

# A Comparative Study of the Effect of Parameter Scalability in Multi-Objective Metaheuristics

Juan J. Durillo, Antonio J. Nebro, Carlos A. Coello Coello, Francisco Luna, Enrique Alba

**Abstract**—Some real-world optimization problems have hundreds or even thousands of decision variables. However, the effect that the scalability of parameters has in modern multi-objective metaheuristic algorithms has not been properly studied (the current benchmarks are normally adopted with ten to thirty decision variables). In this paper, we adopt a benchmark of parameter-wise scalable problems (the ZDT test problems) and analyze the behavior of six multi-objective metaheuristics on these test problems when using a number of decision variables that goes from 8 up to 2048. The computational effort required by each algorithm in order to reach the true Pareto front is also analyzed. Our study concludes that a particle swarm algorithm provides the best overall performance, although it has difficulties in multifrontal problems.

## 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). Contrary to single-objective optimization problems, the solution to MOPs is normally never a single solution, but a set of *nondominated* 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 MOPs in the real world tend to be nonlinear and with objective functions that are very expensive to evaluate, motivates the use of *metaheuristics* to deal with these problems [1], [9]. Metaheuristics are a family of techniques comprising *Evolutionary Algorithms* (EA), *Particle Swarm Optimization* (PSO), *Ant Colony Optimization* (ACO), *Tabu Search* (TS), *Scatter Search* (SS) and many others. EAs are among the most popular metaheuristics in current use, and some of the most popular algorithms for multi-objective optimization, such as NSGA-II [6] and SPEA2 [21], are *Genetic Algorithms* (GAs), a subfamily of EAs [3], [5]. Other EAs are the *Evolution Strategies* (ESs) and *Genetic Programming* (GP).

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

Carlos A. Coello Coello is with the Department of Computer Science, CINVESTAV-IPN, Mexico (e-mail: {ccoello@cs.cinvestav.mx})

This work has been partially funded by the Spanish Ministry of Education and Science and FEDER under contract TIN2005-08818-C04-01 (the OPLINK project). Juan J. Durillo is supported by grant AP-2006-03349 from the Spanish Ministry of Education and Science. Carlos A. Coello Coello acknowledges support from CONACYT project no. 45683-Y.

The performance of these algorithms has been typically assessed by using benchmark problems, such as the Zitzler-Deb-Thiele (ZDT) test problems [19], the Deb-Thiele-Laumanns-Zitzler (DTLZ) test problems [7], and the Walking-Fish-Group (WFG) test problems [10]). 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 to compare the values of different quality indicators (e.g., *generational distance* [17] or *hypervolume* [20], among others).

The motivation driving us is that many real-world problems have hundreds or even thousands of decision variables, and the aforementioned benchmarks have been normally adopted using a maximum of up to 30 variables. Thus, the studies currently available do not consider the capability of current multi-objective metaheuristic algorithms to properly scale when dealing with a very large number of decision variables.

Another interesting issue that has been only scarcely covered in the specialized literature is the analysis of the behavior of a multi-objective metaheuristic until reaching the true Pareto front of a problem. Typically, a fixed number of evaluations (and, in consequence, of iterations) is defined by the user, and the performance of the different algorithms studied is compared. However, this sort of comparison does not provide any indication regarding the computational effort that a certain algorithm requires to reach the true Pareto front of a problem. 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 is for a certain algorithm to reach the true Pareto front. That is the reason why we present this sort of analysis in this paper.

In this work, we compare six state-of-the-art multi-objective metaheuristics when solving a set of scalable parameter-wise MOPs, those comprising the ZDT benchmark, considering their formulation ranging from 8 up to 2048 variables. The algorithms are three GAs (NSGA-II [6], SPEA2 [21], and PESA-II [4]), an ES (PAES [12]), a PSO (OMOPSO [15]), and a cellular GA (MOCcell [13]). In our study, we also analyze how fast they provide a satisfactory solution of the problem. Briefly, given that the true Pareto fronts of the ZDT problems are known, we consider that

an algorithm has successfully solved the problem when the hypervolume of its current population (or archive) is higher than 95% of the hypervolume of the true Pareto front. Obviously, statistical tests need to be performed in order to assess the significance of the results, so that chance can be discarded as the responsible for them.

