

# *Restart particle swarm optimization with velocity modulation: a scalability test*

**José García-Nieto & Enrique Alba**

## **Soft Computing**

A Fusion of Foundations,  
Methodologies and Applications

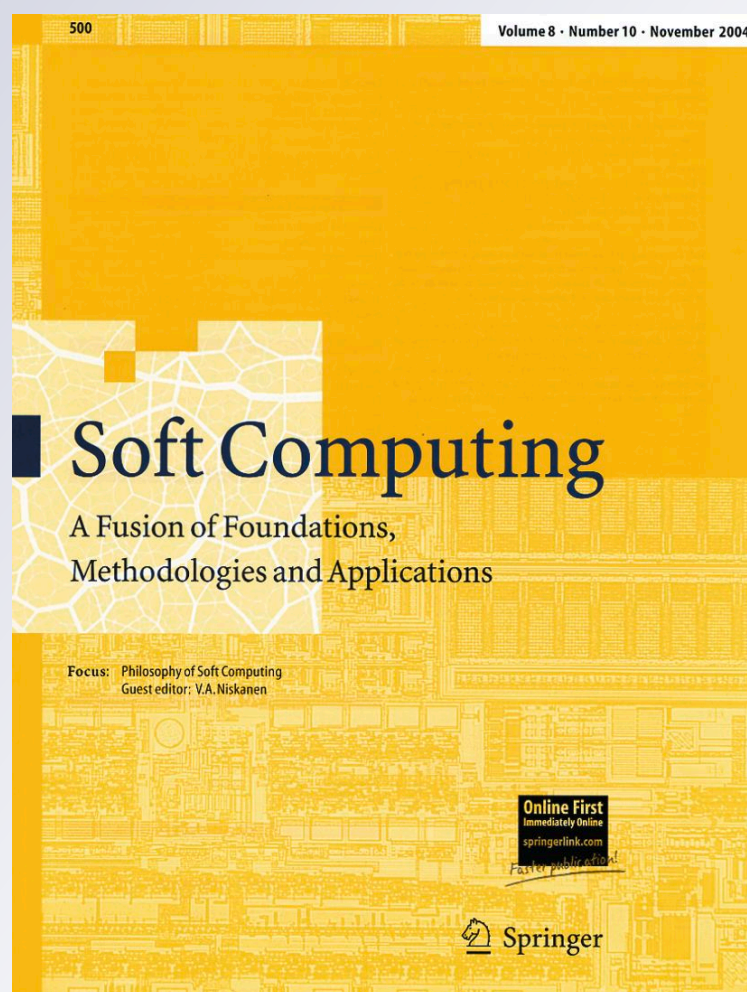
ISSN 1432-7643

Volume 15

Number 11

Soft Comput (2011) 15:2221-2232

DOI 10.1007/s00500-010-0648-1



**Your article is protected by copyright and all rights are held exclusively by Springer-Verlag. This e-offprint is for personal use only and shall not be self-archived in electronic repositories. If you wish to self-archive your work, please use the accepted author's version for posting to your own website or your institution's repository. You may further deposit the accepted author's version on a funder's repository at a funder's request, provided it is not made publicly available until 12 months after publication.**

# Restart particle swarm optimization with velocity modulation: a scalability test

José García-Nieto · Enrique Alba

Published online: 14 September 2010  
© Springer-Verlag 2010

**Abstract** Large scale continuous optimization problems are more relevant in current benchmarks since they are more representative of real-world problems (bioinformatics, data mining, etc.). Unfortunately, the performance of most of the available optimization algorithms deteriorates rapidly as the dimensionality of the search space increases. In particular, particle swarm optimization is a very simple and effective method for continuous optimization. Nevertheless, this algorithm usually suffers from unsuccessful performance on large dimension problems. In this work, we incorporate two new mechanisms into the particle swarm optimization with the aim of enhancing its scalability. First, a *velocity modulation* method is applied in the movement of particles in order to guide them within the region of interest. Second, a *restarting* mechanism avoids the early convergence and redirects the particles to promising areas in the search space. Experiments are carried out within the scope of this Special Issue to test scalability. The results obtained show that our proposal is scalable in all functions of the benchmark used, as well as numerically very competitive with regards to other compared optimizers.

**Keywords** Continuous optimization · Scalability · Particle swarm optimization · Large scale benchmarking

## 1 Introduction

In the evaluation of the search capabilities of a given optimization algorithm the usual approach is to choose a benchmark of known problems, to perform a fixed number of function evaluations, and to compare the results against the ones of other algorithms in the state of art. However, while some real industry problems can have hundreds and thousands of variables, current benchmarks are normally adopted with less than a hundred decision variables (see CEC'05, Suganthan et al. 2005; BBOB'09, Hansen et al. 2009; BBOB'10, Hansen et al. 2010 test beds). Large scale continuous optimization problems have attracted more and more interest (CEC'08, Tang et al. 2007; ISDA'09, Herrera and Lozano 2009; CEC'10, Tang et al. 2010) since they introduce a high complexity to the optimization process. Issues like the exponential increment of the solution space, as well as the change that some problems suffer from their own characteristics with the scale, can deteriorate quickly the performance of our optimization algorithms (Shang and Qiu 2006). This way, we can study certain mechanisms that show the best performance in short scale optimization problems, which is the case of the covariance matrix in G-CMA-ES (Auger and Hansen 2005), but with an unsuitable behavior for high dimensional functions (more than 100 variables). A different performance can be observed in simple algorithms like MTS (Tseng and Chen 2008), which combines several local search strategies using a small population. MTS was the best in the special session of large scale optimization of CEC'08 (Tang et al. 2007), where functions with thousands of variables were tackled.

J. García-Nieto (✉) · E. Alba  
Departamento Lenguajes y Ciencias de la Computación,  
University of Málaga, Campus de Teatinos,  
E.T.S.I. Informática, 29071 Málaga, Spain  
e-mail: jnieto@lcc.uma.es

E. Alba  
e-mail: eat@lcc.uma.es

All this motivates us to deeply analyze the scalable capacities of optimization algorithms. In particular, particle swarm optimization (PSO) (Kennedy and Eberhart 2001) is a very simple and effective method for continuous optimization. Nevertheless, this algorithm is characterized by an early convergence behavior, mainly produced by the overinfluenced best solution and its relative facility to fall in local optima (Liang et al. 2006; van den Bergh and Engelbrecht 2004). For this reason, PSO usually suffers from an unsuccessful performance on large dimension problems.

In this work, we have incorporated two mechanisms to the PSO with the aim of enhancing its scalability. First, a *velocity modulation* method is applied in the movement of particles in order to guide them within the feasible region. Second, a *restarting* mechanism avoids the early convergence and redirects the particles to promising areas in the search space. To evaluate the scalability of the resulting approach, we have followed the experimental framework proposed in this Special Issue on *Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems* (in URL <http://www.sci2s.ugr.es/eamhco/CFP.php>). We also studied the influence of both *velocity modulation* and *restarting* mechanisms to show real insights of the improvement of our proposal, called Restart PSO with Velocity Modulation (RPSO-vm), regarding the basic PSO. The results obtained confirm us that RPSO-vm is scalable in all functions of the benchmark used, as well as highly competitive in comparison with PSO and other well-known efficient optimizers.

The remaining of this paper is organized as follows. The next section presents basic preliminary concepts. In Sect. 3, the RPSO-vm algorithm is introduced. Section 4 describes the experimentation procedure with the benchmark of functions and the parameter settings. In Sect. 5, experimental results are reported with comparisons, analyses, and discussions. Finally, concluding remarks are given in Sect. 6.

## 2 Preliminaries

Particle swarm optimization (Montes de Oca et al. 2009; Kennedy and Eberhart 2001) has been successfully used in many problems of real parameter optimization (Das et al. 2008; García-Nieto et al. 2009; Hsieh et al. 2008; Liang et al. 2006; Liang and Suganthan 2005) since it is a well adapted algorithm for continuous solution encoding. Basically, a continuous optimization problem consists of:

find  $\mathbf{x}^*$  such that  $\forall \mathbf{x} f(\mathbf{x}^*) \leq f(\mathbf{x})$  (minimization).

Here,  $f(\cdot)$  is a function in a real space domain that models an optimization problem,  $\mathbf{x} = \{x_1, x_2, \dots, x_{DIM}\}$  is a solution for such problem, and  $DIM$  is the number of variables with  $x_i \in [x_{low}, x_{upp}]$  ( $1 \leq i \leq DIM$ ). Finally,  $x_{low}, x_{upp} \in \mathbb{R}$  correspond to lower (low) and upper (upp) limits of the variable domain, respectively.

