where all the algorithms have a hit rate of 1.0 except for PAES in the 2048 instance, which has 0.95. This means that in five out of the 100 executions carried out for this instance, PAES reached the maximum number of evaluations before obtaining a Pareto front with the desired HV value.

We pay attention now to the speed, i.e., the number of function evaluations needed by the metaheuristics to find a Pareto front according to our success condition. In Fig. 4, we plot the results using a logarithmic scale. We have connected by a line the symbols of the two algorithms yielding the best values. Thus, we can observe that OMOPSO (dotted line) is the fastest algorithm up to 128 variables, while GDE3 scales better from 256 to 2048 variables. The lines clearly depict that there is a tendency change in these two algorithms at 256 variables, indicating that GDE3 tends to be faster than the other techniques as the number of decision variables increases, while OMOPSO exhibits the opposite behavior. This suggests that GDE3 could be the most appropriate algorithm to solve ZDT1 with more than 2048 variables. Accordingly, we determine that GDE3 is the algorithm that scales the best in problem ZDT1. Considering the rest of techniques, MOCell is the second fastest algorithm up to 32 variables and in the case of 2048 variables. NSGA-II, SPEA2, and AbYSS tend to need a similar number of evaluations when the instances are larger.

TABLE IX
NON-SUCCESSFUL STATISTICAL TESTS OF THE NUMBER OF EVALUATIONS FOR ZDT1

NSGA-II	128, 256, 512, 1024	8, 16	16, 32, 64, 128	2048	-	256	32
SPEA2		16, 32	2048	-	-	-	64
PESA-II			8	-	-	-	128
PAES				512, 1024	-	128, 256, 512, 1024	8, 16, 32
OMOPSO					128, 256	8, 16, 32	2048
GDE3						32	-
MOCell							-
	SPEA2	PESA-II	PAES	OMOPSO	GDE3	MOCell	AbYSS

Finally, we make an analysis considering the outcome of the statistical tests included in Table IX. If we focus on OMOPSO and GDE3, the tests are non-successful only in the instances of 128 and 256 variables, which does not affect the previous analysis. It is interesting to note that PAES, the simplest of the evaluated techniques, presents no differences in many instances compared to NSGA-II and MOCell. We can also observe that NSGA-II and SPEA2 do not present statistical confidence in four cases.

- *ZDT2*: This problem also presents a uniform density of solutions in the search space, but it has a non-convex Pareto front. The number of evaluations needed to solve the ZDT2 problem are included in Table IV and Fig. 5. Table X shows the hit rate of each algorithm. The problems in which each pair of algorithms are statistically independent appear in Table XI.
  We start by commenting that some optimizers have difficulties when solving this problem, as can be seen in Table IV and Fig. 5. In particular, neither PAES nor PESA-II are able to reach a hit rate of 1.0 in instances with more than 64 variables. These results indicate that problems having a non-convex Pareto front can cause difficulties for some algorithms.
  Consequently, the hit rate indicator shows that six algorithms have a 100% success rate in all the instances. Among the algorithms with a hit rate smaller than 1.0, PESA-II is the algorithm having the worst value, 0.49 in the problem with 128 variables, and 0.0 in the next ones. PAES does not achieve a 100% of success after 128 variables but, in constrast to PESA-II, the values are near 100% in the instances ranging from 128 to 1024 variables.
  Let us examine now the speed of the algorithms. A look at Fig. 5 reveals that OMOPSO and GDE3 are the fastest algorithms. The lines connecting the values of these solvers indicate that OMOPSO requires a lower number of function evaluations than GDE3 to get the desired Pareto fronts in all but in the largest instance. In fact, when observing the lines we can see that the one of GDE3 suggests again that it could scale better than the other techniques in order to solve instances of more than 2048 variables. MOCell, AbYSS, SPEA2, and NSGA-II, in this order, are the following metaheuristics in terms of speed, although they tend to get close to each other when the number of decision variables of the problem increases.
  The pairwise tests in Table XI reveal some interesting facts. On the one hand, the results of GDE3 and OMOPSO in the two largest instances are non-significant; on the other hand, the tests show the differences between MOCell and OMOPSO up to 64 variables and between MOCell and GDE3 up to 512 variables are also non-significant.
- *ZDT3*: This problem presents a uniform density of solutions in the search space and its Pareto front is composed of several discontinuous regions; therefore, the main complexity is to find all these discontinuous regions. The evaluations required for solving the different instances are shown in Table V and Fig. 6. Table XII presents the hit rate of the algorithms. The problems in which there are not statistical differences between each pair of algorithms are shown in Table XIII.
  Proceeding as before, we start by analyzing which algorithms are able to solve successfully all the instances of the problem. Therefore, at Table V we see that all the algorithms solve all the instances, except for PAES, which fails in the largest one.
  Concerning the speed of the algorithms, in Table V and Fig. 6 we can see which algorithms need a lower number of function evaluations to reach the target results. We have drawn a dotted line joining the points of GDE3, which shows clearly that this optimizer is the best when solving this problem, considering scalability. The dashed line joins the number of evaluations required by NSGA-II, which is the second fastest algorithm in ZDT3 from 128 to 1024 variables. We can see that NSGA-II, MOCell, and SPEA2 are very close, and we observe in Fig. 6 that they tend to similar values when the number of decision variables of the problem is higher. OMOPSO appears as the worst algorithm with 1024 and 2048 variables, so it scales as well as other algorithms in this problem.
  The pairwise tests in Table XIII do not alter the previous discussion. All the differences between GDE3 and NSGA-II have statistical confidence. Considering MOCell and NSGA-II, the tests are non-successful in only two instances (128 and 1024). We note that SPEA2 does not provide statistical differences with respect to PAES and AbYSS in five instances.
- *ZDT4*: This problem has a total of 100 different Pareto optimal fronts of which only one is global. Consequently, the main difficulty for the different algorithms is to reach this global front. The number of evaluations

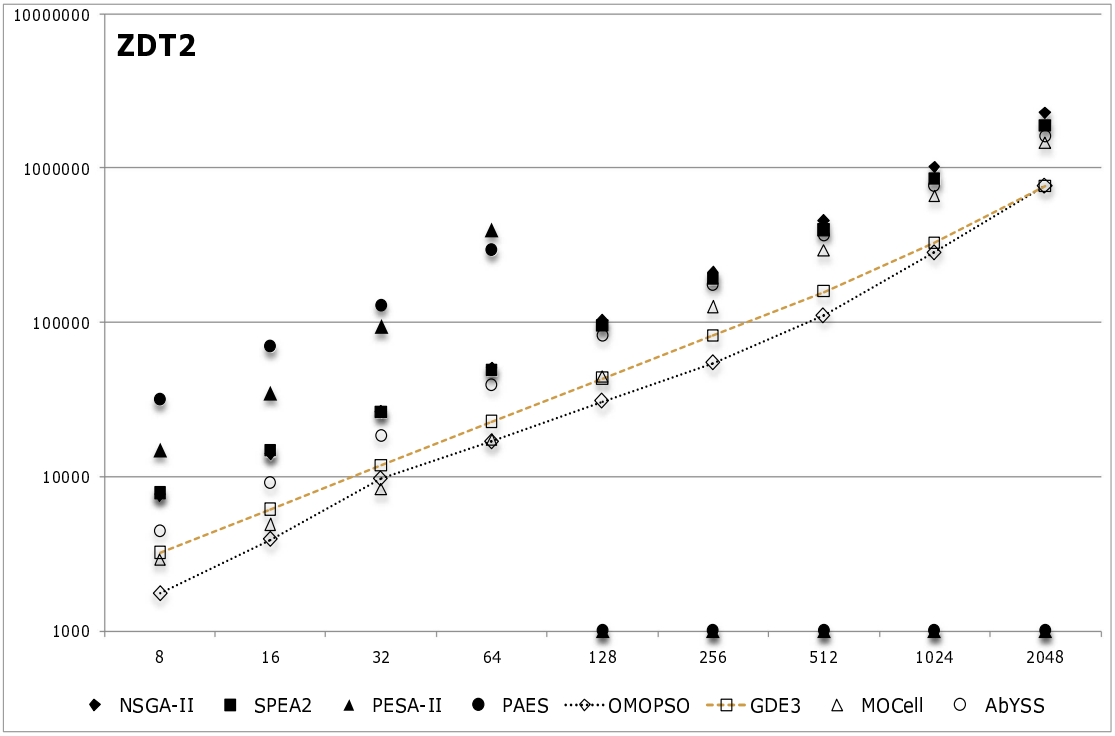

Fig. 5. Number of evaluations when solving ZDT2.

