On the Velocity Update in Multi-Objective Particle Swarm Optimizers

Juan J. Durillo, Antonio J. Nebro, José García-Nieto, and Enrique Alba

Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, University of Málaga, Campus de Teatinos, 29071 Málaga, Spain
{durillo,antonio,jnieto,eat}@lcc.uma.es

Summary. Since its appearance, Particle Swarm Optimization (PSO) has become a very popular technique for solving optimization problems because of both its simplicity and its fast convergence properties. In the last few years there has been a variety of proposals for extending it to handle with multiples objectives. Although many of them keep the same properties of the original algorithm, they face difficulties when tackling the optimization of some multi-modal problems, i.e., those having more than one suboptimal front of solutions. Recent studies have shown that this disadvantage could be related to the velocity of the particles: uncontrolled high velocities may have no effect in particles movements. While many of the contributions on the specialized literature have focused on the selection of the leaders of the swarm, studies about different schemes for controlling the velocity of the particles are scarce in the multi-objective domain. In this work, we study different mechanisms in order to update the velocity of each particle with the idea of enhancing the search capabilities of multi-objective PSO algorithms. Our experiments show that some modifications help to over-coming the difficulties observed in previous proposals when dealing with hard optimization problems.

3.1 Introduction

Particle Swarm Optimization (PSO) has become a popular algorithm due to its relative simplicity and competitive performance when solving a wide range of optimization problems in the continuous domain [11]. A considerable number of these problems has to optimize more than one objective function at the same time which are in conflict with respect to each other, so many proposals of Multi-Objective PSO (MOPSO) techniques have been developed [16]. In the survey presented in [16], the authors considered that the main characteristics of all the existing MOPSOs are the existence of an external archive of non-dominated solutions, the selection strategy for non-dominated solutions as leaders for guiding the swarm, the neighborhood topology, and the existence or not of a mutation operator. In this chapter, our ap-proach is to study a different feature: the scheme for controlling the velocity of the particles. This issue has been studied in single-objective optimization [1, 2, 15, 18], and we are interested in investigating how it can affect the search effectivity in MOPSOs.

In our previous research, we analyzed the performance of six representative MOPSO metaheuristics in [8], concluding that a variant of the OMOPSO algorithm [17] provided the best overall performance over a set of 21 benchmark problems. OMOPSO also shown to be very fast in a comparison with other state-of-the-art multi-objective metaheuristics [14]; however, we realized that OMOPSO had difficulties when facing the solution of some multi-modal problems. We deeply studied this issue in [8], finding that the velocity of the particles in MOPSOs can become too high, and hence resulting in erratic movements towards the upper and lower limits of the positions of the particles. This is an example of the so-called swarm explosion [2], and we found out that it can be prevented by using a velocity constriction mechanism [8].

Our motivation then is, taking OMOPSO as our baseline MOPSO algorithm, and to study different velocity update schemes in order to have an insight of the potential improvements in the search capacity they can lead to MOPSO metaheuristics. The contributions of this chapter can be summarized as follows:

- We propose four velocity schemes to be applied to MOPSO algorithms.
- The resulting techniques are evaluated on a benchmark of 21 problems (those comprising the ZDT [20], DTLZ [5], and WFG [10] problem families).
- Three quality indicators are used to assess the performance of the algorithms (additive unary epsilon (I_ε^1+) [13],spread (Δ) [4], and hypervolume (HV) [19]).
- We use a statistical analysis methodology to ensure the significance of the obtained results.

The remainder of this chapter is organized as follows. Section 3.2 includes basic background about PSO and MOPSO algorithms. Section 3.3 is aimed at describing OMOPSO, the baseline approach considered, and the different velocity schemes that we have applied, leading to four resulting versions of OMOPSO. Section 3.4 is devoted to the experimentation, including the benchmark problems, the quality indicators, the parameter setting, and the methodology adopted in the statistical tests. In Section 3.5, we analyze the obtained results regarding the three used quality indicators. The results are discussed in Section 3.6. Finally, Section 3.7 contains the conclusions and some possible lines for future work.

3.2 PSO Background

In a PSO algorithm, each potential solution to the problem is called *particle* and the population of solutions is called *swarm*. A basic PSO updates the particle x_i at the generation t with the formula:

$$x_i(t) = x_i(t-1) + v_i(t) \tag{3.1}$$

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

$$v_i(t) = \chi[w \cdot v_i(t-1) + C_1 \cdot r_1 \cdot (x_{p_i} - x_i) + C_2 \cdot r_2 \cdot (x_{g_i} - x_i)] \tag{3.2}$$

In this formula, x_{p_i} is the best solution that x_i has viewed (*pbest*), x_{g_i} is the best particle (*gbest*, 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, r_1 and r_2 are two uniformly distributed random numbers in the range $[0, 1]$, and the parameters C_1 and C_2 are specific parameters which control the effect of the personal and global best particles. χ is a constriction coefficient introduced to control the particle's velocity [2].

Algorithm 3.1. 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
```

Algorithm 3.1 describes the pseudocode 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). Then, 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 is usually the best particle in the swarm (i.e., *gbest*). 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. Therefore, issues that have to be considered now are [16]:

1. How to select the 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 single solutions?

The pseudocode of a general MOPSO is included in Algorithm 3.2. After initialising 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 kind of 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 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 [16].

Algorithm 3.2. 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()
```

3.3 Velocity Schemes and Resulting Algorithms

In this section we describe the MOPSO variants we have developed for our study. We start by giving details of OMOPSO, the baseline approach.

3.3.1 OMOPSO

As commented in the introduction, our base MOPSO algorithm is a variant of OMOPSO (Optimized MOPSO), proposed by Reyes-Sierra and Coello Coello in [17]. This algorithm is characterized by using 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, but in our experiments we always discard this feature, being the leaders archive the result of the execution of the technique.

The velocity scheme in OMOPSO, following Equation 3.2, is defined as follows:

- The inertia weight w is a uniformly distributed random number in the range $[0,1]$.
- The coefficients C_1 and C_2 are two uniformly distributed random numbers in the range $[1.5,2.0]$.
- The constriction coefficient χ takes the value 1.0.

In addition, we introduce a mechanism in such a way that the accumulated velocity of each variable j (in each particle) is further bounded by means of the following *velocity constriction* equation:

$$v_{i,j}(t) = \begin{cases} delta_j & \text{if } v_{i,j}(t) > delta_j \\ -delta_j & \text{if } v_{i,j}(t) \leq -delta_j \\ v_{i,j}(t) & \text{otherwise} \end{cases} \tag{3.3}$$

where

$$delta_j = \frac{(upper\_limit_j - lower\_limit_j)}{2} \tag{3.4}$$

After applying Equation 3.1, OMOPSO checks whether the resulting positions are out of the bounds of the variables of the problem. In that case, the positions are assigned the corresponding upper or lower bound value; additionally, the direction of the velocity is reversed by multiplying it by -1.0.