In PSO, each potential solution to the problem is given by a particle *position* and the population of particles is called *swarm*. In this algorithm, each particle position  $\mathbf{x}_i$  is updated each generation  $t$  by means of Eq. 1.

$$\mathbf{x}_i(t+1) \leftarrow \mathbf{x}_i + \mathbf{v}_i(t+1) \quad (1)$$

where factor  $\mathbf{v}_i(t+1)$  is the velocity of the particle and is given by

$$\begin{aligned} \mathbf{v}_i(t+1) \leftarrow & \mathbf{v}_i(t) + \varphi_1 \cdot UN(0,1) \cdot (\mathbf{p}_i(t) - \mathbf{x}_i(t)) \\ & + \varphi_2 \cdot UN(0,1) \cdot (\mathbf{b}_i(t) - \mathbf{x}_i(t)) \end{aligned} \quad (2)$$

In this formula,  $\mathbf{p}_i(t)$  is the personal best solution that the particle  $i$  has stored so far,  $\mathbf{b}_i(t)$  is the global best particle (*leader*) that the entire swarm has ever generated. Finally,  $\varphi_1$  and  $\varphi_2$  are specific parameters which control the relative effect of the personal and global best particles, and  $UN(0,1)$  is a uniform random value in  $[0,1]$  which is sampled anew for each component of the velocity vector.

Velocity constriction is one of the main mechanisms used for controlling the movement of particles through the search space and for balancing the exploration-exploitation trade-off of the algorithm. Therefore, an efficient movement strategy of particles could help the PSO to find an optimum even in large scale problems. We can find several velocity constriction mechanisms in the literature. Three of the most popular are the following ones:

- $V_{MAX}$  factor. The simplest method for regulating the velocity lies in the maximum (and minimum) velocity delimitation. This mechanism uses a given value  $V_{max}$  for adjusting the maximum velocity each particle undergoes each generation step. According to this method, if the new velocity exceeds  $V_{max}$  then this value is aggregated to the new position calculation (Eq. 1) instead of the corresponding new velocity.
- *Inertia weight* ( $\omega$ ) (Shi and Eberhart 1998; Suresh et al. 2008) is one of the most used methods in PSO for controlling the velocity of particles in their movement. This parameter controls the trade-off between global and local search. Then, a high inertia value provides the algorithm with exploration capability and a low inertia



promotes the exploitation. The inertia weight linearly changes during the optimization process (of the algorithm) by using the following equation:

$$\omega \leftarrow \omega_{\max} - \frac{(\omega_{\max} - \omega_{\min}) \cdot \#g_{\text{current}}}{\#g_{\text{total}}} \quad (3)$$

This way, at the beginning of the process a high inertia ( $\omega_{\max}$ ) value is introduced (for exploration) which decreases until reaching the lowest value ( $\omega_{\min}$ ). The inertia value is incorporated in the velocity calculation as follows:

$$\mathbf{v}_i(t+1) \leftarrow \omega \cdot \mathbf{v}_i(t) + \varphi_1 \cdot UN(0,1) \cdot (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + \varphi_2 \cdot UN(0,1) \cdot (\mathbf{b}_i(t) - \mathbf{x}_i(t)) \quad (4)$$

- A third velocity constriction method was introduced in (Clerc and Kennedy 2003). In that work, the author indicates that the use of a *constriction factor* ( $\chi$ ) may be necessary to ensure convergence of the particle swarm algorithm. A detailed discussion of the constriction factor is beyond the scope of this work in (Clerc and Kennedy 2003), but a simplified method of incorporating it appears in Eq. 5, where  $\chi$  is a function of  $\varphi_1$  and  $\varphi_2$  as reflected in Eq. 6.

$$\mathbf{v}_i(t+1) \leftarrow \chi [\mathbf{v}_i(t) + \varphi_1 \cdot UN(0,1) \cdot (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + \varphi_2 \cdot UN(0,1) \cdot (\mathbf{b}_i(t) - \mathbf{x}_i(t))] \quad (5)$$

$$\chi = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}}, \varphi = \varphi_1 + \varphi_2, \varphi > 4 \quad (6)$$

### 3 The algorithm

Our proposal, RPSO-vm, consists in running a PSO algorithm in which we have incorporated two main ideas: *velocity modulation* and *restarting* mechanisms.

Using the *velocity modulation*, the algorithm controls that the overall movement calculated in each evolution step and for each particle position does not exceed the limits ( $x_{\text{low}}$ ,  $x_{\text{upp}}$ ) of the problem domain. First, after calculating the new velocity value ( $v_{\text{aux}}^j$ ) RPSO-vm performs a modulation procedure as shown in Algorithm 1. The velocity vector magnitude ( $\hat{\mathbf{v}}_i(t)$ ) is then bounded, which limits the given particle to move far from the interest area. These steps are calculated in Algorithm 2 in lines 7 and 8. Second, once obtained the new velocity  $v_i^j(t+1)$ , the overall movement is calculated, also controlling that the new particle position ( $x_{\text{aux}}^j$ ) does not exceed the problem limits. If this happens, the new position is recalculated by subtracting the new velocity to the old particle position (lines 10–14 in Algorithm 2).

#### Algorithm 1 Pseudocode of velmod procedure

---

```

1: if  $x_{\text{low}}^j > v_{\text{aux}}^j$  then
2:    $v_i^j(t+1) \leftarrow x_{\text{low}}^j$ 
3: else if  $v_{\text{aux}}^j \geq x_{\text{upp}}^j$  then
4:    $v_i^j(t+1) \leftarrow x_{\text{upp}}^j$ 
5: end if
6: Output:  $v_i^j(t+1)$  /*constricted velocity*/

```

---

A second phase of RPSO-vm concerns the *restarting* strategy. Similar to other known algorithms like CHC (Eshelman 1991) and G-CMA-ES (Auger and Hansen 2005), our proposal is stopped whenever one stopping criterion described below is met, and a restart is launched. The decision on when to restart the algorithm is made according to two independent criteria:

1. Stop if the standard deviation of the fitness values of particles in the entire swarm is smaller than  $1e - 3$ . In this case, the particles are restarted by randomly initializing their positions with a probability of  $1/DIM$  (lines 18–26 in Algorithm 2).
2. Stop if the overall change in the objective function value is below  $1e - 8$  for  $10 \cdot DIM/\text{size}(S)$  generations. In this case, the particles are restarted by calculating their derivatives to the global best position  $b$  and dividing them into two (lines 27–33 in Algorithm 2). This way, we force the particles to go to the best but avoiding the global convergence.

Applying the first restarting criteria, our algorithm tries to mitigate the early stagnation that basic PSO usually suffers from, especially in multimodal functions. In spite of working with high inertia and/or high social influences ( $\varphi_1$  and  $\varphi_2$ ), which moves the particles to distant positions, the PSO tends to be easily trapped in unproductive regions. This drawback is specially sensitive in functions with multiple local optima such as Rastrigin and its hybrids.

The second restarting criterion is based on the existence of plateaus and quite regular regions in functions like Rosenbrock, Schwefel, and their hybrids that make the PSO to spend a number of function evaluations (with time and computing resources) without an effective improvement. In this case, particles tend to spread them in the search space avoiding the influence of the best particle. Therefore, after a certain number of function evaluations without improvement, the particles are moved to their derivatives with regard to the best position.

**Algorithm 2** Pseudocode of RPSO-vm

---

```

1:  $t \leftarrow 0$ 
2: initialize( $S(t)$ ) /* Swarm  $S(0)$  */
3: while not stop condition is met ( $MAXIMUM(t)$ ) do
4:   /****** Particle Swarm *****/
5:   for each particle position  $\mathbf{x}_i(t)$  of the swarm  $S(t)$  do
6:     for each variable  $j$  of the particle position  $\mathbf{x}_i(t)$  do
7:        $v_{aux}^j \leftarrow \omega \cdot v_i^j(t) + \phi_1 \cdot UN(0, 1) \cdot (p_i^j(t) - x_i^j(t))$ 
         $\quad + \phi_2 \cdot UN(0, 1) \cdot (b^j(t) - x_i^j(t))$ 
8:        $v_i^j(t+1) \leftarrow velmod(v_{aux}^j)$ 
9:        $x_{aux}^j \leftarrow x_i^j(t) + v_i^j(t+1)$ 
10:      if  $x_{low}^j < x_{aux}^j \leq x_{upp}^j$  then
11:         $x_i^j(t+1) \leftarrow x_{aux}^j$ 
12:      else
13:         $x_i^j(t+1) \leftarrow x_i^j(t) - v_i^j(t+1)$ 
14:      end if
15:    end for
16:  end for
17:  /****** Restarting *****/
18:  if  $std(S) < 1e-3$  then
19:    for each particle position  $\mathbf{x}_i(t)$  of  $S(t)$  (with  $\mathbf{x}_i(t) \neq b(t)$ ) do
20:      for each variable  $j$  of the particle position  $\mathbf{x}_i(t)$  do
21:        if  $r^j(t) < 1/DIM$  (with  $r^j(t) \in [0, 1]$ ) then
22:           $x_i^j(t+1) \leftarrow x_{low}^j + UN(0, 1) \cdot (x_{upp}^j - x_{low}^j)$ 
23:        end if
24:      end for
25:    end for
26:  end if
27:  if  $change(fit(b)) < 1e-8$  for  $(10 \cdot DIM)/size(S)$  steps then
28:    for each particle position  $\mathbf{x}_i(t)$  of  $S(t)$  (with  $\mathbf{x}_i(t) \neq b(t)$ ) do
29:      for each variable  $j$  of the particle position  $\mathbf{x}_i(t)$  do
30:         $x_i^j(t+1) \leftarrow (b^j(t) - x_i^j(t))/2$ 
31:      end for
32:    end for
33:  end if
34:   $t \leftarrow t + 1$ 
35: end while
36: Output:  $b$  /*The best solution found*/

```

---

Algorithm 2 shows the complete pseudo-code of the RPSO-vm algorithm developed for this work. First, an initialization process of all particles in the swarm  $S$  is carried out. After this, each evolution step the particle's positions are updated following the velocity variation model of the equations previously explained (lines 5–16). If stopping criteria are reached, the algorithm restarts modifying the particles, excepting for the best one (lines 18–33). Finally, the algorithm returns the best solution found during the whole process.