## II. RELATED WORK

Many real world MOPs have many decision variables and objective functions. To study the search capabilities of multi-objective metaheuristics to solve these types of problems, benchmarks of scalable MOPs have been defined: the ZDT set [19] is composed of six parameter-wise scalable MOPs, and the DTLZ [7] and WFG [10] test suites include scalable problems in both the number of parameters and the number of objectives.

In the multi-objective research community, the study of objective function scalability is a hot topic (leading to the so-called *many-objective optimization*). In [11], the performance of three multi-objective genetic algorithms (PESA, SPEA, and NSGA-II) is studied when solving four DTLZ problems with a number of objectives ranging from 2 to 8. In a similar context, the behavior of NSGA-II when solving the same problems and the same number of objectives is analyzed in [14], where the concept of classifying nondominated solutions is introduced to improve the search capabilities of the NSGA-II.

Due to difficulties to solve high dimensional MOPs, some authors introduce ideas to reduce the number of objectives, as the work of Brockhoff and Zitzler [2]; here, they investigate whether all the objectives are necessary to preserve the problem characteristics. In [16], Saxena and Deb propose techniques for dimensionality reduction and apply them to up to 50-objective MOPs.

While these and other works are focused in objective function scalability, the study of parameter scalability has not been considered before in multi-objective optimization, to the best of the authors' knowledge. As far as we know, only in [18] a small study using the ZDT1 problem with up to 100 variables is included. Thus, the main motivation of this study has been precisely to present an in-depth study on this important topic.

## III. SCALABLE PARAMETER-WISE MULTI-OBJECTIVE OPTIMIZATION PROBLEMS

To carry out our study, it is 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 [19] fulfills this requirement. Additionally, it offers, a group of problems with different properties: convex, non-convex, disconnected, multi-frontal, many-to-one problems. These problems have been widely used in many studies in the field since they were formulated. We omitted problem ZDT5 because it uses binary encoding.

Since we were interested in studying the behavior of multi-objective metaheuristics when solving scalable parameter-wise problems, we have evaluated each ZDT problem with

TABLE I  
PARAMETERIZATION ( $L =$  INDIVIDUAL LENGTH)

Parameterization used in MOCeII	
<i>Population Size</i>	100 individuals ( $10 \times 10$ )
<i>Neighborhood</i>	1-hop neighbours (8 surrounding solutions)
<i>Selection of Parents</i>	binary tournament + binary tournament
<i>Recombination</i>	simulated binary, $p_c = 0.9$
<i>Mutation</i>	polynomial, $p_m = 1.0/L$
<i>Archive Size</i>	100 individuals
Parameterization used in NSGA-II	
<i>Population Size</i>	100 individuals
<i>Selection of Parents</i>	binary tournament + binary tournament
<i>Recombination</i>	simulated binary, $p_c = 0.9$
<i>Mutation</i>	polynomial, $p_m = 1.0/L$
Parameterization used in SPEA2	
<i>Population Size</i>	100 individuals
<i>Selection of Parents</i>	binary tournament + binary tournament
<i>Recombination</i>	simulated binary, $p_c = 0.9$
<i>Mutation</i>	polynomial, $p_m = 1.0/L$
Parameterization used in PESA-II	
<i>Population Size</i>	100 individuals
<i>Selection of Parents</i>	region based selection + region based selection
<i>Recombination</i>	simulated binary, $p_c = 0.9$
<i>Mutation</i>	polynomial, $p_m = 1.0/L$
<i>Archive Size</i>	100 individuals
Parameterization used in OMOPSO	
<i>Particles</i>	100 particles
<i>Mutation</i>	uniform + non-uniform
<i>Leaders Size</i>	100
Parameterization used in PAES	
<i>Mutation</i>	polynomial, $p_m = 1.0/L$
<i>Archive Size</i>	100

8, 16, 32, 64, 128, 256, 512, 1024, and 2048 variables. This way, we can study not only what techniques produce more accurate fronts when solving problems having many variables, but also if their search capabilities remain constant or not when the number of problem variables augments.