Finally, OMOPSO applies a combination of uniform and non-uniform mutation to the particle swarm (uniform mutation to the 30% of the swarm, non-uniform to other 30%, and no mutation to the rest of the particles).

Once we have defined our base MOPSO algorithm, we present next the four velocity schemes which will lead to the same number of algorithms. Each scheme affects to each different component of Equation 3.2: the constriction coefficient χ, the inertia weigth w, the coefficients C_1 and C_2, and the component $v_i(t-1)$ (the current velocity).

3.3.2 SMPSO

Our previous research in [8] indicated that OMOPSO had difficulties when solving some multi-modal problems (e.g., ZDT4, DTLZ1, and DTLZ3). Our analysis of this issue showed that including a constriction coefficient similar to the one proposed in [2] the resulting algorithm could successfully solve these problems. We called this algorithm SMPSO (Speed contrained Multi-objective PSO). The constriction coeffient applied in SMPSO is defined as follows:

$$\chi = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \tag{3.5}$$

where

$$\varphi = \begin{cases} C_1 + C_2 & \text{if } C_1 + C_2 > 4 \\ 4 & \text{if } C_1 + C_2 \leq 4 \end{cases} \tag{3.6}$$

Besides using this velocity scheme, the coeffients C_1 and C_2 are random numbers in the range $[1.5, 2.5]$; the range used in OMOPSO, $[1.5, 2.0]$ would lead Equation 3.6 to always return a value of 4.

3.3.3 MOPSO_TVAC

Ratnaweera et *al.* proposed in [15] the use of *Time-Varying Acceleration Coefficients* (TVAC) with the idea of enhancing the search in the early part of the optimization and to encourage the particles to converge toward the global optima at the end of the search process. This can be carried out by linearly changing the coefficients C_1 and C_2 through the time (number of iterations). They suggest the following definitions of C_1 and C_2:

$$C_1 = (C_{1f} - C_{1i}) \frac{iter}{MAXITR} + C_{1i} \tag{3.7}$$

$$C_2 = (C_{2f} - C_{2i}) \frac{iter}{MAXITR} + C_{2i} \tag{3.8}$$

where C_{1i}, C_{1f}, C_{2i}, and C_{2f} are constants, $iter$ is the current iteration number and $MAXITR$ is the maximum number of iterations of the PSO algorithm. As suggested in [15], we use the values of C_1 and C_2 changing from 2.5 to 0.5 and from 0.5 to 2.5, respectively. The inertia weight w is not considered, so it takes the value 1.0.

The resulting algorithm after applying this velocity scheme to OMOPSO is named MOPSO_TVAC. In [15], a mutation operator is introduced, but we have omitted it due to the fact that OMOPSO already includes its own mutation mechanism.

3.3.4 MOHPSO

The *self-Organizing Hierarchical Particle Swarm Optimization optimizer* (HPSO) was also proposed in [15]. The authors of this work observed that in the absence of the previous velocity term, particles rapidly rush to a local optimum and stagnate due to the lack of momentum. To cope this issue, they proposed a reinitialization scheme proportional to the maximum allowable velocity. Taking these ideas, the MOHPSO algorithm is characterized by the following equation defining the velocity:

$$v_i(t) = C_1 \cdot r_1 \cdot (x_{p_i} - x_i) + C_2 \cdot r_2 \cdot (x_{g_i} - x_i) \tag{3.9}$$

If a given velocity element $v_{i,j}(t)$ gets 0, then it is reinitialized according to:

$$v_{i,j}(t) = \begin{cases} rand1 \cdot delta_j & if\ rand2 < 0.5 \\ rand3 \cdot -delta_j & if\ rand2 >= 0.5 \end{cases} \tag{3.10}$$

where $rand1$, $rand2$, and $rand3$ are separately generated uniformly distributed random numbers in $[0,1]$. The limit $delta_j$ results from Equation 3.4.

3.3.5 MOPSO_TVIW

The last variant we have considered is based on the *Time-Varying Inertia Weight* (TVIW) proposed by Shi and Eberhart in [18]. They found that the performance of a PSO method could improve by linearly varying the inertia weigh w. We adopt this

scheme in OMOPSO and the result is MOPSO_TVIW. The inertia weight is defined as follows:

$$w = (w_1 - w_2)\frac{MAXITR - iter}{MAXITR} + w_2 \tag{3.11}$$

where w_1 and w_2 are the initial and final values of the inertia weight, respectively, $iter$ is the current iteration number, and $MAXITR$ is the maximum number of iterations of the PSO algorithm. The study carried out in [18] yielded that the most promising results were obtained by varying w from 0.9 (w_1) at the beginning of the search to 0.4 (w_2) at the end for most of the studied problems; here, we use the same values of w_1 and w_2.

3.4 Experimentation

In this section we explain the benchmark problems used to evaluate the algorithms, the quality indicators used to assess their performance, the parameter settings used, and the statistical tests carried out.

3.4.1 Benchmark Problems

Here, we describe the different sets of problems addressed in this work. These problems are well-known, and they have been used in many studies in this area.

The problems families are the following:

- **Zitzler-Deb-Thiele (ZDT):** This benchmark is composed of five bi-objective problems [20]: ZDT1 (convex), ZDT2 (nonconvex), ZDT3 (nonconvex, disconnected), ZDT4 (convex, multimodal), and ZDT6 (nonconvex, nonuniformly spaced). These problems are scalable according to the number of decision variables.
- **Deb-Thiele-Laumanns-Zitzler (DTLZ):** The problems of this family are scalable both in the number of variables and objectives [5]. It is composed of the following seven problems: DTLZ1 (linear), DTLZ2-4 (nonconvex), DTLZ5-6 (degenerate), and DTLZ7 (disconnected).
- **Walking-Fish-Group (WFG):** This set is composed of nine problems, WFG1 - WFG9, that have been constructed using the WFG toolkit [10]. The properties of these problems are detailed in Table 3.1. They all are scalable both in the number of variables and the number of objectives.

In this work we have used the bi-objective formulation of the DTLZ and WFG problem families. A total of 21 MOPs are used to evaluate the six metaheuristics.