## 4 Experimental setup

In this section, we present the experimental methodology and statistical procedure followed to evaluate and to compare our proposal. This experimentation has been defined in the scope of the Special Issue on *Scalability of Evolutionary Algorithms and other Metaheuristics for*

*Large Scale Continuous Optimization Problems* (SOCO'10), available in URL <http://www.sci2s.ugr.es/eamhco/CFP.php>.

We have implemented our RPSO-vm in C++ using the MALLBA library (Alba et al. 2007), a framework of metaheuristics. The benchmark of functions was tackled including the C-code provided in this special issue to our implementation of RPSO-vm. A complete package of this software is available in the new version release of MALLBA<sup>1</sup>. Following the specifications of the SOCO'10 experimental procedure, we have performed 25 independent runs of RPSO-vm for each test function and dimension. The study has been made with dimension  $D = 50, 100, 200, 500$ , and  $1,000$  continuous variables. The measures provided are the Average, the Maximum, the Minimum, and the Median of error of the best individuals found in the 25 runs. For a solution  $x$ , the error measure is defined as:  $f(x) - f^*$ , where  $f^*$  is the optimum fitness of the function. The maximum number of fitness evaluations has been stated to  $5,000 \cdot D$ , which constitutes the stop condition of each run.

To analyze the results we have used non-parametric (Sheskin 2003) tests. These tests use the mean ranking of each algorithm. We have applied them since several times the functions might not follow the conditions of normality and homoscedasticity to apply parametric tests with security (García et al. 2009). In particular, we have considered the application of the Iman and Davenport test, and Holm's test as post-hoc procedure. The former is used to know beforehand if there are statistically relevant differences in compared algorithms. In that case, a post-hoc procedure, the Holm's test, is then employed to know which algorithms are statistically worse than the reference algorithm with the best ranking.

### 4.1 Benchmark functions

The test suite elaborated for this Special Issue is composed by 19 functions with different properties (Herrera et al. 2010): unimodal, multimodal, separable, non-separable, shifted, and hybrid composed. Functions f1 to f6 were defined for CEC'08 (Tang et al. 2007) and functions f7–f11 were defined for ISDA'09 (Herrera and Lozano 2009) (and shifted for SOCO'10), where the previous ones were also used. Finally, functions f12–f19 have been created specifically for this Special Issue. Table 1 shows their names, bounds, and optimum values. We can describe several properties of the functions that we consider interesting.

<sup>1</sup> MALLBA Library <http://www.neo.lcc.uma.es/mallba/easy-mallba/html/mallba.html>. Directory Mallba/rep/PSO/soco2010.

- Functions f1 and f2 are shifted unimodal, functions f3–f6 are shifted multimodal and functions f7–f11 are shifted unimodal.
- Functions f2, f3, f5, f9, and f10 are non-separable. That is specially interesting to analyze if our proposal obtains good results in non-separable functions since we can observe its capacity of managing correlated variables, a typical property in real world problems.
- Functions f12–f19 are hybrid composition functions. They have been generated by composing ( $\oplus$ ) two functions, one or both of them non-separable. For these compositions, functions f7–f11 have been used in their non-shifted versions (NS). A composition uses a splitting mechanism to graduate the proportion (in parentheses in Table 1) of non-separable variables in the complete search space.

#### 4.2 Parameter settings

Table 2 shows the parameter settings used to configure our proposal, RPSO-vm. These parameters were tuned in the context of the ISDA'09 special session of real parameter optimization (Herrera and Lozano 2009) reaching results statistically similar to the best participant algorithm in that special session. These values of parameters were kept the same for all the experiments. The inertia weight changes linearly by following Eq. 3.

**Table 1** SOCO'10 test suite of functions

Number	Name	Intervals	$f^*$
f1	Shifted Sphere	$[-100, 100]$	-450
f2	Shifted Schwefel 2.21	$[-100, 100]$	-450
f3	Shifted Rosenbrock	$[-100, 100]$	390
f4	Shifted Rastrigin	$[-5, 5]$	-330
f5	Shifted Griewank	$[-600, 600]$	-180
f6	Shifted Ackley	$[-32, 32]$	-140
f7	Shifted Schwefel 2.22	$[-10, 10]$	0
f8	Shifted Schwefel 1.2	$[-65.536, 65.536]$	0
f9	Shifted Extended f10	$[-100, 100]$	0
f10	Shifted Bohachevsky	$[-15, 15]$	0
f11	Shifted Schaffer	$[-100, 100]$	0
f12	Hybrid NS f9 $\oplus$ f1 (0.25)	$[-100, 100]$	0
f13	Hybrid NS f9 $\oplus$ f3 (0.25)	$[-100, 100]$	0
f14	Hybrid NS f9 $\oplus$ f4 (0.25)	$[-5, 5]$	0
f15	Hybrid NS f10 $\oplus$ NS f7 (0.25)	$[-10, 10]$	0
f16	Hybrid NS f9 $\oplus$ f1 (0.50)	$[-100, 100]$	0
f17	Hybrid NS f9 $\oplus$ f3 (0.75)	$[-100, 100]$	0
f18	Hybrid NS f9 $\oplus$ f4 (0.75)	$[-5, 5]$	0
f19	Hybrid NS f10 $\oplus$ NS f7 (0.75)	$[-10, 10]$	0

**Table 2** Parameter setting used in RPSO-vm

Description	Parameter	Value
Swarm size	size ( $S$ )	10
Inertia weight	$\omega$	$0.0 \leftarrow 0.1$
Individual coefficient	$\varphi_1$	1.5
Social coefficient	$\varphi_2$	1.5

#### 5 Analysis of results

In this section, the results are presented and several analyses are made as follows: first, we carry out a brief analysis of the performance of our proposal in terms of the improvement obtained by both, velocity modulation and restarting mechanisms. In this sense, an additional comparison is made concerning the neighborhood topology of RPSO-vm in terms of global best versus local best guidance of particles. Second, the scalability analysis is tackled in comparison with provided results of other algorithms (DE, CHC, and G-CMA-ES) for all dimensions.

Finally, we present the computational effort required in terms of average running time.

##### 5.1 RPSO-vm performance results

As specified in benchmarking requirements of this Special Issue, we show in Table 3 the average, the maximum, the minimum, and the median of the best error values found in 25 independent runs of our RPSO-vm, for each function and for each dimension. In this table, we have marked in bold face the average error values since they will be used in advance for comparisons (as recommended in this test bed). Nevertheless, we can notice that median values are frequently better than average values, especially in shifted Extended\_f10 (f9) and several hybrid functions (f14, f16 and f17) where the distribution of results are scattered.

A first analysis consists of studying the improvement obtained by RPSO-vm with regards to basic PSO algorithm. Table 4 shows the mean errors (in 25 runs) obtained by RPSO-vm in comparison with the ones of PSO only with restarting (RPSO), PSO only with velocity modulation (PSO-vm), and the basic PSO. Additionally, we have included to this comparison the standard version of PSO (SPSO 2007) consisting of the *best* PSO. This version uses a variable random topology for selecting the best neighbor (*b*) for each particle (Ghosh et al. 2009). The resulting algorithm (*lb*)RPSO-vm incorporates both, modulation velocity and restart mechanisms in order to obtain an as fair as possible comparison. For this specific analysis, we have only focused on 1,000 variables dimension since it allows the most interesting analysis.

