Multi-Objective Particle Swarm Optimizers: An Experimental Comparison

Juan J. Durillo[1], José García-Nieto[1], Antonio J. Nebro[1]
Carlos A. Coello Coello[2⋆], Francisco Luna[1], and Enrique Alba[1]

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

Abstract. Particle Swarm Optimization (PSO) has received increasing attention in the optimization research community since its first appearance in the mid-1990s. Regarding multi-objective optimization, a considerable number of algorithms based on Multi-Objective Particle Swarm Optimizers (MOPSOs) can be found in the specialized literature. Unfortunately, no experimental comparisons have been made in order to clarify which MOPSO version shows the best performance. In this paper, we use a benchmark composed of three well-known problem families (ZDT, DTLZ, and WFG) with the aim of analyzing the search capabilities of six representative state-of-the-art MOPSOs, namely, NSPSO, SigmaMOPSO, OMOPSO, AMOPSO, MOPSOpd, and CLMOPSO. We additionally propose a new MOPSO algorithm, called SMPSO, characterized by including a velocity constraint mechanism, obtaining promising results where the rest perform inadequately.

Key words: Particle Swarm Optimization, Multi-Objective Optimization, Comparative Study

1 Introduction

The relative simplicity and competitive performance of the Particle Swam Optimization (PSO) [11] algorithm as a single-objective optimizer have favored the use of this bio-inspired technique when dealing with many real-word optimization problems [17]. A considerable number of these optimization problems require the optimization of more than one objective at the same time which are in conflict with respect to each other. These properties, along with the fact that PSO is a population-based metaheuristic, have made it a natural candidate to be extended for multi-objective optimization. Since the first proposed Multi-Objective Particle Swarm Optimizer (MOPSO) developed by Moore and Chapman in 1999 [15], more than thirty different MOPSOs have been reported in the specialized literature. Reyes and Coello [17] carried out a survey of the

⋆ The fourth author is also affiliated to the UMI 1375 CNRS.

existing MOPSOs, providing a complete taxonomy of such algorithms. In that work, the authors considered as the main features of all existing MOPSOs the following ones: the existence of an external archive of non-dominated solutions, the selection strategy of non-dominated solutions as leaders for guiding the swarm, the neighborhood topology, and the existence or not of a mutation operator.

In this work, we are interested in analyzing six representative state-of-the-art MOPSOs in order to provide hints about their search capabilities. Five of them were selected from Reyes and Coello's survey, namely: NSPSO [14], SigmaMOPSO [16], OMOPSO [18], AMOPSO [19], and MOPSOpd [1]. An approach not covered in the survey is also compared: MOCLPSO [9].

With the aim of assessing the performance of these algorithms, we have used three benchmarks of multi-objective functions covering a broad range of problems with different features (concave, convex, disconnected, deceptive, etc.). These benchmarks include the test suites Zitzler-Deb-Thiele (ZDT) [20], the Deb-Thiele-Laumanns-Zitzler (DTLZ) problem family [5], and the Walking-Fish-Group (WFG) test problems [10]. The experimental methodology we have followed consists of computing a pre-fixed number of function evaluations and then comparing the obtained results by considering three different quality indicators: additive unary epsilon [13], spread [4], and hypervolume [21]. The results of our study reveal that many MOPSOs have difficulties when facing some multi frontal problems. We analyze this issue and propose a new algorithm, called SMPSO, which incorporates a velocity constraint mechanism. We find that SMPSO shows a promising behavior on those problems where the other algorithms fail.

The remainder of this paper is organized as follows. Section 2 includes basic background about PSO and MOPSO algorithms. In Section 3, we briefly review the studied approaches focusing on their main features. Section 4 is devoted to the experimentation, including the parameter settings and the methodology adopted in the statistical tests. In Section 5, we analyze the obtained results regarding the three quality indicators mentioned before. The results are discussed in Section 6, where a new MOPSO based on a constraint velocity mechanism is introduced. Finally, Section 7 contains the conclusions and some possible paths for future work.

2 PSO Background

PSO is a population-based metaheuristic inspired on the social behavior of birds within a flock. In a PSO algorithm each potential solution to the problem is called a *particle* and the population of solutions is called a *swarm*. The way in which PSO updates the particle $\boldsymbol{x}_i$ at the generation t is through the formula:

$$\boldsymbol{x}_i(t) = \boldsymbol{x}_i(t-1) + \boldsymbol{v}_i(t) \tag{1}$$

where the factor $\boldsymbol{v}_i(t)$ is known as velocity and it is given by

$$\boldsymbol{v}_i(t) = w * \boldsymbol{v}_i(t-1) + C1 * r1 * (\boldsymbol{x}_{pbest_i} - \boldsymbol{x}_i) + C2 * r2 * (\boldsymbol{x}_{gbest_i} - \boldsymbol{x}_i) \tag{2}$$

Algorithm 1 Pseudocode of a general PSO algorithm.

```
1: initializeSwarm()
2: locateLeader()
3: generation = 0
4: while generation < maxGenerations do
5:    for each particle do
6:       updatePosition() // flight (Formulas 1 and 2)
7:       evaluation()
8:       updatePbest()
9:    end for
10:   updateLeader()
11:   generation ++
12: end while
```

In this formula, $\boldsymbol{x}_{pbest_i}$ is the best solution that $\boldsymbol{x}_i$ has viewed, $\boldsymbol{x}_{gbest_i}$ is the best particle (also known as the *leader*) 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.

Algorithm 1 describes the pseudo-code of a general single-objective PSO. The algorithm starts by initializing the swarm (Line 1), which includes both the positions and velocities of the particles. The corresponding *pbest* of each particle is initialized, as well as the leader (Line 2). Then, during a maximum number of iterations, each particle *flies* through the search space updating its position (Line 6), it is evaluated (Line 7), and its *pbest* is also calculated (Lines 6-8). At the end of each iteration, the leader is updated. As commented before, the leader can be the *gbest* particle in the swarm. However, it can be a different particle depending on the *social structure* of the swarm (i.e., the topology of the neighborhood of each particle) [12].

To apply a PSO algorithm in multi-objective optimization the previous scheme has to be modified to cope with the fact that the solution of a problem with multiple objectives is not a single one but a set of non-dominated solutions. Issues that have to be considered are [17]:

1. How to select the set of particles to be used as leaders?
2. How to retain the non-dominated solutions found during the search?
3. How to maintain diversity in the swarm in order to avoid convergence to a single solution?

