A Study of Convergence Speed in Multi-Objective Metaheuristics

A. J. Nebro[1], J. J. Durillo[1], C. A. Coello Coello[2], F. Luna[1], and E. Alba[1]

[1] Department of Computer Science, University of Málaga (Spain)
{antonio,durillo,flv,eat}@lcc.uma.es
[2] Department of Computer Science, CINVESTAV-IPN, Mexico
ccoello@cs.cinvestav.mx

Abstract. An open issue in multi-objective optimization is designing metaheuristics that reach the Pareto front using a low number of function evaluations. In this paper, we adopt a benchmark composed of three well-known problem families (ZDT, DTLZ, and WFG) and analyze the behavior of six state-of-the-art multi-objective metaheuristics, namely, NSGA-II, SPEA2, PAES, OMOPSO, AbYSS, and MOCell, according to their convergence speed, i.e., the number of evaluations required to obtain an accurate Pareto front. By using the hypervolume as a quality indicator, we measure the algorithm that converges faster, as well as their hit rate over 100 independent runs. We also determine the robustness of the metaheuristics analyzed. Our study reveals that modern multi-objective metaheuristics such as MOCell, OMOPSO, and AbYSS provide the best overall performance, while NSGA-II and AbYSS again are the most robust solvers.

1 Introduction

Many real-world optimization problems require to optimize 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 of MOPs is not given by a single solution, but by 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 a well-distributed Pareto front is the main goal in multi-objective optimization. This means that multi-objective optimizers need to explore larger portions of the search space because they search for the Pareto front, i.e., not a single optimum but a set of Pareto optima. Additionally, many real-world MOPs typically need computationally expensive methods for computing the objective functions and constraints. In this context, deterministic techniques are generally not applicable, which leads therefore to using approximate methods [7]. Among them, metaheuristics [1, 7] are nowadays used extensively to deal with MOPs.

The performance of these algorithms is normally assessed using benchmarks, such as the Zitzler-Deb-Thiele (ZDT) test suite [19], the Deb-Thiele-Laumanns-

Zitzler (DTLZ) test suite [3], and the Walking-Fish-Group (WFG) test problems [9]). The experimentation methodology typically lies in computing a prefixed number of function evaluations and then comparing the obtained results by considering different quality indicators [11].

The motivation driving us is that the objective functions of many real-world problems are hard to compute, so applying metaheuristics requiring a high number of function evaluations to solve them is not a satisfactory approach in practice. So therefore, it can be more important to obtain a reasonably good approximation to the Pareto front of a given MOP faster than to search for a high quality solution set but requiring more time. Thus, an open research area is to design techniques with this goal in mind, and some works that address this issue have recently appeared. Santana-Quintero et al. propose in [17] a PSO algorithm using rough sets theory, and it is used to solve problems using 4,000 fitness function evaluations, which is a low number compared to today's standards in the specialized literature. In a related paper, Hernández-Díaz et al. [8] propose a hybrid algorithm between a multi-objective differential evolution approach and rough sets theory, which only performs 3,000 fitness function evaluations. In [18], a more efficient multi-objective PSO algorithm is presented; this algorithm is able to provide accurate Pareto fronts of MOPs computing only 2,000 fitness function evaluations. Eskandari et al. explore in [6] the use of dynamic population sizing to design an algorithm called FastPGA, which outperforms NSGA-II when computing less than 10,000 solution evaluations. Knowles [10] studies multi-objective optimization calculating only 260 function evaluations; he proposes an algorithm called ParEGO, which outperforms NSGA-II using such a low number of evaluations.

