**PSO2: Particle Swarm Optimization with PSO-Based Local Search**

Mohamed Khairy, Magda B. Fayek, Elsayed E. Hemayed  
Computer Engineering Dept., Faculty of Engineering,  
Cairo University  
Egypt  
khairy.mohamed.a@gmail.com, magdafayek@ieee.org, hemayed@ieee.org

**Abstract**  
Several attempts have been made to enhance PSO performance by combining it with a local search method. Following the same track, we present in this paper local search in PSO performed by smaller independent swarms of PSO producing PSO2. Different modifications are made to help basic PSO2 enhance performance. PSO2-RS and PSO2-SA are 2 modified versions of PSO2 that targeted to increase the swarm diversity. Increasing the local search swarms sizes as the search progresses is another modification made to basic PSO2 in order to change the algorithm behavior to be more exploitive. The final algorithm is examined against 4 functions of the CEC-2005 benchmark suite and results are reported.

**Keywords-component:** Optimization; Swarm Intelligence, Particle Swarm Optimization; Local Search; Hybridization; PSO2.

I. INTRODUCTION

Particle Swarm Optimization (PSO) is a stochastic population-based search algorithm that was first introduced by Kennedy and Eberhart in 1995 [1]. PSO is inspired by the social interactions of birds in a flock.

The algorithm consists of simple entities, called particles that are placed in the search space. Each particle determines its next movement by combining some aspects of its own history and its neighborhood history with some random perturbations. The next iteration takes place after all particles move to their next positions and the swarm as a whole moves closer to a potential optimum.

Each particle is represented by its position in the search space as \( X_i^d = (x_{i1}^d, x_{i2}^d, \ldots, x_{iD}^d) \) and the rate of position change (Velocity) is given by \( V_i^d = (v_{i1}^d, v_{i2}^d, \ldots, v_{iD}^d) \) where \( i \) is the particle index and \( D \) is the number of dimensions. The particles are manipulated according to the following equations:

\[
\begin{align*}
    v_{id}^{t}(t) &= v_{id}^{t}(t-1) + c_1 \cdot \text{rand1}_i \cdot (pbest_{id}^{t} - x_{id}^{t}(t-1)) + c_2 \cdot \text{rand2}_i \cdot (gbest_{id}^{t} - x_{id}^{t}(t-1)) \\
    x_{id}^{t}(t) &= x_{id}^{t}(t-1) + v_{id}^{t}(t)
\end{align*}
\]

where \( c_1 \) and \( c_2 \) are the acceleration constants, \( \text{rand1}_i \) and \( \text{rand2}_i \) are two uniformly distributed random numbers in the range \([0, 1]\). \( gbest_i^d \) and \( pbest_i^d \) are the global and personal best positions respectively. Three components affect the particle’s velocity. The first component is the inertia. The second component is the cognitive component, which is represented by the second part in (1). Cognitive component represents the private thinking of the particle. The third component is the social part, third part in (1), and it represents the social collaboration between particles.

Since it was introduced, the algorithm has undergone several modifications that targeted to damp the velocity and prevent explosion. Inertia weight was one of the first of these modifications introduced by Shi and Eberhart [2]. Clerc analysis of a simplified PSO algorithm in [3] led him to propose a well explained method for ensuring convergence and eliminating the need for Vmax parameter by the introduction of constriction coefficients.

Other techniques were proposed e.g. Fully Informed Particle Swarm (FIPS) [4], Guaranteed Convergence PSO (GCPSO) [5], Bare Pones [11], sub-swarms [12], dynamic topologies [6], TRIBES [13] and others to enhance canonical PSO algorithm. In FIPS Mendes proposed to have the particle affected by its entire neighborhood instead of only the best particle in canonical PSO. GCPSO prevents stagnation by forcing the best particle to move once stagnation is detected. Dynamic topologies represent another direction that seems also to be promising.

Another trend of PSO research was to hybridize PSO with a local search method to enhance its local search ability when approaching potential optima. The search methods used with PSO in literature range from stochastic methods e.g. GA, simulated annealing, hill climbing and the Harmony search (HS) [7] [10] to exact methods e.g. Sequential Quadratic Programming (SQP) and Quasi-Newton [6]. In this paper, we propose PSO with local search that uses smaller independent swarms of PSO.

Similar to the work presented in this paper, Clerc in [9] argued the claims raised by many PSO researchers to balance between exploration and exploitation. Clerc proved that the good balance often claimed for PSO never happens merely because exploitation for problems with dimension more than 6 does not exist. In his work, he achieved exploitation by forcing some particles from the existing swarm to move towards more promising areas. Different from Clerc, in the work presented in
this paper, we achieve exploitation using smaller swarms that are independent from the main swarm.