3.4.2 Quality Indicators

To assess the search capabilities of multi-objective metaheuristics on the test problems, two different issues are normally taken into account: the distance between the

Table 3.1. Properties of the MOPs created using the WFG toolkit

Problem	Separability	Modality	Bias	Geometry
WFG1	separable	uni	polynomial, flat	convex, mixed
WFG2	non-separable	f_1 uni, f_2 multi	no bias	convex, disconnected
WFG3	non-separable	uni	no bias	linear, degenerate
WFG4	non-separable	multi	no bias	concave
WFG5	separable	deceptive	no bias	concave
WFG6	non-separable	uni	no bias	concave
WFG7	separable	uni	parameter dependent	concave
WFG8	non-separable	uni	parameter dependent	concave
WFG9	non-separable	multi, deceptive	parameter dependent	concave

solution set generated by the proposed algorithm to the optimal Pareto front should be minimized (convergence) and the spread of solutions should be maximized in order to obtain as smooth and uniform a distribution of solutions as possible (diversity). To measure these two criteria it is necessary to know the exact location of the optimal Pareto front; the benchmark problems used in this work have known Pareto fronts.

The quality indicators can be classified into three categories depending on whether they evaluate the closeness to the Pareto front, the diversity in the solutions obtained, or both [3]. We have adopted one indicator of each type.

- **Unary Epsilon Indicator ($I^1_{\varepsilon+}$).** This indicator was proposed by Zitzler et al. [21] and makes direct use of the principle of Pareto-dominance. Given an approximation set, A, of a problem, the $I^1_{\varepsilon+}$ indicator is a measure of the smallest distance one would need to translate every point in A so that it dominates the optimal Pareto front of the problem. More formally, given $z^1 = (z^1_1, ..., z^1_n)$ and $z^2 = (z^2_1, ..., z^2_n)$, where n is the number of objectives:

$$I^1_{\varepsilon+}(A) = inf_{\varepsilon \in \mathbb{R}} \left\{ \forall z^2 \in Pareto\ Optimal\ Front\ \exists z^1 \in A : z^1 \prec_\varepsilon z^2 \right\} \qquad (3.12)$$

  where, $z^1 \prec_\varepsilon z^2$ if and only if $\forall 1 \leq i \leq n : z^1_i < \varepsilon + z^2_i$.
- **Spread (Δ).** The *Spread* indicator [4] measures the extent of spread achieved among the obtained solutions. This indicator (illustrated in Fig. 3.1) is defined as:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1)\bar{d}} \quad , \qquad (3.13)$$

  where d_i is the Euclidean distance between consecutive solutions, $\bar{d}$ is the mean of these distances, and d_f and d_l are the Euclidean distances to the *extreme* (bounding) solutions of the optimal Pareto front in the objective space (see [4] for the details). Δ takes a value of zero for an ideal distribution, pointing out a

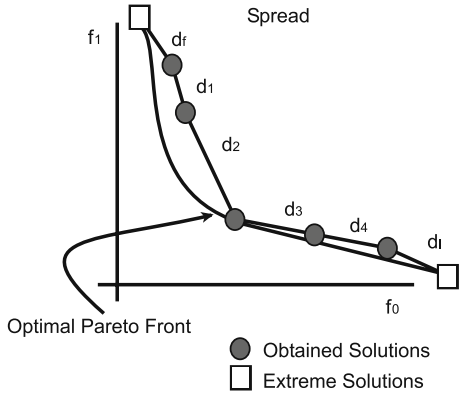

Fig. 3.1. Calculating the Spread quality indicator

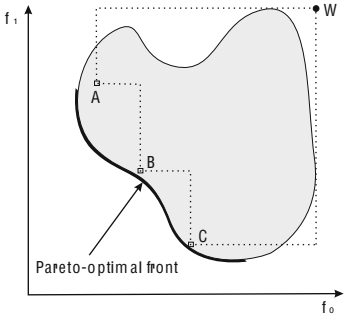

Fig. 3.2. The hypervolume enclosed by the non-dominated solutions

perfect spread out of the solutions in the Pareto front. We apply this indicator after a normalization of the objective function values.

- **Hypervolume (HV).** The HV indicator calculates the volume, in the objective space, covered by members of a non-dominated set of solutions Q for problems where all objectives are to be minimized [19]. In the example depicted in Fig. 3.2, the HV is the region enclosed within the discontinuous line, where $Q = \{A, B, C\}$ (in the figure, the grey area represents the objective space that has been explored). Mathematically, for each solution $i \in Q$, a hypercube vol_i is constructed with a reference point W and the solution i as the diagonal corners of the hypercube. The reference point can be found simply by constructing a vector of worst objective function values. Thereafter, a union of all hypercubes is found and its hypervolume (HV) is calculated:

$$HV = volume\left(\bigcup_{i=1}^{|Q|} vol_i\right). \tag{3.14}$$

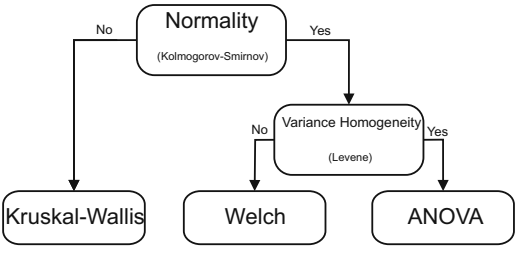

Fig. 3.3. Statistical analysis performed in this work

Algorithms with larger HV values are desirable. Since this indicator is not free from arbitrary scaling of objectives, we have evaluated the metric by using normalized objective function values.

3.4.3 Parameter Settings

Given that we are studying variants of a same technique, the main parameters take the same values in all of them. All the MOPSOs studied here were implemented using the Java-based framework jMetal [7] for solving multi-objective optimization problems[1]. These algorithms have been configured with swarms of 100 particles and the archive size is also 100. The specific parameters of each algorithm are specified in previous sections.

The stopping criterion is to reach $25,000$ function evaluations in the experiments performed for assessing the quality of the obtained solution sets.

3.4.4 Statistical Tests

Since we are dealing with stochastic algorithms we have made 100 independent runs of each experiment, and we show the median, $\tilde{x}$, and interquartile range, IQR, as measures of location (or central tendency) and statistical dispersion, respectively. The following statistical analysis has been performed throughout this work [6]. Firstly, a Kolmogorov-Smirnov test was performed 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. Fig. 3.3 summarizes the statistical analysis.