The pseudo-code of a general MOPSO is included in Algorithm 2. After initializing the swarm (Line 1), the typical approach is to use an external archive to store the leaders, which are taken from the non-dominated particles in the swarm. After initializating the leaders archive (Line 2), some quality measure has to be calculated (Line 3) for all the leaders to select usually one leader for each particle of the swarm. In the main loop of the algorithm, the flight of each particle is performed after a leader has been selected (Lines 7-8) and, optionally, a mutation or *turbulence* operator can be applied (Line 9); then, the particle is evaluated and its corresponding *pbest* is updated (Lines 10-11). After each

Algorithm 2 Pseudocode of a general MOPSO algorithm.

```
1: initializeSwarm()
2: initializeLeadersArchive()
3: determineLeadersQuality()
4: generation = 0
5: while generation < maxGenerations do
6:   for each particle do
7:     selectLeader()
8:     updatePosition() // flight (Formulas. 1 and 2)
9:     mutation()
10:     evaluation()
11:     updatePbest()
12:   end for
13:   updateLeadersArchive()
14:   determineLeadersQuality()
15:   generation ++
16: end while
17: returnArchive()
```

iteration, the set of leaders is updated and the quality measure is calculated again (Lines 13-14). After the termination condition, the archive is returned as the result of the search. For further details about the operations contained in the MOPSO pseudocode, please refer to [17].

3 Studied Approaches

The studied approaches we have considered in this work can be classified as *Pareto-based* MOPSOs [17]. The basic idea, commonly found in all these algorithms, is to select as leaders the particles that are non-dominated with respect to the swarm. However, this leader selection scheme can be slightly different depending on the additional information each algorithm includes on its own mechanism (e.g., information provided by a density estimator). We summarize next the main features of the considered MOPSOs:

- **Non-dominated Sorting PSO:** NSPSO [14] incorporates the main mechanisms of NSGA-II [4] to a PSO algorithm. In this approach, once a particle has updated its position, instead of comparing the new position only against the *pbest* position of the particle, all the *pbest* positions of the swarm and all the new positions recently obtained are combined in just one set (given a total of $2N$ solutions, where N is the size of the swarm). Then, NSPSO selects the best solutions among them to conform the next swarm (by means of a non-dominated sorting). This approach also selects the leaders randomly from the leaders set (stored in an external archive) among the best of them, based on two different mechanisms: a niche count and a nearest neighbor density estimator. This approach uses a mutation operator that is applied at each iteration step only to the particle with the smallest density estimator value.
- **SigmaMOPSO:** In SigmaMOPSO [16], a sigma value is assigned to each particle of the swarm and of an external archive. Then, a given particle of the swarm selects as its leader to the particle of the external archive with the

closest sigma value. The use of the sigma values makes the selection pressure of PSO even higher, which may cause premature convergence in some cases. To avoid this, a turbulence operator is used, which is applied on the decision variable space.

- **Optimized MOPSO:** The main features of OMOPSO [18] include the use of the crowding distance of NSGA-II to filter out leader solutions and the combination of two 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, being the leaders archive the result of the execution of the technique.

- **Another MOPSO:** AMOPSO [19] uses the concept of Pareto dominance to determine the flight direction of a particle. The authors adopt clustering techniques to divide the population of particles into several swarms. This aims at providing a better distribution of solutions in the decision variable space. Each sub-swarm has its own set of leaders (non-dominated particles). In each sub-swarm, a PSO algorithm is executed (leaders are randomly chosen) and, at some point, the different sub-swarms exchange information: the leaders of each swarm are migrated to a different swarm in order to variate the selection pressure. Also, this approach does not use an external archive since elitism in this case is an emergent process derived from the migration of leaders.

- **Pareto Dominance MOPSO:** in MOPSOpd [1], the authors propose methods based exclusively on Pareto dominance for selecting leaders from a non-dominated external archive. Three different selection techniques are presented: one technique that explicitly promotes diversity (called Rounds by the authors), one technique that explicitly promotes convergence (called Random), and finally one technique that is a weighted probabilistic method (called Prob) reaching a compromise between Random and Rounds. Additionally, MOPSOpd uses a turbulence factor that is added to the position of the particles with certain probability; we have used the same operator applied in SigmaMOPSO.

- **Comprehensive Learning MOPSO:** MOCLPSO [9] incorporates a Pareto dominance mechanism to the CLPSO algorithm for selecting leaders from non-dominated external archive. In this approach, a crowding distance method is used to estimate the density of the solutions once the external archive reaches its maximum allowable size. The distance values of all the archive members are calculated and sorted from large to small. The first $Nmax$ (maximum size of archive) members are kept whereas the remaining ones are deleted from the archive. The leaders are randomly chosen from this external archive of non-dominated solutions. In MOCLPSO, no perturbation methods are applied to keep the diversity through the evolution steps.

4 Experimentation

In this section, we detail the parameter settings we have used, as well as the methodology followed in the experiments.

The benchmarking MOPs chosen to evaluate the six MOPSOs have been the aforementioned ZDT [20], DTLZ [5], and WFG [10] test suites, leading to a total number of 21 problems. The two latter families of MOPs have been used with their bi-objective formulation. For assessing the performance of the algorithms, we have considered three quality indicators: additive unary epsilon indicator ($I^1_{\epsilon+}$) [13], spread (Δ) [4], and hypervolume (HV) [21]. The two first indicators measure, respectively, the convergence and the diversity of the resulting Pareto fronts, while the last one measures both convergence and diversity.

All the algorithms have been implemented using jMetal [7], a Java-based framework for developing metaheuristics for solving multi-objective optimization problems.

4.1 Parameterization

We have chosen a common subset of parameter settings which are the same to all the algorithms. Thus, the size of the swarm and the leader archive, when applicable, is fixed to 100 particles, and the stopping condition is always to perform 250 iterations (yielding a total of 25,000 function evaluations). If we consider NSPSO, for example, the swarm size and the number of iterations used in [14] is 200 and 100, respectively. Our approach has been to establish common settings in order to make a fair comparison, keeping the rest of the parameters according to the papers where the algorithms were originally described.

The parameter settings are summarized in Table 1. For those particular parameters that have not been explained, please see the references for further details.

4.2 Methodology

To assess the search capabilities of the algorithms, we have made 100 independent runs of each experiment, and we have obtained the median, $\tilde{x}$, and interquartile range, IQR, as measures of location (or central tendency) and statistical dispersion, respectively. Since we are dealing with stochastic algorithms and we want to provide the results with statistical confidence, the following statistical analysis has been performed in all this work [6]. Firstly, a Kolmogorov-Smirnov test is applied in order to check whether the values of the results follow a normal (Gaussian) distribution or not. If the distribution is normal, the Levene test checks for the homogeneity of the variances. If samples have equal variance (positive Levene test), an ANOVA test is done; otherwise a Welch test is performed. For non-Gaussian distributions, the non-parametric Kruskal-Wallis test is used to compare the medians of the algorithms. We always consider a confidence level of 95% (i.e., significance level of 5% or p-value below 0.05) in the statistical tests. Successful tests are marked with '+' symbols in the last column in all the

Table 1. Parameterization.

