Hybrid PSO6 for Hard Continuous Optimization

José García-Nieto, and Enrique Alba

***Abstract*—In our previous works, we empirically showed that a number of $6_{\pm 2}$ informants may endow particle swarm optimization (PSO) with an optimized learning procedure in comparison with other combinations of informants. In this way, the new version PSO6, that evolves new particles from six informants (neighbors), performs more accurately that other existing versions of PSO and is able to generate good particles for a longer time. Despite this advantage, PSO6 may show certain attraction to local basins derived from its moderate performance on non-separable complex problems (typically observed in PSO versions). In this paper, we incorporate a local search procedure to the PSO6 with the aim of correcting this disadvantage. We compare the performance of our proposal (PSO6-Mtsls) on a set of 40 benchmark functions against that of other PSO versions, as well as against the best recent proposals in the current state of the art (with and without local search). The results support our conjecture that the (quasi)-optimally informed PSO, hybridized with local search mechanisms, reaches a high rate of success on a large number of complex (non-separable) continuous optimization functions.**

***Index Terms*—Particle Swarm Optimization, Fully Informed PSO, Multiple Trajectory Search, Benchmarking Functions.**

I. Introduction

IN a previous work [6], we have empirically shown that a number of $6_{\pm 2}$ informants enhances the particle swarm optimization (PSO) with new essential information about the search landscape, leading this algorithm to perform more accurately than other existing versions of it. More recently, using fitness-distance correlation and fitness-fitness cloud analyses, we tested in [7] that this number of $6_{\pm 2}$ informants put into PSO a better learning procedure than other combinations of informants, which makes it able to generate good particles for a longer time. In spite of this advantage, as typically observed in particle swarm versions, the new PSO6 (PSO with six informants) may still show a certain local basin attraction and moderate performance on non-separable complex problems. Therefore, additional mechanisms have to be used to correct this drawback.

In this paper, we add to our PSO6 a local search method, with the aim of improving its performance on complex problems with different landscape features: multimodal, multi-funnel, non-separable, shifted, and rotated. First, as proven in [24], PSO tends to converge quickly to the local basin that contains the majority of particles at initialization. A consequence of this observation is that PSO may exhibit higher tendency to stagnate on multimodal functions than other algorithms. In this case, we use our PSOk with six informants that performs an optimized learning procedure to move particles to more interesting regions, and hence avoid the attraction to non promising local basins. Second, as most of PSO versions work dimension by dimension, it is actually hard to find the problem optima when it is located far from the origin (or when it is on an axis or a diagonal) of coordinates in non-separable problems. To alleviate this last issue, we incorporate a local search method to our PSO6 by means of which, particles move individually, exploring their problem neighborhoods in the context of variations of dependent variables.

Our proposal, called PSO6-Mtsls (PSO6 with Multiple Trajectory Search), is then evaluated on a set of 40 benchmark functions in order to validate whether it is competitive with the current state of the art in continuous optimization, or not. The sequence of tasks that we have performed in this study to validate this hypothesis is as follows:

1) We first experiment with PSO6 to determine which swarm size is better adapted to different problem dimensions. In our previous works [6], and [7], a fixed swarm size (30) was set on a different experimental framework. In this work, we are mainly focused on the experimental procedure proposed in the special issue on Large Scale Continuous Optimization of Soft Computing journal (SOCO'10) [9]. Therefore, we have performed an a-priori analysis concerning the best swarm size tailored to different problem scales.
2) We compare our proposal with related PSO versions concentrating on Standard PSO 2011 [21], Full Informed PSO [14], Full and Informed PSO with Square Neighborhood [14]. We also measure the effects of the local search mechanism (PSO6-Mtsls) with regards to our initial PSO6.
3) We assess the performance of our PSO6-Mtsls with regards to other 15 recent algorithms featured in SOCO'10 (with and without local search methods) from the point of view of the problem scalability, considering dimensions 50, 100, 200, and 500 (#problem variables).
4) We perform a further comparison of PSO6-Mtsls with against similar modern swarm intelligent approaches also hybridized with local search methods: IPSO-Powell [16], IPSO-Mtsls [17], and IACOr-Mtsls [13]. For this experimentation, we use an extended benchmark including functions from the special session of Continuous Optimization of CEC'05 [23] to SOCO'10 functions. In this way, rotated functions are also considered. In all these comparisons, we pay special attention on non-separable problems.

After such a thorough analysis, we show that a slightly improved PSO6 is able of performing equal or better to other techniques, removing the old PSO problems, and providing it with high chance of success on hard continuous optimization.

José García-Nieto and Enrique Alba are with Dept. Lenguajes y Ciencias de la Computación, University of Málaga, Campus de Teatinos, 29071, Málaga-Spain. E-mail: jnieto@lcc.uma.es, eat@lcc.uma.es

The remainder of this paper is organized as follows. We start in Section II by giving some background concepts. In Section III we describe our PSO6 and we detail the main motivations that prompted us to use it in this work. Then, our proposed approach PSO6-Mtsls is introduced in Section IV. The core of the paper with experiments and analyses are given in Section V. Finally, Section VII includes the most interesting concluding remarks and future work.

II. Background

In this section, preliminary concepts concerning the Multiple Trajectory Search algorithms and the Fully Informed PSO are given.

A. Multiple Trajectory Search

The Multiple Trajectory Search (MTS) algorithm was initially designed for multi-objective optimization showing accurate results in this domain [29]. However, it is currently becoming also popular for large scale continuous optimization [28], since it showed the best performance in the special session on Large Scale Global Optimization of CEC'08 [25].

In MTS, after an initialization phase using simulated orthogonal array (SOA), a number of three different local search procedures are applied to each individual, selecting the best from the three new solutions for the new iteration. In our case, we are only interested in the first local search procedure LS1 (out of the three ones used in MTS) since, in most of cases, it achieved larger improvements than the other two (LS2, and LS3) [28], for the seven functions in CEC'08 (note that CEC'08 $\subset$ SOCO'10). Taking into account that, our aim here is to test a new proposal on standard benchmarks using a predefined number of evaluations, thus we have decided to concentrate only on LS1 procedure and provide the PSO6 with more evaluations.

LS1 searches along one dimension, from the first dimension to the last one. As shown in Algorithm 1, LS1 uses three initial parameters: a solution to be optimized X_k, generally initialized to the best solution found in the calling procedure (in our case, the best particle b^t in PSO6), a boolean value $Improve$, and a search range SR. The last two values are initialized in lines 1 to 7. For each dimension concerning the search, the solution's coordinate of this dimension is first subtracted by SR to see if the objective function value is improved (lines 9, and 10). If it is, the best solution is updated and the search proceeds to consider the next dimension. If it is not, the X_k solution is restored to its original value and then the solution's coordinate of this dimension is added by $0.5 \cdot SR$, again to see if the objective function value is improved (lines 13 to 19). If it is, the search proceeds to consider the next dimension. If it is not, the solution is restored and the search proceeds to consider the next dimension (lines 19 to 31). As a result, the global best in the calling process (b^t) is updated just when an improvement is obtained.

Algorithm 1 Pseudocode of MTS: $LS1(X_k, Improve_k, SR_k)$

```
1:  if $Improve_k = False$ then
2:      $SR_k \leftarrow SR_k/2$
3:      if $SR_k < 1e-14$ then
4:          $SR_k \leftarrow (Upper\_Bound - Lower\_Bound) \cdot 0.4$
5:      end if
6:  end if
7:  $Improve_k \leftarrow False$
8:  for $i = 1\ to\ \#Dimensions$ do
9:      $X_k \leftarrow X_k[i] - SR$
10:     if $f(X_k) < f(b^t)$ then
11:         $b^t \leftarrow X_k$
12:     end if
13:     if $f(X_k) = f(b^t)$ then
14:         $Restore(X_k)$
15:     else
16:         if $f(X_k) > f(b^t)$ then
17:             $Restore(X_k)$
18:             $X_k \leftarrow X_k[i] + 0.5 \cdot SR$
19:             if $f(X_k) < f(b^t)$ then
20:                 $b^t \leftarrow X_k$
21:             end if
22:             if $f(X_k) \geq f(b^t)$ then
23:                 $Restore(X_k)$
24:             else
25:                 $Improve_k \leftarrow True$
26:             end if
27:         else
28:             $Improve_k \leftarrow True$
29:         end if
30:     end if
31: end for
32: Output: $SR_k, Improve_k$    //The best solution found
```

B. From Standard PSO to FIPS

The canonical particle swarm optimization (PSO) [10], as well as recent standard versions of this algorithm (Standards 2006, 2007, and 2011) [21], work by iteratively generating new particles' positions located in a given problem search space. Each one of these new particles' positions are calculated using the particle's current position (solution), the particle's previous velocity, and two main informant terms: the particle's best previous location, and the best previous location of any of its neighbors. We could call ir PSO2 for this reason.

Formally, in canonical PSO each particle's position vector $\mathbf{x}_i$ is updated each time step t by means of the Equation 1.