TABLE X
HIT RATE FOR ZDT2

Algorithm/Variables	8	16	32	64	128	256	512	1024	2048
NSGA-II	√	√	√	√	√	√	√	√	√
SPEA2	√	√	√	√	√	√	√	√	√
PESA-II	√	√	√	√	0.43	0.0	0.0	0.0	0.0
PAES	√	√	√	√	0.99	0.98	0.99	0.96	0.79
OMOPSO	√	√	√	√	√	√	√	√	√
GDE3	√	√	√	√	√	√	√	√	√
MOCell	√	√	√	√	√	√	√	√	√
AbYSS	√	√	√	√	√	√	√	√	√

TABLE XI
NON-SUCCESSFUL STATISTICAL TESTS OF THE NUMBER OF EVALUATIONS FOR ZDT2

NSGA-II	8, 16, 32, 64, 128	-	-	-	-	-	
SPEA2		-	-	-	-	-	128, 256, 512
PESA-II			8, 16, 32, 64	-	-	-	-
PAES				-	-	-	-
OMOPSO					32, 1024, 2048	16, 32, 64	-
GDE3						8, 16, 64, 128, 256, 512	-
MOCell							-
	SPEA2	PESA-II	PAES	OMOPSO	GDE3	MOCell	AbYSS

required by the different algorithms is shown in Table VI and Fig. 7. The hit rate indicator is presented in Table XIV. The problems in which there are no statistical differences in each pair of algorithms are shown in Table XV.

The fact that the problem has many sub-optimal Pareto fronts represents a great challenge for the different algorithms, as can be concluded from seeing Table VI and Fig. 7. On the one hand, none of them is able to solve the problem successfully with 2048 variables. Only MOCell needs a number of evaluations lower than the maximum established for 1024 variables. On the other hand, OMOPSO and GDE3, which are among the most effective algorithms for the problems analyzed until now, cannot solve successfully any of the instances (except for the one with 8 variables, in the case of GDE3).

Looking at the hit rate of the different algorithms (Table XIV), the only solver able to reach sometimes the Pareto optimal front in all the instances is AbYSS, the scatter search algorithm. The rest of the algorithms have a hit rate of 0 with 1024 and 2048 (with the exception

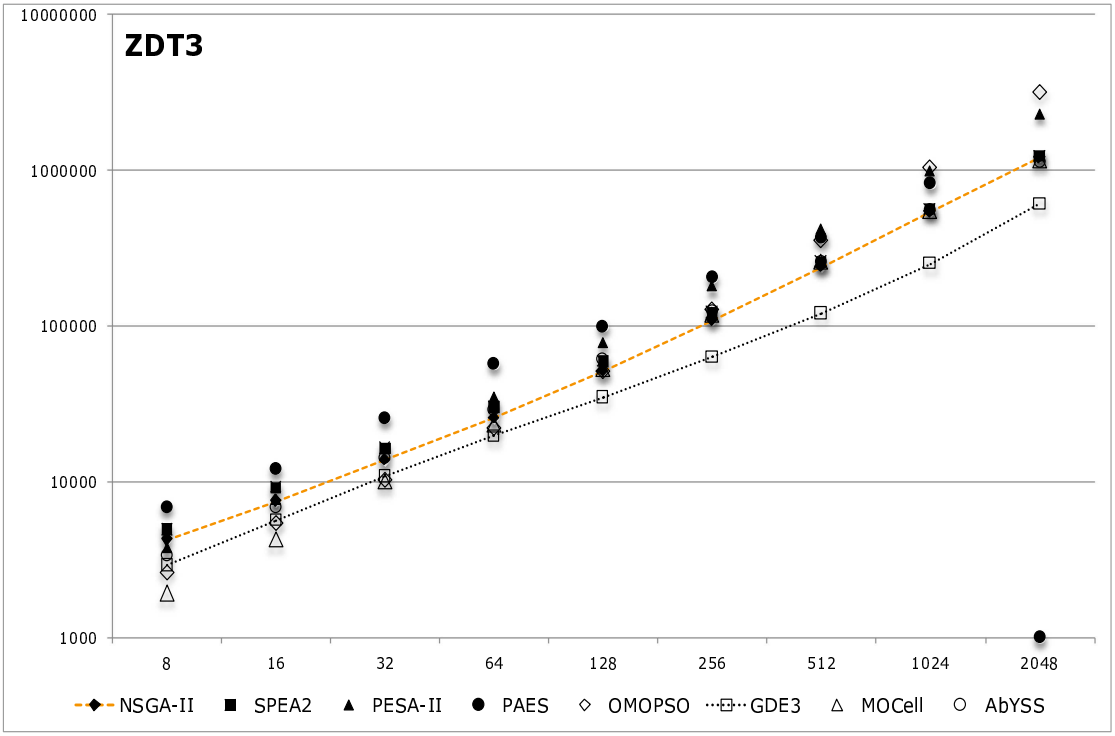

Fig. 6. Number of evaluations when solving ZDT3.

TABLE XII
EVALUATIONS ZDT3

Algorithm/Variables	8	16	32	64	128	256	512	1024	2048
NSGA-II	√	√	√	√	√	√	√	√	√
SPEA2	√	√	√	√	√	√	√	√	√
PESA-II	√	√	√	√	√	√	√	√	√
PAES	√	√	√	√	√	√	√	√	0.88
OMOPSO	√	√	√	√	√	√	√	√	√
GDE3	√	√	√	√	√	√	√	√	√
MOCell	√	√	√	√	√	√	√	√	√
AbYSS	√	√	√	√	√	√	√	√	√

TABLE XIII
NON-SUCCESSFUL STATISTICAL TESTS OF THE NUMBER OF EVALUATIONS FOR ZDT3

NSGA-II	2048	8, 16	-	128	-	128, 1024	8, 16, 32, 1024
SPEA2		32	8, 16, 32, 64, 1024	256	-	256, 512, 1024	64,128, 256, 512, 1024
PESA-II			32, 64	1024	-	-	8,16
PAES				512	-	-	-
OMOPSO					8, 16, 32	8, 32, 64, 128	256
GDE3						32	-
MOCell							512, 1024, 2048
	SPEA2	PESA-II	PAES	OMOPSO	GDE3	MOCell	AbYSS

of MOCell with 1024 variables).

Analyzing the speed of the algorithms, we see in Table VI that MOCell needs less than half of the function evaluations than the next fastest solvers, PESA-II and NSGA-II.

The tests included in Table XV show the instances of the problem where the differences are non-significant. However, they do not change the main conclusion about ZDT4, which is that, given that MOCell needs more than eight million function evaluations to achieve the stopping condition in the 1024 variables instance, it is clear that none of the analyzed techniques scales well on this problem.

- *ZDT6*: This problem presents a non-convex Pareto front, in which the density of solutions across the Pareto-optimal region is not uniform. Table VII and Fig. 8 show the number of evaluations required for reaching a front with the target HV value. Table XVI presents the hit rate of the algorithms. In Table XVII, the problems in which there are no statistical differences for each pair of algorithms appear.