## IV. METAHEURISTICS MULTI-OBJECTIVE ALGORITHMS

As indicated before, we adopted six multi-objective metaheuristics for our study: (1) the Nondominated Sorting Genetic Algorithm II (**NSGA-II**) [6], (2) the Strength Pareto Evolutionary Algorithm (**SPEA2**) [21], (3) the Pareto Archived Evolution Strategy (**PAES**) [12], (4) the Pareto Envelope-based Search Algorithm for multi-objective optimization II (**PESA II**) [4]), (5) the ‘‘Our’’ Multi-Objective Particle Swarm Optimizer (**OMOPSO**) [15], and (6) the Multi-Objective Cellular Genetic Algorithm (**MOCeII**) [13].

The descriptions of these approaches were omitted due to space constraints (interested readers must refer to their corresponding publications). We have used the implementations of these algorithms provided by jMetal [8], which is a Java-based framework for developing metaheuristics for solving multi-objective optimization problems<sup>1</sup>.

## V. EXPERIMENTATION

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

<sup>1</sup>jMetal is freely available for download at the following Web address: <http://neo.lcc.uma.es/metal/>.

### A. Parameterization

We have chosen a set of parameter settings aiming to guarantee a fair comparison among the algorithms. All the GAs (NSGA-II, SPEA2, PESA-II, and MOCcell) use an internal population of size equal to 100; OMOPSO is configured with 100 particles, and the size of the archive in PAES is also 100.

In the GAs, we have used SBX and polynomial mutation [5] as operators for crossover and mutation, respectively. The distribution indexes 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 used the polynomial mutation operator, with the same distribution index. A detailed description of the parameter settings is shown in Table I.

### B. Methodology

We are interested in two goals: the behavior of the algorithms when solving the scalable ZDT benchmark and to know which algorithms are faster in reaching the true Pareto front. Given that the true Pareto fronts of the ZDT problems are known, a strategy could be to run the algorithms until they reach such front, but then it is possible that some of them never achieve the optimal front. Our approach is to establish a stopping condition based on the *high quality* of the Pareto front found, and we have used the hypervolume [20] quality indicator for that purpose.

The hypervolume computes the volume (in the objective space) covered by members of a nondominated set of solutions  $Q$  for problems where 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, a union of all hypercubes is found and its hypervolume ( $HV$ ) is calculated using:

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

Higher values of the hypervolume metrics are desirable. A property of this quality indicator is that it measures both convergence to the true Pareto front and diversity of the obtained fronts.

In our experiments, each algorithm is executed until a maximum of 500,000 function evaluations have been computed. Every 100 evaluations (that is, each iteration in the population based metaheuristics) we measure the hypervolume of the nondominated solutions found so far. Therefore, in NSGA-II and SPEA2 we have considered the nondominated solutions in each generation, in PESA-II, PAES and MOCcell the external population and, in OMOPSO, the leaders archive. We consider as stopping condition to obtain a hypervolume value of 95% of the hypervolume of the true Pareto front, or the reach to the 500,000 evaluations.

Using the hypervolume in the stopping condition allows us to obtain a *hit rate* of the algorithms, i.e., the percentage of successful executions. An execution is successful if the algorithm stops before getting to the 500,000 function evaluations. This way, we can measure the robustness of the techniques when solving the problems.

We have performed 100 independent runs per each algorithm and each problem instance. Since we are dealing with stochastic algorithms, we need to perform a statistical analysis of the obtained results in order to compare them with certain level of confidence. Next, we describe the statistical test that we have carried out for ensuring such statistical confidence. 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., a 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 the tables containing the results; conversely, “-” means that no statistical confidence was found ( $p$ -value  $> 0.05$ ). Looking for homogeneity in the presentation of the results, all the tables include the median,  $\tilde{x}$ , and interquartile range,  $IQR$ , as measures of location (or central tendency) and statistical dispersion, respectively.

### C. Analysis of results

Tables II, III, IV, V, VI show the median and the interquartile range of the number of evaluations needed by the different optimizers for ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6, respectively. When an optimizer is not able to reach an acceptable front in 500,000 function evaluations, its result appears as “-”, and it is not taken into account in the statistical tests. To facilitate the analysis of the 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 the lighter grey points out the second best value.

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, included in 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 study the hit rate values; finally, we consider the speed to obtain the Pareto front.