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

where $\mathbf{v}_i^{t+1}$ is the velocity vector of the particle given by

$$\mathbf{v}_i^{t+1} = \omega \mathbf{v}_i^t + U^t[0, \varphi_1] \cdot (\mathbf{p}_i^t - \mathbf{x}_i^t) + U^t[0, \varphi_2] \cdot (\mathbf{b}_i^t - \mathbf{x}_i^t) \quad (2)$$

In this formula, $\mathbf{p}_i^t$ is the personal best position the particle i has ever stored, $\mathbf{b}_i^t$ is the position found by the member of its neighborhood that has had the best performance so far. Acceleration coefficients φ_1 and φ_2 control the relative effect of the personal and social best particles, and U^t is a diagonal matrix with elements distributed in the interval $[0, \varphi_i]$, uniformly at random. Finally, $\omega \in (0, 1)$ is called the

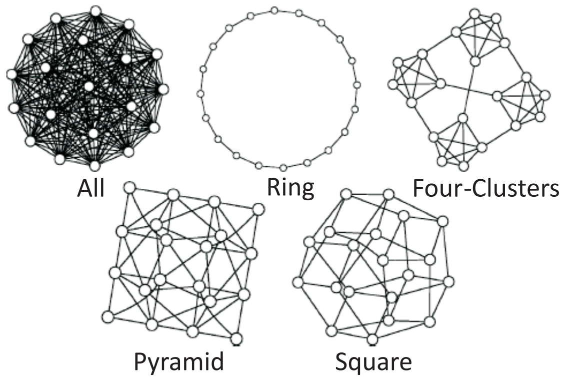

Fig. 1. Topologies used by Mendes et al. [14]. Each particle has a number of fixed neighbors in the swarm (All=N-1; Ring=2; Four-Clusters=4,5; Pyramid=3,5,6; Square=4)

inertia weight and influences the tradeoff between exploitation and exploration.

An equivalent version of the velocity equation was reported in [3], where Clerc's constriction coefficient χ is used instead of inertia weight as shown in Equation 3.

$$\mathbf{v}_i^{t+1} = \chi\left(\mathbf{v}_i^t + U^t[0,\varphi_1]\cdot(\mathbf{p}_i^t - \mathbf{x}_i^t) + U^t[0,\varphi_2]\cdot(\mathbf{b}_i^t - \mathbf{x}_i^t)\right) \tag{3}$$

$$\chi = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|},\ with\ \varphi = \sum_i \varphi_i,\ and\ \varphi > 4 \tag{4}$$

Constriction coefficient χ is calculated, by means of Equation 4, from the two acceleration coefficients φ_1 and φ_2, being the sum of these two coefficients what determines the χ to use. Usually, $\varphi_1 = \varphi_2 = 2.05$, giving as results $\varphi = 4.1$, and $\chi = 0.7298$ [4], [27]. As stated by Mendes et all. [14], [15], this fact implies that the particle's velocity can be adjusted by any number of informant terms, as long as acceleration coefficients sum to an appropriate value, since important information given by other neighbors about the search space may be neglected through overemphasis on the single best neighbor. With this assumption, Mendes et all. [14] proposed the Fully Informed Particle Swarm (FIPS), in which a particle uses information from all its topological neighbors. In FIPS, the value φ, that is, the sum of the acceleration coefficients, is equally distributed among all the neighbors of a particle. Therefore, for a given particle i with position $\mathbf{x}_i$, φ is broken up into several smaller coefficients $\varphi_j = \varphi/|\mathcal{N}_i|, \forall j \in \mathcal{N}_i$. Then, the velocity is updated as follows:

$$\mathbf{v}_i^{t+1} = \chi\left[\mathbf{v}_i^t + \sum_{j\in\mathcal{N}_i} U^t\left[0,\varphi_j\right]\cdot\left(\mathbf{p}_j^t - \mathbf{x}_i^t\right)\right], \tag{5}$$

where $\mathcal{N}_i$ is the set of neighbors of the particle i, and following the neighborhood a given topology. Fig. 1 illustrates the topologies used by Mendes et al. [14] as the ones with most successful performances in a previous work [11]. These topologies are: All, Ring, Square, Four-Clusters, and Pyramid. Their results show that the Square topology (with 4 informants) outperforms the other ones.

Indeed, the fact of defining these neighborhoods in the swarm makes the particles to be influenced only by a certain number of neighbors, and connected with static links in the graph. Once again, important information may be disregarded through overemphasis, in this case, of structured sets of neighbors. The number of informants seems to play also an important role, but with no clue on how many of them is the best choice, or if even the good issue is the neighborhood topology itself or the fact that only a few informants are used, forgetting the intermediate sources of information existing between Canonical/Standard PSO and FIPS versions.

III. PSO6: Quasi-optimal Number of Informants in PSO

As previously commented, the possibility of adjusting the particle's velocity by an arbitrary number of terms enables us to generalize the number (k) of neighbors, from 1 to Ss (being Ss the swarm size). Therefore, a number Ss of different versions of PSO can be generated (selecting k particles of the swarm without replacement), each one of them with neighborhoods containing k particles. Obviously, if $k = Ss$ the resultant version is the FIPS algorithm with neighborhood "ALL", as illustrated in Fig. 1.

Nevertheless, since providing each k neighborhood with structured topologies is impracticable due to the huge number of graph combinations, we have opted in this work to simply selecting k random (uniform) informants of the swarm (S). This way, for each particle i, and at each time step t, a different neighborhood ($\mathcal{N}_i^t$) with k elements is generated, and hence, the number of informants can be analyzed with independence of any structured topology. Formally, we can represent a given neighborhood as follows:

$$\mathcal{N}_i^t = \{n_1,\ldots,n_k\} \mid \mathcal{N}_i^t \subset S^t, \forall n_j, n_h \in \mathcal{N}_i^t, n_h \neq n_j \neq i \tag{6}$$

Following this scheme, we designed in our previous work [6] a new PSO called PSOk, which proceeds as formulated in Equation 5, and using sets of k random (uniform) informant particles as neighborhoods. The pseudocode of PSOk is reproduced in Algorithm 2. After swarm initialization and φ_j value calculation (lines 1 to 3), the optimization process is repeated until reaching the stop condition. In this, at each iteration and for each particle, a new neighborhood is randomly (uniformly) generated by fulfilling conditions of Equation 6 (line 6). Then, particle's velocity, current position, and local best position are updated (lines 7 to 9). Finally, the best so far particle position is returned as output (line 13).

Then, we evaluated all the PSOk versions (with k : $1 \ldots Ss$) [6] in order to discover whether an optimal value, or range of values, exists that allowed to improve over the Standard PSO and avoids the overhead of using topologies or computing contributions from all particles in the swarm. In this evaluation, we tested a number of 30 PSOk versions ($Ss = 30$) on the benchmark of 25 functions provided in the special session of continuous optimization of CEC'05 [23]. As a summary of results, Fig. 2 shows the frequency histogram

Algorithm 2 Pseudocode of PSOk

```
1: $t \leftarrow 0$
2: $\varphi_j \leftarrow \varphi / k$
3: initialize($S^t$) /* Swarm $S^0$ with N particles */
4: while $t < MAXIMUM(t)$) do
5:     for each particle $i^t$ of $S^t$ do
6:         $\mathcal{N}_i^t \leftarrow generate\_neighborhood(k, i, S^t)$   //Eq. 6
7:         $\mathbf{v}_i^{t+1} \leftarrow update\_velocity(\mathbf{v}_i^t, \mathbf{x}_i^t, \varphi_j, \mathcal{N}_i^t)$   //Eq. 5
8:         $\mathbf{x}_i^{t+1} \leftarrow update\_position(\mathbf{x}_i^t, \mathbf{v}_i^{t+1})$   //Eq. 1
9:         $\mathbf{p}_i^{t+1} \leftarrow update\_local\_best(\mathbf{p}_i^t, \mathbf{x}_i^{t+1})$
10:    end for
11:    $t \leftarrow t + 1$
12: end while
13: Output: $b$ /*The best solution found*/
```

of the best median performance (number of Hits) reached by each different PSOk version, for all CEC'05 functions.

The most interesting observation in Fig. 2 is that a number of 6 informants in the neighborhood makes the PSOk to perform with success in practically all evaluated functions (CEC'05). In addition, the range of 6±2 informants concentrated most of the successful runs. This led us to suspect that, on the one hand, less than 4 informants is a deficient value of k, since hard landscapes make regular PSO to stuck in local optima with no enough guide coming from the two best particles in the swarm to scape from them. On the other hand, more than 8 informants is redundant, since lots of particles are just representing the same movements as others, that is, there are classes of equivalence in the swarm (basins of attraction) that are providing redundant samples to interfere with the numerical velocity computation for $k > 8$ (including FIPS). We also made in [6] comparisons of PSO6 with Standard PSO 2007, Standard PSO 2011, FIPS-ALL and FIPS-USquare (the best on in [14]). In addition, we performed a series of analyses focused of the swarm size, the problem dimension (using CEC'08), and the algorithm complexity. All this supported our initial assumptions concerning PSO6, in the scope of the used experimental framework (CE'05).

More recently, continuing with this research line, we carried out a further analysis from the point of view of the evolvability [7]. Our motivation in that work was to find evidences of why such number of 6±2 informants perform better than other neighborhood formulations in PSO. Then, we performed a comprehensive analysis focused on three main metrics: the fitness distance correlation (fdc), the fitness cloud (fc), and the escape probability (ep).

In this study [7], we first observed that using few informants (< 4) leads the PSOk to show a positive fitness-distance correlation, although it evolves solutions with poor fitness values and far from the global optima. With more than 8 informants, solutions are again correlated, although concentrating on small non interesting regions of the landscape. Using 6 informants is the best trade-off between fitness-distance and fitness quality. A illustrative example graphic can be found in Fig. 3, where the median fdc coefficients (out of 25 independent runs) of the different PSOk versions (for the 30 possible values of k) are plotted, for function 25 (Rotated Hybrid Composed in CEC'05). In this figure, for (almost) all combinations of PSOk, solutions are correlated, although with poor fitness and far from the optimum. In this case, PSO6 shows anticorrelated solutions with accurate fitness, although far from the optimum (deceiving function).