In this paper, we are interested in analyzing the convergence speed of six state-of-the-art multi-objective metaheuristics to give hints about their efficiency when solving 21 MOPs comprising the test suites ZDT, DTLZ, and WFG. The algorithms are two Genetic Algorithms (NSGA-II [2], and SPEA2 [20]), an Evolution Strategy (PAES [12, 13]), a Particle Swarm Optimization algorithm (OMOPSO [16]), a Scatter Search method (AbYSS [15]), and a cellular Genetic Algorithm (MOCell [14]). In our study to assess the quality of a front we have employed the hypervolume quality indicator (HV) [21]. To assure that an algorithm has successfully solved a problem it needs reaching a fixed percent of the hypervolume of the true Pareto front. In Fig. 1 we show different approximations of the Pareto front for the problem ZDT1 with different percentages of HV. 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. So, we have taken 98% of the hypervolume of the true Pareto front as a criterion to consider that a MOP has been successfully solved. Thus, those algorithms requiring fewer function evaluations to achieve this termination condition can be consider to be *faster*. Using the hypervolume in the stopping condition also allows us to obtain a hit rate for the algorithms, i.e., their percentage of successful executions. This way, we can not only analyze the *convergence speed* but also the *robustness* of the compared techniques.

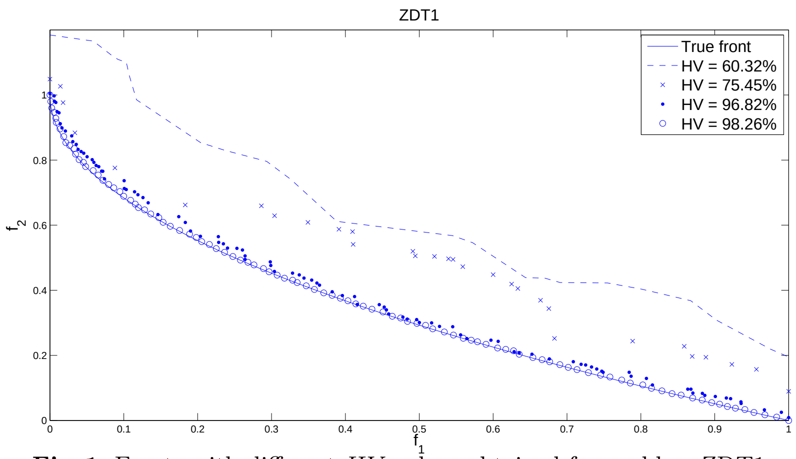

Fig. 1. Fronts with different HV values obtained for problem ZDT1.

The rest of the paper is organized as follows. In Section 2, we describe the multi-objective metaheuristics used in our study, the parameter settings used in the experiments, the benchmark problems, and the methodology adopted in the tests. Section 3 provides an analysis of the results obtained. The conclusions and potential lines for future work are presented in Section 4.

2 Experimentation

In this section, we present the multi-objective metaheuristics tested and describe the parameter settings used in the experiments, the benchmark problems, and the methodology we have followed in the tests.

2.1 Multi-objective Metaheuristics

To carry out our study, we have selected the two most widely known and used metaheuristics in the field: NSGA-II [2] and SPEA2 [20], and we have compared them to other classical algorithms: PAES [12, 13], and to three other modern algorithm: OMOPSO [16], AbYSS [15], and MOCell [5]. We do not describe them here due to space restrictions. Authors unfamiliar with them should revise the indicated references.

We have used the implementation of these algorithms provided by jMetal [5], a Java-based framework for developing metaheuristics for solving multi-objective optimization problems[3].

[3] jMetal is available for download (free of charge) at the following website: `http://neo.lcc.uma.es/metal/`.

2.2 Parameterization

We have chosen a set of parameter settings that aims at guaranteeing a fair comparison among the algorithms. All the evolutionary algorithms (NSGA-II, SPEA2, PESA-II, and MOCell) use an internal population of size equal to 100; OMOPSO is configured with 100 particles and with a maximum number of 100 leaders also. The size of the archive in PAES is 100 as well. AbYSS uses a population size of 20, which is also the size of the RefSet; the size of the external archive is 100. In MOCell a toroidal grid of 100 individuals (10×10) has been chosen for structuring the population, and an archive of 100 individual is used.