Common parameters	
Swarm size	100 Particles
Iterations	250
NSPSO [14]	
Variant	CD (Crowding distance)
C_1, C_2	2.0
w	Decreased from 1.0 to 0.4
SigmaMOPSO [16]	
Archive size	100
C_1, C_2	2.0
w	0.4
Mutation	$newPosition = position + rand(0.0, 1.0) * position$
Mutation probability	0.05
OMOPSO [18]	
Archive size	100
C_1, C_2	$rand(1.5, 2.0)$
w	$rand(0.1, 0.5)$
Mutation	uniform + non-uniform + no mutation
Mutation probability	Each mutation is applied to 1/3 of the swarm
AMOPSO [19]	
Number of subswarms	5
C_1, C_2	2.0
w	0.4
MOPSOpd [1]	
Archive Size	100
C_1, C_2	1.0
w	0.5
Mutation	$newPosition = position + rand(0.0, 1.0) * position$
Mutation probability	0.05
Selection method	Rounds
MOCLPSO [9]	
Archive Size	100
C_1, C_2	N/A
w	0.9 to 0.2

tables containing the results; conversely, '-' means that no statistical confidence was found (p-value > 0.05). The best result for each problem has a gray colored background. For the sake of a better understanding of the results, we have also used a clearer grey background to indicate the second best result.

To further analyze the results statistically, we have also included a post-hoc testing phase which allows for a multiple comparison of samples [8]. We have used the `multcompare` function provided by Matlab© for that purpose.

5 Computational results

This section is devoted to evaluating and analyzing the results of the experiments. We start by discussing the values obtained after applying the $I^1_{\epsilon+}$ quality indicator, which are contained in Table 2. We can observe that OMOPSO clearly outperforms the rest of MOPSOs according to this indicator, achieving the lowest (best) values in 13 out of the 21 problems composing the benchmark. It also obtains six second best values. The next best performing algorithms are SigmaMOPSO, MOPSOpd, and AMOPSO, which get similar numbers of best and second best results. Thus, we can claim that OMOPSO produces solution sets having better convergence to the Pareto fronts in most of the benchmark

Table 2. Median and interquartile range of the $I^1_{\epsilon+}$ quality indicator.

Problem	NSPSO $\bar{x}_{IQR}$	SigmaMOPSO $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	AMOPSO $\bar{x}_{IQR}$	MOPSOpd $\bar{x}_{IQR}$	MOCLPSO $\bar{x}_{IQR}$	
ZDT1	$4.57e-1_{3.7e-1}$	$3.07e-2_{2.6e-2}$	$6.36e-3_{5.1e-4}$	$2.41e-1_{8.0e-2}$	$6.75e-2_{1.6e-2}$	$3.74e-1_{8.8e-2}$	+
ZDT2	$1.54e+0_{8.5e-1}$	$1.00e+0_{0.0e+0}$	$6.19e-3_{5.4e-4}$	$6.33e-1_{8.3e-1}$	$1.00e+0_{8.9e-1}$	$6.45e-1_{1.4e-1}$	+
ZDT3	$9.14e-1_{4.1e-1}$	$9.75e-1_{8.3e-1}$	$1.32e-2_{7.7e-3}$	$7.30e-1_{3.5e-1}$	$1.66e-1_{1.1e-1}$	$5.97e-1_{2.0e-1}$	+
ZDT4	$4.14e+1_{1.6e+1}$	$8.30e+0_{6.8e+0}$	$5.79e+0_{4.3e+0}$	$1.21e+1_{7.6e+0}$	$4.23e+0_{2.1e+0}$	$1.71e+1_{1.3e+1}$	+
ZDT6	$1.81e-1_{3.2e-1}$	$5.91e-3_{1.1e-3}$	$4.65e-3_{4.2e-4}$	$1.69e-1_{6.0e-2}$	$1.21e-1_{7.0e-2}$	$3.38e+0_{3.8e-1}$	+
DTLZ1	$2.30e+1_{8.0e+0}$	$2.54e+1_{1.3e+1}$	$1.92e+1_{1.1e+1}$	$8.46e+0_{1.9e+1}$	$1.72e+1_{1.1e+1}$	$2.12e+1_{8.0e+0}$	+
DTLZ2	$4.41e-2_{6.5e-2}$	$1.13e-1_{9.1e-2}$	$6.72e-3_{9.1e-4}$	$1.25e-1_{3.9e-2}$	$9.26e-2_{5.1e-2}$	$3.95e-2_{3.8e-2}$	+
DTLZ3	$1.04e+2_{6.2e+1}$	$1.79e+2_{7.5e+1}$	$8.86e+1_{9.5e+1}$	$4.41e+1_{9.0e+1}$	$1.23e+2_{6.5e+1}$	$2.37e+2_{5.7e+1}$	+
DTLZ4	$8.91e-2_{5.9e-2}$	$3.00e-1_{4.5e-2}$	$3.18e-2_{1.0e-2}$	$2.20e-1_{1.1e-1}$	$6.33e-2_{3.0e-2}$	$2.56e-2_{8.6e-3}$	+
DTLZ5	$3.92e-2_{3.6e-2}$	$1.11e-1_{9.8e-2}$	$6.62e-3_{8.9e-4}$	$1.22e-1_{4.3e-2}$	$9.10e-2_{4.0e-2}$	$3.31e-2_{3.0e-2}$	+
DTLZ6	$1.47e+0_{7.9e-1}$	$1.00e+0_{2.9e-1}$	$5.36e-3_{4.8e-4}$	$1.75e-1_{9.1e-1}$	$1.57e+0_{1.3e+0}$	$4.77e+0_{3.2e-1}$	+
DTLZ7	$1.33e+0_{1.4e+0}$	$1.27e+0_{2.7e-2}$	$7.13e-3_{6.8e-4}$	$3.00e-1_{1.9e-1}$	$1.65e-1_{1.1e-1}$	$4.94e-1_{1.0e-1}$	+
WFG1	$1.36e+0_{7.7e-2}$	$1.00e+0_{9.3e-2}$	$1.35e+0_{4.9e-2}$	$1.53e+0_{3.0e-2}$	$1.10e+0_{2.0e-1}$	$1.31e+0_{5.1e-2}$	+
WFG2	$1.67e-2_{5.5e-3}$	$4.87e-2_{3.6e-2}$	$1.04e-2_{1.7e-3}$	$3.57e-1_{1.8e-1}$	$7.24e-2_{2.1e-2}$	$5.96e-2_{3.7e-2}$	+
WFG3	$2.00e+0_{5.3e-4}$	$2.00e+0_{4.2e-3}$	$2.00e+0_{1.6e-5}$	$2.10e+0_{1.2e-1}$	$2.00e+0_{4.5e-5}$	$2.12e+0_{2.0e-1}$	+
WFG4	$1.09e-1_{1.8e-2}$	$6.06e-2_{2.7e-2}$	$5.98e-2_{1.5e-2}$	$3.21e-1_{8.1e-2}$	$5.57e-2_{1.8e-2}$	$8.04e-2_{2.4e-2}$	+
WFG5	$8.34e-2_{2.0e-2}$	$6.36e-2_{1.2e-3}$	$6.37e-2_{9.0e-4}$	$6.24e-1_{3.3e-1}$	$3.24e-1_{3.5e-1}$	$2.57e-1_{2.2e-1}$	+
WFG6	$1.04e-1_{6.6e-2}$	$5.60e-1_{3.8e-1}$	$1.79e-2_{2.5e-3}$	$4.63e-1_{1.3e-1}$	$3.30e-1_{2.6e-1}$	$2.40e-1_{2.3e-1}$	+
WFG7	$4.05e+2_{6.1e+3}$	$5.75e+2_{1.8e+2}$	$1.94e+2_{1.7e+3}$	$3.77e+1_{1.5e+1}$	$6.16e+1_{1.1e+1}$	$2.44e+1_{3.4e+1}$	+
WFG8	$5.24e-1_{9.2e-2}$	$5.66e-1_{1.9e-1}$	$5.06e-1_{3.4e-2}$	$8.30e-1_{1.2e-1}$	$5.39e-1_{2.3e-2}$	$7.70e-1_{6.0e-2}$	+
WFG9	$6.38e-2_{2.0e-2}$	$2.89e-2_{1.7e-3}$	$2.95e-2_{2.5e-3}$	$3.25e-1_{2.5e-1}$	$1.11e-1_{4.6e-2}$	$1.49e-1_{2.1e-1}$	+

problems considered in our study. All the results have statistical significance, as it can be seen in the last column, where only '+ ' symbols are found.