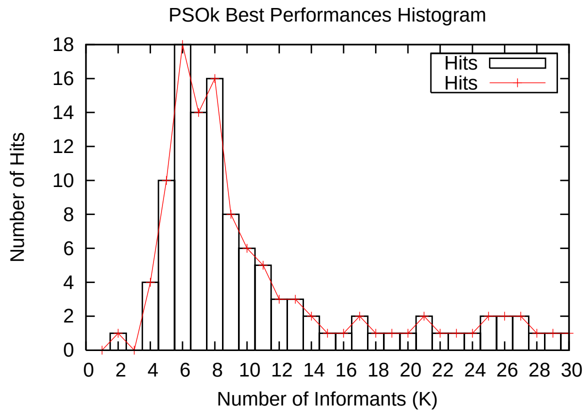

Fig. 2. Frequency histogram of best median performance (number of Hits) reached by each different PSOk version, for all CEC'05 functions. The complete experimentation can be found in [6]

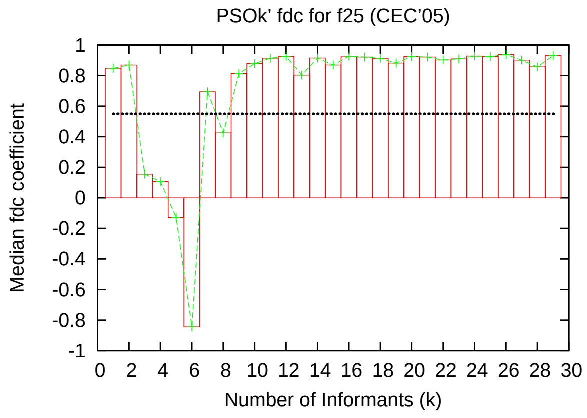

Fig. 3. Median fdc coefficients (out of 25 independent runs) of the different PSOk versions (for the 30 possible values of k), for function 25 (CEC'05). The dotted line represents the fdc coefficients computed from 10,000 uniform random samples for this function. The complete experimentation can be found in [7]

Secondly, we realized that solutions evolved by PSO6 generally show a moderate but maintained escape probability progress, finally reaching deeper basins, e.g., better fitness values than with other number of informants . In general the behavior of all PSOk versions is stable for different problem dimensions (10, 30, and 50). In concrete, PSO6 shows quite similar evolvability indicators for the three studied dimensions.

In sort, after a thorough experimentation, we came to consider PSO6 as a prominent optimizer in continuous optimization, as well as to use it as base algorithm to compose new hybrid approaches.

IV. Algorithm Proposal: PSO6-Mtsls

Our proposal, PSO6 with Multiple Trajectory Search (PSO6-Mtsls), consists in running PSO6 as a baseline method in which we have incorporated a local search mechanism to improve solutions obtained by the particle swarm algorithm. In concrete, we have employed the well-known LS1 [28] of MTS because of three main reasons: (1) LS1 is the responsible of most of the MTS performance, (2) it has been proven to be an efficient optimizer on large scale and non-separable complex problems [28], and (3) it has been successfully used to hybridize other swarm intelligence approaches like IPSO [17], ACO [13], and DE [18]. In the context of the collective learning procedure induced by informant particles in PSOk, the LS1 procedure can be interpreted as a particle's individual learning ability that allows it to explore-explode its immediate area neighborhood in the absence of any social influence. In this sense, the movement of an individual particle depends on the improvement obtained from variations in its adjacent variables (solution dimensions) and hence, the interdependency of variables in non-separable problems can be tackled more effectively than simply using vector operators inducing linear combinations (as in PSO verions).

PSO6-Mtsls invokes a local search routine after a certain number of iterations, performing successive improvements on the global best particle ($\mathbf{b}^t$) obtained by PSO6 at moment (t) of invocation. We have to notice that, despite PSO6 does not work directly with the global best particle (but indirectly if it happen to take part in the current set of six informants), it is kept updated through the iteration procedure, to be used by LS1 as a target particle. If PSO6-Mtsls detects that an application of LS1 does not improve the solution, the local search is stopped. In this way, the additional cost in terms of extra function evaluations performed by LS1 can be lighten.

The pseudocode of PSO6-Mtsls can be observed in Algorithm 3 and is organized as follows: the first phase, from line 1 to 7, corresponds to parameter setting and swarm initialization. For the initialization of particles (line 3), we have partially used the method proposed in [19] to generate good diverse solutions. This method starts with the partition of the range of each dimension to sr subranges of equal size. Then, for each particle, a subrange for each dimension is selected based on the inverse probability of the frequency count associated with the subrange. Finally, a value is uniformly generated within the selected interval and the frequency count associated to the subrange is incremented. In a second phase, the PSO6-Mtsls is then iterated until the stop condition is met: a given number of function evaluations is reached. Finally, the PSO6 performs one iteration (lines 10 to 15), the global best is updated in line 16, and the local search L1 is invoked with a certain frequency according to ls_freq (line 18 to 27). In this case, the local search procedure is repeated a given number of iterations max_ls_iters while the solution is successively improved. In other case, the local search is aborted and the PSO6 follows with a new iteration. Once the stop condition is reached, the algorithm returns the best particle found so far.

Algorithm 3 Pseudocode of PSO6-Mtsls

```
1: $t \leftarrow 0$
2: $\varphi_j \leftarrow \varphi/k$
3: initialize($S^t$) /* Swarm $S^0$ with $Ss$ particles */
4: for $k = 1$ to $size(S^t)$ do
5:    $Improve_k \leftarrow True$
6:    $SR_k \leftarrow (Upper\_Bound - Lower\_Bound) \cdot 0.5$
7: end for
8: while $t < MAXIMUM(t)$ do
9:    /************* PSO$k$, with $k = 6$ *************/
10:   for each particle $i^t$ of $S^t$ do
11:      $\mathcal{N}_i^t \leftarrow generate\_neighborhood(k, i, S^t)$   //Eq. 6
12:      $\mathbf{v}_i^{t+1} \leftarrow update\_velocity(\mathbf{v}_i^t, \mathbf{x}_i^t, \varphi_j, \mathcal{N}_i^t)$   //Eq. 5
13:      $\mathbf{x}_i^{t+1} \leftarrow update\_positon(\mathbf{x}_i^t, \mathbf{v}_i^{t+1})$   //Eq. 1
14:      $\mathbf{p}_i^{t+1} \leftarrow update\_local\_best(\mathbf{p}_i^t, \mathbf{x}_i^{t+1})$
15:   end for
16:   $\mathbf{b}^{t+1} \leftarrow update\_global\_best(\mathbf{b}^t)$
17:   /***************** MTS:LS1 *****************/
18:   if $t\%ls\_freq = 0$ then
19:      $X_k \leftarrow \mathbf{b}^{t+1}$
20:      for $j = 1$ $to$ $\#max\_ls\_iters$ do
21:         $Improve_k, SR_k \leftarrow LS1(X_k, Improve_k, SR_k)$
22:         if $Improve_k = False$ then
23:            $break$
24:         end if
25:      end for
26:      $update(t)$
27:   end if
28:   $t \leftarrow t + 1$
29: end while
30: Output: $\mathbf{b}^t$ /* The best solution found */
```

V. Experiments

In this section, we present the experimental methodology and the statistical procedure applied to evaluate our PSO6-Mtsls and to compare it with other algorithms in the state of the art. The parameter setting is also described, paying special attention to the swarm size and to local search parameters.

A. Experimental Setup

We have implemented our PSO6-Mtsls in C++ following the skeleton architecture of the MALLBA library [1], a framework of metaheuristics. The problem functions were tackled including the C-code provided by each benchmark framework to our implementation of PSO6-Mtsls. A complete package of this software is publicity available in the new version release of MALLBA[1], to allow the reproduction by other researchers. The experiments were performed in computers at 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 Debian O.S. They operate under a Condor [26] middleware platform that acts as a distributed task scheduler (each task dealing with one independent run).

Our experimental study is structured in three different phases. First, we evaluate our PSO6-Mtsls and other related PSO versions concentrating on the original PSO6 without any local search procedure, the Standard PSO 2011 (S2011), the

[1]MALLBA Library, Directory Mallba/rep/PSO/soco2010 `http://neo.lcc.uma.es/mallba/easy-mallba/html/mallba.html`.

TABLE I

SOCO'10 and CEC'05 benchmark test suites with functions' features: unimodal (U), multimodal (M), separable and non-separable, rotated and non-rotated. The problem search ranges and the biases to optima values f^* are also specified