  As before, we start analyzing which algorithms are able

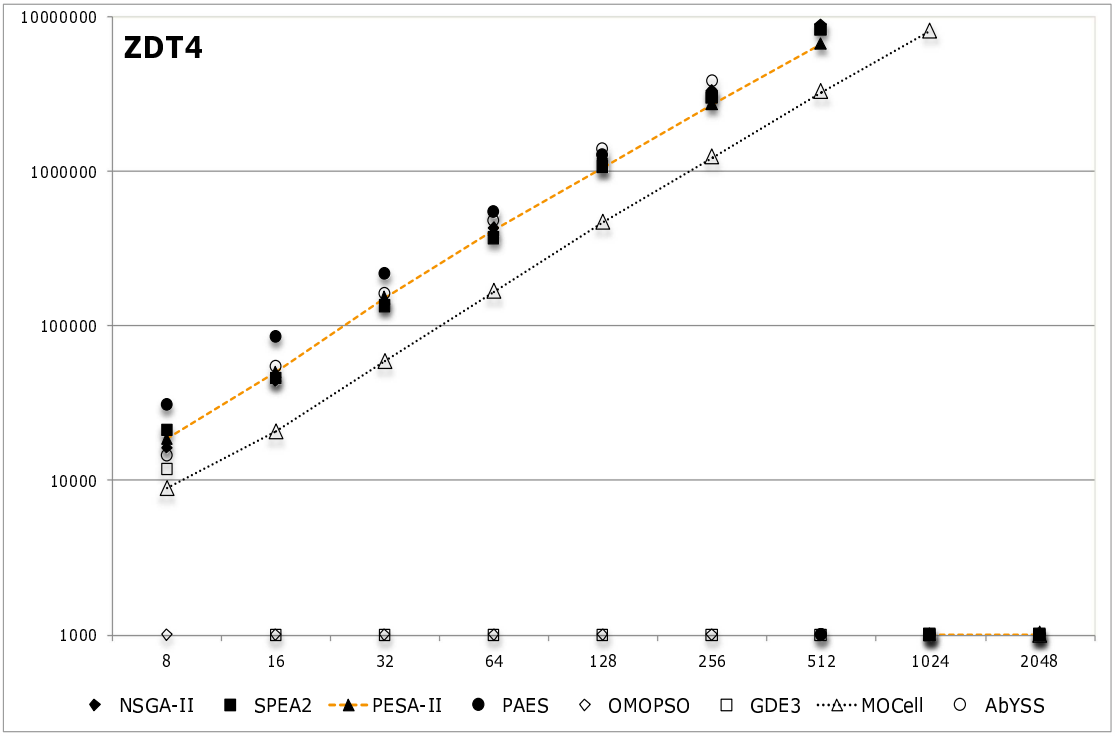

Fig. 7. Number of evaluations when solving ZDT4.

TABLE XIV
HIT RATE FOR ZDT4

Algorithm/Variables	8	16	32	64	128	256	512	1024	2048
NSGA-II	√	√	√	√	√	√	√	0.0	0.0
SPEA2	√	√	√	√	√	√	√	0.0	0.0
PESA-II	√	√	√	√	√	√	√	0.0	0.0
PAES	√	√	√	√	√	√	0.85	0.0	0.0
OMOPSO	0.66	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
GDE3	√	0.96	0.86	0.7	0.0	0.0	0.0	0.0	0.0
MOCell	√	√	√	√	√	√	√	√	0.0
AbYSS	√	√	√	√	√	√	0.40	0.16	0.22

TABLE XV
NON-SUCCESSFUL STATISTICAL TESTS OF THE NUMBER OF EVALUATIONS FOR ZDT4

NSGA-II	16, 32, 64, 512	8, 16, 32, 64	128, 256, 512	-	-	-	8
SPEA2		8, 16, 32, 64, 128	-	-	32	-	16
PESA-II			16	-	-	16	16, 32
PAES				-	-	-	64, 128
OMOPSO					-	-	512
GDE3						-	512
MOCell							-
	SPEA2	PESA-II	PAES	OMOPSO	GDE3	MOCell	AbYSS

to achieve the desired results in the different instances of the problem. Looking at Fig. 8 and Table VII we see that all the optimizers, except for PAES, are able to solve the problem in all cases. Anyway, if we have a look at the hit rate indicator (Table XVI), we can observe that PAES achieves values near 100% in the instances ranging from 64 to 2048 variables.

We focus now on the speed of the algorithms. We can clearly see in Table VII and Fig. 8 that the fastest algorithm is OMOPSO, followed by GDE3. The lines joining the values of these algorithms in Fig. 8 indicate that both of them scale well and they would probably successfully solve larger instances. In fact, this conclusion could be applied to all the algorithms but PAES.

If we analyze the tests included in Table XVII, the most remarkable fact is that the differences between OMOPSO and GDE3 that are non-significant affect only the smallest instances (8, 16, and 32 variables).

VI. DISCUSSION OF RESULTS

In this section, we analyze the results globally, trying to identify the strengths and weaknesses of the algorithms when

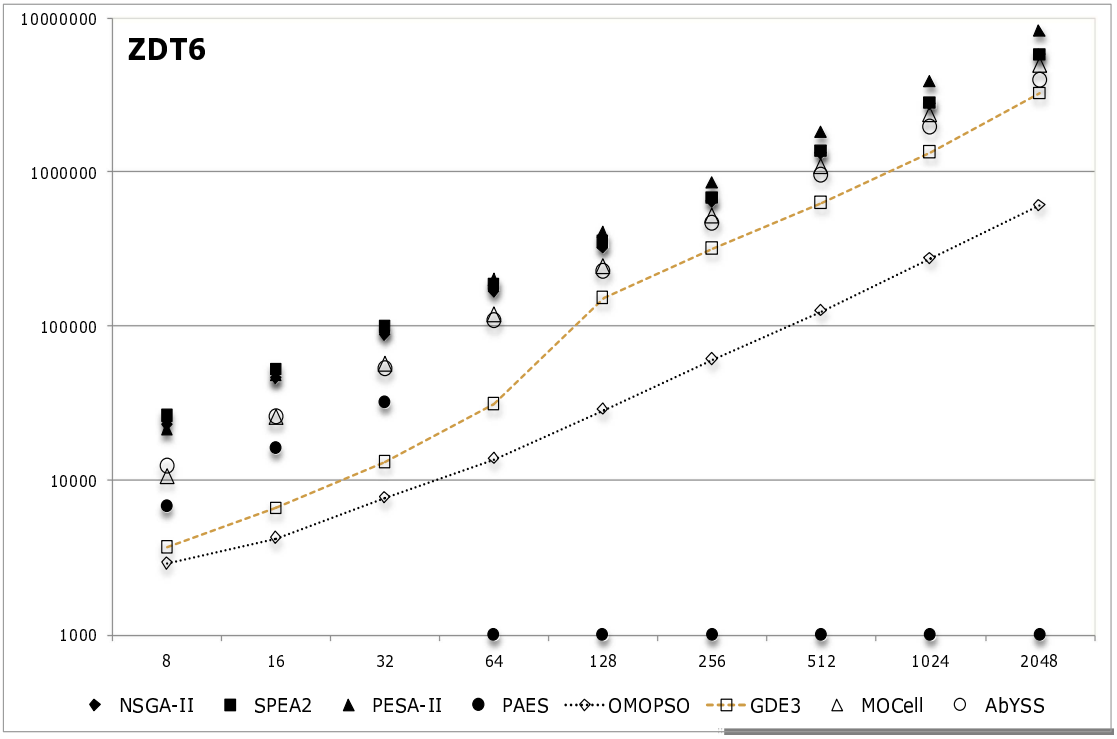

Fig. 8. Number of evaluations when solving ZDT6.