The values obtained after applying the Δ quality indicator are included in Table 3. We can observe again that OMOPSO is clearly the best performing algorithm, yielding the lowest (best) values in 16 out of the 21 problems. Considering the next algorithms according to the best and second best indicator values, we find SigmaMOPSO, NSPSO, and MOCLPSO. AMOPSO is the worst performer according to the Δ indicator, not achieving any best nor second best result.

Table 3. Median and interquartile range of the Δ quality indicator.

Problem	NSPSO $\bar{x}_{IQR}$	SigmaMOPSO $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	AMOPSO $\bar{x}_{IQR}$	MOPSOpd $\bar{x}_{IQR}$	MOCLPSO $\bar{x}_{IQR}$	
ZDT1	$7.19e-1_{1.0e-1}$	$4.11e-1_{3.9e-1}$	$1.00e-1_{1.4e-2}$	$9.57e-1_{1.6e-1}$	$6.03e-1_{1.1e-1}$	$7.70e-1_{6.4e-2}$	+
ZDT2	$9.82e-1_{9.4e-2}$	$1.00e+0_{0.0e+0}$	$9.45e-2_{1.8e-2}$	$1.00e+0_{6.0e-2}$	$1.00e+0_{2.8e-1}$	$8.03e-1_{7.4e-2}$	+
ZDT3	$8.17e-1_{9.7e-2}$	$1.09e+0_{3.6e-1}$	$7.35e-1_{5.2e-2}$	$9.00e-1_{1.5e-1}$	$8.59e-1_{6.7e-2}$	$8.85e-1_{5.7e-2}$	+
ZDT4	$9.53e-1_{8.0e-2}$	$1.00e+0_{3.3e-3}$	$8.78e-1_{5.2e-2}$	$1.03e+0_{2.5e-2}$	$1.00e+0_{2.4e-2}$	$9.32e-1_{8.2e-2}$	+
ZDT6	$1.39e+0_{6.6e-2}$	$2.89e-1_{3.6e-1}$	$8.78e-2_{1.2e+0}$	$1.12e+0_{1.5e-1}$	$1.20e+0_{2.7e-1}$	$9.67e-1_{4.1e-2}$	+
DTLZ1	$8.38e-1_{1.2e-1}$	$1.14e+0_{1.7e-1}$	$7.77e-1_{1.1e-1}$	$1.13e+0_{2.6e-1}$	$8.72e-1_{2.0e-1}$	$7.90e-1_{7.2e-2}$	+
DTLZ2	$6.02e-1_{1.5e-1}$	$1.01e+0_{1.4e-1}$	$1.81e-1_{2.3e-2}$	$1.15e+0_{1.8e-1}$	$1.21e+0_{8.6e-2}$	$7.92e-1_{8.7e-2}$	+
DTLZ3	$9.31e-1_{2.0e-1}$	$1.23e+0_{1.6e-1}$	$7.90e-1_{1.1e-1}$	$1.09e+0_{4.3e-1}$	$8.55e-1_{1.3e-1}$	$7.69e-1_{8.5e-2}$	+
DTLZ4	$7.17e-1_{1.7e-1}$	$1.41e+0_{8.0e-1}$	$6.77e-1_{7.9e-2}$	$1.46e+0_{2.7e-1}$	$1.10e+0_{9.2e-2}$	$7.33e-1_{5.3e-2}$	+
DTLZ5	$5.99e-1_{9.3e-2}$	$1.00e+0_{1.7e-1}$	$1.77e-1_{2.6e-2}$	$1.16e+0_{1.9e-1}$	$1.21e+0_{9.3e-2}$	$7.89e-1_{8.9e-2}$	+
DTLZ6	$8.18e-1_{4.0e-1}$	$1.28e+0_{1.0e+0}$	$1.18e-1_{1.7e-2}$	$1.23e+0_{4.4e-1}$	$8.35e-1_{1.5e-1}$	$8.04e-1_{7.2e-2}$	+
DTLZ7	$9.08e-1_{1.6e-1}$	$7.96e-1_{2.4e-1}$	$5.21e-1_{6.8e-3}$	$1.02e+0_{2.4e-1}$	$7.95e-1_{1.3e-1}$	$8.51e-1_{7.0e-2}$	+
WFG1	$1.14e+0_{5.5e-2}$	$7.50e-1_{1.2e-1}$	$1.17e+0_{6.0e-2}$	$1.30e+0_{3.9e-2}$	$1.16e+0_{7.8e-2}$	$1.12e+0_{4.2e-2}$	+
WFG2	$8.65e-1_{9.0e-2}$	$9.61e-1_{8.5e-2}$	$7.64e-1_{5.5e-3}$	$9.94e-1_{1.9e-1}$	$1.22e+0_{7.0e-2}$	$1.11e+0_{5.8e-2}$	+
WFG3	$5.00e-1_{2.6e-2}$	$4.96e-1_{2.5e-2}$	$3.78e-1_{8.7e-3}$	$1.20e+0_{8.7e-2}$	$1.19e+0_{1.3e-1}$	$9.04e-1_{6.2e-2}$	+
WFG4	$6.25e-1_{5.0e-2}$	$5.01e-1_{7.7e-2}$	$5.06e-1_{6.3e-2}$	$1.14e+0_{1.3e-1}$	$4.83e-1_{4.4e-2}$	$6.18e-1_{4.9e-2}$	+
WFG5	$3.59e-1_{4.5e-2}$	$1.44e-1_{2.0e-2}$	$1.44e-1_{2.0e-2}$	$1.03e+0_{1.7e-1}$	$1.13e+0_{2.3e-1}$	$8.06e-1_{9.7e-2}$	+
WFG6	$5.98e-1_{8.1e-2}$	$6.34e-1_{2.1e-1}$	$1.63e-1_{2.5e-2}$	$1.09e+0_{1.7e-1}$	$1.23e+0_{7.0e-2}$	$8.32e-1_{7.6e-2}$	+
WFG7	$3.71e+1_{5.8e+2}$	$4.07e+1_{5.5e+2}$	$1.59e+1_{2.1e+2}$	$1.13e+0_{1.3e+1}$	$1.31e+0_{7.1e+2}$	$9.13e+1_{8.7e+2}$	+
WFG8	$7.19e-1_{8.4e-2}$	$9.08e-1_{1.7e-1}$	$7.93e-1_{8.8e-2}$	$1.02e+0_{1.4e-1}$	$8.68e-1_{6.6e-2}$	$7.88e-1_{5.3e-2}$	+
WFG9	$5.07e-1_{1.3e-1}$	$2.22e-1_{2.6e-2}$	$2.24e-1_{2.7e-2}$	$1.19e+0_{1.5e-1}$	$7.54e-1_{5.2e-2}$	$7.29e-1_{6.3e-2}$	+

After applying a quality indicator that measures convergence and another one that measures diversity, the HV indicator should confirm the previous results.