f	Name	Unimodal/Multimodal	Separable	Rotated	Search Range	f^*
soco1	Shifted Sphere	U	Y	N	[-100, 100]	-450
soco2	Shifted Schwefel 2.21	U	S	N	[-100, 100]	-450
soco3	Shifted Rosenbrock	M	N	N	[-100, 100]	390
soco4	Shifted Rastrigin	M	Y	N	[-5, 5]	-330
soco5	Shifted Griewank	M	N	N	[-600, 600]	-180
soco6	Shifted Ackley	M	Y	N	[-32, 32]	-140
soco7	Shifted Schwefel 2.22	U	Y	N	[-10, 10]	0
soco8	Shifted Schwefel 1.2	U	N	N	[-65.536, 65.536]	0
soco9	Shifted Extended f10	U	N	N	[-100, 100]	0
soco10	Shifted Bohachevsky	U	N	N	[-15, 15]	0
soco11	Shifted Schaffer	U	N	N	[-100, 100]	0
soco12	Hybr. Comp. soco9 $\bigoplus_{0.25}$ soco1	M	N	N	[-100, 100]	0
soco13	Hybr. Comp. soco9 $\bigoplus_{0.25}$ soco3	M	N	N	[-100, 100]	0
soco14	Hybr. Comp. soco9 $\bigoplus_{0.25}$ soco4	M	N	N	[-5, 5]	0
soco15	Hybr. Comp. soco10 $\bigoplus_{0.25}$ soco7	M	N	N	[-10, 10]	0
soco16	Hybr. Comp. soco9 $\bigoplus_{0.50}$ soco1	M	N	N	[-100, 100]	0
soco17	Hybr. Comp. soco9 $\bigoplus_{0.75}$ soco3	M	N	N	[-100, 100]	0
soco18	Hybr. Comp. soco9 $\bigoplus_{0.55}$ soco4	M	N	N	[-5, 5]	0
soco19	Hybr. Comp. soco10 $\bigoplus_{0.75}$ soco7	M	N	N	[-10, 10]	0
cec3	Shifted Rotated High Conditioned Elliptic	U	S	R	[-100, 100]	-450
cec4	Shifted Schwefel's Problem 1.2 with Noise	U	S	N	[-100, 100]	-450
cec5	Schwefel's Problem 2.6	U	S	N	[-100, 100]	-310
cec7	Shifted Rotated Griewank's. Global Optimum Outside of Bounds	M	S	R	[0, 600]	-180
cec8	Shifted Rotated Ackley's with Optimum on Bounds	M	S	R	[-32, 32]	-140
cec10	Shifted Rotated Rastrigin's	M	S	R	[-5, 5]	-330
cec11	Shifted Rotated Weierstrass	M	N	R	[-0.5, 0.5]	90
cec12	Schwefel's Problem 2.13	M	N	N	[-π, π]	-460
cec13	Shifted Expanded Griewank's plus Rosenbrock's	M	N	N	[-3, 1]	-130
cec14	Shifted Rotated Expanded Scaffer's F6	M	S	R	[-100, 100]	-300
cec15	Hybrid Composition (f1-f2,f3-f4,f5-f6,f7-f8,f9-f10)	M	N	N	[-5, 5]	120
cec16	Rotated Version of Hybrid Composition f15	M	N	R	[-5, 5]	120
cec17	F16 with Noise in Fitness	M	N	R	[-5, 5]	120
cec18	Rot. Hybr. Comp. (f1-f2,f3-f4,f5-f6,f7-f8,f9-f10)	M	N	R	[-5, 5]	10
cec19	Rot. Hybr. Comp. Narrow Basin Global Optimum	M	N	R	[-5, 5]	10
cec20	Rot. Hybr. Comp. Global Optimum on Bounds	M	N	R	[-5, 5]	10
cec21	Rot. Hybr. Comp. (f1-f2,f3-f4,f5-f6,f7-f8,f9-f10)	M	N	R	[-5, 5]	360
cec22	Rot. Hybr. Comp. High Condition Number Matrix	M	N	R	[-5, 5]	360
cec23	Non-Continuous Rotated Hybrid Composition	M	N	R	[-5, 5]	360
cec24	Rot. Hybr. Comp. (f1,f2,f3,f4,f5,f6,f7,f8,f9,f10)	M	N	R	[-5, 5]	260
cec25	Rot. Hybr. Comp. Global Optimum Outside of Bounds	M	N	R	[2, 5]	260

Fully Informed PSO (FIPS-ALL), and the Fully Informed Square Neighborhood (FIPS4, the best one in [14]), by comparing their performances. Second, our proposal is compared against other 15 algorithms featured in SOCO'10, on different problem scales with dimensions 50, 100, 200, and 500 continuous variables. The third experimental phase corresponds to the evaluation of PSO6-Mtsls with regards to other similar modern swarm intelligent approaches also hybridized with local search methods: IPSO-Powell [16], IPSO-Mtsls [17], and IACOr-Mtsls [13].

For the two first phases, we used the 19 functions (labeled $soco*$) provided in SOCO'10. In this benchmark, functions $soco1$ to $soco6$ were originally used in CEC'08 [25]. Functions $soco7$ to $soco11$ were added to the first ones in the special session of ISDA'09 [8], and functions $soco12$ to $soco19$ consist on hybridized functions that combine two others (being one of them non-separable). For the third phase, we extended the working set by including 21 more functions of CEC'05 (labeled as $cec*$) to the previous 19 of SOCO'10, then constituting a set of 40 functions. We have to notice that, as done in [13], from the original 25 functions of CEC'05 we omitted $cec1$, $cec2$, $cec6$, and $cec9$, since they are the same as $soco1$, $soco3$, $soco4$, and $soco8$. Table I shows the set of functions used in this study with their most interesting features: unimodal, multimodal, separable, non-separable, shifted to biased optimum, rotated, and hybrid composed. The respective bounds of search ranges and biases to optima are also indicated. The detailed descriptions of all these functions can be found in [9] and [23].

Following the specifications of the two benchmarks used, we have applied as stop conditions a maximum number of 5,000$\cdot D$ fitness evaluations for SOCO'10, and 10,000$\cdot D$ fitness evaluations for CEC'05 functions. We performed 25 independent runs for each investigated algorithm and problem dimension. We report the error values of the best solutions (x) found defined as: $f(x) - f^*$, where f^* is the optimum fitness of the function f. Error values lower than 10^{-14} (*0-threshold*) are approximated to zero.

To analyze the results, we have used non-parametric statistical tests, since some times the numerical distributions of results did not follow the conditions of normality and homoskedasticity [5]. Therefore, our analyses and comparisons are mainly focused on the whole distribution errors, although paying special attention on the Median errors (and not the Mean error), out of 25 independent runs. In particular, we have considered the application of the Friedman's ranking test, and use the Holm's multicompare test as post-hoc procedure [22] to know which algorithms are statistically worse than the reference algorithm (the one with the best ranking).

TABLE II
PARAMETER SETTINGS

Parameter Value	Algorithms
Swarm size $Ss = D \cdot 0.7$	All
Acceleration coefficient $\varphi = 4.1$	PSO6, PSO6-Mtsls, FIPS-ALL, FIPS4
Acceleration coefficient $\varphi = 0.5 + \ln(2)$	S2011
Inertia weight $\omega = 1/(2 \cdot \ln(2))$	S2011
Constriction coefficient $\chi = 0.7298$	PSO6, PSO6-Mtsls, FIPS-ALL, FIPS4
Number of Informants $k = 6$	PSO6, PSO6-Mtsls
Number of Informants $k = 4$	FIPS4
Number of Informants $k = Ss$	FIPS-ALL
Number of Informed by a give one $k = 3$	S2011
Topology $T = U - Random$	PSO6, PSO6-Mtsls
Topology $T = Square$	FIPS4
Topology $T = Complete$	FIPS-ALL
Topology $T = U - Random$	S2011
Local search frequency $ls\_freq = 5$	PSO6-Mtsls
Max. ls. Iterations $max\_ls\_iters = 70$	PSO6-Mtsls

TABLE III
SWARM SIZE PARAMETER TUNING: SUCCESSFUL RUNS WITH BEST PERFORMANCES RESULTED BY PSO6, WITH DIFFERENT SWARM SIZES ON DIFFERENT PROBLEM DIMENSIONS

	Swarm Size				
Dimension	20	30	60	100	200
50	0	19	0	0	0
100	0	0	19	0	0
200	0	0	0	19	0
500	0	0	0	0	19
Diff.		+	+	+	+

TABLE IV
LS1 PARAMETER TUNING: BEST PERFORMED VALUES ARE IN BOLD

	Values for problem dimension			
LS1 parameter	50	100	200	500
ls_freq	5,10,**30**,50	**5**,10,30,50	**5**,10,30,50	**5**,10,30,50
max_ls_iters	10,30,**50**,70	10,30,50,**70**	10,30,50,**70**	10,30,50,**70**

B. Parameter Settings

The parameter setting applied to PSO6, as well as to the other evaluated PSO versions, are shown in Table II and follow the specification of their original works were they where proposed [6], [14], and [21]. Nevertheless, concerning the swarm size, we have decided to perform an additional parameter tuning with PSO6, since in this work we are using a large set of functions with different dimension scales. Therefore, we have carried out a preliminary experimentation with PSO6 by setting it with different combinations of swarm sizes and problem dimensions, in the context of SOCO'10.

Table III contains the number of functions for which PSO6 obtains the best median results for each combination of swarm size and problem dimension. In this table, we can easily observe that the swarm size seems to be proportional to the problem dimension, since a higher swarm is more accurate for large scales, and opposite. The last row specifies that statistical differences were found in distributions (+). For this reason, and after additional runs, we have opted to use a linear proportion to set the swarm size by using the 70% of the problem dimension as the number of particles in the swarm. In this way, we use $Ss = D \cdot 0.7$ in Table II for PSO6, as well as for all other PSO versions.

In the case of our PSO6-Mtsls, specific parameters to the particle swarm use the same setting as in PSO6, including the proportional swarm size. For specific parameters to LS1, a series of tuning experiments have been also carried out to find an accurate combination of local search frequency and maximum number of iterations in the local search procedure. Table IV shows the experimented values for LS1, where the best parameter combination is in boldface. As expected, the higher frequency and the maximum number of LS1 iterations shows the better performance for almost all problem dimensions. Only in the case of $D = 50$, a different combination performed better with $ls\_freq = 30$ and $max\_ls\_iters = 50$. We suspect that more frequent and large LS1 procedures in PSO6-Mtsls could be costly when $D = 50$, that is, the shorter scale in SOCO'10, for which a lower number of function evaluations are allowed. Nevertheless, for the sake of a homogeneous parameter setting, we have decided to use always the best combination for almost all problem dimensions: $ls\_freq = 5$ and $max\_ls\_iters = 70$. In fact, this combination is close to the ones used in related works in the literature [17], [13], [29]. Parameters of all other compared algorithms can be found in their reference works.