TABLE XVI
HIT RATE FOR ZDT6

Algorithm/Variables	8	16	32	64	128	256	512	1024	2048
NSGA-II	√	√	√	√	√	√	√	√	√
SPEA2	√	√	√	√	√	√	√	√	√
PESA-II	√	√	√	√	√	√	√	√	√
PAES	√	√	√	0.99	0.99	0.98	0.98	0.99	0.94
OMOPSO	√	√	√	√	√	√	√	√	√
GDE3	√	√	√	√	√	√	√	√	√
MOCell	√	√	√	√	√	√	√	√	√
AbYSS	√	√	√	√	√	√	√	√	√

TABLE XVII
NON-SUCCESSFUL STATISTICAL TESTS OF THE NUMBER OF EVALUATIONS FOR ZDT6

NSGA-II	1024, 2048	8, 16	-	-	-	-	-
SPEA2		32, 64	-	-	-	-	-
PESA-II			-	-	-	-	-
PAES				-	-	-	-
OMOPSO					8, 16, 32	-	-
GDE3						-	-
MOCell							8, 16, 32
	SPEA2	PESA-II	PAES	OMOPSO	GDE3	MOCell	AbYSS

executing the full set of experiments. To facilitate this discussion, we have made first a rank of the algorithms according to their scalability and speed. Second, we analyze the poor behavior of OMOSPO and GDE3 when solving problem ZDT4 and indicate ways of improving these algorithms. After that, we analyze the rest of the techniques. Finally, we compare the results of this study with our previous work [11].

A. Scalability and speed

The scalabity ranking is presented in Table XVIII. This ranking considers first those algorithms solving the problems with the highest number of decision variables. The ties are broken considering the number of evaluations in the most difficult instances. To make the discusion clearer, we have marked in boldface those optimizers having a hit rate lower than 1.0 in at least one experiment, which indicates that the algorithm does not scale well.

According to this ranking, GDE3 is the most impressive metaheuristic: it achieves three best and one second best ranks. However, given the difficulties of this algorithm when solving ZDT4, MOCell is the technique that appears as the most reliable, in the sense that it is able to solve all the

instances considered in this study but the largest one on ZDT4, and it occupies the first rank on ZDT4. OMOPSO scales the best in one problem (ZDT6), and it is the second best on ZDT2 (although recall that its results on ZDT2 and 2048 variables and those obtained by GDE3 have no statistical confidence). However, it is unable to solve ZDT4 and it tends to require more evaluations than other algorithms when solving the larger instances of ZDT1 and ZDT3. SPEA2, AbYSS, and NSGA-II are in the middle of the ranking: they never obtain the best result nor they are beyond the sixth position in the ranking. PESA-II is in the lower positions mainly because it does not scale well in ZDT2 and ZDT4. Finally, PAES is the last algorithm in the ranking because of its low hit-rate in many experiments.

TABLE XVIII
RANKING OF THE ALGORITHMS: SCALABILITY

ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
1. GDE3	1. GDE	1. GDE3	1. **MOCell**	1. OMOPSO
2. MOCell	2. OMOPSO	2. AbYSS	2. **PESA-II**	2. GDE3
3. SPEA2	3. MOCell	3. MOCell	3. **SPEA2**	3. AbYSS
4. AbYSS	4. AbYSS	4. NSGA-II	4. **NSGA-II**	4. MOCell
5. OMOPSO	5. SPEA2	5. SPEA2	5. **PAES**	5. NSGA-II
6. NSGA-II	6. NSGA-II	6. PESA-II	6. **AbYSS**	6. SPEA2
7. PESA-II	7. **PAES**	7. OMOPSO	7. **GDE3**	7. PESA-II
8. **PAES**	8. **PESA-II**	8. **PAES**	8. **OMOPSO**	8. **PAES**

The ordering in Table XIX relies on the algorithms requiring globally lower numbers of evaluations to find the target Pareto front, i.e., we sort them according to their speed. To make this ranking, we consider all the instances, not only the largest ones. Thus, for each problem we have sorted the evaluations of the algorithms when solving each of the instances, and the sum of the obtained positions determine the order of the techniques.

If we do not consider the ZDT4 problem, OMOPSO is globally the fastest algorithm: it requires the lowest number of evaluations in problems ZDT1, ZDT2, ZDT6, and it is the fourth one in the ranking of ZDT3. GDE3 is the second algorithm in the ranking, because it is first one in a problem, ZDT3, and the second one in ZDT1, ZDT2, and ZDT6. The next algorithms are MOCell (first rank in ZDT4, a second position, and two third ones), AbYSS, SPEA2 (the first GA in the speed ranking), and NSGA-II. Among the slowest metaheuristics we find again PESA-II and PAES.

An interesting fact is that, if we observe the two tables, the rankings are the same in problems ZDT2, ZDT4, and ZDT6. This suggests that when an algorithm scales well with a problem, it may require a low number of function evaluations to solve it.

TABLE XIX
RANKING OF THE ALGORITHMS: SPEED

ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
1. OMOPSO	1. OMOPSO	1. GDE3	1. MOCell	1. OMOPSO
1. GDE3	2. GDE3	2. MOCell	2. PESA-II	2. GDE3
3. MOCell	3. MOCell	3. NSGA-II	3. SPEA2	3. AbYSS
4. PAES	4. AbYSS	4. OMOPSO	4. NSGA-II	4. MOCell
5. SPEA2	5. SPEA2	5. AbYSS	5. AbYSS	5. NSGA-II
6. NSGA-II	6. NSGA-II	6. SPEA2	6. PAES	6. PESA-II
6. AbYSS	7. PESA-II	7. PESA-II	7. GDE3	7. SPEA2
8. PESA-II	8. PAES	8. PAES	8. OMOPSO	8. PAES

The ZDT benchmark is very well-known, but it is not fully representative in the sense that there are MOPs having features not covered by ZDT [14] (e.g., linear or degenerate geometry, variable linkage, etc.). This work is a first step in the study of the behavior of multi-objective metaheuristics when solving parameter scalable problems; an extension including other problem families, such as DTLZ [9] and WFG [15], would allow more general conclusions to be drawn. However, many researchers may find useful insights from the results obtained in this paper when facing the solution of MOPs having a large number of parameters. For example, the two reference algorithms in the field, NSGA-II and SPEA2, are expected to work well, but MOCell can scale better and provide results faster; and GDE3 and OMOPSO are techniques to consider if we are looking for efficiency, although there is a risk that they cannot solve the problem if this is a multi-frontal one.

B. OMOPSO and Multi-frontal problems

If we do not take into account problem ZDT4, one of the most outstanding algorithms in our study is OMOPSO. In this section, we analyze whether its inefficacy when dealing with ZDT4 is particular of that problem or it happens with multi-frontal problems in general.

To explore this issue, we have defined two multi-modal problems using the methodology described in [7]. In this paper, it is pointed out that given a function $g(\vec{x})$, a two-objective problem can be defined as the minimization of

$$\begin{aligned} f_1(x_1,\vec{x}) &= x_1 \\ f_2(x_1,\vec{x}) &= g(\vec{x})/x_1 \end{aligned}$$

This problem has a local or global Pareto-optimal solution $(x1,\vec{x})$, where $\vec{x}$ is the locally or globally minimum solution of $g(\vec{x})$, respectively, and x_1 can take any value.

TABLE XX
GRIEWANK AND ACKLEY MONO-OBJECTIVE FORMULATION

Problem	Functions	Variables
Griewank	$g(x) = 1 + \sum_1^p \frac{x_i^2}{400} - \prod_1^p \cos\frac{x_i}{\sqrt{i}}$	10
Ackley	$g(x) = 20 + e + -20exp(-0.2\sqrt{\frac{1}{p}\sum_1^p x_i^2})$	3

This way, given a mono-objective function with local optimal solutions, we can construct a multi-frontal bi-objective MOP. We have selected two well-known problems having local minimal solutions, Griewank and Ackley (see Table XX). The resulting problems have been solved by the eight metaheuristics we are dealing with. Our experiments revealed that OMOPSO does not converge to the corresponding Pareto fronts. To illustrate this fact, we include in Figs. 9 and 10 the fronts obtained by OMOPSO and NSGA-II.