Table 4. Median and interquartile range of the HV quality indicator.

Problem	NSPSO $\bar{x}_{IQR}$	SigmaMOPSO $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	AMOPSO $\bar{x}_{IQR}$	MOPSOpd $\bar{x}_{IQR}$	MOCLPSO $\bar{x}_{IQR}$	
ZDT1	$1.54e-1_{2.4e-1}$	$6.54e-1_{8.3e-3}$	$6.61e-1_{1.5e-4}$	$3.81e-1_{9.3e-2}$	$5.94e-1_{1.7e-2}$	$3.28e-1_{4.6e-2}$	+
ZDT2	–	–	$3.28e-1_{2.5e-4}$	$4.10e-2_{1.9e-1}$	$0.00e+0_{2.6e-1}$	$6.54e-2_{3.7e-2}$	+
ZDT3	$1.12e-1_{1.2e-1}$	$3.21e-1_{2.3e-1}$	$5.10e-1_{3.8e-3}$	$2.45e-1_{1.1e-1}$	$4.38e-1_{7.2e-2}$	$2.55e-1_{3.2e-2}$	+
ZDT4	–	–	–	–	–	–	-
ZDT6	$3.09e-1_{1.3e-1}$	$4.01e-1_{3.1e-4}$	$4.01e-1_{1.5e-4}$	$2.31e-1_{4.1e-2}$	$3.50e-1_{5.7e-2}$	–	+
DTLZ1	–	–	–	–	–	–	–
DTLZ2	$1.64e-1_{5.9e-2}$	$1.64e-1_{2.1e-2}$	$2.10e-1_{4.5e-4}$	$1.23e-1_{2.4e-2}$	$1.78e-1_{2.5e-2}$	$2.01e-1_{2.3e-3}$	+
DTLZ3	–	–	–	–	–	–	-
DTLZ4	$1.37e-1_{5.1e-2}$	–	$1.96e-1_{6.1e-3}$	$7.62e-2_{9.8e-2}$	$1.90e-1_{9.8e-3}$	$1.96e-1_{4.0e-3}$	+
DTLZ5	$1.71e-1_{3.5e-2}$	$1.65e-1_{2.3e-2}$	$2.11e-1_{5.4e-4}$	$1.22e-1_{2.9e-2}$	$1.77e-1_{2.0e-2}$	$2.01e-1_{2.1e-3}$	+
DTLZ6	–	–	$2.12e-1_{4.4e-5}$	$8.77e-2_{1.5e-1}$	–	–	+
DTLZ7	$1.59e-2_{9.7e-2}$	$2.18e-1_{1.7e-2}$	$3.34e-1_{3.2e-4}$	$2.00e-1_{7.1e-2}$	$2.53e-1_{5.5e-2}$	$1.01e-1_{1.3e-2}$	+
WFG1	$8.98e-2_{8.3e-3}$	$1.21e-1_{2.2e-3}$	$1.04e-1_{1.0e-2}$	$6.22e-2_{7.4e-3}$	$1.69e-1_{7.2e-2}$	$1.01e-1_{5.1e-3}$	+
WFG2	$5.61e-1_{2.5e-3}$	$5.60e-1_{1.7e-3}$	$5.64e-1_{1.0e-4}$	$4.68e-1_{3.9e-2}$	$5.57e-1_{3.6e-3}$	$5.60e-1_{1.8e-3}$	+
WFG3	$4.40e-1_{3.3e-4}$	$4.38e-1_{8.0e-4}$	$4.42e-1_{5.4e-5}$	$4.04e-1_{1.2e-2}$	$4.27e-1_{1.8e-2}$	$4.30e-1_{1.3e-2}$	+
WFG4	$1.78e-1_{7.0e-3}$	$2.00e-1_{1.6e-3}$	$2.02e-1_{1.6e-3}$	$1.27e-1_{1.2e-2}$	$2.07e-1_{1.3e-3}$	$2.00e-1_{2.3e-3}$	+
WFG5	$1.96e-1_{2.8e-4}$	$1.96e-1_{8.8e-5}$	$1.96e-1_{6.3e-5}$	$1.60e-1_{1.7e-2}$	$1.68e-1_{5.9e-2}$	$1.90e-1_{1.9e-3}$	+
WFG6	$1.75e-1_{2.6e-2}$	$1.90e-1_{1.9e-2}$	$2.09e-1_{3.5e-4}$	$9.88e-2_{2.8e-2}$	$1.60e-1_{4.7e-2}$	$2.01e-1_{1.9e-3}$	+
WFG7	$2.03e+1_{2.7e+3}$	$2.02e+1_{1.1e+3}$	$2.09e+1_{1.7e+4}$	$1.14e+1_{1.4e+2}$	$9.49e+2_{4.2e+2}$	$2.01e+1_{2.7e+3}$	+
WFG8	$1.07e-1_{8.7e-3}$	$1.33e-1_{4.2e-3}$	$1.26e-1_{3.0e-3}$	$6.08e-2_{1.9e-2}$	$1.41e-1_{3.0e-3}$	$1.33e-1_{1.9e-3}$	+
WFG9	$2.24e-1_{6.1e-3}$	$2.34e-1_{4.1e-4}$	$2.34e-1_{6.6e-4}$	$1.87e-1_{1.1e-2}$	$2.29e-1_{4.7e-3}$	$2.30e-1_{1.1e-3}$	+

The HV values, included in Table 4, show that OMOPSO generates solution sets with the highest (best) values in 15 out of the 21 problems. Thus, we can state that according to the parameterization, quality indicators, and benchmark problems considered in this work, OMOPSO is clearly the most salient technique among the six considered in our study.

The results corresponding to problems ZDT4, DTLZ1, and DTLZ3 deserve additional comments. We have used the '–' symbol in Table 4 to indicate those experiments in which the HV value is equal to 0, meaning that the solution sets obtained by the algorithms are outside the limits of the Pareto front; when applying the HV indicator these solutions are not taken into account, because otherwise the obtained results would be unreliable. In the case of the three aforementioned problems, none of the six algorithms is able to achieve a HV greater than 0 over the 100 independent runs. We can also see that other problems are difficult to solve by some techniques, e.g., ZDT2 and DTLZ6. The statistical tests indicate that the results of the Δ and HV indicators have statistical confidence. To provide further statistical information, we show in Table 5 those problems for which no statistical differences appear between OMOPSO and the rest of algorithms considering the three quality indicators. It can be observed that statistical differences exist for most of the pair-wise comparisons.