The null hypothesis is that the means of the obtained results are equivalents. We always consider in this work a confidence level of 95% (i.e., significance level of 5% or p-value under 0.05) in the statistical tests, which means that the differences are

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

unlikely to have occurred by chance with a probability of 95%. Then, if the given test obtains confidence values lower than 0.05 (p-value< 0.05) the null hypothesis is rejected and the compared results are significantly different. Successful tests (null hypotheses rejected) are marked with '+' symbols in the last column in all the tables containing the results; conversely, '-' means that the null hypothesis cannot be rejected, and hence no statistical confidence was found (p-value > 0.05). For the sake of better understanding, the best result for each problem has a gray colored background and the second best one has a clearer gray background.

To further analyze the results statistically, in some cases, we have also included a post-hoc testing phase which allows for a pair-wise comparison of samples [9]. We have used the *Wilcoxon* test for that purpose.

3.5 Analysis of the Obtained Results

Let us start by analyzing the values obtained after applying the $I^1_{\varepsilon+}$ indicator, which are included in Table 3.2. At a first glance, we observe that there is not a clear winner algorithm taking into account the whole 21 benchmark problems. If we consider each problem family, on the one hand, we find that SMPSO and MOPSO_TIVW provide the lowest (best) indicator values on the ZDT problems, but they are the worst techniques on the WFG benchmark; on the other hand, the opposite happens with OMOPSO and MOHPSO: they achieve the lowest values on the WFG benchmark, but not a best nor a second best indicator value on the ZDT problems. Regarding the seven problems composing the DTLZ family, the best results are distributed among the five MOPSOs. All the results in Table 3.2 have statistical confidence, as it can be seen in the last column, where all the cells have a '+' symbol. Let us

Table 3.2. Median and interquartile range of the $I^1_{\varepsilon+}$ quality indicator

Problem	OMOPSO $\tilde{x}_{IQR}$	SMPSO $\tilde{x}_{IQR}$	MOPSO_TVAC $\tilde{x}_{IQR}$	MOHPSO $\tilde{x}_{IQR}$	MOPSO_TVIW $\tilde{x}_{IQR}$	
ZDT1	$6.01e-03_{4.7e-04}$	$5.63e-03_{3.0e-04}$	$5.69e-03_{3.7e-04}$	$6.72e-03_{1.4e-03}$	$5.61e-03_{2.9e-04}$	+
ZDT2	$5.68e-03_{3.7e-04}$	$5.52e-03_{2.4e-04}$	$5.79e-03_{2.9e-04}$	$5.91e-03_{5.4e-04}$	$5.42e-03_{2.3e-04}$	+
ZDT3	$6.73e-03_{3.0e-03}$	$5.61e-03_{9.1e-04}$	$5.44e-03_{1.4e-03}$	$2.63e-02_{3.8e-02}$	$5.84e-03_{1.1e-03}$	+
ZDT4	$6.33e+00_{5.3e+00}$	$6.44e-03_{6.7e-04}$	$7.48e-03_{2.3e-03}$	$3.02e+00_{2.2e+00}$	$8.69e-03_{1.9e-01}$	+
ZDT6	$4.87e-03_{5.1e-04}$	$4.74e-03_{5.1e-04}$	$5.99e-03_{1.5e-03}$	$5.37e-03_{6.4e-04}$	$4.83e-03_{3.6e-04}$	+
DTLZ1	$3.12e+00_{6.5e+00}$	$3.06e-03_{2.7e-04}$	$3.56e-03_{1.1e-03}$	$1.84e+00_{2.1e+00}$	$2.50e-01_{7.5e-01}$	+
DTLZ2	$5.21e-03_{2.3e-04}$	$5.28e-03_{3.3e-04}$	$6.89e-03_{1.3e-03}$	$5.37e-03_{3.3e-04}$	$6.23e-03_{5.7e-04}$	+
DTLZ3	$2.99e+01_{5.9e+01}$	$5.65e-03_{8.8e-04}$	$1.00e-02_{7.0e-01}$	$2.33e+01_{1.9e+01}$	$4.89e+00_{2.2e+01}$	+
DTLZ4	$5.52e-03_{4.2e-04}$	$5.50e-03_{3.7e-04}$	$5.99e-02_{2.5e-02}$	$5.45e-03_{4.9e-04}$	$7.45e-03_{1.1e-03}$	+
DTLZ5	$5.19e-03_{2.6e-04}$	$5.33e-03_{4.2e-04}$	$6.79e-03_{1.2e-03}$	$5.43e-03_{3.7e-04}$	$6.11e-03_{6.0e-04}$	+
DTLZ6	$5.34e-03_{4.4e-04}$	$5.15e-03_{3.8e-04}$	$5.06e-03_{4.4e-04}$	$9.03e-03_{4.6e-03}$	$5.09e-03_{3.2e-04}$	+
DTLZ7	$5.63e-03_{6.8e-04}$	$5.24e-03_{3.7e-04}$	$5.12e-03_{3.9e-04}$	$5.68e-03_{7.7e-04}$	$5.13e-03_{2.8e-04}$	+
WFG1	$1.16e+00_{1.4e-01}$	$1.16e+00_{6.0e-02}$	$1.40e+00_{3.6e-02}$	$7.65e-01_{1.2e-01}$	$1.33e+00_{4.3e-02}$	+
WFG2	$1.23e-02_{2.9e-03}$	$1.78e-02_{6.1e-03}$	$1.04e-01_{3.0e-02}$	$1.56e-02_{2.7e-02}$	$1.76e-02_{4.6e-03}$	+
WFG3	$2.00e+00_{6.0e-04}$	$2.00e+00_{1.4e-03}$	$2.07e+00_{3.1e-02}$	$2.00e+00_{1.4e-03}$	$2.00e+00_{1.3e-03}$	+
WFG4	$4.34e-02_{5.2e-03}$	$5.50e-02_{5.7e-03}$	$7.18e-02_{7.7e-03}$	$2.83e-02_{8.2e-03}$	$6.33e-02_{7.1e-03}$	+
WFG5	$6.36e-02_{5.1e-04}$	$6.38e-02_{1.1e-03}$	$6.54e-02_{2.7e-02}$	$6.35e-02_{5.6e-04}$	$6.37e-02_{7.9e-04}$	+
WFG6	$1.47e-02_{9.4e-04}$	$1.83e-02_{2.0e-03}$	$9.03e-02_{1.9e-02}$	$1.48e-02_{1.1e-03}$	$1.89e-02_{1.7e-03}$	+
WFG7	$1.54e-02_{6.9e-04}$	$1.94e-02_{2.0e-03}$	$1.19e-01_{1.6e-02}$	$1.54e-02_{7.2e-04}$	$1.96e-02_{1.9e-03}$	+
WFG8	$5.11e-01_{2.4e-03}$	$4.09e-01_{6.7e-02}$	$3.88e-01_{8.0e-02}$	$5.07e-01_{2.5e-03}$	$5.13e-01_{5.3e-02}$	+
WFG9	$2.58e-02_{2.6e-03}$	$2.83e-02_{2.6e-03}$	$4.80e-02_{1.2e-02}$	$2.17e-02_{4.2e-03}$	$2.89e-02_{2.6e-03}$	+