In [11] we argued that the reason for this behavior could be related to an inbalance between (low) diversification and (high) intensification, given that OMOPSO appears to be a fast algorithm. Here we go deeper into this issue, in order to find the explanation and propose a solution. Let us recall that OMOPSO is a PSO-based MOEA, in which the

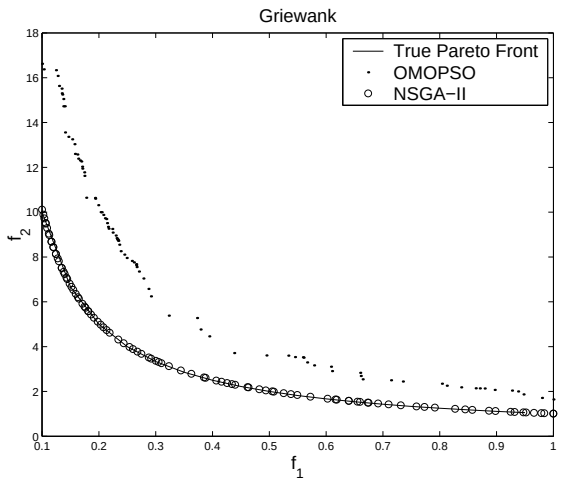

Fig. 9. OMOPSO solving Griewank's test problem.

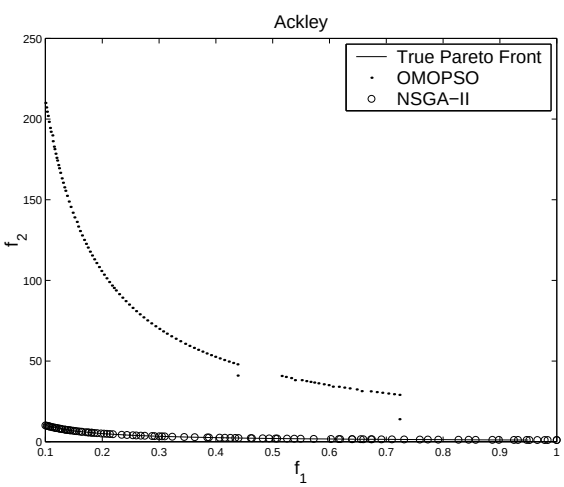

Fig. 10. OMOPSO solving Ackley's test problem.

potential solutions to the problem are called *particles* and the population of solutions is called the *swarm*. The way in which PSO updates particle x_i at generation t is by applying the following formula:

$$x_i(t) = x_i(t+1) + v_i(t) \tag{2}$$

where the factor $v_i(t)$ is known as velocity and it is given by

$$\begin{aligned} v_i(t) = w * v_i(t-1) \quad &+ \quad C1 * r1 * (x_{ibest} - x_i) \\ &+ \quad C2 * r2 * (x_{global} - x_i) \end{aligned} \tag{3}$$

In this formula, x_{ibest} is the best solution stored by x_i, x_{global} is the best particle that the entire swarm has viewed, w is the inertia weight of the particle and controls the trade-off between global and local experience, $r1$ and $r2$ are two uniformly distributed random numbers in the range $[0, 1]$, and $C1$ and $C2$ are specific parameters which control the effect of the personal and global best particles. If the resulting position of a particle is out of the limits of its allowable values, the approach taken in OMOPSO is to assign the limit value to the particle and to change the direction of the velocity.

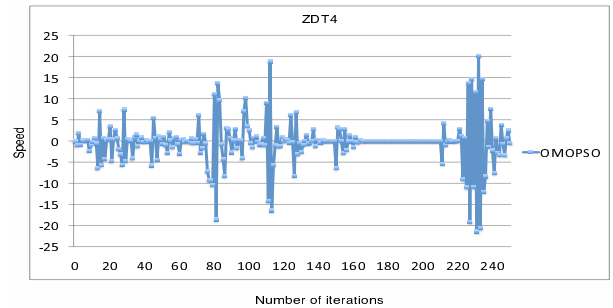

Fig. 11. OMOPSO: Velocity of one particle in the swarm.

We have monitored the velocity of the particle representing the second decision variable in ZDT4 (this variable takes values in the interval $[-5, +5]$, which provides a better illustration of the following analysis than using the first variable, which ranges in $[-1, +1]$). Fig. 11 depicts the velocity of OMOPSO in the 250 iterations. The x-axis represents the number of iterations. We can observe that the velocity values suffer a kind of erratic behavior, alternating very high with very low values, in some points of the execution. Let us note that the limits of the second variable in ZDT4 are $[-5, +5]$, and the velocity takes values higher than ± 20. Thus, as a consequence, this particle is moving to its extreme values continuously, so it is not contributing to guide the search.

To determine whether this is the reason making OMOPSO unable to solve multi frontal MOPs, we have modified it including a velocity constraint mechanism, similar to the one proposed in [4]. In addition, the accumulated velocity of each variable j (in each particle) is also bounded by means of the following equation:

$$v_{i,j}(t) = \begin{cases} delta_j & \text{if } v_{i,j}(t) > delta_j \\ -delta_j & \text{if } v_{i,j}(t) \leq -delta_j \\ v_{i,j}(t) & \text{otherwise} \end{cases} \tag{4}$$

where

$$delta_j = \frac{(upper_limit_j - lower_limit_j)}{2} \tag{5}$$

This way, we can ensure an effective new position calculation, and hence avoid erratic movements. We have called the resulting algorithm SMPSO (Speed-constrained Multi-objective PSO).

In Fig. 12, we show again the velocity of the particle representing the second parameter of ZDT4. We can observe that the erratic movements of the velocity have disappeared, so the particle has taken new values and thus it has explored different regions of the search space.

To evaluate the effect of the changes in SMPSO, we have used it to solve all the problems, following the same methodology. The results are included in Table XXI. To illustrate its search capabilities we have included in Fig. 13

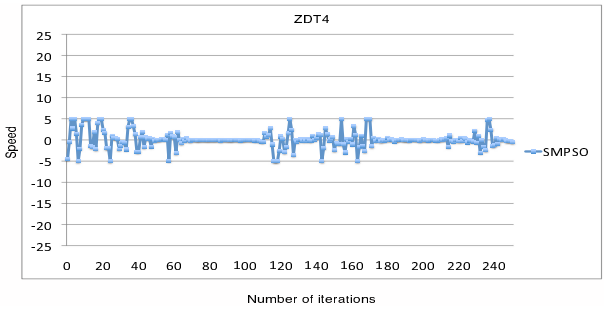

Fig. 12. SMPSO: Velocity of one particle in the swarm.

five pictures showing the values of SMPSO and OMOPSO, except for ZDT4, where the evaluations of MOCell are included instead of those of OMOPSO. Let us comment on this last case first. We can observe that the results of SMPSO on ZDT4 are surprisingly good: SMPSO is not only capable of solving ZDT4, but it scales up to 2048 variables, requiring a very low number of evaluations (which are several orders of magnitude lower than those required by MOCell). If we consider the other problems, we see that in general SMPSO is faster than OMOPSO up to 64 variables, but OMOPSO scales better. So, with SMPSO we have a PSO metaheuristic which is more robust than OMOPSO (in the sense that it can solve all the problems considered in this work) and faster when the problems have few decision variables, but at the price of not being able to scale as well as OMOPSO.

We would like to remark that we have not tried to find the best possible configuration for SMPSO, i.e., SMPSO has the same parameter settings as OMOPSO, except for those affecting the velocity constraint. Thus, there is clearly room for improvement here.