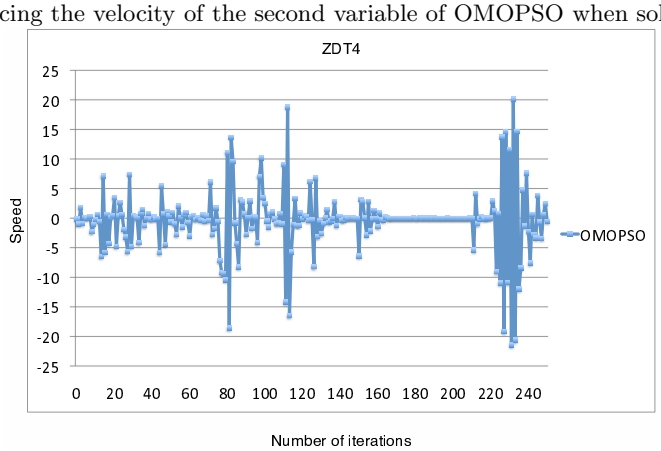
6 Discussion

The conclusion drawn from the analysis of the results in the previous section is that OMOPSO performs the best in our study. In this section, we carry out the same experiments but using OMOPSO and NSGA-II in order to put the results of the first one in context. Such a comparison will allow us to know how competitive OMOPSO is. Before that, we investigate why OMOPSO (as well as the rest of MOPSOs) is unable to solve the ZDT4, DTLZ1, and DTLZ3 problems. If we consider ZDT4, it is a well-known problem characterized by having many

Table 5. Non-successful statistical tests between OMOPSO and the rest of the algorithms.

	$I^1_{\epsilon+}$	Δ	HV
AMOPSO	- DTLZ3 -	- - -	- - -
MOCLPSO	- DTLZ1, DTLZ4 -	ZDT6 DTLZ1, DTLZ3 WFG8	- DTLZ4 WFG1, WFG4
MOPSOpd	ZDT4 DTLZ1, DTLZ3 WFG3, WFG4	- - WFG1, WFG4	- - -
NSPSO	- DTLZ3 WFG1, WFG8	- DTLZ4 -	- - -
SigmaMOPSO	- - WFG4, WFG5, WFG9	ZDT6 - WFG4, WFG5, WFG9	- - WFG5, WFG9

Fig. 1. Tracing the velocity of the second variable of OMOPSO when solving ZDT4.



local optima (it is a multifrontal problem). We have traced the velocity of the second variable in the first particle in OMOPSO when facing the solution of ZDT4 (the second 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 $[0, 1]$). The obtained values after the 250 iterations are depicted in Fig. 1. We can observe that the velocity values suffer a kind of erratic behavior in some points of the execution, alternating very high with very low values. Let us note that the limits of the second variable in ZDT4 are $[-5, +5]$, and the velocity takes values higher than ± 20. The consequence is that this particle is moving to its extreme values continuously, so it is not contributing to guide the search.

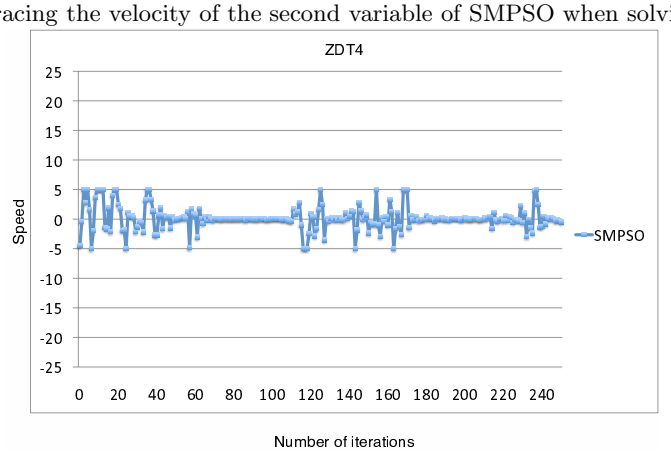
To find out whether this is one of the reasons making OMOPSO unable to solve multi frontal MOPs, we have modified it by including a velocity constraint mechanism, similar to the one proposed in [2]. In addition, the accumulated velocity of each variable j (in each particle) is also bounded by means of the following equation:

$$v_{i,j}(t) = \begin{cases} delta_j & \text{if } v_{i,j}(t) > delta_j \\ -delta_j & \text{if } v_{i,j}(t) \leq -delta_j \\ v_{i,j}(t) & \text{otherwise} \end{cases} \tag{3}$$

where

$$delta_j = \frac{(upper_limit_j - lower_limit_j)}{2} \tag{4}$$

Fig. 2. Tracing the velocity of the second variable of SMPSO when solving ZDT4.



This way, we can ensure an effective new position calculation. We have called the resulting algorithm SMPSO (Speed-constrained Multi-objective PSO). In Fig. 2 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 vanished, so the particle is taking values inside the bounds of the variable and thus it is moving along different regions of the search space. To evaluate the effect of the changes in SMPSO, we have included this algorithm in the comparison between OMOPSO and NSGA-II. We have solved all the problems again, following the same methodology. The parameter settings of NSGA-II are: the population size is 100 individuals, we have used SBX and polynomial mutation [3] as operators for crossover and mutation operators, respectively, and 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 Table 6, we include the median and interquartile range of NSGA-II, OMOPSO, and SMPSO corresponding to the $I^1_{\epsilon+}$ quality indicator. We observe that SMPSO yields the best values in 11 out of the 12 problems comprising the ZDT and DTLZ benchmarks. If we focus on the WFG problems, the lowest

Table 6. NSGA-II vs OMOPSO vs SMPSO: Median and interquartile range of the $I^1_{\epsilon+}$ quality indicator.