**Table 3** Maximum, minimum, median, and mean errors obtained by RePSOVM for all dimensions

Dimension	Value	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
50	Maximum	2.84E-14	1.58E-02	1.86E+04	2.84E-14	3.20E-01	1.78E-11	1.78E-14	2.13E+03	6.77E+00	0.00E+00
	Median	2.68E-14	6.60E-03	1.90E+01	2.66E-14	7.36E-02	2.66E-13	0.00E+00	1.04E+03	3.12E-06	0.00E+00
	Minimum	1.87E-14	4.36E-03	5.21E-02	0.00E+00	0.00E+00	1.24E-13	0.00E+00	4.37E+02	0.00E+00	0.00E+00
	Mean	<b>2.62E-14</b>	<b>7.54E-03</b>	<b>1.75E+03</b>	<b>2.30E-14</b>	<b>9.53E-02</b>	<b>1.49E-12</b>	<b>1.14E-15</b>	<b>1.06E+03</b>	<b>2.94E-01</b>	<b>0.00E+00</b>
100	Maximum	2.84E-14	2.92E-01	9.65E+03	2.84E-14	3.18E-01	9.52E-11	0.00E+00	1.89E+04	2.80E+00	0.00E+00
	Median	2.80E-14	2.01E-01	1.19E+02	2.77E-14	8.80E-02	6.32E-13	0.00E+00	1.05E+04	2.58E-05	0.00E+00
	Minimum	1.57E-14	1.17E-01	3.57E-01	1.17E-14	1.31E-14	2.42E-13	0.00E+00	6.82E+03	5.93E-07	0.00E+00
	Mean	<b>2.66E-14</b>	<b>1.98E-01</b>	<b>1.42E+03</b>	<b>2.61E-14</b>	<b>1.07E-01</b>	<b>4.76E-12</b>	<b>0.00E+00</b>	<b>1.09E+04</b>	<b>1.85E-01</b>	<b>0.00E+00</b>
200	Maximum	2.84E-14	2.49E+00	6.60E+03	2.84E-14	7.74E-01	8.20E-12	0.00E+00	7.06E+04	2.51E+01	0.00E+00
	Median	2.75E-14	2.01E+00	6.80E+01	2.74E-14	1.18E-01	1.89E-12	0.00E+00	5.05E+04	1.48E-03	0.00E+00
	Minimum	0.00E+00	1.65E+00	8.75E-02	0.00E+00	1.35E-14	6.96E-13	0.00E+00	3.81E+04	1.35E-06	0.00E+00
	Mean	<b>2.53E-14</b>	<b>2.00E+00</b>	<b>1.03E+03</b>	<b>2.43E-14</b>	<b>1.73E-01</b>	<b>2.95E-12</b>	<b>0.00E+00</b>	<b>5.23E+04</b>	<b>1.67E+00</b>	<b>0.00E+00</b>
500	Maximum	2.84E-14	1.81E+01	1.47E+04	2.84E-14	6.01E-01	7.01E-12	1.13E-14	3.62E+05	2.28E+01	0.00E+00
	Median	2.82E-14	1.71E+01	1.38E+02	2.77E-14	2.52E-01	2.59E-12	0.00E+00	3.00E+05	1.81E+00	0.00E+00
	Minimum	1.73E-14	1.49E+01	2.70E-02	0.00E+00	1.42E-14	1.21E-12	0.00E+00	2.47E+05	1.90E-03	0.00E+00
	Mean	<b>2.65E-14</b>	<b>1.67E+01</b>	<b>1.13E+03</b>	<b>2.44E-14</b>	<b>2.54E-01</b>	<b>3.14E-12</b>	<b>0.00E+00</b>	<b>3.00E+05</b>	<b>4.85E+00</b>	<b>0.00E+00</b>
1,000	Maximum	2.84E-14	4.69E+01	2.07E+03	3.03E-13	8.68E-01	2.41E-11	1.55E-14	1.14E+06	3.17E+01	0.00E+00
	Median	2.82E-14	4.29E+01	1.37E+02	2.78E-14	1.12E-01	3.75E-12	0.00E+00	9.21E+05	9.84E+00	0.00E+00
	Minimum	2.07E-14	3.99E+01	1.85E+01	2.27E-14	1.22E-14	2.60E-12	0.00E+00	7.39E+05	6.45E-02	0.00E+00
	Mean	<b>2.72E-14</b>	<b>4.29E+01</b>	<b>3.21E+02</b>	<b>4.81E-14</b>	<b>2.14E-01</b>	<b>4.92E-12</b>	<b>0.00E+00</b>	<b>9.35E+05</b>	<b>1.17E+01</b>	<b>0.00E+00</b>
Dimension	Value	f11	f12	f13	f14	f15	f16	f17	f18	f19	
50	Maximum	3.07E-01	2.14E+00	1.49E+04	1.04E+00	0.00E+00	1.05E-09	9.08E+03	1.26E+00	0.00E+00	
	Median	3.84E-06	0.00E+00	1.43E+01	1.72E-14	0.00E+00	1.68E-11	5.06E+01	3.18E-09	0.00E+00	
	Minimum	0.00E+00	0.00E+00	2.67E-02	0.00E+00	0.00E+00	2.20E-12	9.41E-01	4.60E-10	0.00E+00	
	Mean	<b>1.68E-02</b>	<b>8.58E-02</b>	<b>6.57E+02</b>	<b>6.81E-02</b>	<b>0.00E+00</b>	<b>7.88E-11</b>	<b>8.73E+02</b>	<b>5.05E-02</b>	<b>0.00E+00</b>	
100	Maximum	3.83E+00	2.30E-13	2.56E+04	1.20E+00	0.00E+00	1.11E-06	3.56E+04	1.61E+00	0.00E+00	
	Median	1.84E-05	0.00E+00	1.06E+02	4.38E-14	0.00E+00	4.64E-10	1.51E+02	5.04E-08	0.00E+00	
	Minimum	2.58E-08	0.00E+00	5.36E-02	0.00E+00	0.00E+00	5.17E-12	1.71E-01	1.46E-09	0.00E+00	
	Mean	<b>4.61E-01</b>	<b>1.60E-14</b>	<b>2.25E+03</b>	<b>1.27E-01</b>	<b>0.00E+00</b>	<b>4.87E-08</b>	<b>1.76E+03</b>	<b>1.36E-01</b>	<b>0.00E+00</b>	
200	Maximum	3.73E+00	1.16E+00	1.23E+05	9.95E-01	0.00E+00	1.29E+01	2.40E+05	3.73E+00	0.00E+00	
	Median	5.47E-02	2.79E-14	1.48E+02	3.23E-12	0.00E+00	1.02E-09	2.77E+02	2.60E-08	0.00E+00	
	Minimum	4.37E-05	0.00E+00	7.56E-01	1.50E-14	0.00E+00	5.35E-11	1.31E-01	3.33E-09	0.00E+00	
	Mean	<b>5.66E-01</b>	<b>4.64E-02</b>	<b>1.17E+04</b>	<b>7.96E-02</b>	<b>0.00E+00</b>	<b>5.40E-01</b>	<b>2.08E+04</b>	<b>1.50E-01</b>	<b>0.00E+00</b>	
500	Maximum	1.56E+01	2.98E-07	1.79E+04	1.36E+01	0.00E+00	3.04E+01	8.31E+03	1.41E+01	0.00E+00	
	Median	3.95E+00	2.75E-13	7.20E+01	2.50E-11	0.00E+00	3.46E-08	2.16E+01	7.80E-01	0.00E+00	
	Minimum	1.70E-03	0.00E+00	2.74E-01	2.05E-13	0.00E+00	6.84E-10	7.85E-01	1.44E-08	0.00E+00	
	Mean	<b>4.88E+00</b>	<b>1.32E-08</b>	<b>1.33E+03</b>	<b>1.29E+00</b>	<b>0.00E+00</b>	<b>2.12E+00</b>	<b>5.72E+02</b>	<b>2.47E+00</b>	<b>0.00E+00</b>	
1,000	Maximum	3.67E+01	1.16E-08	2.78E+04	2.60E+00	0.00E+00	7.53E+00	3.28E+04	6.52E+00	0.00E+00	
	Median	9.61E+00	2.02E-12	2.27E+02	4.07E-08	0.00E+00	2.19E-06	4.02E+01	1.33E+00	0.00E+00	
	Minimum	7.67E-02	2.98E-14	4.16E+01	5.39E-13	0.00E+00	6.27E-09	1.79E+00	2.93E-07	0.00E+00	
	Mean	<b>1.10E+01</b>	<b>1.00E-09</b>	<b>1.93E+03</b>	<b>5.27E-01</b>	<b>0.00E+00</b>	<b>9.50E-01</b>	<b>2.82E+03</b>	<b>1.80E+00</b>	<b>0.00E+00</b>	

As we can see in Table 4, RPSO-vm obtains the higher number of best error values (15 out of 19 in bold), and followed by RPSO. The other versions are clearly worse than the formers. In addition, after applying the Iman-Davenport test to see whether there are significant

differences between them we obtained a test value of 166.07 with a critical value of 3.77 (with  $\alpha = 0.05$ ), which proves that there is an evident improvement of RPSO-vm over PSO.