C. Improving GDE3

GDE3 shows similarities to OMOPSO in the sense that it is among the top techniques in four problems, but it fails in ZDT4. It is clear that the factors leading to the poor performance of GDE3 in ZDT4 are different to those affecting OMOPSO, especially because the search capabilities of a differential evolution metaheuristic depend on the values of the parameters CR and F. A deeper study that attempts to find the best values for these parameters is beyond the scope of this paper; instead, we take the approach of keeping CR and F unchanged and applying polynomial mutation to the new generated solutions, as is suggested in [21]. The obtained results are included in Table XXII.

We proceed now as for OMOPSO, showing a figure containing a comparison between GDE3 and its variant using mutation, except for ZDT4, where MOCell is included. The results indicate that GDE3 with mutation is able to solve ZDT4 instances of up to 128 variables, while in the rest of the problems it needs more evaluations than the original GDE3, although it can be observed in the figures that the two algorithms have a very similar behavior.

As in the case of PSO and OMOPSO/SMPSO, differential evolution appears to be a technique worth considering to solve problems like those considered in this work.

D. About the Rest of Algorithms

In the two previous sections we have analyzed OMOPSO and GDE3, aiming to improve their search capabilities. We discuss here the rest of the techniques adopted in our comparative study. Although they are different algorithms (NSGA-II, SPEA2, and PESA-II are GAs, MOCell is a cellular GA, PAES is an evolution strategy, and AbYSS is a scatter search approach), all of them share the use of polynomial mutation and, except for PAES, the SBX crossover operator. This means that the main way to modify the search capabilities of these metaheuristics is by modifying the distribution indices that govern these operators.

We have made a number of preliminary experiments with MOCell considering different values, ranging from 5.0 to 200.0, in both indexes, and we have not observed noticeable changes in the behavior of the technique. This, however, does not mean that there is no room for improvement; the use of different mutation and crossover operators may change the search capabilities of a GA.

E. Comparison with Previous Work

In this section we compare the results of this paper with those obtained in some of our previous work [11]. In that paper, the main conclusions were that, considering scalability, PAES was the most competitive algorithm followed by OMOPSO, while the latter appeared as the fastest algorithm. In the present work, however, PAES appears in the last positions in the scalabily ranking. This is explained by the fact that we have not considered here the impact of not having a 100% hit rate in the tables containing the computed evaluations, which undoubtely has penalized PAES in practically all the problems considered.

If we consider OMOPSO, its results in [11] remain basically the same. The inclusion in the present work of GDE3 and AbYSS have affected OMOPSO only in the scalabity ranking, and OMOPSO is still among the fastest algorithms assessed (excluding the ZDT4 test problem).

Finally, we would like to mention the effort that has been required to carry out this work. If we consider that we have studied eight algorithms to solve nine instances of five problems, the number of experiments is 360. As we have executed one hundred independent runs per experiment, this means a total of 36,000 runs. Furthermore, the experiments with SMPSO and GDE3 with mutation have required 9,000 additional runs. Taking into account also the pilot tests with the other algorithms to study the influence of the distribution indices in the mutation and crossover operators, the total number of runs have been in the order of 50,000. The fact that we have used a stronger termination condition (98% instead of 95% of the hypervolume of the Pareto front) and that the maximum number of evaluations have been raised from 500,000 in [11] to 10,000,000 in this paper, has had

TABLE XXI
SMPSO: EVALUATIONS

Problem	8 variables	16 variables	32 variables	64 variables	128 variables	256 variables	512 variables	1024 variables	2048 variables
ZDT1	1.40e+3 $_{3.5e+2}$	2.50e+3 $_{7.0e+2}$	5.20e+3 $_{1.3e+3}$	1.33e+4 $_{3.0e+3}$	3.37e+4 $_{6.2e+3}$	9.17e+4 $_{1.7e+4}$	2.31e+5 $_{2.7e+4}$	6.44e+5 $_{7.4e+4}$	1.81e+6 $_{9.4e+4}$
ZDT2	1.40e+3 $_{3.0e+2}$	2.30e+3 $_{6.0e+2}$	4.60e+3 $_{1.5e+3}$	1.16e+4 $_{3.8e+3}$	2.80e+4 $_{6.0e+3}$	7.26e+4 $_{1.4e+4}$	1.77e+5 $_{2.8e+4}$	4.55e+5 $_{5.9e+4}$	1.20e+6 $_{2.2e+5}$
ZDT3	1.90e+3 $_{6.0e+2}$	3.60e+3 $_{9.0e+2}$	8.00e+3 $_{3.0e+3}$	2.18e+4 $_{6.2e+3}$	6.30e+4 $_{1.2e+4}$	1.72e+5 $_{2.5e+4}$	4.85e+5 $_{6.3e+4}$	1.43e+6 $_{1.2e+5}$	4.38e+6 $_{1.7e+5}$
ZDT4	3.90e+3 $_{8.0e+2}$	4.65e+3 $_{1.0e+3}$	5.25e+3 $_{1.4e+3}$	5.90e+3 $_{1.2e+3}$	6.75e+3 $_{1.4e+3}$	6.95e+3 $_{1.4e+3}$	7.40e+3 $_{1.8e+3}$	8.15e+3 $_{2.0e+3}$	9.10e+3 $_{2.6e+3}$
ZDT6	2.45e+3 $_{8.5e+2}$	3.40e+3 $_{1.4e+3}$	7.15e+3 $_{2.7e+3}$	1.60e+4 $_{4.4e+3}$	3.65e+4 $_{8.0e+3}$	8.67e+4 $_{1.6e+4}$	1.94e+5 $_{3.1e+4}$	4.45e+5 $_{5.6e+4}$	1.03e+6 $_{1.0e+5}$

TABLE XXII
GDE3 WITH POLYNOMIAL MUTATION: EVALUATIONS

Problem	8 variables	16 variables	32 variables	64 variables	128 variables	256 variables	512 variables	1024 variables	2048 variables
ZDT1	3.30e+3 $_{3.0e+2}$	6.50e+3 $_{4.0e+2}$	1.18e+4 $_{5.0e+2}$	2.12e+4 $_{8.0e+2}$	3.80e+4 $_{9.0e+2}$	6.98e+4 $_{1.4e+3}$	1.33e+5 $_{2.4e+3}$	2.74e+5 $_{3.5e+3}$	6.45e+5 $_{8.4e+3}$
ZDT2	4.10e+3 $_{3.0e+2}$	8.00e+3 $_{5.0e+2}$	1.54e+4 $_{7.5e+2}$	2.88e+4 $_{9.0e+2}$	5.41e+4 $_{1.6e+3}$	1.02e+5 $_{1.8e+3}$	1.98e+5 $_{3.4e+3}$	4.14e+5 $_{4.4e+3}$	9.91e+5 $_{1.4e+4}$
ZDT3	2.60e+3 $_{1.0e+3}$	5.40e+3 $_{2.4e+3}$	1.01e+4 $_{3.3e+3}$	2.20e+4 $_{4.0e+3}$	5.06e+4 $_{8.1e+3}$	1.26e+5 $_{2.3e+4}$	3.53e+5 $_{3.9e+4}$	1.03e+6 $_{8.3e+4}$	3.17e+6 $_{1.9e+5}$
ZDT4	2.11e+4 $_{1.4e+3}$	5.98e+4 $_{4.8e+3}$	1.88e+5 $_{1.4e+4}$	7.78e+5 $_{6.9e+4}$	3.41e+6 $_{4.6e+5}$	-	-	-	-
ZDT6	4.50e+3 $_{4.0e+2}$	9.10e+3 $_{7.5e+2}$	1.90e+4 $_{2.0e+3}$	5.02e+4 $_{5.2e+3}$	2.24e+5 $_{8.0e+3}$	4.67e+5 $_{5.0e+3}$	9.39e+5 $_{7.4e+3}$	2.02e+6 $_{1.4e+4}$	4.97e+6 $_{3.3e+4}$