- *ZDT1*: Table II and Figure 1 show the evaluations needed to find the Pareto fronts for problem ZDT1. We can observe that all the optimizers successfully solved this problem up to 512 variables, four out the



TABLE VII  
HIT RATE FOR ZDT1

Algorithm	8	16	32	64	128	256	512	1024	2048
MOCcell	1	1	1	1	1	1	1	0	0
NSGA-II	1	1	1	1	1	1	1	1	0
SPEA2	1	1	1	1	1	1	1	0.97	0
PESA-II	1	1	1	1	1	1	1	0	0
OMOPSO	1	1	1	1	1	1	1	1	0
PAES	1	1	1	1	1	0.98	0.98	0.89	0.56

six algorithms solved the problem with 1024 variables, and only PAES was able to find solutions with 2048 variables. This last result is certainly surprising, given that PAES is the simplest algorithm in our experiments. If we analyze the hit rate (see Table VII), all the algorithms excepting PAES achieved a 100% up to 512 variables. The hit rate of PAES when solving the 2048 instance (0.56), indicates that the problem was solved only in about half of the 100 independent runs executed. Attending to the speed of the techniques, OMOPSO reached first the desired results in the instances up to 256 variables, and it ranked second when dealing with both 512 and 1024 variables. PAES was the fastest approach in the largest instances and the second fastest algorithm in the instances ranging from 32 to 256 variables. For all cases, except the last column, Table II indicates that the results have statistical confidence (see the “+” in the last row). The “-” in the last column is due to the fact that PAES was the only algorithm that obtained a solution, and thus, the other algorithms were not considered.

TABLE VIII  
HIT RATE FOR ZDT2

Algorithm	8	16	32	64	128	256	512	1024	2048
MOCcell	1	1	1	1	1	1	1	0.02	0
NSGA-II	1	1	1	1	1	1	1	0	0
SPEA2	1	1	1	1	1	1	1	0	0
PESA-II	1	1	1	1	1	1	1	0	0
OMOPSO	1	1	1	1	1	1	1	1	0
PAES	1	1	1	1	1	1	0.97	0.87	0.49

- *ZDT2*: The evaluations needed to solve the *ZDT2*

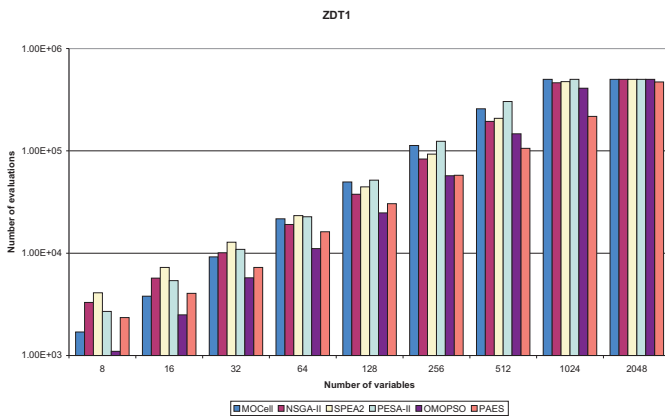


Fig. 1. Number of evaluations when solving ZDT1.

ZDT2

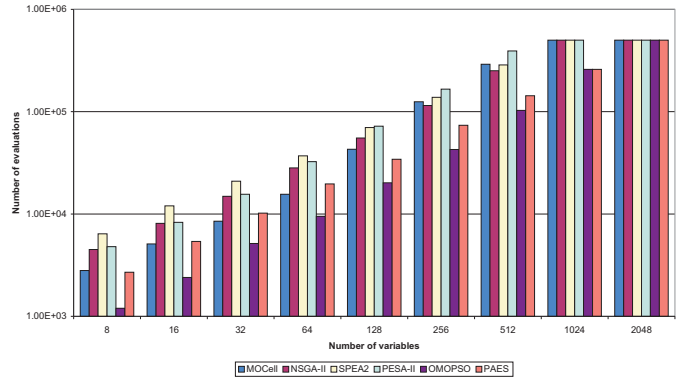


Fig. 2. Number of evaluations when solving ZDT2.

problem are included in Table III and Figure 2. In this problem, only OMOSPO and PAES found the solution with 1024 variables, and again PAES is the only technique that was successful with the largest instance. The hit rate (see Table VIII) produces results similar to those obtained in ZDT1. All the algorithms found the Pareto front up to 256 variables, and PAES obtained a value of 0.49 in the instance with 2048 variables. OMOPSO was the fastest algorithm in the instances up to 512 variables, and it ranked second with 1024 variables. PAES was the second best technique, achieving the best values in the two most difficult instances and was the second best in four other instances. As in ZDT1, all the results have statistical confidence.