More precisely, Table 5 contains the results of a multicomparison Holm test where we can see that RPSO-vm is



**Table 4** Mean errors obtained by RPSO-vm, RPSO, PSO-vm, and PSO for dimension 1,000

F/D	RPSO-vm	RPSO	PSO-vm	PSO	(lb)RPSO-vm
f1	2.72E-14	<b>2.69E-14</b>	5.61E+06	5.55E+06	6.31E+06
f2	<b>4.29E+01</b>	4.38E+01	1.72E+02	1.72E+02	1.89E+02
f3	<b>3.21E+02</b>	1.26E+04	5.43E+12	5.54E+12	7.65E+12
f4	<b>4.81E-14</b>	3.98E-02	2.32E+04	2.32E+04	2.56E+04
f5	2.14E-01	<b>1.77E-01</b>	4.98E+04	5.05E+04	5.65E+04
f6	<b>4.92E-12</b>	5.13E-12	2.14E+01	2.14E+01	2.15E+01
f7	<b>0.00E+00</b>	2.00E-14	4.93E+03	4.98E+03	3.34E+27
f8	<b>9.35E+05</b>	9.38E+05	2.03E+07	2.53E+07	8.33E+07
f9	<b>1.17E+01</b>	1.34E+01	1.20E+04	1.20E+04	1.28E+04
f10	<b>0.00E+00</b>	<b>0.00E+00</b>	2.16E+05	2.16E+05	2.57E+05
f11	<b>1.10E+01</b>	1.33E+01	1.20E+04	1.20E+04	1.28E+04
f12	<b>1.00E-09</b>	1.15E-01	4.20E+06	4.18E+06	4.73E+06
f13	1.93E+03	<b>7.38E+02</b>	4.13E+12	4.01E+12	5.86E+12
f14	<b>5.27E-01</b>	6.28E-01	1.76E+04	1.78E+04	1.96E+04
f15	<b>0.00E+00</b>	<b>0.00E+00</b>	3.81E+04	3.67E+04	5.37E+18
f16	9.50E-01	<b>7.39E-01</b>	2.67E+06	2.70E+06	3.14E+06
f17	<b>2.82E+03</b>	1.42E+04	9.53E+11	9.33E+11	1.64E+12
f18	<b>1.80E+00</b>	3.35E+00	7.20E+03	7.20E+03	8.29E+03
f19	<b>0.00E+00</b>	<b>0.00E+00</b>	1.14E+05	1.11E+05	6.59E+12

**Table 5** Comparison of RPSO-vm versus (lb)RPSO-vm, RPSO, PSO-vm, and PSO according to Holm's multicompare test ( $\alpha = 0.05$ )

<i>i</i>	Algorithm	<i>z</i>	<i>p</i> value	$\alpha/i$	Sig. dif?
4	(lb)RPSO-vm	7.02	2.09E-12	0.012	Yes
3	PSO	4.10	4.06E-05	0.016	Yes
2	PSO-vm	4.10	4.06E-05	0.025	Yes
1	RPSO	0.41	6.81E-01	0.050	No

statistically better than all PSO versions, excepting RPSO. In this case, RPSO-vm obtained a better ranking than RPSO but without significant differences. Therefore, the main consequence is that velocity modulation (PSO-vm) can improve the performance of basic PSO, although it is in the case of PSO with restarting method (RPSO) where a significant improvement is obtained. In the case of (lb)RPSO-vm, we suspect that the fact of using the same parameter setting specifically fine-tuned for RPSO-vm (global best) could lead this version of PSO to perform inadequately in our experiments.

These preliminary results lead us to definitively use both, the velocity modulation and the restarting method to design our proposed PSO for large scale optimization (RPSO-vm).

## 5.2 Scalability analysis

This section is focused on analyzing the capability of our RPSO-vm to scale with the dimension of the search space of each function. As proposed in this Special Issue, the scalability study is made in comparison with other well-known algorithms in the state of the art. These algorithms are a version of DE (DE/1/exp) (Price et al. 2005), CHC (Eshelman 1991), and G-CMA-ES (Auger and Hansen 2005). The descriptions and the parameter settings of these algorithms can be found in (Herrera et al. 2010). Therefore, we first analyze the results of RPSO-vm dimension by dimension, and secondly, we made a brief study from a general point of view of the scalability behavior of RPSO-vm regarding several selected functions (f2, f9, f14, and f19) of the SOCO'10 benchmark.

An initial study with all these results consists of applying an Iman-Davenport test to see if there exist significant differences between them for all considered dimensions. Table 6 shows the results of this test, where we can effectively notice that there are statistical differences in compared results. In fact, for almost all cases the test values (ID value) increase with the dimension, which means that there are higher differences between compared algorithms in large scales (500 and 1,000) than in small dimensions (50, 100 and 200). Hence, we can know beforehand that there is an algorithm with poor scalability behavior, at least.

Following this general point of view, Table 7 shows the results of applying a multicomparison Holm's test to all mean fitness values obtained by each algorithm for each dimension. We must notice that G-CMA-ES could not obtained any result for dimension 1,000 (Table 12), hence it has not been considered for comparisons regarding the largest scale.

The main observation we can draw from Table 7 is that there are two algorithms: DE, and RPSO-vm that clearly show a better average distribution than the remaining ones. In addition, these ranks are kept for all dimensions. Concretely, DE reached the best rank and for this reason it has been considered as the reference algorithm for this

**Table 6** Results of the Iman-Davenport's (ID) test of RPSO-vm and all compared algorithms for each dimension ( $\alpha = 0.05$ )

Dimension	ID value	Critical value	Sig. dif?
50	13.80	2.53	Yes
100	13.43	2.53	Yes
200	12.76	2.53	Yes
500	14.37	2.53	Yes
1,000	30.04	2.84	Yes

**Table 7** Comparison of DE versus RPSO-vm, CHC, and G-CMA-ES according to Holm's multicompare test ( $\alpha = 0.05$ )

DIM	$i$	Algorithm	$z$	$p$ value	$\alpha/i$	Sig. dif?
50	3	CHC	4.77	1.79E-06	0.016	Yes
	2	G-CMA-ES	2.95	3.14E-03	0.025	Yes
	1	<b>RPSO-vm</b>	1.57	1.16E-01	0.050	No
100	3	CHC	4.71	2.45E-06	0.016	Yes
	2	G-CMA-ES	2.89	3.85E-03	0.025	Yes
	1	<b>RPSO-vm</b>	1.44	1.48E-01	0.050	No
200	3	CHC	4.649	3.33E-06	0.016	Yes
	2	G-CMA-ES	2.76	5.70E-03	0.025	Yes
	1	<b>RPSO-vm</b>	1.38	1.66E-01	0.050	No
500	3	CHC	4.52	6.07E-06	0.016	Yes
	2	G-CMA-ES	3.70	2.09E-04	0.025	Yes
	1	<b>RPSO-vm</b>	1.57	1.16E-01	0.050	No
1,000	2	CHC	4.62	3.77E-06	0.025	Yes
	1	<b>RPSO-vm</b>	0.97	3.30E-01	0.050	No

statistical test. Nevertheless, our RPSO-vm is the only algorithm that does not shown significant statistical differences (sig. dif) with regard to DE (the reference), and resulting the lower difference precisely in the largest dimension (1,000 variables). This is an important indicator that confirms us the successful performance of our proposal in terms of scalability.

The following Tables 8, 9, 10, 11 and 12 contain the results of all compared algorithms for dimensions, 50, 100, 200, 500, and 1,000, respectively. The last column in these tables shows the results of RPSO-vm, indicating in bold face such values for which the mean error is the best found. A detailed study leads us to analyze the results dimension by dimension in the following.

### 5.2.1 Dimension 50

Table 8 shows mean errors (of 25 runs) obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 50. In this case, DE obtained the best results in 13 functions, 7 of them in hybrid composition functions.

RPSO-vm obtained the best results in 7 functions, and G-CMA-ES obtained the best results in three functions.