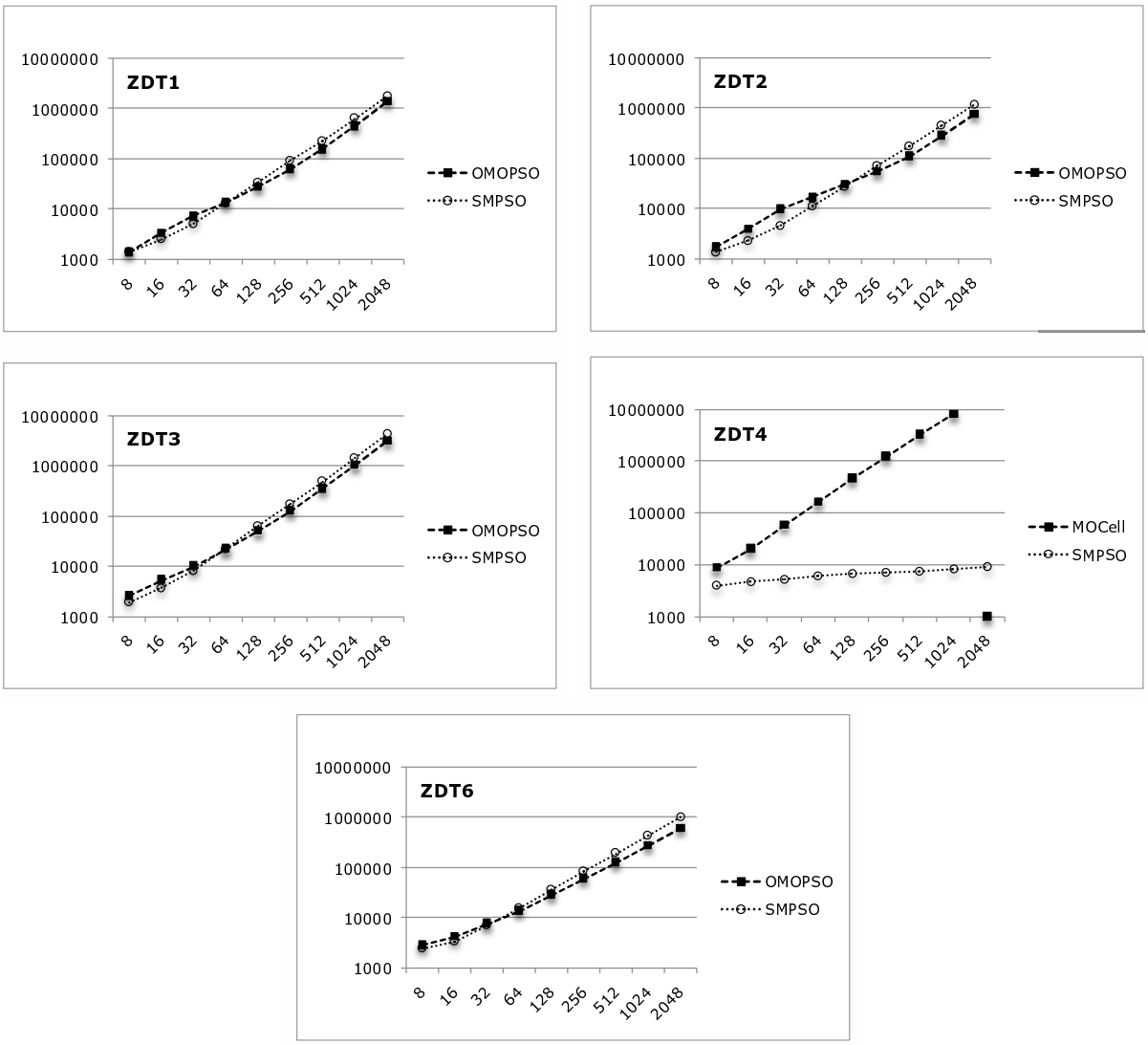

Fig. 13. Results of SMPSO on the ZDT problem family.

as a consequence the fact that the computational time of the algorithms unable to solve the problems with 1024 and 2048 variables was higher than one hour.

To execute this large amount of experiments, we have used the computers of the laboratories of the Departament of Computer Science of the University of Málaga, in Spain. Most of them are equipped with modern dual core processors so, taking into account that there are more than 180 computers, that means that up to 360 cores have been available. To run all the programs, we have used Condor [31], a middleware that acts as a distributed scheduler, which has proven to be an ideal tool to cope with the large amount of tasks we have dealt with.

VII. Conclusions and Future Work

We have evaluated eight state-of-the-art metaheuristics over a set of parameter scalable problems in order to study the behavior of the algorithms concerning their capabilities to solve problems having a large number of decision variables. The benchmark has been composed of five problems from the ZDT family, using instance sizes ranging from 8 to 2048 variables. We have also studied the speed of the techniques when solving the problems. The stopping condition has been to reach a front with a hypervolume higher than the 98% of the hypervolume of the true Pareto front, or to compute 10,000,000 function evaluations.

Our study has revealed that differential evolution and particle swarm optimization are the most promising approaches to deal with the scalable problems used in this work. GDE3 and OMOPSO do not only scale well, but they are among the fastest algorithms. Furthemore, we have shown that their search capabilities can be improved to solve ZDT4, the problem which has appeared as the most difficult one to solve.

Two modern optimizers, MOCell and AbYSS, have shown a high degree of regularity in the tests. With the exception