This paper is organized as follows. In section II, a brief overview of PSO2 is given. In section III, the proposed algorithm and different modifications to it are introduced, benchmark test functions, and experimental setup and evaluation criteria are stated. Section IV summarizes the results and discusses the findings of the proposed technique. In section V, a conclusion and aspects for future work are stated.

II. PSO2: THE PROPOSED ALGORITHM

A. Basic Particle Swarm Optimization with PSO-Based Local Search (PSO2)

Like canonical PSO, PSO2 particles fly through a D dimensional space by learning from the historical information of the particle itself and neighbor particles. In PSO2, after the completion of each iteration, a number of particles (best performing) in the swarm are selected for applying local search. Each of the selected particles sets a small perimeter around its current position and starts applying the local search algorithm. The local search algorithm is performed by placing small independent swarms around the selected particles within the specified range. The small swarms are initialized and run a small number of iterations.

Once the local search is ended, the selected particle compares its pbest with its local search swarm global best gbest. If the pbest is larger (assuming a minimization problem) the local search gbest becomes the particle pbest. This process is performed for each of the selected particles. The algorithm then continues as the canonical PSO algorithm, and each particle updates its velocity and position. The algorithm steps are illustrated in figure 1. The dotted arrow in figure 1 shows the original transition taken by canonical PSO. While the new box inserted into the flowchart shows the added step of PSO based local search.

B. PSO2-RS and PSO2-SA

Two modifications are made to PSO2. In PSO2-RS, we select random particles for local search instead of selecting top performing particles in basic PSO2. The purpose of this modification is to increase the diversity in the swarm by driving the search away from currently best particles to randomly chosen particles. In PSO2-SA, we incorporate Simulated Annealing (SA) in PSO2 producing PSO2-SA. PSO2 originally accepts the current iteration best solution as global best, gbest, solution if and only if it is better (less in minimization problems) than the current global best. What SA adds to the algorithm is that it forces PSO2 to accept new global best solutions even if it is worse than the current global solution. Accepting less value solutions happens with a probability that depends on simulated annealing temperature. Temperature is a simulated annealing parameter that has a relatively high initial value and decreases along the run. The probability also depends on the cooling schedule which determines the rate by which the temperature decreases.

C. PSO2-SA-DYSS

The third update to PSO2 is PSO2-SA with Dynamic Sized Swarms (PSO2-SA-DYSS). In order to successfully achieve exploitation at the end of the run, the main PSO2-SA swarm size is reduced and the local search swarms’ sizes are increased as shown in figure 2. This process shifts the PSO2 behavior from exploration to a finer grained search at the end of the runs when the search is hopefully approaching a global optimum.

III. EXPERIMENTAL RESULTS

In this section, two sets of experiments are presented. The first set examines different factors that are known to influence PSO performance in order to reach the best combination possible for PSO2. The second set targets to compare the best algorithm candidate that emerged from the first experimental set, to a similar PSO algorithm called Dynamic Multi-Swarm with local search algorithm DMS-L-PSO [6]. The second set
targets as well to compare PSO2 to one of the best performing algorithms in the CEC-2005 special session on real-parameter competition, a restart evolution strategy with increasing population size (G-CMA-ES) [14].

Table 1. Experiments Sets

<table>
<thead>
<tr>
<th>Exp Set #</th>
<th>Objectives</th>
<th>Test Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Improve PSO2, by finding best combination and incorporating other techniques</td>
<td>Sphere, Rosenbrock, Rastrigin and Griewank</td>
</tr>
<tr>
<td>2</td>
<td>Comparing PSO2 to DMS-L-PSO and G-CMA-ES</td>
<td>4 functions from the CEC-2005 benchmark suite</td>
</tr>
</tbody>
</table>

Different factors are known to influence PSO performance including the swarm size, the neighborhood size, the acceleration coefficients values, the velocity update method and the swarm topology used.