**Table 8** Mean errors obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 50

f/Alg.	DE	CHC	G-CMA-ES	RPSO-vm
f1	<b>0.00E+00</b>	1.67E-11	0.00E+00	2.62E-14
f2	3.60E-01	6.19E+01	<b>2.75E-11</b>	7.54E-03
f3	2.89E+01	1.25E+06	<b>7.97E-01</b>	1.75E+03
f4	3.98E-02	7.43E+01	1.05E+02	<b>2.30E-14</b>
f5	<b>0.00E+00</b>	1.67E-03	2.96E-04	9.53E-02
f6	<b>1.43E-13</b>	6.15E-07	2.09E+01	1.49E-12
f7	<b>0.00E+00</b>	2.66E-09	1.01E-10	<b>0.00E+00</b>
f8	3.44E+00	2.24E+02	<b>0.00E+00</b>	1.06E+03
f9	2.73E+02	3.10E+02	1.66E+01	<b>2.94E-01</b>
f10	<b>0.00E+00</b>	7.30E+00	6.81E+00	<b>0.00E+00</b>
f11	<b>6.23E-05</b>	2.16E+00	3.01E+01	1.68E-02
f12	<b>5.35E-13</b>	9.57E-01	1.88E+02	8.58E-02
f13	<b>2.45E+01</b>	2.08E+06	1.97E+02	6.57E+02
f14	<b>4.16E-08</b>	6.17E+01	1.09E+02	6.81E-02
f15	<b>0.00E+00</b>	3.98E-01	9.79E-04	<b>0.00E+00</b>
f16	1.56E-09	2.95E-09	4.27E+02	<b>7.88E-11</b>
f17	<b>7.98E-01</b>	2.26E+04	6.89E+02	8.73E+02
f18	<b>1.22E-04</b>	1.58E+01	1.31E+02	5.05E-02
f19	<b>0.00E+00</b>	3.59E+02	4.76E+00	<b>0.00E+00</b>

**Table 9** Mean errors obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 100

f/Alg.	DE	CHC	G-CMA-ES	RPSO-vm
f1	<b>0.00E+00</b>	3.56E-11	0.00E+00	2.66E-14
f2	4.45E+00	8.58E+01	<b>1.51E-10</b>	1.98E-01
f3	8.01E+01	4.19E+06	<b>3.88E+00</b>	1.42E+03
f4	7.96E-02	2.19E+02	2.50E+02	<b>2.61E-14</b>
f5	<b>0.00E+00</b>	3.83E-03	1.58E-03	1.07E-01
f6	<b>3.10E-13</b>	4.10E-07	2.12E+01	4.76E-12
f7	<b>0.00E+00</b>	1.40E-02	4.22E-04	<b>0.00E+00</b>
f8	3.69E+02	1.69E+03	<b>0.00E+00</b>	1.09E+04
f9	5.06E+02	5.86E+02	1.02E+02	<b>1.85E-01</b>
f10	<b>0.00E+00</b>	3.30E+01	1.66E+01	<b>0.00E+00</b>
f11	<b>1.28E-04</b>	7.32E+01	1.64E+02	4.61E-01
f12	5.99E-11	1.03E+01	4.17E+02	<b>1.60E-14</b>
f13	<b>6.17E+01</b>	2.70E+06	4.21E+02	2.25E+03
f14	<b>4.79E-02</b>	1.66E+02	2.55E+02	1.27E-01
f15	<b>0.00E+00</b>	8.13E+00	6.30E-01	<b>0.00E+00</b>
f16	<b>3.58E-09</b>	2.23E+01	8.59E+02	4.87E-08
f17	<b>1.23E+01</b>	1.47E+05	1.51E+03	1.76E+03
f18	<b>2.98E-04</b>	7.00E+01	3.07E+02	1.36E-01
f19	<b>0.00E+00</b>	5.45E+02	2.02E+01	<b>0.00E+00</b>

The Holm's test with  $\alpha = 0.05$  (Table 7) showed that DE, the algorithm with best ranking is statistically better than all algorithms, excepting RPSO-vm with  $p$  value=0.05.

**Table 10** Mean errors obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 200

f/Alg.	DE	CHC	G-CMA-ES	RPSO-vm
f1	<b>0.00E+00</b>	8.34E-01	0.00E+00	2.53E-14
f2	1.92E+01	1.03E+02	<b>1.16E-09</b>	2.00E+00
f3	1.78E+02	2.01E+07	<b>8.91E+01</b>	1.03E+03
f4	1.27E-01	5.40E+02	6.48E+02	<b>2.43E-14</b>
f5	<b>0.00E+00</b>	8.76E-03	<b>0.00E+00</b>	1.73E-01
f6	<b>6.54E-13</b>	1.23E+00	2.14E+01	2.95E-12
f7	<b>0.00E+00</b>	2.59E-01	1.17E-01	<b>0.00E+00</b>
f8	5.53E+03	9.38E+03	<b>0.00E+00</b>	5.23E+04
f9	1.01E+03	1.19E+03	3.75E+02	<b>1.67E+00</b>
f10	<b>0.00E+00</b>	7.13E+01	4.43E+01	<b>0.00E+00</b>
f11	<b>2.62E-04</b>	3.85E+02	8.03E+02	5.66E-01
f12	<b>9.76E-10</b>	7.44E+01	9.06E+02	4.64E-02
f13	<b>1.36E+02</b>	5.75E+06	9.43E+02	1.17E+04
f14	1.38E-01	4.29E+02	6.09E+02	<b>7.96E-02</b>
f15	<b>0.00E+00</b>	2.14E+01	1.75E+00	<b>0.00E+00</b>
f16	<b>7.46E-09</b>	1.60E+02	1.92E+03	5.40E-01
f17	<b>3.70E+01</b>	1.75E+05	3.36E+03	2.08E+04
f18	<b>4.73E-04</b>	2.12E+02	6.89E+02	1.50E-01
f19	<b>0.00E+00</b>	2.06E+03	7.52E+02	<b>0.00E+00</b>

**Table 11** Mean errors obtained by DE, CHC, G-CMA-ES, and RPSO-vm for dimension 500

f/Alg.	DE	CHC	G-CMA-ES	RPSO-vm
f1	<b>0.00E+00</b>	2.84E-12	0.00E+00	2.65E-14
f2	5.35E+01	1.29E+02	<b>3.48E-04</b>	1.67E+01
f3	4.76E+02	1.14E+06	<b>3.58E+02</b>	1.13E+03
f4	3.20E-01	1.91E+03	2.10E+03	<b>2.44E-14</b>
f5	<b>0.00E+00</b>	6.98E-03	2.96E-04	2.54E-01
f6	<b>1.65E-12</b>	5.16E+00	2.15E+01	3.14E-12
f7	<b>0.00E+00</b>	1.27E-01	7.21E+153	<b>0.00E+00</b>
f8	6.09E+04	7.22E+04	<b>2.36E-06</b>	3.00E+05
f9	2.52E+03	3.00E+03	1.74E+03	<b>4.85E+00</b>
f10	0.00E+00	1.86E+02	1.27E+02	<b>0.00E+00</b>
f11	<b>6.76E-04</b>	1.81E+03	4.16E+03	4.88E+00
f12	<b>7.07E-09</b>	4.48E+02	2.58E+03	1.32E-08
f13	<b>3.59E+02</b>	3.22E+07	2.87E+03	1.33E+03
f14	<b>1.35E-01</b>	1.46E+03	1.95E+03	1.29E+00
f15	<b>0.00E+00</b>	6.01E+01	2.82E+262	<b>0.00E+00</b>
f16	<b>2.04E-08</b>	9.55E+02	5.45E+03	2.12E+00
f17	<b>1.11E+02</b>	8.40E+05	9.59E+03	5.72E+02
f18	<b>1.22E-03</b>	7.32E+02	2.05E+03	2.47E+00
f19	<b>0.00E+00</b>	1.76E+03	2.44E+06	<b>0.00E+00</b>

### 5.2.2 Dimension 100

In this case, in spite of reaching DE the best mean error in more functions than RPSO-vm and G-CMA-ES (see Table 9), both Friedman's and Holm's test showed similar results to dimension 50. That is, RPSO-vm and DE are statistically similar to themselves, but better than CHC, and G-CMA-ES.

### 5.2.3 Dimension 200

As shown in Table 10, the results are quite similar to the previous ones of dimension 100. In fact, the same algorithms (RPSO-vm, DE, and G-CMA-ES) obtained the best mean errors practically in the same functions. Holm's test also obtained that DE is statistically similar to RPSO-vm ( $p$  value = 0.05), and better than the rest of algorithms.

### 5.2.4 Dimension 500

Table 11 contains the mean errors of all algorithms in dimension 500. We can see the set functions for which RPSO-vm always obtained the best mean fitness: f4, f7, f9, f10, f15, and f19. In particular Shifted Schwefel 2.22 (f7) and its hybrids (f15 and f19) are optimized for all dimensions. These functions are unimodal separable (f7) and unimodal non-separable (f9, f10, f15, and 19) which could lead us to think that RPSO-vm only has successful

**Table 12** Mean errors obtained by DE, CHC, and RPSO-vm for dimension 1,000

f/Alg.	DE	CHC	RPSO-vm
f1	<b>0.00E+00</b>	1.36E-11	2.72E-14
f2	8.46E+01	1.44E+02	<b>4.29E+01</b>
f3	9.69E+02	8.75E+03	<b>3.21E+02</b>
f4	1.44E+00	4.76E+03	<b>4.81E-14</b>
f5	<b>0.00E+00</b>	7.02E-03	2.14E-01
f6	<b>3.29E-12</b>	1.38E+01	4.92E-12
f7	<b>0.00E+00</b>	3.52E-01	<b>0.00E+00</b>
f8	<b>2.46E+05</b>	3.11E+05	9.35E+05
f9	5.13E+03	6.11E+03	<b>1.17E+01</b>
f10	<b>0.00E+00</b>	3.83E+02	<b>0.00E+00</b>
f11	<b>1.35E-03</b>	4.82E+03	1.10E+01
f12	1.68E-08	1.05E+03	<b>1.00E-09</b>
f13	<b>7.30E+02</b>	6.66E+07	1.93E+03
f14	6.90E-01	3.62E+03	<b>5.27E-01</b>
f15	<b>0.00E+00</b>	8.37E+01	<b>0.00E+00</b>
f16	<b>4.18E-08</b>	2.32E+03	9.50E-01
f17	<b>2.36E+02</b>	2.04E+07	2.82E+03
f18	<b>2.37E-03</b>	1.72E+03	1.80E+00
f19	<b>0.00E+00</b>	4.20E+03	<b>0.00E+00</b>

performance with unimodal functions, but we can easily check that our proposal obtained the best results for f4 (multimodal), and for all dimensions. In addition, RPSO-

vm obtained the second best mean fitness for the remaining of hybrid functions (dimension 500). Statistically, Holm's test confirms our initial hypothesis since it showed (Table 7) that the results of RPSO-vm are not significantly different to the ones of DE ( $p$  value = 0.05), and they are statistically better than the results of the rest of algorithms.