VI. PERFORMANCE COMPARISONS

This section is devoted to show all the performance results of our PSO6-Mtsls. A series of comparisons with other PSO versions, as well as with other modern proposals in the current state of the art are carried out from different points of view. Our goal is to solve different problems as well as highlighting its advantages.

A. Comparison of PSO versions

Figures 4 and 5 show the boxplots representing que distributions of error fitness obtained by Standard PSO 2011 (S2011), Fully Informed PSO with complete neighborhood (FIPS-ALL), FIPS with Squared Neighborhood (FIPS4), PSO6, and our proposal here PSO6-Mtsls, for the 19 functions of SOCO'10 benchmark. Table V shows the average ranking of compared algorithms resulted from the Friedman's statistical test and applying a post-hoc Holm's correction for multiple comparisons ($\alpha = 0.05$), for problem dimensions 50 and 100. In this table, the algorithm with the best ranking is used as control method (marked in boldface). Thus, those algorithms with adjusted Holms's *p-values* < 0.05 are statistically outperformed by the control method.

In general, we can observe in Figs. 4 and 5 that PSO6-Mtsls shows the best performance in (almost) all functions, and for the two analyzed problem dimensions. As shown in Table V, our proposal is the best ranked algorithm and therefore, it is set as control method for the post-hoc Holm's test. For dimension 50, all compared PSO versions excepting PSO6 are statistically outperformed by PSO6-Mtsls. We have to notice that LS1 parameters were set using a homogeneous tuning for all dimensions, although being slightly disadvantageous in the particular case of dimension 50 (Section V-B). We suspect that using specific parameter setting for this dimension could lead our PSO6-Mtsls to be statistically better than PSO6. In

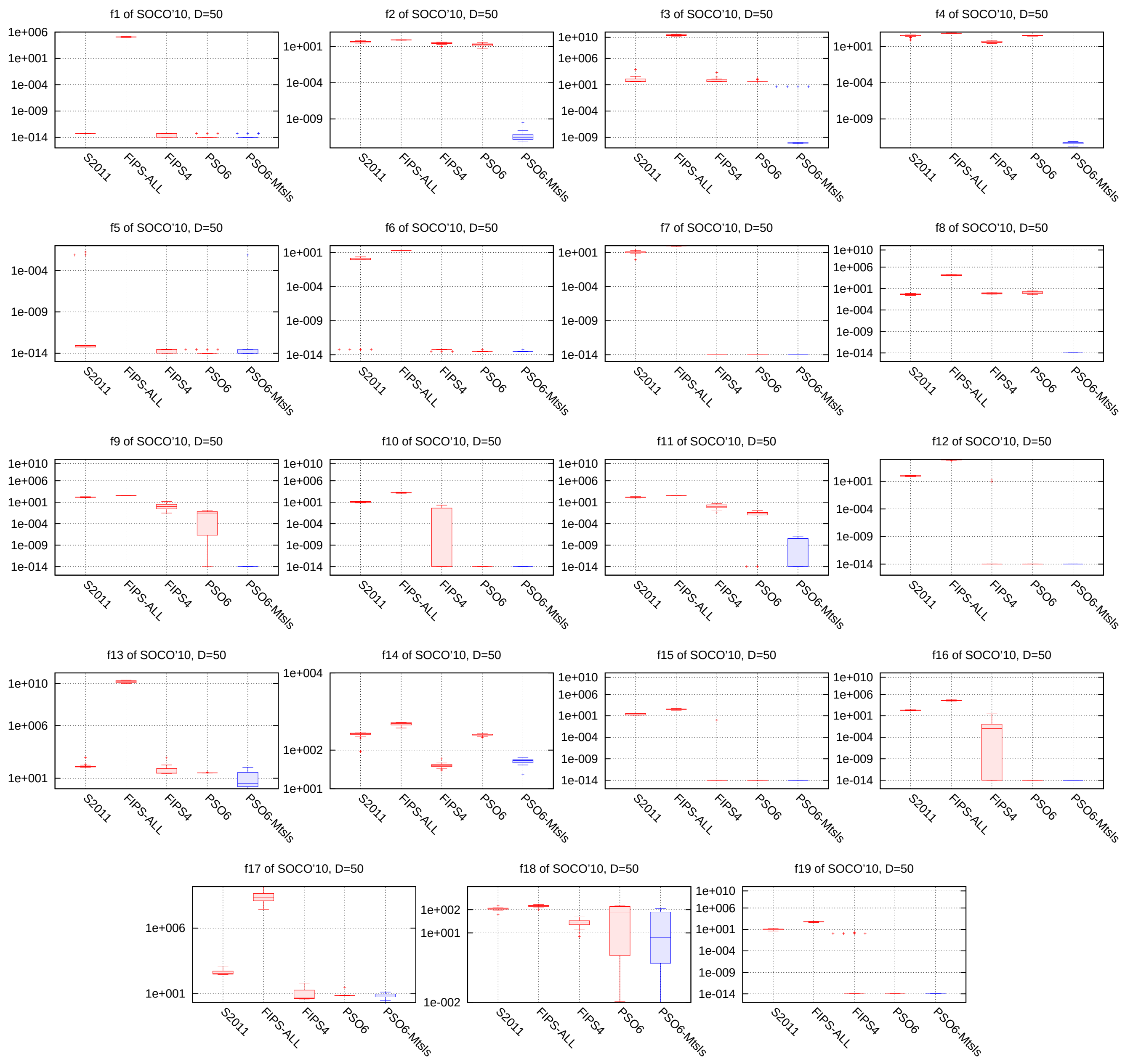

Fig. 4. SOCO'10 function's fitness distributions of S2011, FIPS-ALL, FIPS4, PSO6, and PSO6-Mtsls, for dimension 50

fact, in the case of dimension 100, we can effectively observe that our proposal outperforms all other compared algorithms, including PSO6, with statistical confidence.

If we examine non composed functions (*soco*1 to *soco*11) in Figs. 4 and 5, we can clearly observe that PSO6-Mtsls always shows the best results, followed by PSO6, FIPS4, S2011, and FIPS-ALL. Nevertheless, there are several functions: *soco*1, *soco*6, and *soco*7, for which PSO6-Mtsls obtained similar distributions to the ones of PSO6, and FIPS4. Not surprisingly, these functions are characterized as separable in SOCO'10, and hence, PSO6 and FIPS4 are also able to show accurate performances with regards to our proposal. Therefore, the possible benefits induced by the local search method could be said to pay in non-separable functions.

TABLE V
AVERAGE FRIEDMAN'S RANKINGS WITH HOLM'S CORRECTION ($\alpha = 0.05$) FOR SOCO'10 FUNCTIONS WITH DIMENSIONS 50 AND 100

	50		100	
Algorithm	Rank	$Holm's_{p-value}$	Rank	$Holm's_{p-value}$
PSO6-Mtsls	**1.367**	-	**1.152**	-
PSO6	2.149	0.16E-01	2.331	4.36E-02
FIPS4	2.915	0.44E-02	2.952	2.05E-02
S2011	3.684	2.34E-04	3.763	6.55E-05
FIPS-ALL	5.000	2.90E-10	5.000	7.26E-11

Concerning composed functions (*soco*12 to *soco*19), the error distributions obtained by PSO6-Mtsls are in general better than the ones of compared PSO versions. Therefore, we can claim that the use of local search (LS1) in PSO6