In the first set of experiment, two of those aspects were examined, namely, the velocity update method and the topology structure. These are specified as follows:

1- Velocity update methods:
   a. Inertia weight
   b. Constriction Coefficient
   c. Fully Informed FIPS

2- Topology
   a. Fully connected (gbest)
   b. Ring (lbest)
   c. Von Neumann

Equations 3, 4 and 5 represent the three velocity update methods a, b and c under examination respectively.

\[ v_i^d(t) = \omega \cdot v_i^d(t-1) + c_1 \cdot rand(1) \cdot (pbest_i^d - x_i^d(t-1)) + c_2 \cdot rand(2) \cdot (gbest_i^d - x_i^d(t-1)) \]  \(3\)

\[ v_i^d(t) = \nu \cdot v_i^d(t-1) + c_1 \cdot rand(1) \cdot (pbest_i^d - x_i^d(t-1)) + c_2 \cdot rand(2) \cdot (gbest_i^d - x_i^d(t-1)) \]  \(4\)

\[ v_i^d(t) = \nu \cdot v_i^d(t-1) + \frac{1}{K_i} \sum_{k=1}^{K_i} c \cdot rand(1) \cdot (pbrn_i^d - x_i^d(t-1)) \]  \(5\)

Focusing on improving PSO2 performance, every possible combination of the aspects under study was examined. These different combinations resulted in 9 different configurations of PSO2. A set of standard test functions were used (Sphere, Rosenbrock, Rastrigin and Griewank) to examine each combination performance. Every combination has been applied to PSO2 basic form as well as PSO2-RS, PSO2-SA and PSO2-SA-DYSS.

Swarm size is set to 20, inertia weight in (3) is set to 1.05, constriction coefficient in (4 and 5) is set to 0.729, and acceleration coefficients \((c1, c2)\) are set both to 2.05 [3]. All algorithms are run 25 runs, 4,000 iterations each.

The number of selected particles, for local search, is set to 5. The local search swarms’ sizes is set to 10. The inner iteration max number is set to 10. The local search particle’s perimeter is fixed and set to 5. These settings are suggested from initial examinations.

Results showed that PSO2 with constriction coefficient and gbest topology produced best performance as shown in figure 3. Moreover, random selection of particle for local search that was introduced in PSO2-RS did not introduce significant improvements. Its performance was inferior for most functions and it was not investigated further. Selecting less performing particles for local search appears to be inconsistent with PSO basic concept of following the best after all.

On the contrary to random selection of PSO2-RS, simulated annealing used in PSO2-SA, helped PSO2 escape from inferior local optima regions in two of functions, Rosenbrock and Greiwank, as shown in figures 3 and 4, while maintaining performance in the other functions.

Figure 3 shows that PSO2 supersede PSO2-SA for FES between 20,000 and 60,000. This finding is due to the high probability at the beginning of the iterations to accepting a worse position as best position (taking uphill steps). For FES between 60,000 and 100,000, PSO2-SA presents better results by skipping the local optimum reached at that region. On the other hand, PSO2-SA-DYSS at all FES presents better results than both PSO2 and PSO2-SA.

Similar results can be shown in figure 5 for Griewank function. PSO2-SA and PSO2-SA-DYSS both reached the global optimum before FES reaches 10,000 FES as for PSO2, the algorithm reaches the global optimum at 30,000 FES.
search swarms’ sizes is set to 1 and is kept fixed along the run. The inner iteration max number is set to 5 along all iterations. The local search particle’s perimeter is fixed and set to 5. These settings are suggested from initial examinations.

Figure 6 shows that PSO2 and PSO2-SA nearly have identical results which indicates that simulated annealing did not introduce significant improvements for Rastrigin function. Conversely, PSO2-SA-DYSS has produced better results than both PSO2 and PSO2-SA.

In the second set of experiments, we examined PSO2-SA-DYSS algorithm against four functions from the CEC-2005 benchmark suite [8]. The selected functions have different properties (namely, optima shifts, rotation, and multi-modality) which represent challenges that we need to examine the algorithm against. These functions are shown in table 2.

Swarm size is set to 20 and is reduced to 10 as iterations proceed, acceleration coefficient (c1, c2) are set both to 2.05, constriction coefficient (X) is set to 0.729, simulated annealing initial T0 is set to 100 with an exponential cooling schedule, Vmax is set to 50% of Xmax, and the max number of function evaluations (Max_FES) is set to 100,000.

The number of selected particles, for local search, is set to 1 and increased abruptly at 85% of the Max_FES to 3. The local search swarms’ sizes is set to 1 and is kept fixed along the run. The inner iteration max number is set to 5 along all iterations. The local search particle’s perimeter is fixed and set to 5. These settings are suggested from initial examinations.

For each function PSO2-SA-DYSS is run for 25 times. Best functions error values achieved when Function Evaluations FES = 1000, FES = 10000 and FES = 100000 for the four functions is recorded in table 2. Runs are then sorted in descending order with respect to best fitness found (1st represents the best results, 13th is the median, etc).

