Málaga, 22 de noviembre de 2008

Informe Ejecutivo

Título: MULTIOBJ-1.1: : Un estudio sobre la velocidad de convergencia de metaheurísticas multiobjetivo

Resumen: Este informe recoge los resultados más relevantes de un estudio sobre la velocidad de convergencia de un conjunto de metaheurísticas de optimización multiobjetivo que representan el estado del arte (NSGA-II, SPEA2, PAES, OMOPSO, AbYSS y MOCell). Tomando un conjunto de 21 problemas pertenecientes a tres conocidos bancos de pruebas (ZDT, DTLZ y WFG), se aborda en primer lugar el fijar un criterio de convergencia basado en el indicador de calidad hypervolumen, para posteriormente realizar un estudio comparativo de todos los algoritmos.

Objetivos:

1. Estudiar la velocidad de convergencia de un seis metaheurísticas multiobjetivo.
2. Establecer un criterio para medir la cómo de rápido converge un algoritmo multiobjetivo y otro para determinar la robustez de los mismos (*hit rate*).

Conclusiones:

1. MOCell, OMOPSO y AbYSS son los algoritmos más rápidos
2. NSGA-II y MOCell son las técnicas más robustas
3. Las metaheurśticas clásicas PAES y SPEA2 son las que peor rendimiento global ofrecen.
4. Los problemas del banco de prueba WFG son los más difíciles de resolver.

Relación con entregables:

Málaga, November 22nd, 2008

Executive Summary

TITLE: MULTIOBJ-1.1: : Studying the Convergence Speed of Multi-Objective Optimization Metaheuristics

ABSTRACT: In report we include the most salient results of studying the convergence speed of six multi-objective metaheuristics representative of the state-of-the-art (NSGA-II, SPEA2, PAES, OMOPSO, AbYSS, and MOCell). A total of 21 problems belonging to three well-known benchkmarks (ZDT, DTLZ, and WFG) have been considered. The study discusses first a convergence speed criterium base on the hypervolume quality indicator; after that, a comparitive study of the considered algorithms is performed.

GOALS:

1. Study the convergence speed of six multi-objective metaheuristics
2. Define criteria to measure convergence speed in the multi-objective context and to decide about the robustness (*hit rate*) of the techniques.

CONCLUSIONS:

1. MOCell, OMOPSO, and AbYSS are the fastest algorithms.
2. NSGA-II and MOCell are the solvers providing better hit rates.
3. SPEA2 and PAES, two classical algorithms, ocuppy the last position in the considered criteria.
4. The WFG problems are harder to solve than those belonging to the ZDT and DTLZ benchmarks.

RELATION WITH DELIVERABLES:

MULTIOBJ-1.1: Studying the Convergence Speed of Multi-Objective Optimization Metaheuristics

DIRICOM

November 2008

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 the objective space, they are collectively known as the Pareto front. Obtaining the Pareto front is the main goal in multi-objective optimization. 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 [**?**]. Among them, metaheuristics [**?**, **?**] 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 [**?**], the Deb-Thiele-Laumanns-Zitzler (DTLZ) problem family [**?**], and the Walking-Fish-Group (WFG) test problems [**?**]. The experimentation methodology typically lies in computing a pre-fixed number of function evaluations and then comparing the obtained results by considering different quality indicators [**?**].

In this report, we study 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 [**?**], and SPEA2 [**?**]), an Evolution Strategy (PAES [**?**]), a Particle Swarm Optimization algorithm (OMOPSO [**?**]), a Scatter Search method (AbYSS [**?**]), and a cellular Genetic Algorithm (MOCell [**?**]). This report summarizes the work presented in [**?**].

2 Measuring Convergence Speed

In our study, we define a convergence criterium based the hypervolume quality indicator [**?**]. 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 fronts obtained for problem ZDT1 with different percentages of hypervolume. We can observe that the front with a hypervolume of 98.26% represents a reasonable approximation to the true Pareto frond in terms of convergence and diversity of solutions. The same conclusions can ve drawn when applying this process to the rest of ZDT problems.

Consequently, 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.

3 Estudied Algorithms

In this section we describe briefly the six metaheuristics that we have considered in this study. We have used the implementation of these algorithms provided by jMetal [**?**], a Java-based framework aimed at multi-objective optimization[1].