ZDT3

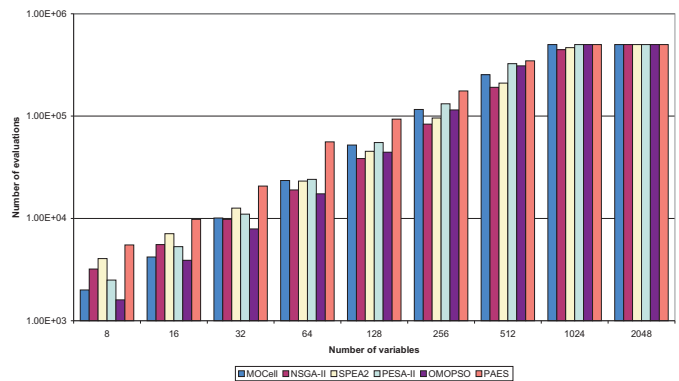


Fig. 3. Number of evaluations when solving ZDT3.

TABLE IX  
HIT RATE FOR ZDT3

Algorithm	8	16	32	64	128	256	512	1024	2048
MOCcell	1	1	1	1	1	1	1	0	0
NSGA-II	1	1	1	1	1	1	1	1	0
SPEA2	1	1	1	1	1	1	1	1	0
PESA-II	1	1	1	1	1	1	1	0	0
OMOPSO	1	1	1	1	1	1	1	0	0
PAES	1	1	1	1	0.96	0.83	0.64	0.36	0.21

- **ZDT3**: Table IV and Figure 3 show the evaluations needed to solve the ZDT3 problem. In this case, none of the metaheuristics solved the problem with 2048 variables, and only NSGA-II, SPEA2, and PAES succeeded with the instance having 1024 variables. However, if we take a look to the hit rates in Table IX, we observe that PAES had a 0.21 of success when dealing with the 2048 instance; as we are considering the median of 100 runs, the problem appears as not solved in Table IX (we have to note that the *IQR* only considers the values between the 25<sup>th</sup> and the 75<sup>th</sup> percentiles).

Concerning the speed of the algorithms, OMOPSO is the fastest in the instances ranging from 8 to 64 variables, while NSGA-II is the best technique in the instances from 128 to 1024 variables. The last row in Table 3 indicates that all the results are significant.

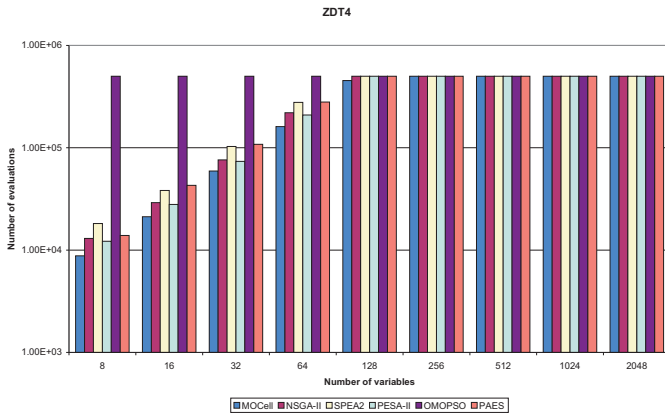


Fig. 4. Number of evaluations when solving ZDT4.

TABLE X  
HIT RATE FOR ZDT4

Algorithm	8	16	32	64	128	256	512	1024	2048
MOCeII	1	1	1	1	0.84	0	0	0	0
NSGA-II	1	1	1	1	0.01	0	0	0	0
SPEA2	1	1	1	1	0	0	0	0	0
PESA-II	1	1	1	1	0.02	0	0	0	0
OMOPSO	0	0	0	0	0	0	0	0	0
PAES	1	1	1	1	0.02	0	0	0	0

- **ZDT4**: Table V and Figure 4 include the results for the ZDT4 problem. This problem is particularly interesting for us, since several multi-objective metaheuristics have difficulties to solve it due to its multifrontal nature. Indeed, for this problem, none of the optimizers produced an acceptable front in instances having more than 128 variables. It is remarkable that OMOPSO is unable to find the Pareto front for this problem in any configuration. The most salient algorithm is MOCeII, which is the only metaheuristic yielding satisfactory results for 128 variables.