For the GAs, we have used SBX and polynomial mutation [4] 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 $p_m = 1/n$, where n is the number of decision variables. In PAES, we have also used the polynomial mutation operator, with the same distribution index. OMOPSO makes use of two types of mutation operators: uniform and non-uniform. AbYSS uses polynomial mutation in the local search procedure and SBX as its solution recombination method.

2.3 Test Problems

The benchmarking MOPs used to evaluate the six metaheuristic algorithms have been proposed in the especialized literature of the area. They are the ZDT [19], DTLZ [3], and WFG [9] test suites. The two latter families of MOPs have been used with their bi-objective formulation.

2.4 Methodology

Since our main interest is to analyze the convergence speed of the metaheuristics studied, it is important to define first what we mean by convergence in this case, and to ensure that such definition allows us to measure it in a quantitative and meaningful way. Our proposal is to establish a stopping condition based on the *high quality* of the approximation of the Pareto front found, and we have used the hypervolume [21] quality indicator for that purpose.

In our experiments, each algorithm is executed until a maximum of 1,000,000 function evaluations have been performed. At every 100 evaluations (that is, at 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 at each generation, whereas in PAES, AbYSS, and MOCell, we have used the external population and, in OMOPSO, the leaders archive. We consider as stopping condition either reaching a hypervolume value of 98% of the hypervolume of the true Pareto front or computing the 1,000,000 evaluations previously indicated.

Using the hypervolume as the stopping condition allows us to obtain a *hit rate* for the algorithms, i.e., the percentage of successful executions. An execution is successful if the algorithm stops before reaching 1,000,000 function evaluations.

This way, we can measure the robustness of the techniques when solving the test problems adopted.

We have performed 100 independent runs of each algorithm for 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 a certain level of confidence. Next, we describe the statistical tests 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 they follow a normal distribution, the Levene test checks for the homogeneity of the variances. If the 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 in order 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 the '+' symbol in the last column in the tables containing the results; conversely, a '$-$' symbol 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, since some samples are normal and others are not. We have also performed a post-hoc testing phase (but not included because of space constrains) using the `multcompare` function provided by Matlab ©, which allows for a multiple comparison of samples. In general, it can be said that the differences of the best algorithms with respect of the others for each MOP are statistically significant at 95% of confidence level.

3 Analysis of results

In this section, we analyze the obtained results. Table 1 shows the median, $\tilde{n}$, and the interquartile range, IQR, of the number of evaluations needed by the different optimizers when solving all the problems. When an optimizer is not able to reach acceptable fronts upon performing 1,000,000 function evaluations after the 100 independent runs, its result appears as '$-$', and it is not taken into account in the statistical tests. Concretely, the '$-$' symbol means that the median of the number of function evaluations is 1,000,000 and the IQR is 0. However, it it worth mentioning that the IQR only considers the values between the 25th and the 75th percentiles, so it is possible that the algorithm was successful only in a few executions (less than 25% of the independent runs executed). To ease the analysis of the results in Table 1, 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. The hit rate is reported in Table 2. A '$\surd$' in a cell means a 100% hit rate, while a '$-$' indicates that the problem could not be solved in none of the 100 independent runs.

Table 1. Median and IQR of the number of evaluations computed by the algorithms.