The NSGA-II algorithm was proposed by Deb *et al.* [**?**]. It is 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 the 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.

[1] jMetal is freely available to download at the following Web address: `http://neo.lcc.uma.es/metal/`.

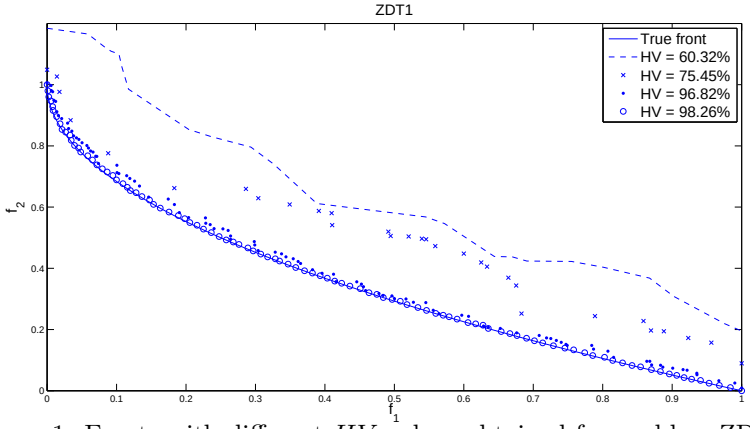

Figure 1: Fronts with different HV values obtained for problem ZDT1

SPEA2 was proposed by Zitler *et al.* in [**?**]. In this algorithm, each individual has assigned a fitness value that is the sum of its strength raw fitness and 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.

PAES is a metaheuristic proposed by Knowles and Corne [**?**]. 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 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 [**?**]. Its main features include the use of the crowding distance of 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 ϵ-dominance, being the leader population obtained after the algorithm has finished the result of the execution of the technique.

MOCell [**?**] is a cellular genetic algorithm (cGA). As other 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 [**?**], in which the cells are explored sequentially (asynchronously). The selection is based on taking an individual from the neighborhood of the current cell and another one chosen randomly from the archive. After applying the genetic crossover and mutation operators, the new offspring is compared with the current one, replacing it if better; in the case of both solution be non-dominated, the worst individual in the neighborhood is replaced by the current one. In this two cases, the new individual is inserted into the archive.

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

4 Experiments

In our experiments, each algorithm has been 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 to the desired hypervolume value 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 metaheuristic stops before reaching 1,000,000 function evaluations.

We have performed 100 independent runs of each algorithm for each problem instance. To present the results, we use the median, $\tilde{x}$, and interquartile range, IQR, as measures of location (or central tendency) and statistical dispersion, respectively. For further details about the parameter settings of the algorithms and the statistical methodology followed, please refer to [**?**].

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

Problem	NSGA-II $\bar{x}_{IQR}$	SPEA2 $\bar{x}_{IQR}$	PAES $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	AbYSS $\bar{x}_{IQR}$	MOCell $\bar{x}_{IQR}$	
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 $_{4.9e+4}$	–	–	+

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

Table 1 shows the median and the interquartile range 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 25^{th} and the 75^{th} 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 rates are reported in Table 2. A '√' 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.

The results presented in these tables can be summarized in Table 3, which includes a ranking of the techniques. If we consider the 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), and AbYSS (five best results, two second best ones). The hit rate ranking would be headed by NSGA-II. It performs the best in the DTLZ test suite, as MOCell and AbYSS do in the WFG family. In the case of the ZDT problems, these three approaches perform equally well. It is worth mentioning that AbYSS is the only metaheuristic providing a hit rate higher than 0% in all the problems.

Table 3: Ranking of the algorithms

Rank	Convergence speed	Hit rate
1	MOCell	NSGA-II
2	OMOPSO	MOCell
3	AbYSS	AbYSS
4	NSGA-II	OMOPSO
5	PAES	PAES
6	SPEA2	SPEA2

5 Conclusions

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.

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, followed by NSGA-II and PAES. SPEA2 is the last algorithm in the ranking, so it can be considered as the slowest of the compared techniques.

As to the hit rate, NSGA-II is the most salient algorithm in the DTLZ test suite, followed by AbYSS and MOCell. In the WFG family, AbYSS and MOCell are the first one in the ranking, followed by NSGA-II. The algorithm providing the worst behavior are SPEA2 (fails in 14 problems) and PAES (fails in 8 problems).