The analysis of the hit rate (see Table X) confirms that the problem is solved in the 100 independent runs performed by all the algorithms, excepting OMOPSO, up to 64 variables. The value obtained by MOCeII with

128 variables, 0.84, indicates that not all its executions were successful.

Attending to the search speed, MOCeII is the best option for all problem instances, followed by PESA-II. However, if we observe the last row of Table V, we see that there is no statistical confidence in the experiments, so we cannot make any assumptions about the differences in the speed of the metaheuristics.

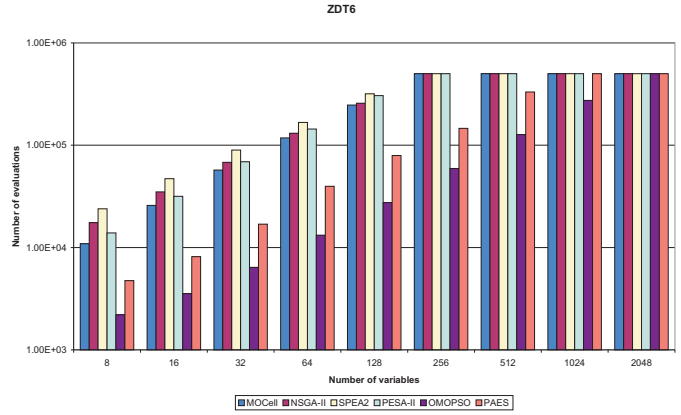


Fig. 5. Number of evaluations when solving ZDT6.

TABLE XI  
HIT RATE FOR ZDT6

Algorithm	8	16	32	64	128	256	512	1024	2048
MOCeII	1	1	1	1	1	0	0	0	0
NSGA-II	1	1	1	1	1	0	0	0	0
SPEA2	1	1	1	1	1	0	0	0	0
PESA-II	1	1	1	1	1	0	0	0	0
OMOPSO	1	1	1	1	1	1	1	1	0
PAES	1	0.98	0.96	0.96	0.96	0.92	0.74	0.33	0

- **ZDT6**: Table VI and Figure 5 include the evaluations needed to find the Pareto fronts for problem ZDT6. We observe that all the algorithms successfully solved this problem up to 128 variables, and only OMOPSO and PAES solved the problem up to 1024 variables. The hit rate (see Table XI) clearly shows that OMOPSO is the best algorithm in this problem, achieving the 100% of success in all the experiments. PAES, however, is not so robust, and its hit rate values decreases progressively with the number of variables. Concerning the search speed, OMOPSO is the fastest algorithm, followed by PAES. For all cases, the results have statistical confidence.

## VI. DISCUSSION OF THE RESULTS

In this section, we analyze the results globally, with the goal of identifying the strengths and weaknesses of the algorithms analyzed, when solving the full set of test problems. To facilitate this discussion, we have made a rank of the three best algorithms according to their ability to solve the problems with a higher number of decision variables (scalability) and to the number of evaluations required to find

TABLE XII  
RANKING OF THE ALGORITHMS: SCALABILITY

Ranking	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
1	PAES	PAES	NSGA-II	MOCeII	OMOPSO
2	OMOPSO	OMOPSO	SPEA2	PESA-II	PAES
3	NSGA-II	MOCeII	PAES	NSGA-II	MOCeII

TABLE XIII  
RANKING OF THE ALGORITHMS: SPEED

Ranking	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
1	OMOPSO	OMOPSO	NSGA-II	MOCeII	OMOPSO
2	PAES	PAES	OMOPSO	PESA-II	PAES
3	NSGA-II	MOCeII	MOCeII	NSGA-II	MOCeII

the Pareto fronts (speed). The two rankings are presented in Tables XII and XIII, respectively.

#### A. Scalability and speed

The ranking in Table XII considers the algorithms solving the problems with a higher number of decision variables. The ties are broken considering the hit rate and the iterations in the most difficult instances. According to this ranking, PAES is the most salient technique: it achieves the best results in ZDT1 and ZDT2, the second best in ZDT6, and the third best in ZDT3. OMOPSO is the second technique according to this ordering; however, its noticeable inability to deal with ZDT4 makes it the less robust algorithm considering the whole benchmark. Concerning the rest of the approaches evaluated, NSGA-II stands out in the disconnected problem ZDT3 and MOCeII in the multifrontal problem ZDT4.