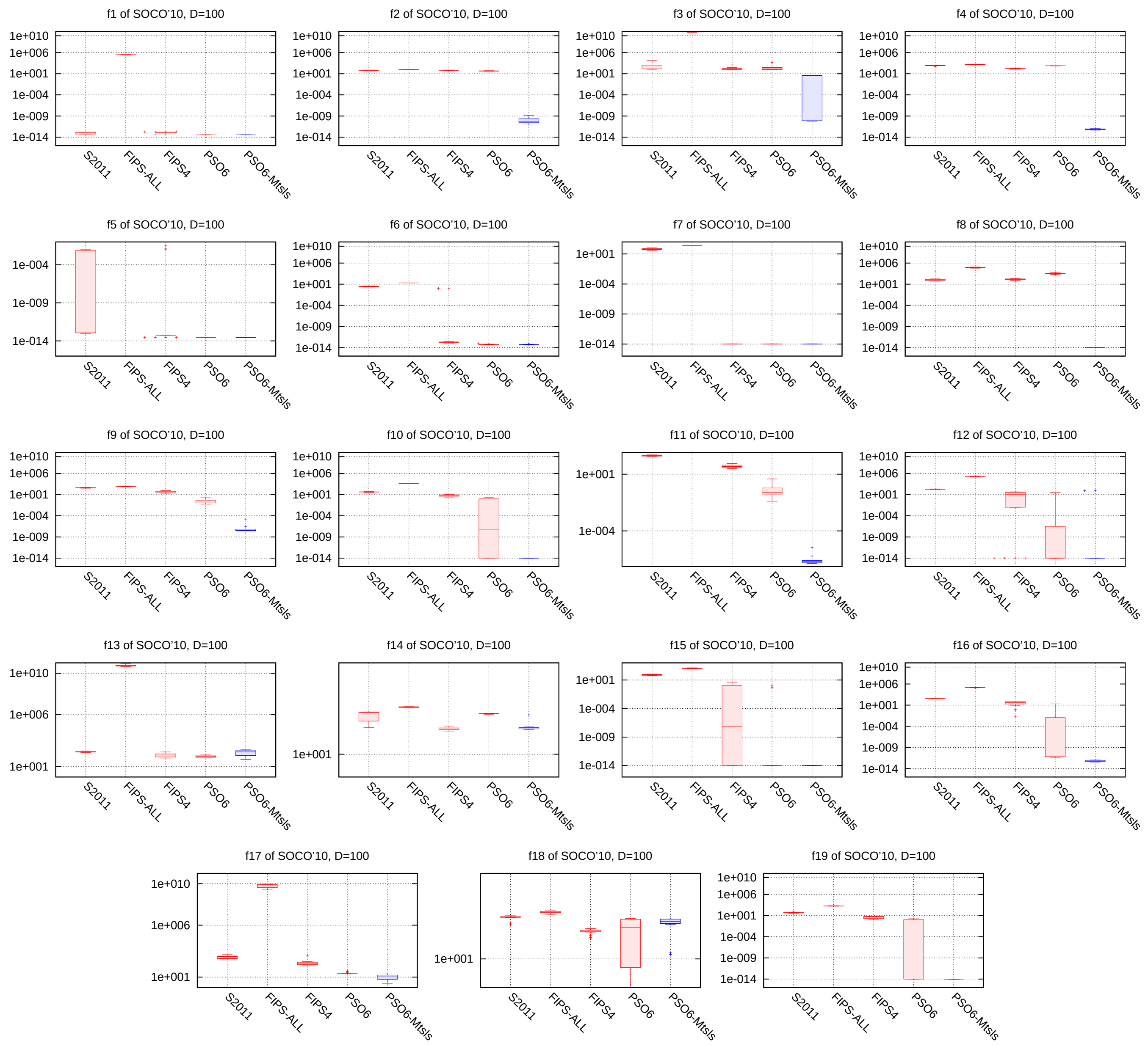

Fig. 5. SOCO'10 function's fitness distributions of S2011, FIPS-ALL, FIPS4, PSO6, and PSO6-Mtsls, for dimension 100

is advantageous in the context of SOCO'10 benchmark of functions. However, a single exception can be observed for function *soco*18 with dimension 100, where the high proportion of *soco*4 (separable) variables in the composition with *soco*9 is the probable reason for PSO6 to show a better error distribution, even without any local search procedure. In this sense, a secondary observation is that PSO6 generally shows a better performance than FIPS4 (the best algorithm in [14]), and is statistically better than FIPS-ALL and S2011. This result was also founded in our previous work [6], although in the context of CEC'05 benchmark of functions, with dimension 30. Therefore, in the context of large scale SOCO'10 functions with more than 50 dimensions, we can also claim that PSO6 is able to perform an optimized learning procedure.

B. *Comparisons with other algorithms in the state of the art*

Fig. 6 shows the boxplots representing the median error distributions of the 19 SOCO functions obtained with PSO6, PSO6-Mtsls, and the algorithms [2] featured in the special issue of SOCO'10, for dimensions 50, 100, 200, and 500. From these last algorithms, the results of DE, CHC, and G-CMA-ES were provided as base-reference techniques to compare with, previous to the global comparisons. In relation with this figure, Table VI contains the results of applying the Friedman's test and Holm's corrections to the aforementioned distributions of median errors, for all compared algorithms and dimensions.

In general, our PSO6-Mtsls is statistically better than PSO6, and shows more accurate distributions than RPSO-

[2]The complete information about featured algorithms in SOCO'10 is available in `http://sci2s.ugr.es/EAMHCO/`

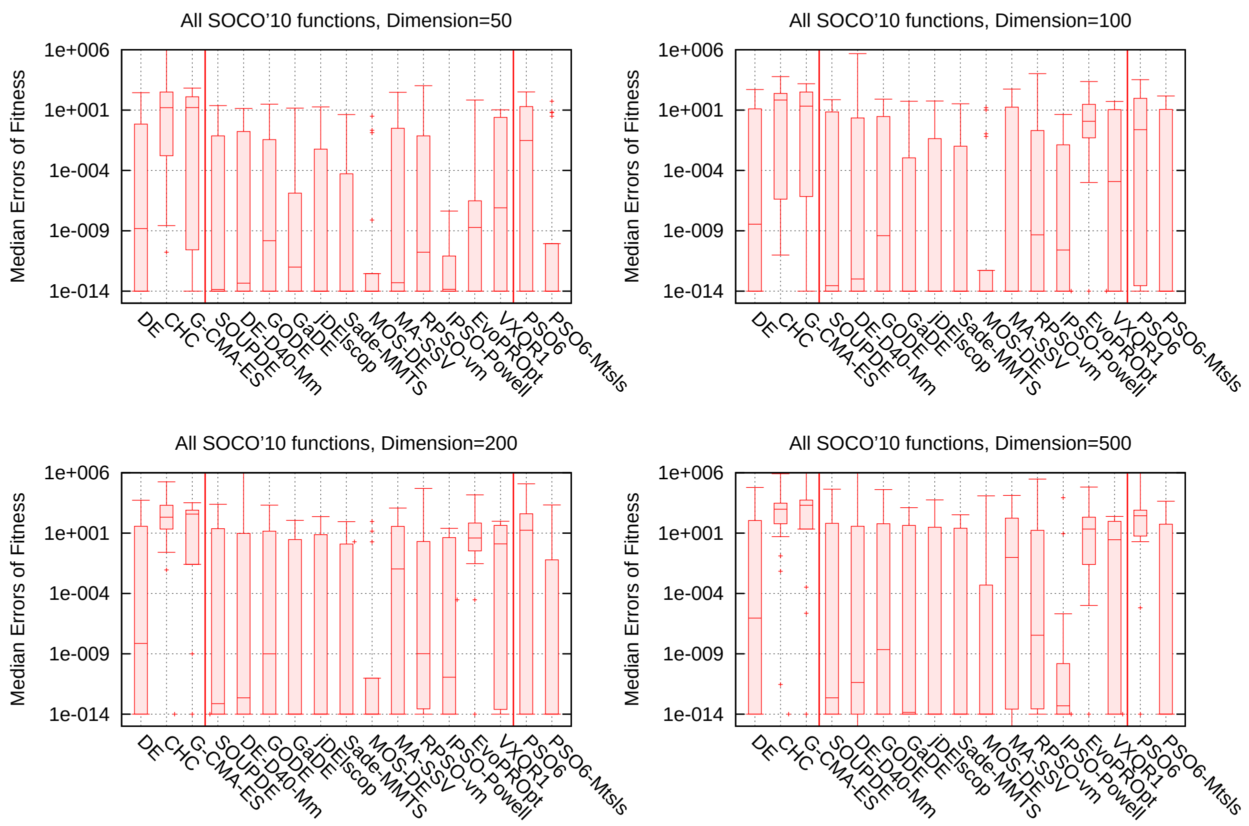

Fig. 6. SOCO'10 function's fitness distributions of all featured algorithms and PSO6 with and without Mtsls, for dimensions 50, 100, 200, and 500

TABLE VI
AVERAGE FRIEDMAN'S RANKINGS WITH HOLM'S CORRECTION ($\alpha = 0.05$) FOR SOCO'10 FUNCTIONS AND FEATURED ALGORITHMS

	50		100		200		500	
Algorithm	Rank	$Holm'_{sp}$	Rank	$Holm'_{sp}$	Rank	$Holm'_{sp}$	Rank	$Holm'_{sp}$
PSO6-Mtsls	5.894	0.16E+01	7.929	1.24E-01	6.952	2.40E-1	7.236	2.14E-01
PSO6	10.263	0.13E-02	11.763	7.12E-05	12.315	4.61E-06	12.947	8.40E-08
SOUPDE	7.578	0.62E+00	6.763	5.40E-01	7.553	1.44E-01	7.631	1.41E-01
DE-D40-Mm	9.342	5.53E-01	8.710	5.85E-02	8.578	3.90E-02	8.657	3.45E-02
GODE	8.657	0.15E+00	7.894	1.44E-01	7.973	1.02E-01	7.921	1.02E-01
GaDE	6.868	0.11E+01	5.631	1.26E+00	5.631	6.23E-01	5.552	6.38E-01
jDElscop	5.868	0.16E+01	5.315	1.26E+00	5.315	6.23E-01	5.052	6.38E-01
Sade-MMTS	6.263	0.16E+01	5.631	1.26E+00	6.157	5.47E-01	6.368	5.40E-01
MOS-DE	**4.921**	-	**4.315**	-	**3.973**	-	**3.921**	-
MA-SSV	11.736	4.45E-04	10.421	2.33E-03	9.131	1.64E-02	11.578	3.54E-05
RPSO-vm	9.552	0.52E-02	9.631	1.17E-02	9.210	1.29E-01	8.236	6.74E-02
IPSO-Powell	6.078	0.16E+01	8.210	1.22E-01	7.736	1.53E-02	6.173	6.30E-01
EvoPROpt	10.184	1.43E-01	12.421	1.05E-05	13.026	4.60E-07	12.526	1.95E-05
VXQR1	10.447	0.96E-03	9.868	7.71E-03	10.815	3.55E-04	10.000	2.27E-03
DE	10.026	0.18E-02	9.210	2.53E-02	9.131	1.64E-02	8.815	2.81E-02
CHC	16.157	1.11E-11	15.736	5.03E-11	16.052	2.67E-12	16.105	1.64E-12
G-CMA-ES	13.157	7.45E-06	13.368	4.93E-07	13.342	1.61E-07	13.473	7.73E-08

vm and IPSO-Powell, the two other PSO versions evaluated in SOCO'10. An exception can be found for dimension 500 where IPSO-Powell also shows an accurate distribution, although with worse median value than our proposal. Concerning the remaining algorithms, PSO6-Mtsls significantly outperforms G-CMA-ES, CHC, DE, VXQR1, EvoPROpt, and MA-SSW. In 200 and 500 dimensions, our proposal also outperforms GODE and DE-D40-Mm. Nevertheless, the best ranked distributions correspond to the ones of MOS-DE, which is established as control method in Table VI. In spite of this, we can effectively check that adjusted p-values of PSO6-Mtsls are always higher than 0.05 (confidence level), which leads us to ensure that no statistical differences can be found between our proposal and MOS-DE. The remaining techniques featured in SOCO'10: IPSO-Powell, Sade-MMTS, jDElscop, GaDE, and SOUPDE are also in the group of similar algorithms (without statistical differences in their performance) with regards to our proposal and the control algorithm (MOS-DE).