	NSGA-II	SPEA2	PAES	OMOPSO	AbYSS	MOCell	
Problem	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	
ZDT1	1.43e+4 $_{8.0e+2}$	2.12e+4 $_{1.6e+3}$	1.32e+4 $_{1.1e+4}$	6.80e+3 $_{2.0e+3}$	1.37e+4 $_{1.6e+3}$	1.30e+4 $_{1.2e+3}$	+
ZDT2	2.43e+4 $_{1.8e+3}$	–	1.71e+5 $_{2.0e+5}$	8.90e+3 $_{3.6e+3}$	1.71e+4 $_{2.8e+3}$	1.17e+4 $_{4.0e+3}$	+
ZDT3	1.27e+4 $_{9.0e+2}$	1.72e+4 $_{1.5e+3}$	2.56e+4 $_{2.3e+4}$	9.85e+3 $_{2.7e+3}$	1.27e+4 $_{2.0e+3}$	1.30e+4 $_{1.3e+3}$	+
ZDT4	2.13e+4 $_{5.0e+3}$	2.46e+5 $_{2.6e+5}$	4.41e+4 $_{1.8e+4}$	–	2.28e+4 $_{1.1e+4}$	1.63e+4 $_{5.0e+3}$	+
ZDT6	2.88e+4 $_{1.2e+3}$	5.27e+4 $_{5.5e+3}$	9.95e+3 $_{1.2e+4}$	2.80e+3 $_{1.5e+3}$	1.56e+4 $_{1.2e+3}$	2.09e+4 $_{1.3e+3}$	+
DTLZ1	2.51e+4 $_{9.4e+3}$	–	8.73e+4 $_{1.3e+5}$	1.00e+6 $_{4.7e+4}$	2.37e+4 $_{1.2e+4}$	2.01e+4 $_{7.7e+3}$	+
DTLZ2	8.10e+3 $_{1.2e+3}$	–	3.07e+4 $_{2.0e+4}$	8.20e+3 $_{3.1e+3}$	4.70e+3 $_{9.0e+2}$	5.60e+3 $_{9.0e+2}$	+
DTLZ3	1.18e+5 $_{5.7e+4}$	–	1.00e+6 $_{2.7e+5}$	–	1.19e+5 $_{7.5e+4}$	6.73e+4 $_{2.3e+4}$	+
DTLZ4	8.50e+3 $_{1.4e+3}$	–	–	1.25e+4 $_{3.8e+3}$	4.80e+3 $_{7.5e+2}$	1.00e+6 $_{9.9e+5}$	+
DTLZ5	7.95e+3 $_{1.1e+3}$	–	3.14e+4 $_{2.9e+4}$	8.45e+3 $_{2.9e+3}$	4.65e+3 $_{8.0e+2}$	5.80e+3 $_{8.5e+2}$	+
DTLZ6	1.00e+6 $_{9.7e+5}$	–	7.89e+4 $_{1.5e+5}$	4.10e+3 $_{1.5e+3}$	–	–	+
DTLZ7	1.36e+4 $_{1.0e+3}$	2.35e+4 $_{2.6e+3}$	–	6.15e+3 $_{2.6e+3}$	1.06e+4 $_{1.7e+3}$	1.11e+4 $_{1.6e+5}$	+
WFG1	4.38e+4 $_{1.1e+5}$	2.27e+5 $_{8.2e+5}$	–	–	–	4.16e+4 $_{1.7e+4}$	+
WFG2	1.75e+3 $_{4.5e+2}$	2.40e+3 $_{8.0e+2}$	1.32e+5 $_{1.6e+5}$	1.80e+3 $_{4.0e+2}$	1.85e+3 $_{2.4e+3}$	1.40e+3 $_{8.0e+2}$	+
WFG3	–	–	–	–	–	–	-
WFG4	1.84e+4 $_{6.2e+3}$	–	–	2.23e+5 $_{1.3e+5}$	6.75e+3 $_{2.4e+3}$	1.05e+4 $_{3.1e+3}$	+
WFG5	–	–	–	–	–	–	-
WFG6	1.00e+6 $_{5.2e+5}$	9.05e+4 $_{8.0e+4}$	–	7.30e+3 $_{1.2e+3}$	–	1.00e+6 $_{5.5e+5}$	+
WFG7	1.62e+5 $_{2.7e+5}$	–	–	1.49e+4 $_{2.6e+3}$	9.60e+3 $_{3.4e+3}$	1.21e+4 $_{3.4e+3}$	+
WFG8	–	–	–	–	–	–	+
WFG9	–	–	–	8.93e+4 $_{4.9e+4}$	–	–	+