Table 3.3. Median and interquartile range of the Δ quality indicator

Problem	OMOPSO $\tilde{x}_{IQR}$	SMPSO $\tilde{x}_{IQR}$	MOPSO_TVAC $\tilde{x}_{IQR}$	MOHPSO $\tilde{x}_{IQR}$	MOPSO_TVIW $\tilde{x}_{IQR}$	
ZDT1	$7.98e-02_{1.4e-02}$	$7.66e-02_{1.4e-02}$	$1.01e-01_{1.3e-02}$	$1.10e-01_{2.7e-02}$	$8.39e-02_{1.6e-02}$	+
ZDT2	$7.46e-02_{1.6e-02}$	$7.33e-02_{1.6e-02}$	$8.71e-02_{1.3e-02}$	$9.01e-02_{1.9e-02}$	$7.09e-02_{2.0e-02}$	+
ZDT3	$7.13e-01_{1.0e-02}$	$7.10e-01_{7.2e-03}$	$7.81e-01_{7.1e-02}$	$7.71e-01_{5.9e-02}$	$7.12e-01_{9.5e-03}$	+
ZDT4	$8.69e-01_{5.9e-02}$	$9.81e-02_{1.4e-02}$	$2.05e-01_{3.6e-02}$	$9.02e-01_{1.6e-01}$	$1.29e-01_{3.4e-01}$	+
ZDT6	$2.90e-01_{1.1e+00}$	$2.83e-01_{1.2e+00}$	$1.33e+00_{5.7e-02}$	$1.29e+00_{4.3e-02}$	$1.11e+00_{1.2e+00}$	+
DTLZ1	$8.30e-01_{1.8e-01}$	$7.71e-02_{1.4e-02}$	$1.62e-01_{2.4e-02}$	$8.01e-01_{3.7e-01}$	$6.00e-01_{6.2e-01}$	+
DTLZ2	$1.29e-01_{1.5e-02}$	$1.32e-01_{1.7e-02}$	$2.29e-01_{5.1e-02}$	$1.30e-01_{1.4e-02}$	$1.68e-01_{2.2e-02}$	+
DTLZ3	$8.06e-01_{2.2e-01}$	$1.43e-01_{3.5e-02}$	$3.06e-01_{2.7e-01}$	$8.83e-01_{1.7e-01}$	$6.88e-01_{2.9e-01}$	+
DTLZ4	$1.28e-01_{1.9e-02}$	$1.26e-01_{1.7e-02}$	$7.78e-01_{1.2e-01}$	$1.23e-01_{2.0e-02}$	$2.02e-01_{3.0e-02}$	+
DTLZ5	$1.32e-01_{1.6e-02}$	$1.34e-01_{1.7e-02}$	$2.34e-01_{3.6e-02}$	$1.29e-01_{1.6e-02}$	$1.66e-01_{2.0e-02}$	+
DTLZ6	$1.18e-01_{2.3e-02}$	$1.11e-01_{2.2e-02}$	$1.10e-01_{2.0e-02}$	$2.72e-01_{1.3e-01}$	$1.09e-01_{1.9e-02}$	+
DTLZ7	$5.20e-01_{3.7e-03}$	$5.19e-01_{2.2e-03}$	$5.21e-01_{3.1e-03}$	$5.20e-01_{2.5e-03}$	$5.19e-01_{8.3e-04}$	+
WFG1	$1.15e+00_{9.4e-02}$	$1.01e+00_{5.1e-02}$	$1.14e+00_{4.6e-02}$	$9.69e-01_{2.4e-01}$	$1.11e+00_{3.5e-02}$	+
WFG2	$7.76e-01_{1.7e-02}$	$8.27e-01_{4.3e-02}$	$8.59e-01_{8.9e-02}$	$7.91e-01_{8.6e-02}$	$8.36e-01_{4.5e-02}$	+
WFG3	$3.67e-01_{7.0e-03}$	$3.85e-01_{6.3e-03}$	$6.36e-01_{3.6e-02}$	$3.68e-01_{7.3e-03}$	$3.89e-01_{7.4e-03}$	+
WFG4	$3.98e-01_{4.4e-02}$	$4.88e-01_{5.8e-02}$	$4.77e-01_{7.4e-02}$	$2.06e-01_{5.3e-02}$	$5.29e-01_{5.3e-02}$	+
WFG5	$1.29e-01_{1.9e-02}$	$1.44e-01_{1.6e-02}$	$1.63e-01_{3.8e-02}$	$1.33e-01_{2.0e-02}$	$1.45e-01_{1.8e-02}$	+
WFG6	$1.24e-01_{1.6e-02}$	$1.62e-01_{2.2e-02}$	$6.23e-01_{6.3e-02}$	$1.24e-01_{1.6e-02}$	$1.74e-01_{2.8e-02}$	+
WFG7	$1.22e-01_{1.9e-02}$	$1.60e-01_{1.8e-02}$	$6.27e-01_{6.9e-02}$	$1.30e-01_{1.6e-02}$	$1.66e-01_{1.9e-02}$	+
WFG8	$5.69e-01_{4.6e-02}$	$7.48e-01_{5.6e-02}$	$7.78e-01_{7.2e-02}$	$5.49e-01_{5.2e-02}$	$7.67e-01_{9.9e-02}$	+
WFG9	$1.99e-01_{1.7e-02}$	$2.17e-01_{2.8e-02}$	$3.43e-01_{5.7e-02}$	$1.69e-01_{2.0e-02}$	$2.26e-01_{2.5e-02}$	+

start by analyzing the values obtained after applying the $I^1_{\varepsilon+}$) indicator, which are included in Table 3.2. At a first glance, we observe that there is not a clear winner algorithm taking into account the whole 21 benchmark problems. If we consider each problema family, on the one hand we find that SMPSO and MOPSO_TIVW provide the lowest (best) indicator values on the ZDT problems, but they are the worst techniques on the WFG benchmark; on the other hand, the opposite happens with OMOPSO and MOHPSO: they achieve the lowest values on the WFG benchmark, but not a best nor a second best indicator value on the ZDT problems. Regarding the seven DTLZ family, the best results are distributed among the five MOPSOs. All the results in Table 3.2 have statistical confidence, as it can be seen in the last column, where all the cells have a '+' symbol.