Problem	NSGA-II $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	SMPSO $\bar{x}_{IQR}$	
ZDT1	$1.37e-2_{3.0e-3}$	$6.36e-3_{5.1e-4}$	$5.78e-3_{3.8e-4}$	+
ZDT2	$1.28e-2_{2.3e-3}$	$6.19e-3_{5.4e-4}$	$5.66e-3_{3.0e-4}$	+
ZDT3	$8.13e-3_{1.9e-3}$	$1.32e-2_{7.7e-3}$	$6.09e-3_{1.3e-3}$	+
ZDT4	$1.49e-2_{3.0e-3}$	$5.79e+0_{4.3e+0}$	$7.93e-3_{1.4e-3}$	+
ZDT6	$1.47e-2_{2.8e-3}$	$4.65e-3_{4.2e-4}$	$4.87e-3_{4.8e-4}$	+
DTLZ1	$7.13e-3_{1.6e-3}$	$1.92e+1_{1.1e+1}$	$3.73e-3_{5.4e-4}$	+
DTLZ2	$1.11e-2_{2.7e-3}$	$6.72e-3_{9.1e-4}$	$5.81e-3_{6.0e-4}$	+
DTLZ3	$1.04e+0_{1.2e+0}$	$8.86e+1_{9.5e+1}$	$6.57e-3_{1.0e-2}$	+
DTLZ4	$1.13e-2_{9.9e-1}$	$3.18e-2_{1.0e-2}$	$6.54e-3_{8.8e-4}$	+
DTLZ5	$1.05e-2_{2.5e-3}$	$6.62e-3_{8.9e-4}$	$5.77e-3_{6.1e-4}$	+
DTLZ6	$4.39e-2_{3.4e-2}$	$5.36e-3_{4.8e-4}$	$5.22e-3_{4.4e-4}$	+
DTLZ7	$1.04e-2_{2.8e-3}$	$7.13e-3_{6.8e-4}$	$5.46e-3_{4.3e-4}$	+
WFG1	$3.52e-1_{4.6e-1}$	$1.35e+0_{4.9e-2}$	$1.34e+0_{4.6e-2}$	+
WFG2	$7.10e-1_{7.0e-1}$	$1.04e-2_{1.7e-3}$	$1.40e-2_{3.4e-3}$	+
WFG3	$2.00e+0_{5.8e-4}$	$2.00e+0_{1.6e-5}$	$2.00e+0_{3.9e-4}$	+
WFG4	$3.26e-2_{6.7e-3}$	$5.98e-2_{1.5e-2}$	$6.46e-2_{6.0e-3}$	+
WFG5	$8.41e-2_{8.3e-3}$	$6.37e-2_{9.0e-4}$	$6.40e-2_{2.0e-3}$	+
WFG6	$4.14e-2_{1.6e-2}$	$1.79e-2_{2.5e-3}$	$2.56e-2_{3.8e-3}$	+
WFG7	$3.47e+2_{8.1e+3}$	$1.94e+2_{1.7e+3}$	$2.67e+2_{3.8e+3}$	+
WFG8	$3.38e-1_{2.3e-1}$	$5.06e-1_{3.4e-2}$	$4.32e-1_{7.8e-2}$	+
WFG9	$3.73e-2_{7.5e-3}$	$2.95e-2_{2.5e-3}$	$3.15e-2_{3.3e-3}$	+

(best) metric values are shared between OMOPSO (six problems) and NSGA-II (three problems), while SMPSO obtains the second lowest values in 8 out of the 9 WFG problems. These results indicate first, that OMOPSO is competitive when compared against NSGA-II concerning convergence and, second, that the velocity constraint mechanism included in SMPSO improves globally the behavior of OMOPSO considering all the benchmark problems.

The values obtained when applying the Δ and HV indicators are included in Tables 7 and 8, respectively. We can observe that we can practically draw the same conclusions obtained from the $I^1_{\epsilon+}$ indicator, i.e., the algorithms obtain the lowest values in the same problems according to the convergence and diversity indicators. In all the experiments included in this section all the statistical tests are significant, which actually grounds our claims. If we focus in the HV and in those problems in which OMOPSO obtained a value of 0 (ZDT4, DTLZ1, and DTLZ3), we see that the velocity constraint mechanism added to SMPSO allows it to successfully solve them. NSGA-II also outperforms OMOPSO in this sense, only presenting difficulties in DTLZ3.

Table 9 contains those problems for which no statistical confidence exist considering the three algorithms and the three quality indicators. The results of OMOPSO against NSGA-II are significant in all the problems but DTLZ3 with respect to the Δ indicator. Concerning SMPSO, there a few cases where the results are not significant, but they do not alter the analysis carried out.

We can summarize this section by stating that OMOPSO, the most salient of the six MOPSOs studied in this work, is a competitive algorithm when compared with NSGA-II, and we have shown that its search capabilities can be improved by including a velocity constraint mechanism. However, although SMPSO out-

Table 7. NSGA-II vs OMOPSO vs SMPSO: Median and interquartile range of the Δ quality indicator.

Problem	NSGA-II $\tilde{x}_{IQR}$	OMOPSO $\tilde{x}_{IQR}$	SMPSO $\tilde{x}_{IQR}$	
ZDT1	$3.70e-1_{4.2e-2}$	$1.00e-1_{1.4e-2}$	$8.66e-2_{1.6e-2}$	+
ZDT2	$3.81e-1_{4.7e-2}$	$9.45e-2_{1.8e-2}$	$7.46e-2_{1.5e-2}$	+
ZDT3	$7.47e-1_{1.8e-2}$	$7.35e-1_{5.2e-2}$	$7.17e-1_{1.7e-2}$	+
ZDT4	$4.02e-1_{5.8e-2}$	$8.78e-1_{5.2e-2}$	$1.53e-1_{2.2e-2}$	+
ZDT6	$3.56e-1_{3.6e-2}$	$8.78e-2_{1.2e+0}$	$7.28e-1_{1.2e+0}$	+
DTLZ1	$4.03e-1_{6.1e-2}$	$7.77e-1_{1.1e-1}$	$1.14e-1_{1.8e-2}$	+
DTLZ2	$3.84e-1_{3.8e-2}$	$1.81e-1_{2.3e-2}$	$1.59e-1_{2.3e-2}$	+
DTLZ3	$9.53e-1_{1.6e-1}$	$7.90e-1_{1.1e-1}$	$1.98e-1_{3.3e-1}$	+
DTLZ4	$3.95e-1_{6.4e-1}$	$6.77e-1_{7.9e-2}$	$1.70e-1_{2.5e-2}$	+
DTLZ5	$3.79e-1_{4.0e-2}$	$1.77e-1_{2.6e-2}$	$1.58e-1_{2.2e-2}$	+
DTLZ6	$8.64e-1_{3.0e-1}$	$1.18e-1_{1.7e-2}$	$1.14e-1_{2.1e-2}$	+
DTLZ7	$6.23e-1_{2.5e-2}$	$5.21e-1_{6.8e-3}$	$5.20e-1_{2.0e-3}$	+
WFG1	$7.18e-1_{5.4e-2}$	$1.17e+0_{6.0e-2}$	$1.12e+0_{5.0e-2}$	+
WFG2	$7.93e-1_{1.7e-2}$	$7.64e-1_{5.5e-3}$	$8.26e-1_{3.5e-2}$	+
WFG3	$6.12e-1_{3.6e-2}$	$3.78e-1_{8.7e-3}$	$3.84e-1_{6.4e-3}$	+
WFG4	$3.79e-1_{3.9e-2}$	$5.06e-1_{6.3e-2}$	$5.51e-1_{7.0e-2}$	+
WFG5	$4.13e-1_{5.1e-2}$	$1.44e-1_{2.0e-2}$	$1.50e-1_{2.8e-2}$	+
WFG7	$3.79e+1_{4.6e+2}$	$1.59e+1_{2.1e+2}$	$2.44e+1_{3.1e+2}$	+
WFG6	$3.90e-1_{4.2e-2}$	$1.63e-1_{2.5e-2}$	$2.47e-1_{4.1e-2}$	+
WFG8	$6.45e-1_{5.5e-2}$	$7.93e-1_{8.8e-2}$	$8.08e-1_{5.4e-2}$	+
WFG9	$3.96e-1_{4.1e-2}$	$2.24e-1_{2.7e-2}$	$2.46e-1_{2.8e-2}$	+

performs both NSGA-II and OMOPSO in the ZDT and DTLZ problems, it does not achieve the best result in the WFG benchmark. This indicates that more research has to be done. It is also necessary to consider a broader set of problems as well as studying in more depth the effect of modulating the speed in a MOPSO.