3.1 ZDT Problems

We start by analyzing the results obtained when solving the ZDT test suite. In this benchmark, OMOPSO is the most outstanding algorithm, because it is the fastest optimizer in four out of the five MOPs of the family. Furthermore, except for the ZDT4 problem, the differences are noticeable when compared with the second best performer, particularly in ZDT1 (52%) and ZDT2 (76%). Despite its good global results, OMOPSO is the only metaheuristic unable to solve ZDT4, a multifrontal problem, in less than 1,000,000 function evaluations, so it is not as robust as the other algorithms. This assessment is corroborated if we examine the hit rate results (see Table 2). MOCell, the cellular GA, is the fastest metaheuristic solving ZDT4. Both NSGA-II and MOCell are the second fastest algorithms in two problems. It is remarkable that PAES, the simplest of the compared algorithms, is the second fastest in ZDT6. Concerning SPEA2 and AbYSS, they do not obtain the best or second best result in any problem.

3.2 DTLZ Test Problems

The DTLZ test suite is composed of seven MOPs, some of them including properties not found in any of the ZDT family. For example, DTLZ1 is a linear problem, and both DTLZ5 and DTLZ6 have degenerate Pareto fronts.

If we focus on convergence speed, AbYSS is the fastest algorithm at reaching 98% the HV of the true Pareto fronts on three out of the seven MOPs from this benchmark, and the second fastest in other two. The second fastest algorithm is MOCell, which requires the lowest number of evaluations on DTLZ3, and DTLZ1, and also is the second fastest solver in two out of the seven problems. OMOPSO obtains the best results in two problems: DTLZ6 and DTLZ7. After them, NSGA-II is the fastest one in two cases and PAES in one.

Table 2. Hit Rate.

Problem	NSGA-II	SPEA2	PAES	OMOPSO	AbYSS	MOCell
ZDT1	√	√	√	√	√	√
ZDT2	√	–	0.99	√	√	√
ZDT3	√	√	√	√	√	√
ZDT4	√	0.99	√	–	√	√
ZDT6	√	√	0.96	√	√	√
DTLZ1	√	–	0.91	0.28	√	√
DTLZ2	√	–	√	√	√	√
DTLZ3	√	–	0.30	0.01	√	√
DTLZ4	0.89	–	–	√	√	0.38
DTLZ5	√	–	√	√	√	√
DTLZ6	0.40	–	0.97	√	0.19	0.10
DTLZ7	0.99	√	0.16	√	0.89	0.76
WFG1	0.83	0.73	–	–	0.21	√
WFG2	√	√	0.99	√	0.98	√
WFG3	–	–	–	–	0.17	–
WFG4	√	–	–	√	√	√
WFG5	–	–	–	–	0.11	–
WFG6	0.34	–	√	√	0.13	0.37
WFG7	0.99	–	–	√	√	√
WFG8	–	–	–	–	0.18	0.12
WFG9	–	–	–	0.99	0.24	0.24

We examine now the robustness of the metaheuristics. According to Table 1, it is noticeable that SPEA2 is only able to solve the DTLZ7 problem in less than 1,000,000 functions evaluations. The second less robust algorithm is PAES, which is unable to solve the DTLZ4 problem satisfactorily. Also, it is remarkable that OMOPSO only finds an accurate front in one of the 100 independent runs performed on problem DTLZ3 (see Table 2). Conversely, according to the hit rate obtained, AbYSS and NSGA-II seem to be the most robust algorithms on this family, followed by MOCell.

3.3 WFG Test Problems

The WFG test suite is composed of nine MOPs having different properties. A first look at the number of evaluations required by the six studied metaheuristics shows that none of them is able to provide accurate Pareto fronts of three problems (WFG3, WFG5, and WFG8) in less than 1,000,000 function evaluations, and many algorithms have difficulties when solving the others, too. This clearly indicates that this benchmark is harder to be solved than both the ZDT and the DTLZ problem families.