The values obtained after applying the Δ quality indicator are shown in Table 3.3, where the lower the value the better. The results show that, as happened with the previous analyzed indicator, SMPSO and MOPSO_TVIW are the best algorithms in the ZDT benchmark, and they are the worst algorithms in the WFG family, in which OMOPSO and MOHPSO has obtained the best values. As to the DTLZ benchmark, SMPSO has been the best algorithm in this indicator: it has obtained the best and second best value in, respectively, two and three out of the seven DTLZ problems. OMOPSO and MOPSO_TVIW have also obtained the best value in two problems. Statistical confidence has been found in all the comparisons.

Finally, we pay attention to the results obtained after applying the HV indicator (Table 3.4). Higher values of HV mean better results. In this case, MOPSO_TVAC has been clearly the best algorithm in the ZDT family: it has obtained the highest (best) value in four out of the problems composing this benchmark. Regarding the WFG family, the results have confirmed the conclusions obtained for this family in the two previous analyzed indicators: OMOPSO and MOHPSO have obtained the

Table 3.4. Median and interquartile range of the HV quality indicator

Problem	OMOPSO $\bar{x}_{IQR}$	SMPSO $\bar{x}_{IQR}$	MOPSO_TVAC $\bar{x}_{IQR}$	MOHPSO $\bar{x}_{IQR}$	MOPSO_TVIW $\bar{x}_{IQR}$	
ZDT1	$6.61e-01_{4.5e-04}$	$6.62e-01_{1.6e-04}$	$6.62e-01_{2.7e-05}$	$6.61e-01_{9.0e-04}$	$6.62e-01_{1.4e-04}$	+
ZDT2	$3.28e-01_{3.3e-04}$	$3.29e-01_{9.3e-05}$	$3.29e-01_{2.8e-05}$	$3.28e-01_{3.7e-04}$	$3.29e-01_{5.3e-05}$	+
ZDT3	$5.15e-01_{9.2e-04}$	$5.15e-01_{4.1e-04}$	$5.16e-01_{3.0e-04}$	$5.12e-01_{2.9e-03}$	$5.15e-01_{4.7e-04}$	+
ZDT4	$0.00e+00_{0.0e+00}$	$6.61e-01_{2.8e-04}$	$6.62e-01_{1.7e-04}$	$0.00e+00_{0.0e+00}$	$6.59e-01_{2.4e-01}$	+
ZDT6	$4.01e-01_{1.3e-04}$	$4.01e-01_{1.3e-04}$	$4.01e-01_{2.3e-04}$	$4.01e-01_{1.5e-04}$	$4.01e-01_{9.3e-05}$	+
DTLZ1	$0.00e+00_{0.0e+00}$	$4.94e-01_{2.5e-04}$	$4.95e-01_{9.5e-05}$	$0.00e+00_{0.0e+00}$	$2.50e-01_{4.9e-01}$	+
DTLZ2	$2.12e-01_{1.8e-04}$	$2.12e-01_{1.6e-04}$	$2.12e-01_{4.1e-04}$	$2.12e-01_{5.0e-04}$	$2.11e-01_{2.9e-04}$	+
DTLZ3	$0.00e+00_{0.0e+00}$	$2.12e-01_{3.8e-04}$	$2.12e-01_{1.3e-01}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{8.6e-02}$	+
DTLZ4	$2.10e-01_{2.8e-04}$	$2.10e-01_{1.6e-04}$	$1.79e-01_{1.5e-02}$	$2.10e-01_{4.5e-04}$	$2.08e-01_{9.0e-04}$	+
DTLZ5	$2.12e-01_{1.7e-04}$	$2.12e-01_{1.8e-04}$	$2.12e-01_{3.7e-04}$	$2.11e-01_{5.0e-04}$	$2.11e-01_{3.0e-04}$	+
DTLZ6	$2.12e-01_{9.7e-05}$	$2.12e-01_{8.1e-05}$	$2.12e-01_{3.3e-05}$	$2.12e-01_{4.4e-04}$	$2.12e-01_{3.5e-05}$	+
DTLZ7	$3.34e-01_{2.6e-04}$	$3.34e-01_{9.1e-05}$	$3.34e-01_{1.5e-05}$	$3.34e-01_{1.9e-04}$	$3.34e-01_{5.5e-05}$	+
WFG1	$1.46e-01_{6.8e-02}$	$1.14e-01_{5.4e-03}$	$8.86e-02_{5.1e-03}$	$2.62e-01_{4.7e-02}$	$9.98e-02_{4.7e-03}$	+
WFG2	$5.63e-01_{7.7e-04}$	$5.60e-01_{1.6e-03}$	$5.28e-01_{6.6e-03}$	$5.62e-01_{2.1e-03}$	$5.61e-01_{8.0e-04}$	+
WFG3	$4.42e-01_{1.4e-04}$	$4.41e-01_{3.0e-04}$	$4.14e-01_{3.3e-03}$	$4.41e-01_{2.5e-04}$	$4.41e-01_{2.7e-04}$	+
WFG4	$2.06e-01_{1.8e-03}$	$2.01e-01_{2.2e-03}$	$1.91e-01_{9.2e-04}$	$2.13e-01_{3.3e-03}$	$1.97e-01_{1.7e-03}$	+
WFG5	$1.96e-01_{6.2e-05}$	$1.96e-01_{7.6e-05}$	$1.96e-01_{7.9e-05}$	$1.96e-01_{6.2e-05}$	$1.96e-01_{7.4e-05}$	+
WFG6	$2.10e-01_{2.2e-04}$	$2.08e-01_{5.4e-04}$	$1.86e-01_{5.1e-03}$	$2.10e-01_{3.7e-04}$	$2.08e-01_{4.9e-04}$	+
WFG7	$2.10e-01_{1.6e-04}$	$2.09e-01_{3.2e-04}$	$1.73e-01_{4.2e-03}$	$2.10e-01_{3.7e-04}$	$2.09e-01_{4.1e-04}$	+
WFG8	$1.44e-01_{1.0e-03}$	$1.47e-01_{1.7e-03}$	$1.26e-01_{6.8e-03}$	$1.50e-01_{1.2e-03}$	$1.40e-01_{2.3e-03}$	+
WFG9	$2.36e-01_{6.8e-04}$	$2.35e-01_{6.3e-04}$	$2.26e-01_{8.7e-04}$	$2.38e-01_{1.5e-03}$	$2.34e-01_{4.4e-04}$	+

best or second best value in most of the problems of this benchmark. Meanwhile, SMPSO is the best choice in the DTLZ problems: it has obtained the best value in three problems, and the second best value in another two. MOPSO_TVAC has obtained similar figures in this family: two best and two second best values. As in the previous cases, we have found statistical confidence in all the experiments carried out.