The ordering in Table XIII relies on the algorithms requiring (globally) the lowest number of evaluations to find the Pareto front. In the case of incomparable results, the number of evaluations when solving the most difficult instances is taken into account. We do not consider here the hit rate. If we omit the ZDT4 problem, OMOPSO is clearly the best algorithm: it requires the lowest number of evaluations in problems ZDT1, ZDT2, ZDT6, and it is the second one in ZDT3. PAES is the second best algorithm in terms of the speed.

An interesting fact is that, if we observe the two tables, the rankings are similar with only few exceptions. This seems to imply that when an algorithm scales well with a problem, it also requires a low number of function evaluations to obtain the desired front.

#### B. OMOPSO and Multifrontal problems

If we do not take into account problem ZDT4, the most outstanding algorithm in our study is, considering both scalability and speed, OMOPSO. In this section we analyze whether its inefficacy when dealing with ZDT4 is particular to that problem, or if it happens with multifrontal problems in general.

To explore this issue, we have defined two multifrontal problems using the methodology described in [5]. Deb points out that given a function  $g(\vec{x})$ , a two-objective problem is defined as the minimization of

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

TABLE XIV  
GRIEWANK AND ACKLEY SINGLE-OBJECTIVE FORMULATION

Problem	Functions	Variables
Griewank	$f(x) = 1 + \sum_1^p \frac{x_i^2}{4000} - \prod_1^p \cos \frac{x_i}{\sqrt{i}}$	10
Ackley	$f(x) = 20 + e + -20exp(-0.2\sqrt{\frac{1}{p}\sum_1^p x_i^2})$	3

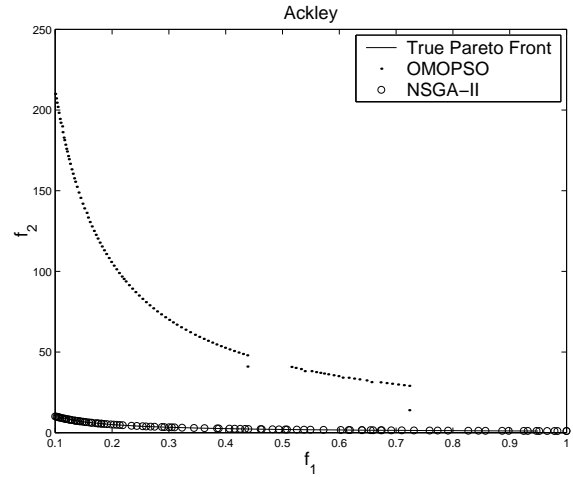


Fig. 6. OMOPSO solving Ackley test problem.

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

This way, given a single-objective problem with local optimal solutions, we can construct a multifrontal bi-objective MOP. We have selected two well-known problems having local minimal solutions, Griewank and Ackley (see Table XIV). The resulting problems have been solved by the six metaheuristics we are dealing with. The conclusion is that all the algorithms but OMOPSO converge to the true Pareto front. To illustrate this fact, we include in Figures 7 and 6 the fronts obtained by OMOPSO and NSGA-II.

An explanation of this behavior of OMOPSO with multifrontal problems could be due to an unbalance between diversification and intensification. The fact that OMOPSO is the fastest algorithm can be due to the high intensification capabilities of the technique; this produces a negative effect in the diversification when searching. This issue is particularly harmful when trying to solve multifrontal problems. However, this hypothesis do not seem to be satisfactory enough to explain the good behavior of OMOPSO when the problems have a high number of decision variables. A deeper study should be carried out in order to draw a more solid conclusion on this behavior.

## VII. CONCLUSIONS AND FUTURE WORK

We have evaluated six metaheuristics over a set of parameter-wise scalable MOPs in order to study the performance of the algorithms concerning their capabilities to solve problems having a large number of decision variables. We have also studied their velocity to obtain a good quality

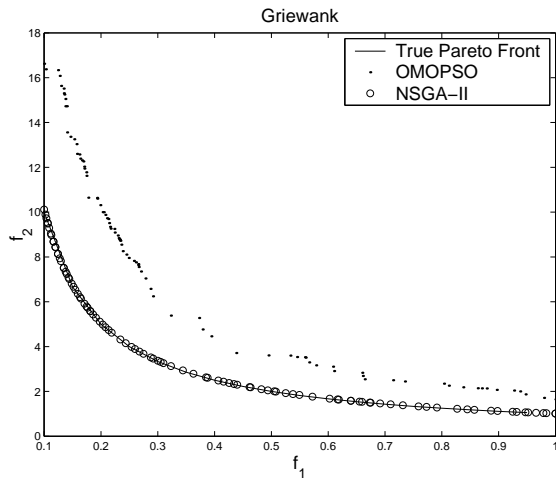


Fig. 7. OMOPSO solving Griewank test problem.