From the point of view of the problem scalability, we can observe in Fig. 6 that the performance of PSO6-Mtsls keeps competitive results even in dimension 500 with regards to the group of best compared algorithms. In fact, we have to notice that the median errors of PSO6-Mtsls are below the 0-threshold at least for 12 functions out of 19 (SOCO'10). Therefore, as shown in boxplots (Fig. 6), our proposal resulted with a global median of 1.00E-14 for all dimensions. In this way, we can claim that our approach is also competitive as the problem dimensionality increases.

An interesting observation in this comparison concerns the performance of G-CMA-ES, which is relatively limited on SOCO'10 functions. Although this algorithm shows accurate results on non-separable functions $soco3$, $soco5$, and $soco8$, it has a moderate performance on separable unimodal and multimodal ones, such as $soco4$, $soco6$, $soco7$, as well as on non-separable hybrid composed (from $soco12$ to $soco19$). Taking into account that G-CMA-ES obtained the best results in the special session of CEC'05, we suspect that the existence (or not) of rotated functions, on which this algorithm shows highly accurate results, could influence its global performance with regards to other compared algorithms. We have to notice that a number of rotated functions (21 out of 25) are included in CEC'05, whereas practically none of them can be found in SOCO'10. A similar observation was made in [13], where the authors argued that the global performance of a given algorithm can be biased to certain function feature more expressed in the tackled benchmark. This motivated us to use an extended benchmark composed of 40 problem functions from CEC'05 and SOCO'10 (previously described in Table I) to compare our proposal with G-CMA-ES, as well as with other related swarm intelligent approaches with local search methods. The results of this comparison are analyzed in the following section.

C. Comparisons on an extended benchmark

Table VII contains the median of distribution errors obtained by G-CMA-ES [2], IPSO-Powell [16], IPSO-Mtsls [17], IACOr-Mtsls [13], and PSO6-Mtsls, out of 30 independent runs on the extended benchmark of 40 functions taken from SOCO'10 and CEC'05, for dimension 50. IPSO-Powell and IPSO-Mtsls are PSO versions that perform an incremental social learning mechanism for swarm size adaptation on continuous optimization functions. These two IPSO algorithms are hybridized with Powell's direction set [20] and Mtsls (LS1) [28] local search procedures, respectively. IACOr-Mtsls consists of an Ant Colony Optimization algorithm also performing an incremental social learning mechanism as in the previous PSO versions, and also hybridized with Mtsls (LS1). In this way, we can compare our proposal with modern swarm intelligent approaches hybridized with the same and different local search methods. G-CMA-ES is a covariance matrix adaptation evolution strategy that performs frequent restarts with increasing population size. As commented before, we also compare our PSO6-Mtsls with G-CMA-ES on CEC'05 non-separable/rotated functions, where this last algorithm shows an impressive performance. In this table, the best median values for each function are represented in boldface, and the last row contains the global count of the number of best medians for each compared algorithm.

TABLE VII
MEDIAN ERRORS OBTAINED BY G-CMA-ES, IPSO-POWELL, IPSO-MTSLS, IACOR-MTSLS, AND PSO6-MTSLS, FOR DIMENSION 50

F/A	G-CMA-ES	IPSO-Powell	IPSO-Mtsls	IACOr-Mtsls	PSO6-Mtsls
fsoco1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco2	2.64E-11	**1.42E-14**	4.12E-13	4.41E-13	2.96E-12
fsoco3	**0.00E+00**	**0.00E+00**	6.38E+00	4.83E+01	8.47E-11
fsoco4	1.08E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco6	2.11E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco7	7.67E-11	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco8	**0.00E+00**	1.75E-09	2.80E-10	2.66E-05	**0.00E+00**
fsoco9	1.61E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco10	6.71E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco11	2.83E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco12	1.87E+02	1.02E-12	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco13	1.97E+02	**2.00E-10**	5.39E-01	6.79E-01	3.09E+00
fsoco14	1.05E+02	1.77E-12	**0.00E+00**	**0.00E+00**	5.37E+01
fsoco15	8.12E-04	1.07E-11	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco16	4.22E+02	3.08E-12	**0.00E+00**	**0.00E+00**	**0.00E+00**
fsoco17	6.71E+02	**4.35E-08**	1.47E+01	6.50E+00	6.68E+00
fsoco18	1.27E+02	8.06E-12	**0.00E+00**	**0.00E+00**	6.21E+00
fsoco19	4.03E+00	1.83E-12	**0.00E+00**	**0.00E+00**	**0.00E+00**
fcec3	**0.00E+00**	8.72E+03	1.59E+04	8.40E+05	1.73E+02
fcec4	4.27E+05	2.45E+02	3.88E+03	**5.93E+01**	2.08E+02
fcec5	5.70E-01	4.87E-07	**7.28E-11**	9.44E+00	2.98E+03
fcec7	3.85E-14	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
fcec8	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**
fcec10	**9.97E-01**	8.96E+02	8.92E+02	2.69E+02	1.84E+02
fcec11	**1.21E+00**	6.90E+01	6.64E+01	5.97E+01	3.38E+01
fcec12	2.36E+03	5.19E+04	3.68E+04	1.37E+04	**6.97E+02**
fcec13	4.71E+00	3.02E+00	3.24E+00	**2.14E+00**	6.70E+00
fcec14	2.30E+01	2.35E+01	2.36E+01	2.33E+01	**2.27E+01**
fcec15	2.00E+02	2.00E+02	2.00E+02	**0.00E+00**	3.06E+02
fcec16	**2.15E+01**	4.97E+02	4.10E+02	3.00E+02	1.95E+02
fcec17	**1.61E+02**	4.54E+02	4.11E+02	4.37E+02	2.20E+02
fcec18	9.13E+02	1.22E+03	1.21E+03	9.84E+02	**8.25E+02**
fcec19	9.12E+02	1.23E+03	1.19E+03	9.93E+02	**8.25E+02**
fcec20	9.12E+02	1.22E+03	1.19E+03	9.93E+02	**8.25E+02**
fcec21	1.00E+03	1.19E+03	1.03E+03	**5.00E+02**	7.18E+02
fcec22	8.03E+02	1.43E+03	1.45E+03	1.13E+03	**5.00E+02**
fcec23	1.01E+03	**5.39E+02**	**5.39E+02**	**5.39E+02**	7.24E+02
fcec24	9.86E+02	1.31E+03	1.30E+03	1.11E+03	**2.17E+02**
fcec25	**2.15E+02**	1.50E+03	1.59E+03	9.38E+02	**2.15E+02**
#bests	11/40	14/40	18/40	21/40	**23/40**

TABLE VIII
AVERAGE FRIEDMAN'S RANKINGS WITH HOLM'S CORRECTION ($\alpha = 0.05$) FOR SOCO'10 AND CEC'05 FUNCTIONS

Algorithm	Rank	$Holm's_p$
PSO6-Mtsls	**2.48**	-
IACOr-Mtsls	2.68	5.71E-01
IPSO-Mtsls	3.08	1.79E-01
IPSO-Powell	3.35	4.41E-02
G-CMA-ES	3.38	4.36E-02

A first observation in Table VII is that PSO6-Mtsls obtains the highest number of best median errors (23 out of 40), followed by IACOr-Mtsls, IPSO-Mtsls, IPSO-Powell, and finally G-CMA-ES. The statistical tests associated to these results are presented in Table VIII, where we can effectively validate that our proposal is the best ranked algorithm, then working in this case as control method for the post-hoc Holm's correction. According to this, we can even ensure that IPSO-Powell and G-CMA-ES are statistically outperformed by our PSO6-Mtsls.

In this sense, a second observation is that our approach does not show statistical differences with regards to the other two swarm intelligent methods hybridized with Mtsls, that is, IACOr-Mtsls and IPSO-Mtsls. This led us to suspect that Mtsls is largely responsible of the accurate performance of these three approaches, in comparison with IPSO-Powell and G-CMA-ES. However, the difference in their ranking values

TABLE IX
NUMBER OF BEST MEDIAN ERRORS WITH REGARDS TO DIFFERENT FUNCTIONS FEATURES WHEN COMPARING PSO6-MTSLS VERSUS G-CMA-ES, IPSO-POWELL, IPSO-MTSLS, AND IACOR-MTSLS. THE RESULTS ARE PRESENTED IN FORM OF (WIN, DRAW, LOSE)

Function's	PSO6-Mtsls versus			
features	G-CMA-ES	IPSO-Powell	IPSO-Mtsls	ACOr-Mtsls
Separable	(6, 1, 0)	(0, 4, 0)	(0, 3, 0)	(0, 4, 0)
Non sep.	(12, 4, 6)	(13, 9, 2)	(9, 12, 3)	(9, 12, 5)
Unimodal	(1, 2, 1)	(1, 5, 1)	(1, 5, 1)	(1, 5, 1)
Multimodal	(17, 3, 5)	(12, 8, 1)	(8, 10, 2)	(8, 11, 4)
Rotated	(7, 2, 5)	(7, 3, 0)	(7, 3, 0)	(7, 4, 0)
Non rot.	(10, 3, 1)	(6, 10, 2)	(2, 12, 3)	(2, 12, 5)
SOCO'10	(10, 3, 1)	(5, 10, 2)	(1, 12, 3)	(1, 12, 2)
CEC'05	(8, 2, 5)	(8, 3, 0)	(8, 3, 1)	(8, 4, 3)
Total	(18, 5, 6)	(13, 13, 2)	(9, 15, 3)	(9, 16, 5)

seems to be mostly due to the contribution of their base methods: PSO6, IPSO, and IACO.