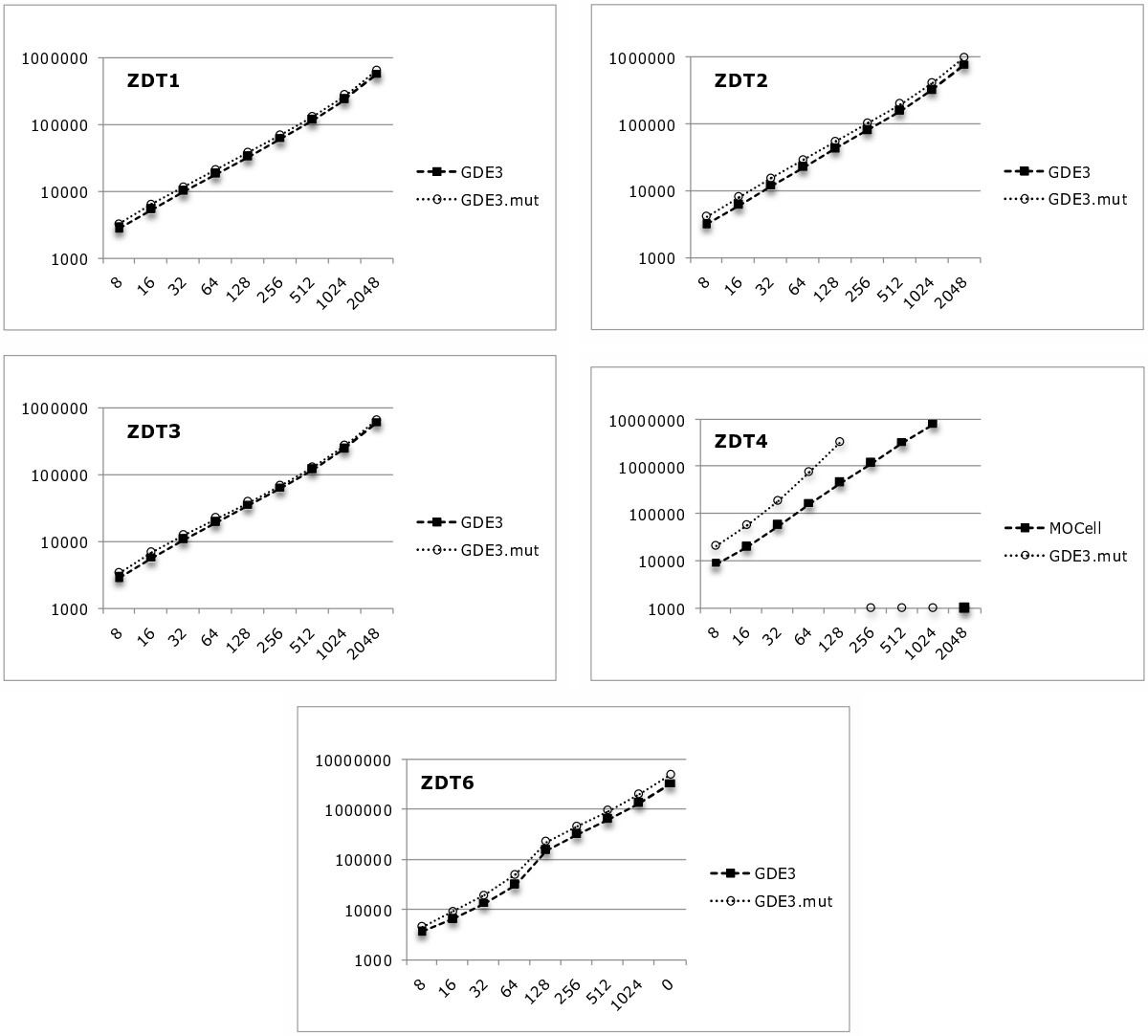

Fig. 14. Results of GDE3 with polynomial mutation on the ZDT problem family.

of MOCell in ZDT4 (where it is the overall best technique), they are not in the first position in the scalabily and the speed rankings, but also they are always around the third and fifth positions. Both metaheuristics are in the group of algorithms having solved a higher number of instances. In this group we find NSGA-II and SPEA2, which are, in general, very close in the rankings, but they usually appear after MOCell. PESA-II has difficulties in ZDT2 and it normally appears among the algorithms requiring higher numbers of function evaluations to reach a front with the target HV value.

Finally, PAES, the simplest of the optimizers in the study, is the algorithm scaling the worst, due to the low hit rates it obtains in many instances.

We have presented a study about the behavior of eight multi-objective metaheuristics concerning their scalability and speed when solving a scalable benchmark. The next step is an extension including more scalable problems (e.g., DTLZ and WFG) to assess whether or not the features of these problems confirm the results obtained in this work. Our analysis of OMOPSO and GDE3 has also shown that an open research line is to study variations and different parameter settings of the existing multi-objective metaheuristics in order to improve their scalability, and that is another path for future research that we aim to explore.

REFERENCES

[1] C. Blum and A. Roli. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

[2] D. Brockhoff and E. Zitzler. Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In Thomas Philip Runarsson, Hans-Georg Beyer, Edmund Burke, Juan J. Merelo-Guervós, L. Darrell Whitley, and Xin Yao, editors, *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference*, pages 533–542. Springer. Lecture Notes in Computer Science Vol. 4193, Reykjavik, Iceland, September 2006.

[3] E.K. Burke and G. Kendall. *Search methodologies: Introductory tutorials in optimization and decision support Techniques*. Springer, 2005.

[4] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[5] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, second edition, September 2007. ISBN 978-0-387-33254-3.

[6] D.W. Corne, N.R. Jerram, J.D. Knowles, and M.J. Oates. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 283–290. Morgan Kaufmann, 2001.

[7] K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[9] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multiobjective optimization. In Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105–145. Springer, USA, 2005.

[10] Francesco di Pierro, Shoon-Thiam Khu, and Dragan A. Savić. An Investigation on Preference Order Ranking Scheme for Multiobjective Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 11(1):17–45, February 2007.

[11] J.J. Durillo, A.J. Nebro, C.A. Coello Coello, F. Luna, and E. Alba. A comparative study of the effect of parameter scalability in multiobjective metaheuristics. In *2008 Congress on Evolutionary Computation (CEC'2008)*, pages 1893–1900, Hong Kong, June 2008. IEEE Service Center.

[12] J.J. Durillo, A.J. Nebro, F. Luna, B. Dorronsoro, and E. Alba. 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.

[13] F. W. Glover and G. A. Kochenberger. *Handbook of Metaheuristics*. Kluwer, 2003.

[14] S. Huband, L. Barone, R.L. While, and P. Hingston. A scalable multi-objective test problem toolkit. In C.A. Coello, A. Hernández, and E. Zitler, editors, *Third International Conference on Evolutionary MultiCriterion Optimization, EMO 2005*, volume 3410 of *Lecture Notes in Computer Science*, pages 280–295. Springer, 2005.

[15] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, October 2006.

[16] V. Khare, X. Yao, and K. Deb. Performance scaling of multi-objective evolutionary algorithms. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 376–390, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[17] 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 98–105, Piscataway, NJ, 1999. IEEE Press.

[18] J. D. Knowles and D. W. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[19] S. Kukkonen and J. Lampinen. GDE3: The third evolution step of generalized differential evolution. In *IEEE Congress on Evolutionary Computation (CEC'2005)*, pages 443 – 450, 2005.

[20] J. Lampinen. DE's selection rule for multiobjective optimization. Technical report, Lappeenranta University of Technology, Department of Information Technology, 2001.

[21] H. Li and Q. Zhang. Multiobjective Optimization Problems with Complicated Pareto Set, MOEA/D and NSGA-II. Accepted for publication in the *IEEE Transactions on Evolutionary Computation*, 2009.

[22] A. J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. J. Durillo, and A. Beham. AbYSS: Adapting scatter search to multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 12(4):439–457, 2008.

[23] A.J. Nebro, J.J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. A cellular genetic algorithm for multiobjective optimization. In David A. Pelta and Natalio Krasnogor, editors, *Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2006)*, pages 25–36, Granada, Spain, 2006.

[24] A.J. Nebro, J.J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. Design issues in a multiobjective cellular genetic algorithm. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization. 4th International Conference, EMO 2007*, volume 4403 of *Lecture Notes in Computer Science*, pages 126–140. Springer, 2007.

[25] K. Praditwong and X. Yao. How well do multi-objective evolutionary algorithms scale to large problems. In *2007 IEEE Congress on Evolutionary Computation (CEC'2007)*, pages 3959–3966, Singapore, September 2007. IEEE Press.

[26] Robin C. Purshouse and Peter J. Fleming. On the Evolutionary Optimization of Many Conflicting Objectives. *IEEE Transactions on Evolutionary Algorithms*, 11(6):770–784, December 2007.

[27] M. Reyes and C.A. Coello Coello. Improving PSO-based multi-objective optimization using crowding, mutation and ϵ-dominance. In C.A. Coello, A. Hernández, and E. Zitler, editors, *Third International Conference on Evolutionary MultiCriterion Optimization, EMO 2005*, volume 3410 of *LNCS*, pages 509–519. Springer, 2005.

[28] D.K. Saxena and K. Deb. Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: Employing correntropy and a novel maximum variance unfolding. In Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata, editors, *Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007*, pages 772–787, Matshushima, Japan, March 2007. Springer. Lecture Notes in Computer Science Vol. 4403.

[29] D.J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC; 4 edition, 2007.

[30] Li-Sun Shu Shinn-Ying Ho and Jian-Hung Chen. Intelligent Evolutionary Algorithms for Large Parameter Optimization Problems. *IEEE Transactions on Evolutionary Computation*, 8(6):522–541, 2004.

[31] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the Condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.

[32] 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., Graduate School of Eng., Air Force Inst. Technol., Wright-Patterson, AFB, OH, 1998.

[33] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 8(11):712–731, 2008.

[34] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

[35] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. In K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2002.

[36] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.