We summarize in Table 3.5 the comparison of OMOPSO against the different alternatives implemented. In this table, each cell represents the result of the comparison between OMOPSO and another algorithm in the quality indicator represented by the column which contains the cell. In this comparison using the Wilcoxon test, a symbol "▲" means that the corresponding algorithm is significantly better than OMOPSO in that quality indicator, a symbol "▽" means that OMOPSO is better, and a symbol "–" means that no statistical differences were found. We see that SMPSO, MOPSO_TVAC, and MOPSO_TVIW outperform the results of OMOPSO in most of the problems belonging to the ZDT familiy. SMPSO is the only technique which clearly improves OMOPSO in the DTLZ benchmark, while MOHPSO is the only algorithm yielding competitive results compared with OMOPSO in the WFG family.

Figure 3.4 shows some examples of fronts obtained by the different approaches when solving the ZDT4 problem. We see that OMOPSO, and MOHPSO are unable to converge to the optimal Pareto front on ZDT4. The rest of evaluated algorithms has obtained similar results: they converge to the optimal Pareto front and they have obtained a uniform distribution of solutions.

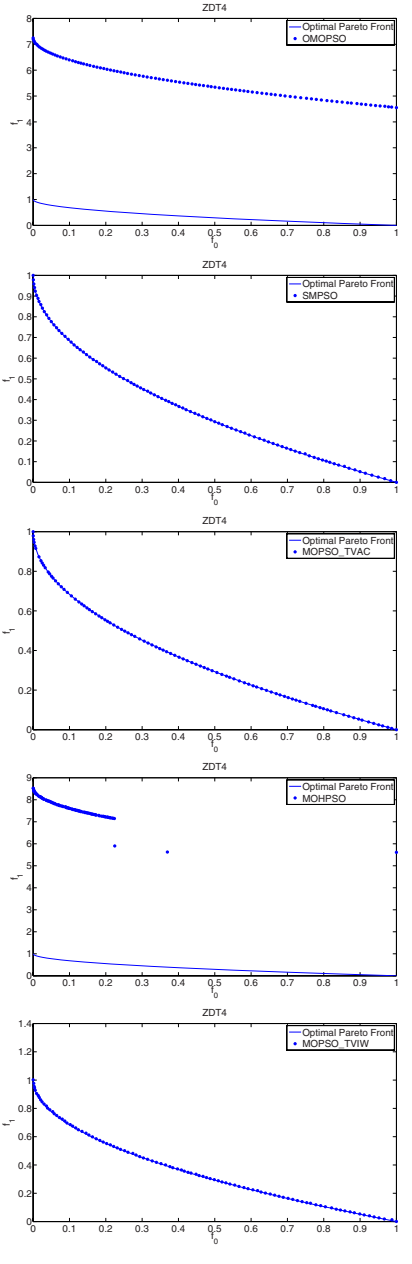

Fig. 3.4. Pareto fronts obtained by the different approaches when solving the ZDT4 problem. From top to bottom: OMOPSO, SMPSO, MOPSO_TVAC, MOHPSO, MOPSO_TVIW.

Table 3.5. OMOPSO vs other Approaches

Problem	SMPSO			MOPSO_TVAC			MOHPSO			MOPSO_TVIW		
	$I^1_{\varepsilon+}$	Δ	HV	$I^1_{\varepsilon+}$	Δ	HV	$I^1_{\varepsilon+}$	Δ	HV	$I^1_{\varepsilon+}$	Δ	HV
ZDT1	▲	–	▲	▲	▽	▲	▽	▽	▽	▲	▽	▲
ZDT2	▲	–	▲	–	▽	▲	▽	▽	▲	▲	–	▲
ZDT3	▲	▲	▲	▲	▽	▲	▽	▽	▽	▲	–	▲
ZDT4	▲	▲	▲	▲	▲	▲	▲	–	–	▲	▲	▲
ZDT6	–	–	▲	▽	▽	▲	▽	▽	▲	–	▽	▲
DTLZ1	▲	▲	▲	▲	▲	▲	▲	–	–	▲	▲	▲
DTLZ2	▽	▽	▲	▽	▽	▽	▽	–	▽	▽	▽	▽
DTLZ3	▲	▲	▲	▲	▲	▲	▲	▽	▽	▲	▲	▲
DTLZ4	–	–	▲	▽	▽	▽	–	–	▽	▽	▽	▽
DTLZ5	▽	▽	▲	▽	▽	▽	▽	–	▽	▽	▽	▽
DTLZ6	▲	▲	▲	▲	▲	▲	▽	▽	▽	▲	▲	▲
DTLZ7	▲	▲	▲	▲	▽	▲	–	–	▲	▲	▲	▲
WFG1	–	▲	▽	▽	–	▽	▲	▲	▲	▽	▲	▽
WFG2	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
WFG3	▽	▽	▽	▽	▽	▽	▽	–	▽	▽	▽	▽
WFG4	▽	▽	▽	▽	▽	▽	▲	▲	▲	▽	▽	▽
WFG5	▽	▽	▲	▽	▽	▲	–	–	▲	–	▽	▲
WFG6	▽	▽	▽	▽	▽	▽	–	–	▽	▽	▽	▽
WFG7	▽	▽	▽	▽	▽	▽	–	▽	▽	▽	▽	▽
WFG8	▲	▽	▲	▲	▽	▽	▲	▲	▲	▲	▽	▽
WFG9	▽	▽	▽	▽	▽	▽	▲	▲	▲	▽	▽	▽

3.6 Discussion

In the previous section we have seen that there is not an algorithm which can be considered the best one in all the evaluated problems. We have observed that there are some alternatives which are specially suited for solving one or two problem families, but they fail when they are evaluated using other benchmarks. The arising question is whether it could be possible to combine the features of different MOPSOs, each of them standing out in a concrete problem family, in order to combine the better characteristic of them into a unique MOPSO able of outperforming the others in most of the problems.

Taking a look to Table 3.5, we see that SMPSO has improved OMOPSO in a higher number of cases than the other approaches in the problems belonging to the ZDT and DTLZ families, while MOHPSO obtains remarkable figures in the WFG test suite. Thus, the ideal algorithm should perform like SMPSO on the ZDT and DTLZ problems, and like MOHPSO in the WFG family. As both algorithms are based in different ideas, we have combined them to propose a new algorithm: SMHPSO.