### 5.2.5 Dimension 1000

For the largest scale, Table 12 shows the mean errors where RPSO-vm obtained the best results in 10 out of 19 functions. As aforementioned, G-CMA-ES did not obtained any value for dimension 1,000. Regarding dimension 500, the set of functions for which RPSO-vm obtained the best mean fitness has been increased with f2, f3, f12, and f14, having these functions different properties of modality and separability. As happened in all dimensions, in spite of having DE the best average ranking, the Holm's test (Table 7) showed RPSO-vm is statistically similar to DE. In comparison with CHC, our proposal is statistically the best algorithm.

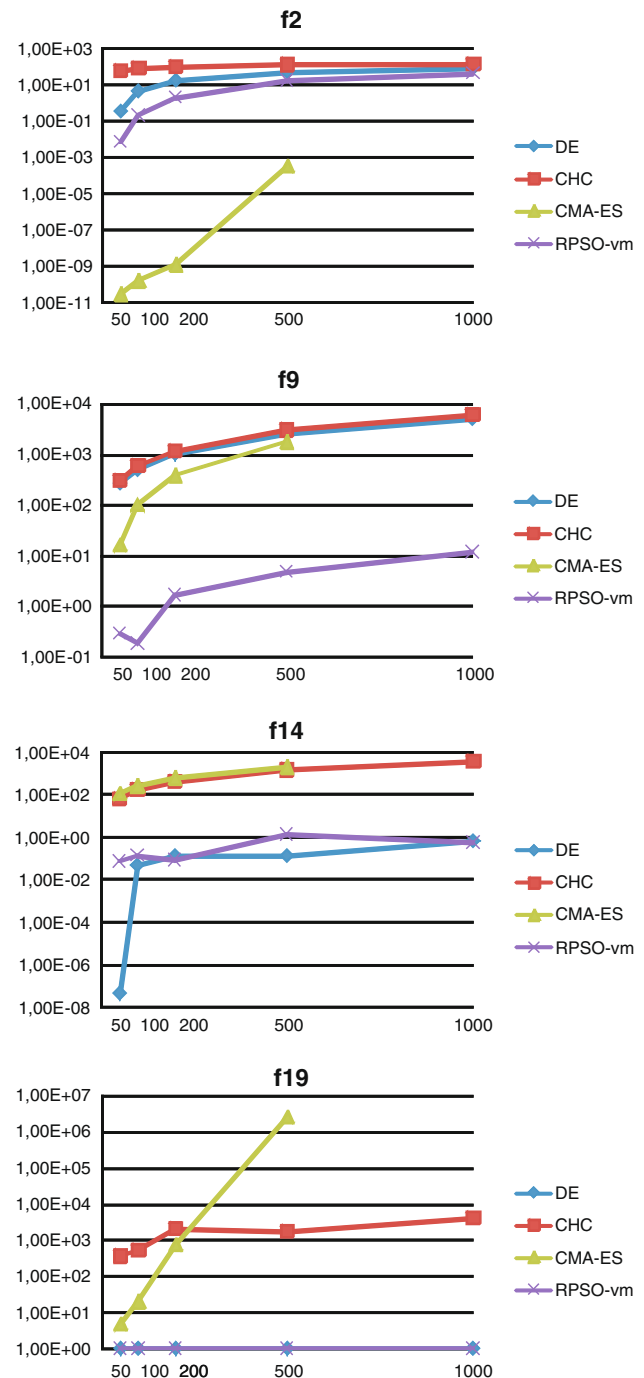
From a graphical point of view, Fig. 1 illustrates the tendency of results of the compared algorithms and RPSO-vm for functions f2, f9, f14, and f19 through the different dimensions. We have chosen these functions since they showed a representative behavior in terms of scalability.

Thus, we can observe in Fig. 1 that the performance of all algorithms deteriorates with the increment of the dimension. Nevertheless, this degradation is slight in almost all cases, and even nonexistent in others, as happened in functions f2 and f19 for algorithms RPSO-vm and DE. A different and anomalous behavior is observed in G-CMA-ES for functions f2 and f19, where it diminishes quickly. We suspect that the use of covariance matrix mechanism of G-CMA-ES is unsuitable for large dimensions due to the great amount of resources it requires (Hansen et al. 2003; Knight and Lunacek 2007).

### 5.3 Computational effort

Finally, we present in this section some remarks about the computational effort. To execute these experiments, we have used the computers of the laboratories of the Department of Computer Science of the University of Málaga (Spain). Most of them are equipped with modern dual-core processors, 1GB RAM, and Linux so, having into account that there are more than 180 computers, that means that up to 360 cores have been available. To run all the programs, we have used the Condor (Thain et al. 2005) middleware that acts as a distributed task scheduler (each task dealing with one independent run of RPSO-vm).

In Table 13, we present the average running time (seconds) in which RPSO-vm has found the best mean error for



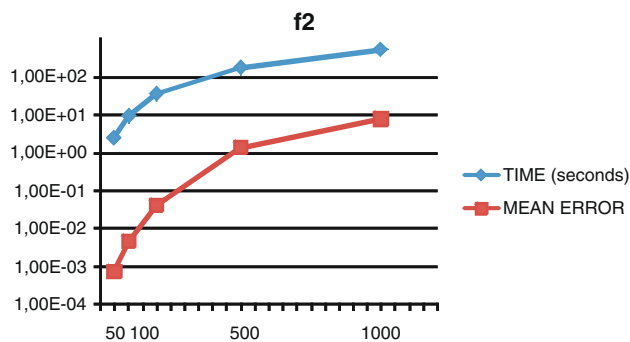
**Fig. 1** Scalable results of DE, CHC, G-CMA-ES, and RPSO-vm for functions f2, f9, f14, and f19. Y axis shows the results in logarithmic scale. X axis shows the problem dimensions

all functions and for all dimensions. As expected, the running time increases with the number of variables, specially in non-separable functions. Specifically, f1 (Shifted Sphere) required the lowest time to be optimized for all dimensions, and f17 (Hybrid NS f9⊕f3) took the longest time. In general, hybrid composition functions required more time to reach their best value than simple functions.



**Table 13** Average running time (ART), in seconds, of the 25 runs of RPSO-vm for all functions and for all dimensions D

ART/D	50	100	200	500	1,000
f1	7.91E-01	3.33E+00	1.36E+01	8.42E+01	3.60E+02
f2	2.72E+00	9.67E+00	4.10E+01	2.55E+02	9.37E+02
f3	6.30E+00	2.66E+01	9.85E+01	5.71E+02	2.50E+03
f4	2.82E+00	1.26E+01	5.50E+01	3.81E+02	1.52E+03
f5	2.19E+00	8.06E+00	3.82E+01	2.24E+02	8.60E+02
f6	4.82E+00	1.65E+01	7.43E+01	4.14E+02	1.76E+03
f7	2.42E+00	8.62E+00	3.58E+01	2.11E+02	8.72E+02
f8	2.27E+00	9.49E+00	3.41E+01	2.24E+02	8.02E+02
f9	9.05E+00	3.28E+01	1.32E+02	1.11E+03	3.19E+03
f10	4.14E+00	1.79E+01	6.40E+01	4.12E+02	1.74E+03
f11	9.23E+00	3.61E+01	1.43E+02	9.64E+02	3.92E+03
f12	4.17E+00	1.59E+01	6.63E+01	4.04E+02	1.33E+03
f13	7.44E+00	2.76E+01	9.97E+01	7.62E+02	3.03E+03
f14	5.91E+00	2.50E+01	9.30E+01	5.34E+02	2.20E+03
f15	2.78E+00	1.16E+01	4.43E+01	2.67E+02	8.90E+02
f16*	5.49E+00	2.20E+01	9.45E+01	6.09E+02	2.25E+03
f17*	8.48E+00	3.13E+01	1.41E+02	7.00E+02	3.24E+03
f18*	7.18E+00	3.05E+01	1.23E+02	7.47E+02	2.93E+03
f19*	4.01E+00	1.38E+01	6.15E+01	3.61E+02	1.43E+03

**Fig. 2** Differences in times versus mean error magnitudes of f2 for each dimension. Y axis contains values in logarithmic scale and X axis contains dimensions

In this sense, an interesting observation consists in comparing the increment of both, the processing time and the optimum mean error found, through the different scales of the search space. This way, we can obtain insights about the computational effort required with regards to the quality of solutions obtained. Figure 2 shows a representative case observed in function f2, where the increment of the processing time as well as the mean error is practically linear. If we take into account that the search space grows exponentially with the dimension  $[x_{low}, x_{upp}]^{DIM}$  in all functions, we can claim that our proposal scales successfully. Concerning the quality of solutions, the deterioration that the mean error suffers is higher in comparison with the

processing time. Specifically, from dimension 200–500, the mean error increases in two orders of magnitude while the time required takes less than one order of magnitude. Curiously, the difference in the mean error between 500 and 1,000 dimensions is not bigger than one order of magnitude which leads us to suspect that our proposal performs relatively better in larger dimensions than in smaller ones.