Proceeding as in the two previous benchmarks, we start by analyzing the convergence speed. Clearly, the fastest algorithm is MOCell, which requires a lower number of evaluations in two cases, and the second best in two of the problems studied. OMOPSO and AbYSS are the fastest in two out of the nine MOPs. NSGA-II obtains the second lowest number of evaluations in two problems and PAES does the same in one.

If we do not consider the aforementioned unsolved problems, the most robust algorithm is MOCell, which has a 100% hit rate on WFG1, WFG2, WFG4, and WFG7, and hit rates of 0.37, 0.12 and 0.24 on WFG6, WFG8 and WFG9, respectively. It is followed by AbYSS (100% on WFG4 and WFG7) which is the only solver able of finding some accurate fronts in problems WFG3 and WFG5. It is remarkable the behavior of OMOPSO, which obtains a 100% hit rate in practically five problems, but fails when solving WFG1 and WFG8. The less robust algorithms for this benchmark are SPEA2 and PAES. As commented before, those results below a hit rate of 25% are reported in Table 1 as '−', as it happens with AbYSS and WFG1, WFG3, WFG5, WFG7, WFG8, and WFG9.

3.4 Discussion

If we merely make a global ranking of the fastest algorithms in our study, it would be led by MOCell (five best results, six second best ones), OMOPSO (eight best results, one second best), and AbYSS (five best results, two second best ones). The robustness ranking would be headed by NSGA-II in the DTLZ test suite, and by MOCell and AbYSS in the WFG family. In the case of the ZDT problems, these three approaches perform equally well.

These conclusions are relevant, and, we believe that they are the most important contributions of our study. However, although from a practical point of view, the hints of the type "if you want to solve a problem fast, try first MOCell and OMOPSO" or "if you need a robust algorithm, try NSGA-II", can be useful, it would be more interesting to provide some hints (given the characteristics of a given MOP) regarding the algorithm that is more suitable to solve it. The three benchmarks we have used provide us with a range set of problems, having each of them different features. Unfortunately, their analysis in [9] indicate that it is far from simple to make a clear classification of the 21 problems according to their properties (convex, concave, linear, disconnected, multifrontal, separable, etc). In any case, we attempt to draw some (more general) conclusions based on our study, subject to the evident limitations previously indicated.

If we focus on the modality feature, it is present in the following problems: ZDT3, ZDT4, ZDT6, DTLZ1, DLTZ3, DTLZ7, WFG4, and WFG9 (this is also deceptive) [9]. An analysis of the evaluations required to solve these problems, shows that MOCell is the fastest algorithm in ZDT4, DTLZ3, WFG4, and WFG9 and the second fastest in DTLZ1 and DLZ7. On the other hand, OMOPSO fails when solving ZDT4, DTLZ1, and DTLZ3. However, OMOPSO is the fastest algorithm to solve problems ZDT3, ZDT6 and DTLZ7, and it is one of the only two solvers able to achieve a hit rate of 100% on problem WFG9. According to these results, it is not clear whether OMOPSO should be discarded or not when dealing with this type of problems.

There are three problems having disconnected Pareto fronts: ZDT3, DTLZ7, and WFG2. Our study reveals that OMOPSO is the fastest in the first two and the third fastest in the last one. The second algorithm is MOCell, which requires the lowest number of evaluations in problem WFG2, followed by NSGA-II which is the second fastest in problems ZDT3 and WFG2.

4 Conclusions and Future Work

We have evaluated six metaheuristics over a set of 21 MOPs in order to study the performance of the algorithms concerning their convergence speed, i.e., their velocity to reach an accurate Pareto front using a stopping condition based on the hypervolume of the true Pareto front. We have also evaluated the percentage of successful executions or hit rate, which has allowed us to decide about the robustness of the techniques.