A final interesting observation concerns the different function features that our proposal can successfully tackle with regards to the four compared techniques. Table IX shows a detailed comparison presented in form of (win, draw, lose) according to different features of the extended benchmark of 40 functions. In comparison with G-CMA-ES, our approach obtained a higher number of "wins" (better medians) on non-separable, multimodal, and rotated functions (as well as in non-separable and non-rotated). We have to notice that rotated functions correspond to CEC'05 benchmark on which G-CMA-ES was the best algorithm. In fact, our PSO6-Mtsls obtained 8 "wins" in CEC'05 and 10 in SOCO'10, in contrast with 5 and 1 "loses" in these two benchmarks with regards to G-CMA-ES. If we have a look on non-separable functions, our proposal obtains a higher number of "wins" than G-CMA-ES and IPSO-Powell, although the number of "draws" is higher in comparison with IACOr-Mtsls and IPSO-Mtsls. Once again, the effect that Mtsls induces on non-separable functions leads hybridized algorithms with this local search method to outperform other compared techniques. A similar behavior can be observed concerning multimodal functions with a high number of "draws" when comparing hybridized algorithms with Mtsls. Nevertheless, it is on rotated functions where PSO6-Mtsls shows a higher number of "wins" in comparison with all other algorithms. In this case, the base method PSO6 is responsible of the accurate performance, since it never obtained "loses" on rotated functions, excepting 5 in comparison with G-CMA-ES.

In summary, the local search method Mtsls (LS1) seems to be responsible of the successful performance of our proposal on non-separable and multimodal functions, whereas the learning procedure of PSO6 takes mostly part in rotated ones. PSO6-Mtsls shows more "wins" than "loses" in all comparisons, although the number of "draws" is higher when it is compared with IACOr-Mtsls and IPSO-Mtsls, e. g., the other swarm intelligent approaches hybridized with Mtsls.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we incorporate a local search procedure (Mtsls) to our PSO6 with the aim of improving the performance of this algorithm mainly in non-separable and rotated continuous optimization functions. Our proposal, called PSO6-Mtsls, is then empirically tested and compared on a set of 40 benchmark functions with that of other PSO versions, as well as with that of other recent proposals in the current state of the art (with and without local search methods). From this experimentation, the following conclusions can be extracted:

1) Comparisons of our proposal against other PSO base methods: S2011, FIPS-ALL, FIPS4, and PSO6, lead us to claim that the use of the local search method Mtsls (LS1) in combination with the learning procedure performed by PSO6 is advantageous in the context of SOCO'10 benchmark. In addition, PSO6 generally shows better performance than FIPS4 (the best algorithm in [14]), and is statistically better than S2011 and FIPS-ALL for this benchmark of functions and for different problem dimensions.
2) With regards to the current state of the art, PSO6-Mtsls statistically outperforms G-CMA-ES, CHC, DE, VXQR1, EvoPROpt, and MA-SSW. In 200 and 500 dimensions, our proposal also outperforms GODE and DE-D40-Mm. In comparison with MOS-DE, the best ranked algorithm in SOCO'10, PSO6-Mtsls always show similar distributions to this approach for all studied dimensions.
3) The median error of PSO6-Mtsls is below the 0-threshold at least for 12 functions out of 19 (SOCO'10). Therefore, as shown in boxplots (Fig. 6), our proposal resulted with a global median of 1.00E-14 for all dimensions. In this way, we can claim that our approach is also scalable.
4) In the scope of the extended benchmark with 40 functions used here, we can ensure that PSO6-Mtsls statistically outperforms IPSO-Powell and G-CMA-ES, and is better ranked than IACOr-Mtsls and IPSO-Mtsls. The local search method Mtsls (LS1) seems to be responsible of the successful performance on non-separable and multimodal functions, whereas the learning procedure of PSO6 takes mostly part in rotated ones.

We here state that PSO is a first class optimizer able of the best performance in present benchmarking for the continuous optimization.

As future work, we are interested in investigating other elemental features and learning procedures [12] of the PSO algorithm, as well as to study other complementary methods to construct satisfactory hybrid approaches, capable of solving highly complex functions. Besides, we plan to perform analytical investigations on new benchmarks (BBOB, CEC'11, etc.) with different function characteristics and dimensions.

ACKNOWLEDGMENTS

Authors acknowledge funds from the CICE of the Junta de Andalucia, under contract P07-TIC-03044 (DIRICOM `http://diricom.lcc.uma.es`) and Spanish Ministry of Sciences and Innovation (MEC) and FEDER under contracts TIN2011-28194 (RoadMe `http://roadme.lcc.uma.es`). José García-Nieto is supported by grant BES-2009-018767 from MEC.

References

[1] E. Alba, G. Luque, J. García-Nieto, G Ordoñez, and G Leguizamón. MALLBA: A software library to design efficient optimisation algorithms. *Int. Journal of Innovative Computing and Applications (IJICA)*, 1(1):74–85, 2007.

[2] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. *IEEE Congress on Evolutionary Computation*, 2:1769–1776, 2005.

[3] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58 – 73, Feb 2002.

[4] R. Eberhart and Y. Shi. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation CEC'00*, volume 1, pages 84–88, La Jolla, CA, USA, 2000.

[5] S. García, D. Molina, M. Lozano, and F. Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005. *Journal of Heuristics*, 15(6):617–644, 2009.

[6] J. García-Nieto and E. Alba. Empirical computation of the quasi-optimal number of informants in particle swarm optimization. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 147–154, New York, NY, USA, 2011. ACM.

[7] J. García-Nieto and E. Alba. Why six informants is optimal in pso. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, pages 25–32, New York, NY, USA, 2012. ACM.

[8] F. Herrera and M. Lozano. Workshop for evolutionary algorithms and other metaheuristics for continuous optimization problems - a scalability test. Technical report, SCI2S, University of Granada, Pisa, Italy, December 2019.

[9] F. Herrera, M. Lozano, and D. Molina. 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, May 2010.

[10] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, California, 2001.

[11] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proceedings of the Congress of Evolutionary Computation CEC'02*, volume 2, pages 1671–1676, Washington, DC, USA, 2002. IEEE Computer Society.

[12] C. Li, S. Yang, and T. T. Nguyen. A self-learning particle swarm optimizer for global optimization problems. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42(3):627 –646, june 2012.

[13] T. Liao, M. A. Montes de Oca, D. Aydin, T. Stützle, and M. Dorigo. An incremental ant colony algorithm with local search for continuous optimization. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 125–132, New York, NY, USA, 2011. ACM.

[14] R. Mendes, J. Kennedy, and J. Neves. The Fully Informed Particle Swarm: Simpler, Maybe Better. *IEEE Transactions on Evolutionary Computation*, 8(3):204 – 210, June 2004.

[15] A. S. Mohais, R. Mendes, C. Ward, and C. Posthoff. Neighborhood restructuring in particle swarm optimization. In *LNCS 3809. Proceedings of the 18th Australian Joint Conference on Artificial Intelligence*, pages 776–785. Springer, 2005.

[16] M. A. Montes de Oca, D. Aydin, and T. Stützle. An incremental particle swarm for large-scale continuous optimization problems: an example of tuning-in-the-loop (re)design of optimization algorithms. *Soft Computing*, 15:2233–2255, 2011.

[17] M. A. Montes de Oca, T. Stützle, K. Van den Enden, and M. Dorigo. Incremental social learning in particle swarms. *Trans. Sys. Man Cyber. Part B*, 41(2):368–384, April 2011.

[18] S. Muelas, A. La Torre, and J. Peña. A memetic differential evolution algorithm for continuous optimization. In *Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications*, ISDA '09, pages 1080–1084, Washington, DC, USA, 2009. IEEE Computer Society.

[19] S. Muelas, J. Peña, A. La Torre, and V. Robles. A new initialization procedure for the distributed estimation of distribution algorithms. *Soft Computing*, 15(4):713–720, April 2010.

[20] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964.

[21] PSO-Central-Group. Standard PSO 2006, 2007, and 2011. Technical Report [online] http://www.particleswarm.info/, Particle Swarm Central, January 2011.

[22] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, 2007.

[23] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. 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, 2005.

[24] A. M. Sutton, D. Whitley, M. Lunacek, and A. Howe. Pso and multi-funnel landscapes: how cooperation might limit exploration. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO '06, pages 75–82, New York, NY, USA, 2006. ACM.

[25] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang. 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, November November 2007.

[26] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.

[27] I. C. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Inf. Process. Lett.*, 85:317–325, March 2003.

[28] L. Tseng and Chun C. Multiple trajectory search for Large Scale Global Optimization. In *IEEE Congress on Evolutionary Computation*, pages 3052–3059, 2008.

[29] L. Tseng and C. Chen. Multiple trajectory search for unconstrained/constrained multi-objective optimization. In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, CEC'09, pages 1951–1958, Piscataway, NJ, USA, 2009. IEEE Press.

PLACE
PHOTO
HERE

José García-Nieto Biography text here.

PLACE
PHOTO
HERE

Enrique Alba Biography text here.