## 6 Conclusions

In this work, we have incorporated both *velocity modulation* and *restarting* mechanisms to the PSO with the aim of enhancing its scalability. Our hypothesis is that these two new mechanisms can help the PSO to avoid the early convergence and redirects the particles to promising areas in the search space. The experimentation phase has been carried out in the scope of this special issue to test the ability of being scalable. The results obtained show that our proposal is scalable in all functions of the benchmark used, as well as highly competitive with regard to other compared optimizers. In concrete, we can remark the following:

the new proposal, called Restarting PSO with Velocity Modulation (RPSO-vm), outperforms the basic PSO, as well as PSO with each new mechanism separately, for all dimensions. Additionally, the RPSO-vm algorithm with *global best* neighborhood topology outperforms (*lb*)RPSO-vm: they two are the same algorithm but one has a global best (*gbest*) topology while the other has a variable neighborhood (*lbest*) topology.

RPSO-vm shows a competitive performance in terms of its scalability. In fact, it is the second best algorithm for all dimensions and statistically similar to the best one in comparison with provided algorithms in this Special Issue. These algorithms: DE, CHC, and G-CMA-ES, are well-known optimizers traditionally used for continuous optimization and showing an excellent performance in other benchmarks (CEC'05, CEC'08, BBOB'09, etc.).

RPSO-vm obtained the best results in functions f4, f7, f9, f10, f15, and f19 for all the dimensions. These functions are all shifted and they have different properties of modality, separability and composition. For the largest dimension (1,000), the set of functions in which our algorithm obtained the best results is increased with f2, f3, f12, and f14.

In terms of computational effort, the running time increases with the number of variables, specially in non-separable and hybrid composition functions. Additionally, we observed that from dimension 200–500 the mean error increased in two orders of magnitude while the time required takes less than one order of magnitude. The difference in the mean error between 500 and 1000 dimensions is not bigger than one order of magnitude. This leads

us to suspect that our proposal performs relatively better in larger dimensions than in smaller ones.

In general, we can conclude that modifying PSO, a simple well-known algorithm, we have reached a highly accurate performance even in large scale environments. In the light of these results, we are encouraged to follow betting on PSO based algorithms in future works.

**Acknowledgments** Authors acknowledge funds from the Spanish Ministry of Sciences and Innovation (MICINN) and FEDER under contract TIN2008-06491-C04-01 (M\* project <http://www.mstar.lcc.uma.es>) and CICE, Junta Andalucía, under contract P07-TIC-03044 (DIRICOM project <http://www.diricom.lcc.uma.es>). José García-Nieto is supported by grant BES-2009-018767 from the MICINN.

## References

- Alba E, Luque G, García-Nieto J, Ordonez G, Leguizamón G. (2007) Mallba: a software library to design efficient optimisation algorithms. *Int J Innov Comput Appl* 2007 (IJICA) 1(1):74–85
- Auger A, Hansen N (2005) A restart cma evolution strategy with increasing population size. *IEEE Congr Evol Comput* 2:1769–1776
- Clerc M, Kennedy J (2003) The particle swarm: explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans Evol Comput* 6(1):58–73
- Das S, Abraham A, Konar A (2008) Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. Springer, Berlin
- Eshelman LJ (1991) The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination. In: Rawlins G, Kaufmann M (eds) *Proceedings of foundations of genetic algorithms Conference*, vol 1, pp 265–283
- García S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the ea's behaviour: a case study on the CEC'2005 special session on real parameter optimization, vol 15, pp 617–644
- García-Nieto J, Apolloni J, Alba E, Leguizamón G (2009) Algoritmo basado en cúmulos de partículas y evolución diferencial para la resolución de problemas de optimización continua. In: *MAEB 2009 (VI Congreso español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados)*, Málaga, p 433–440
- Ghosh S, Kundu D, Suresh K, Das S, Abraham A, Panigrahi BK, Snel V (2009) On some properties of the lbest topology in particle swarm optimization. In: *Hybrid intelligent systems, 2009. HIS '09. Ninth international conference*, vol 3, pp 370–375, 12–14
- Hansen N, Auger A, Finck S, Ros R (2009) BBOB'09: Real-parameter black-box optimization benchmarking. Technical report RR-6828, INRIA.
- Hansen N, Auger A, Finck S, Ros R (2010) BBOB'10: Real-parameter black-box optimization benchmarking. Technical report RR-7215, INRIA.
- Hansen N, Müller SD, Petros K (2003) Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol Comput* 11(1):1–18
- Herrera F, Lozano M, Molina D (2010) Components and parameters of de, real-coded chc, and g-cmaes. Technical report, SCI2S, University of Granada, Spain
- Herrera F, Lozano M, Molina D (2010) Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems. Technical report, SCI2S. University of Granada, Spain
- Herrera F, Lozano M (2009) ISDA'09 workshop on evolutionary algorithms and other metaheuristics for continuous optimization problems—a scalability test. Technical report. University of Granada, Pisa, Italy
- Hsieh S, Sun T, Liu C, Tsai S (2008) Solving large scale global optimization using improved particle swarm optimizer. *Evolutionary Computation*, 2008. CEC 2008. IEEE world congress on computational intelligence. IEEE congress, pp 1777–1784
- Kennedy J, Eberhart RC (2001) *Swarm intelligence*. Morgan Kaufmann, San Francisco
- Knight JN, Lunacek M (2007) Reducing the space-time complexity of the CMA-ES. In: *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, New York, pp 658–665
- Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *Evol Comput IEEE Trans* 10(3):281–295
- Liang JJ, Suganthan PN (2005) Dynamic multi-swarm particle swarm optimizer with local search. *IEEE Congr Evol Comput* 1:522–528
- Montes de Oca MA, Stützle T, Birattari M, Dorigo M (2009) Frankenstein's pso: a composite particle swarm optimization algorithm. *Trans Evol Comput* 13(5):1120–1132
- Price KV, Storn R, Lampinen J (2005) *Differential evolution: a practical approach to global optimization*. Springer, London
- Shang Y, Qiu Y (2006) A note on the extended rosenbrock function. *Evol Comput* 14(1):119–126
- Sheskin DJ (2003) *Handbook of parametric and nonparametric statistical procedures*. Chapman & Hall/CRC Statistics, London/Boca Raton
- Shi Y, Eberhart RC (1998) Parameter selection in particle swarm optimization. In: *EP '98: Proceedings of the 7th international conference on evolutionary programming VII*, Springer, London, pp 591–600
- Suganthan PN, Hansen N, Liang JJ, Deb K, Chen Y-P, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC'05 special session on real-parameter optimization. Technical report KanGAL report 2005005, Nanyang Technological University, Singapore and Kanpur, India
- Suresh K, Ghosh S, Kundu D, Sen A, Das S, Abraham A (2008) Inertia-adaptive particle swarm optimizer for improved global search. In: *Proceedings of ISDA '08*, IEEE Computer Society, Washington, DC, pp 253–258
- Tang K, Xiaodong Li, Suganthan PN, Yang Z, Weise T (2010) Benchmark functions for the CEC'2010 special session and competition on large scale global optimization. Technical report Nature Inspired Computation and Applications Laboratory, USTC, Nanyang Technological University, anyang Technological University, China
- Tang K, Yao X, Suganthan PN, MacNish C, Chen YP, Chen CM, Yang Z (2007) Benchmark functions for the CEC'08 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China
- Thain D, Tannenbaum T, Livny M (2005) Distributed computing in practice: the condor experience. *Concurr Pract Exp* 17(2–4): 323–356
- Tseng L, Chen, C (2008) Multiple trajectory search for large scale global optimization. *Evolutionary computation*, 2008. CEC 2008, IEEE world congress on computational intelligence. IEEE Congress, pp 3052–3059
- van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. *Evol Comput IEEE Trans* 8(3):225–239