To evaluate the performance of SMHPSO, the natural approach is to compare it against SMPSO and MHPSO. Table 3.6 summarizes the comparison between SMHPSO, SMPSO, and MOHPSO. As in Table 3.5, we have also used the Wilcoxon test to check the statistical significance of the results. Here, a symbol "▲" means that the

Table 3.6. SMHPSO vs SMPSO vs MOHPSO

Problem	SMPSO			MOHPSO		
	$I^1_{\varepsilon+}$	Δ	HV	$I^1_{\varepsilon+}$	Δ	HV
ZDT1	▲	▲	▲	▲	▲	▲
ZDT2	▲	▲	▲	▲	▲	▲
ZDT3	▲	▲	▲	▲	▲	▲
ZDT4	▲	–	▲	▽	▽	▽
ZDT6	▲	▲	▲	▲	–	▲
DTLZ1	–	▽	▲	▽	▽	▽
DTLZ2	–	–	–	▽	▲	▽
DTLZ3	–	–	▽	▽	▽	▽
DTLZ4	–	–	–	–	–	▽
DTLZ5	–	–	–	▽	▲	▽
DTLZ6	▲	▲	▲	▽	▽	▲
DTLZ7	▲	▲	▲	▲	▲	▲
WFG1	▽	▽	▽	▲	▽	▲
WFG2	▽	▽	▽	▽	▽	▲
WFG3	–	▽	▽	▲	▲	▲
WFG4	▽	▽	▽	▲	▲	▲
WFG5	▲	▲	–	▲	▲	–
WFG6	▽	–	▽	▲	▲	▲
WFG7	▽	–	▽	▲	▲	▲
WFG8	▽	▲	▽	▽	▲	–
WFG9	▽	–	▽	▲	▲	▲

corresponding algorithm is significantly better than SMHPSO in the indicator specified by the column containing that symbol, a symbol "▽" means that SMHPSO is perform better, and a symbol "–" means that no statistical differences were found.

The results in Table 3.6 indicate that SMHPSO outperforms SMPSO in the WFG family and, in a similar way, it improves the values obtained by MOHPSO also in the DTLZ problems. Thus, we can state that SMHPSO has improved the results of SMPSO and MOHPSO in those problems in which these two algorithms encounter difficulties, but it has not been able to improve the results in those problems in which SMPSO and MOHPSO are the best algorithms.

This leads us to propose a research line, related to investigate how to use the proper velocity update scheme in order to design a MOPSO able of improve the performance of those studied in this chapter. Besides analyzing other update strategies (e.g., those proposed in [1]), finding out hybrid approaches as well as designing adaptative mechanisms to vary the update scheme during the search are promising ideas.

3.7 Conclusions and Future Work

In this chapter we have studied the effect of applying different velocity schemes to OMOPSO, a multi-objective PSO algorithm which has proven to be competitive against a set of state-of-the-art multi-objective optimizers in previous works.

Concretely, we have developed and evaluated four alternatives called SMPSO, MOPSO_TVAC, MOHPSO, and MOPSO_TVIW, each of them characterized by modifying a different component of the formula defining the velocity scheme in a PSO. As bechmark problems, we have used 21 instances corresponding to the well-known ZDT, DTLZ, and WFG test suites.

The obtained results have shown that, in the context of the problems, the quality indicators, and the parameter settings considered, SMPSO, MOPSO_TVAC, and MOPSO_TVIW improves the results obtained by OMOPSO in the ZDT and DTLZ families, whereas MOHPSO is well suited for solving the problems composing the WFG benchmark. We have also carried out a first attempt to hybridize two of the most promising MOPSOs trying to combine their search habilities but, although the results are promising, it is a matter of further research.

Other future research topics in this line are related to study how the velocity update scheme may affect the MOPSOs concerning issues such as their speed to converge faster to the true Pareto front, or their ability to solve scalable problems in the number of variables and/or objectives.

Acknowledgement. Authors acknowledge funds from the Spanish Ministry of Sciences and Innovation European FEDER under contract TIN2008-06491-C04-01 (M* project, available at `http://mstar.lcc.uma.es`) and CICE, Junta de Andalucía under contract P07-TIC-03044 (DIRICOM project, `http://diricom.lcc.uma.es`). Juan J. Durillo is supported by grant AP-2006-003349 from the Spanish Ministry of Education and Science. José García-Nieto is supported by grant BES-2009-018767 from the Spanish Ministry of Sciences and Innovation.

References

1. Bui, L., Soliman, O., Abbass, H.: A modified strategy for the constriction factor in particle swarm optimization. In: Randall, M., Abbass, H.A., Wiles, J. (eds.) ACAL 2007. LNCS (LNAI), vol. 4828, pp. 333–344. Springer, Heidelberg (2007)
2. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
3. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
5. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In: Abraham, A., Jain, L., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization. Theoretical Advances and Applications, pp. 105–145. Springer, USA (2005)
6. Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. J. Mach. Learn. Res. 7, 1–30 (2006)
7. Durillo, J., Nebro, A., Luna, F., Dorronsoro, B., Alba, E.: jMetal: a Java Framework for Developing Multi-objective Optimization Metaheuristics. Tech. Rep. ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos (2006)

8. Durillo, J., García-Nieto, J., Nebro, A., Coello Coello, C., Luna, F., Alba, E.: Multi-objective particle swarm optimizers: An experimental comparison. Accepted for publication in EMO 2009 (2009)
9. Hochberg, Y., Tamhane, A.C.: Multiple Comparison Procedures. Wiley, Chichester (1987)
10. Huband, S., Barone, L., While, R.L., Hingston, P.: A scalable multi-objective test problem toolkit. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 280–295. Springer, Heidelberg (2005)
11. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
12. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco (2001)
13. Knowles, J., Thiele, L., Zitzler, E.: A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Tech. Rep. 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich (2006)
14. Nebro, A.J., Durillo, J.J., Coello Coello, C., Luna, F., Alba, E.: A study of convergence speed in multi-objective metaheuristics. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 763–772. Springer, Heidelberg (2008)
15. Ratnaweera, A., Halgamuge, S., Watson, H.: Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. International Journal of Computational Intelligence Research 8(3), 240–255 (2004)
16. Reyes-Sierra, M., Coello Coello, C.: Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. International Journal of Computational Intelligence Research 2(3), 287–308 (2006)
17. Reyes-Sierra, M., Coello Coello, C.A.: Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 505–519. Springer, Heidelberg (2005)
18. Shi, Y., Eberhart, R.: Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 1999, pp. 1945–1950 (1999)
19. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)
20. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. IEEE Transactions on Evolutionary Computation 8(2), 173–195 (2000)
21. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation 7, 117–132 (2003)