In the context of the problems analyzed, the experimentation methodology, and the parameter settings used, we can state that, regarding convergence speed, MOCell, OMOPSO, and AbYSS are the most competitive algorithms. They are the fastest in six and seven out the 21 analyzed problems, respectively, followed by the NSGA-II (the second best in six problems) and PAES (the second best in one). SPEA2 is the only metaheuristic which does not achieve a best or second best result, so it can be considered as the slowest of the compared techniques.

As to the hit rate, NSGA-II is the most robust algorithm in the DTLZ test suite, followed by AbYSS and MOCell. In the WFG family AbYSS and MOCell are the most robust ones, followed by the NSGA-II. The least robust ones are SPEA2 (fails in 14 problems) and PAES (fails in 8 problems).

Taking into account problem properties, we have found out that OMOPSO and MOCell perform well in disconnected problems. Concerning multifrontality, MOCell provides again good values, while OMOPSO has an "all or nothing" behavior: it is either among the best when solving a problem, or it has a hit rate of zero (i.e., it is the worst).

We have presented a first study of the behavior of multi-objective metaheuristics concerning their convergence speed as well as their robustness. A line of future work is to deepen in the study of the features of the problems, to try to determine more precisely which algorithms are more suited to solve a certain type of MOP. Other interesting research line is to analyze the best parameter settings of the algorithms in order to make them to converge faster while maintaining a high degree of robustness.

Acknowledgements

The third author gratefully acknowledges support from CONACyT project no. 45683-Y.

References

1. C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.
2. 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.
3. K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable Test Problems for Evolutionary Multiobjective Optimization. In A. Abraham, L. Jain, and R. Goldberg, editors, *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105–145. Springer, 2005.
4. Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, 2001.

5. 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, Dpto. de Lenguajes y Ciencias de la Computación, University of Málaga, 2006.
6. H. Eskandari, C. D. Geiger, and G. B. Lamont. FastPGA: A dynamic population sizing approach for solving expensive multiobjective optimization problems. In *Evolutionary Multi-Criterion Optimization (EMO 2007)*, LNCS 4403, pages 141–155, 2006.
7. F. Glover and G. A. Kochenberger. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003.
8. A. G. Hernández-Díaz, L. V. Santana-Quintero, C. Coello Coello, R. Caballero, and J. Molina. A New Proposal for Multi-Objective Optimization using Differential Evolution and Rough Sets Theory. In Maarten Keijzer et al., editor, *Genetic and Evolutionary Computation Conference (GECCO'2006)*, pages 675–682, 2006.
9. 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.
10. J. Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.
11. J. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 2006.
12. J. D. Knowles and D. W. Corne. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation. In *1999 Congress on Evolutionary Computation*, pages 98–105, 1999.
13. Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
14. A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. A cellular genetic algorithm for multiobjective optimization. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2006)*, pages 25–36, 2006.
15. 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 (to appear), 2008.
16. M. Reyes Sierra and C. A. Coello Coello. Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance. In *Evolutionary Multi-Criterion Optimization (EMO 2005)*, LNCS 3410, pages 505–519, 2005.
17. L. V. Santana-Quintero, N Ramírez-Santiago, C. A. Coello Coello, J. Molina Luque, and A García Hernández-Díaz. A New Proposal for Multiobjective Optimization Using Particle Swarm Optimization and Rough Sets Theory. In *Parallel Problem Solving from Nature (PPSN IX)*, LNCS 4193, pages 483–492. 2006.
18. G. Toscano-Pulido, C. A. Coello Coello, and L. V. Santana-Quintero. EMOPSO: A Multi-Objective Particle Swarm Optimizer with Emphasis on Efficiency. In *Evolutionary Multi-Criterion Optimization (EMO 2007)*, LNCS 4403, pages 272–285, 2007.
19. E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.
20. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In *EUROGEN 2001*, pages 95–100, 2002.

21. 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.