Pareto front using a stopping condition based on the hypervolume of the true Pareto front.

Regarding scalability, PAES seems to be the most competitive algorithm. This is a surprising result, because PAES is the simplest algorithm used in our study. It also provides good results in terms of speed, but its hit rate values indicate that it is not as robust as the other approaches analyzed.

OMOPSO is the best algorithm in terms of speed and the second best in terms of scalability. Unfortunately, it is unable to solve any instance of the ZDT4 problem. We have studied this issue by defining two additional multifrontal problems, and we found that OMOPSO exhibited in them the same poor behavior as before.

The three genetic algorithms, NSGA-II, SPEA2, and PESA-II, have not produced particularly remarkable results, in particular SPEA2. Notwithstanding, NSGA-II is the best when solving ZDT3, the partitioned problem.

MOCcell, the cellular genetic algorithm, stands out in problem ZDT4. It is the fastest algorithm in that problem, and the only one able to produce results with 128 variables. In terms of scalability, it is behind NSGA-II and SPEA2; concerning speed, it ranked third.

We have presented a first study of the behavior of multi-objective metaheuristics concerning their parameter scalability and convergence speed. The next step is an extension of this study, which includes more scalable problems (e.g., the DTLZ and WFG test problems) in order to assess whether the features of the problems (convexity, non-convexity, etc.) affect, in some way, the results obtained in this work. This sort of study can also lead to the design of new multi-objective metaheuristics that can overcome the limitations of those in current use.

## REFERENCES

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

[2] Dimo Brockhoff and Eckart Zitzler. Are all objectives necessary? on dimensionality reduction in evolutionary multiobjective optimization. In *PPSN*, pages 533–542, 2006.

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

[4] David W. Corne, Nick R. Jerram, Joshua D. Knowles, and Martin 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.

[5] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

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

[7] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Test Problems for Evolutionary Multi-Objective Optimization. Technical Report 112, Zurich, Switzerland, 2001.

[8] Juan J. Durillo, Antonio J. Nebro, Francisco Luna, Bernabé Dorronsoro, and Enrique 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.

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

[10] Simon Huband, Luigi Barone, R. Lyndon While, and Phil Hingston. A scalable multi-objective test problem toolkit. In C.A. Coello, A. Hernández, and E. Zitzler, editors, *Third International Conference on Evolutionary Multi-Criterion Optimization, EMO 2005*, volume 3410 of *Lecture Notes in Computer Science*, pages 280–295. Springer, 2005.

[11] Vineet Khare. Performance scaling of multi-objective evolutionary algorithms. MSc Thesis, University of Birmingham, September 2002.

[12] Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[13] Antonio J. Nebro, Juan J. Durillo, Francisco Luna, Bernabé Dorronsoro, and Enrique Alba. A cellular genetic algorithm for multi-objective 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.

[14] K. Praditwong and X. Yao. How well do multi-objective evolutionary algorithms scale to large problems. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation CEC'07*. IEEE Press, September 2007.

[15] Margarita Reyes and Carlos A. Coello Coello. Improving pso-based multi-objective optimization using crowding, mutation and  $\epsilon$ -dominance. In C.A. Coello, A. Hernández, and E. Zitzler, editors, *Third International Conference on Evolutionary Multi-Criterion Optimization, EMO 2005*, volume 3410 of *LNCS*, pages 509–519. Springer, 2005.

[16] Dhish Kumar Saxena and Kalyanmoy Deb. Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: Employing coreentropy and a novel maximum variance unfolding. 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 772–787. Springer, 2007.

[17] 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.

[18] Qingfu Zhang and Hui Lim. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. December 2007.

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

[20] 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.

[21] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailiou, and T. Fogarty, editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2002.