7 Conclusions and Further Work

We have evaluated six MOPSO algorithms over a set of three well-known benchmark problems by using three different quality indicators. For each experiment, 100 independent runs have been carried out, and statistical tests have been applied to know more about the confidence of the obtained results. In the context of the problems analyzed, the experimentation methodology, and the parameter settings used, we can state that OMOPSO is clearly the most salient of the six compared algorithms. The results have also shown that all the algorithms are unable to find accurate Pareto fronts for three multi frontal problems. We have studied this issue and we have proposed the use of a velocity constraint mechanism to enhance the search capability in order to solve these problems. The resulting algorithm, SMPSO, shows significant improvements when compared with respect to OMOPSO and NSGA-II. As part of our future work, we plan to study the convergence speed of MOPSO algorithms in order to determine whether they are faster than other multi-objective evolutionary algorithms in reaching the Pareto front of a problem.

Table 8. NSGA-II vs OMOPSO vs SMPSO: Median and interquartile range of the HV quality indicator.

Problem	NSGA-II $\tilde{x}_{IQR}$	OMOPSO $\tilde{x}_{IQR}$	SMPSO $\tilde{x}_{IQR}$	
ZDT1	$6.59e-1_{4.4e-4}$	$6.61e-1_{1.5e-4}$	$6.62e-1_{1.5e-4}$	+
ZDT2	$3.26e-1_{4.3e-4}$	$3.28e-1_{2.5e-4}$	$3.28e-1_{1.1e-4}$	+
ZDT3	$5.15e-1_{2.3e-4}$	$5.10e-1_{3.8e-3}$	$5.15e-1_{5.1e-4}$	+
ZDT4	$6.56e-1_{4.5e-3}$	–	$6.61e-1_{3.8e-4}$	+
ZDT6	$3.88e-1_{2.3e-3}$	$4.01e-1_{1.5e-4}$	$4.01e-1_{1.0e-4}$	+
DTLZ1	$4.88e-1_{5.5e-3}$	–	$4.94e-1_{3.4e-4}$	+
DTLZ2	$2.11e-1_{3.1e-4}$	$2.10e-1_{4.5e-4}$	$2.12e-1_{2.3e-4}$	+
DTLZ3	–	–	$2.12e-1_{2.8e-3}$	+
DTLZ4	$2.09e-1_{2.1e-1}$	$1.96e-1_{6.1e-3}$	$2.09e-1_{3.3e-4}$	+
DTLZ5	$2.11e-1_{3.5e-4}$	$2.11e-1_{5.4e-4}$	$2.12e-1_{2.1e-4}$	+
DTLZ6	$1.75e-1_{3.6e-2}$	$2.12e-1_{4.4e-5}$	$2.12e-1_{4.8e-5}$	+
DTLZ7	$3.33e-1_{2.1e-4}$	$3.34e-1_{3.2e-4}$	$3.34e-1_{7.3e-5}$	+
WFG1	$5.23e-1_{1.3e-1}$	$1.04e-1_{1.0e-2}$	$9.70e-2_{5.3e-3}$	+
WFG2	$5.61e-1_{2.8e-3}$	$5.64e-1_{1.0e-4}$	$5.62e-1_{5.7e-4}$	+
WFG3	$4.41e-1_{3.2e-4}$	$4.42e-1_{5.4e-5}$	$4.41e-1_{1.1e-4}$	+
WFG4	$2.17e-1_{4.9e-4}$	$2.02e-1_{1.6e-3}$	$1.96e-1_{2.0e-3}$	+
WFG5	$1.95e-1_{3.6e-4}$	$1.96e-1_{6.3e-5}$	$1.96e-1_{5.8e-5}$	+
WFG6	$2.03e-1_{9.0e-3}$	$2.09e-1_{3.5e-4}$	$2.05e-1_{1.1e-3}$	+
WFG7	$2.09e+1_{3.3e+4}$	$2.09e+1_{1.7e+4}$	$2.06e+1_{8.2e+4}$	+
WFG8	$1.47e-1_{2.1e-3}$	$1.26e-1_{3.0e-3}$	$1.40e-1_{1.9e-3}$	+
WFG9	$2.37e-1_{1.7e-3}$	$2.34e-1_{6.6e-4}$	$2.33e-1_{4.1e-4}$	+

Table 9. Non-successful statistical tests among NSGA-II, OMOPSO, and SMPSO.

$I^1_{\epsilon+}$	OMOPSO	SMPSO
NSGA-II		WFG3, WFG8
OMOPSO	N/A	ZDT6, DTLZ6, WFG1, WFG4

Δ	OMOPSO	SMPSO
NSGA-II	DTLZ3	WFG2
OMOPSO	N/A	ZDT6, DTLZ6

HV	OMOPSO	SMPSO
NSGA-II		ZDT6
OMOPSO	N/A	DTLZ6, DTLZ7, WFG8

Acknowledgments. 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 Ministry of Education and Science. Francisco Luna acknowledges support from the Spanish Ministry of Education and Science and FEDER under contract TIN2005-08818-C04-01 (the OPLINK project).

References

1. J. E. Álvarez-Benítez, R. M. Everson, and J. E. Fieldsend. A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts. In C. A. Coello Coello et al., editor, *EMO 2005*, number 3410 in LNCS, pages 459–473, 2005.
2. 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.
3. K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

4. 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.
5. 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.
6. J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.
7. 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, December 2006.
8. Y. Hochberg and A. C. Tamhane. *Multiple Comparison Procedures*. Wiley, 1987.
9. V. L. Huang, P. N. Suganthan, and J. J. Liang. Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems. *Int. J. Intell. Syst.*, 21(2):209–226, 2006.
10. 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.
11. J. Kennedy and R. Eberhart. Particle swarm optimization. In *Fourth IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.
12. James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, California, 2001.
13. J. Knowles, L. Thiele, and E. Zitzler. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Technical Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 2006.
14. X. Li. A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization. In *Genetic and Evolutionary Computation - GECCO 2003*, volume 2723 of *LNCS*, pages 37–48, 2003.
15. J. Moore and R. Chapman. Application of particle swarm to multiobjective optimization. Technical report, Department of Computer Science and Software Engineering, Auburn University, 1999.
16. S. Mostaghim and J. Teich. Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In *Proceedings of the IEEE Swarm Intelligence Symposium, 2003. SIS '03*, pages 26–33, 2003.
17. M. Reyes-Sierra and C. Coello. Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.
18. 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.
19. G. Toscano and C. Coello. Using Clustering Techniques to Improve the Performance of a Multi-objective Particle Swarm Optimizer. In *Genetic and Evolutionary Computation - GECCO 2004*, volume 3102/2004 of *LNCS*, pages 225–237, 2004.
20. E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.
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.