Success rate is also recorded for each of the functions. Success rate is defined as number of successful runs divided by the total number of runs. A successful run is defined as the run during which the algorithm achieves a certain accuracy level within the Max_FES. The accuracy level for functions 1 and 3 is 1.0 E-06. The target accuracy level for functions 6 and 10 is 1.0 E-02 [8].

Table 2. Benchmark Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Shifted Sphere Function</td>
<td>( F(x) = \sum_{i=1}^{n} x_i^2 + f_{bias}, \quad z = x \cdot \theta )</td>
</tr>
<tr>
<td>2: Shifted Rotated High CONDITIONED Elliptic Function</td>
<td>( F(x) = \sum_{i=1}^{n} (10^\phi (z_i^2 + x_i^2) + f_{bias}), \quad z = (x - \theta) \cdot M )</td>
</tr>
<tr>
<td>3: Shifted Rotated Elliptic Function</td>
<td>( F(x) = \sum_{i=1}^{n} (400(z_i^2 - x_i^2) + f_{bias}), \quad z = x \cdot \theta )</td>
</tr>
<tr>
<td>4: Shifted Rotated Rastrigin Function</td>
<td>( F(x) = \sum_{i=1}^{n} (x_i^2 - 10 \cos(2\pi x_i) + 10) + f_{bias}, \quad z = (x - \theta) \cdot M )</td>
</tr>
</tbody>
</table>

PSO2-SA-DYSS is compared with Dynamic Multi-swarm with Local Search (DMS-L-PSO) algorithm [6] and G-CMA-ES [14]. DMS-L-PSO is a PSO algorithm variant that uses dynamic topology and Quasi-Newton as a local search method. DMS-L-PSO was one of the algorithms of the CEC-2005 special session on real-parameter optimization competition.
CEC-2005 special session on real-parameter optimization competition was selected because of the suitability of the target optimization problems type and the availability of both known benchmark suite and data sets. DMS-L-PSO was chosen because of its similarity to PSO2. As for G-CMA-ES, it is an evolutionary strategy algorithm. It was selected for its competitive performance over the set of functions provided by CEC-2005.

PSO2-SA-DYSS results for the four functions are plotted in figures 7 and 8. A more detailed discussion is provided below. First, generally looking at PSO2-SA-DYSS and DMS-L-PSO results in table 3 we can make the following observations: at low FES PSO2-SA-DYSS outperforms DMS-L-PSO especially for uni-modal functions (F1 and F3). As FES increases, both PSO2-SA-DYSS and DMS-L-PSO improve regarding accuracy and stability. Stability improvement is reflected by the decrease in standard deviation. However, due to the higher rate of improvements of DMS-L-PSO it tends to exceed PSO2-SA-DYSS for all runs’ grades for the different functions.

Now, looking at the different functions responses specifically:

Considering F1 and F3, which are both uni-modal, with F3 being subject to rotation, results show that PSO2-SA-DYSS presents better results than DMS-L-PSO at all FES except for the worst run (25th) and FES = 100,000. These results comply with our expectations due to the fact that gbest topologies are best suited for uni-modal functions. DMS-L-PSO results also does meet our expectations due to the fact that the diversity resulted from the dynamic topologies used in DMS-L-PSO slows the convergence.

On the other hand, G-CMA-ES performance on F1 is exceptionally better than both algorithms at FES=1000, while it suffers from stagnation afterwards for both F1 and F3.

For F6 (shifted Rosenbrock), PSO2-SA-DYSS is better than DMS-L-PSO for FES = 1000 which is again the effect of gbest topology in fast convergence. For FES = 10,000, PSO2-SA-DYSS presents also better results than DMS-L-PSO for the best and till the run ranked 13th. In the runs ranked 19th and 25th DMS-L-PSO gave better results. These findings lead the mean function error of both algorithms to be close. For FES = 100,000 DMS-L-PSO always presents better results. DMS-L-PSO owes that to Quasi-Newton local search ability. G-CMA-ES when applied on F6 presented superior performance to both DMS-L-PSO and PSO2-SA-DYSS for all FES.

Now we look at the results for F 10, a shifted rotated Rastrigin with a huge number of local optima which makes it a challenging function. Looking at the last three columns in table 3 and considering the best runs in FES = 1,000 and FES = 10,000 we find that PSO2-SA-DYSS reaches its best result of 1.63 E+01 which outperforms DMS-L-PSO. On the other hand G-CMA-ES reaches its best result of 6.11E+00 that is better than both of the other algorithms. Nonetheless, PSO2-SA-DYSS inferior runs’ (7th, median, 19th and 25th) results and stability is best.

At FES = 10,000 DMS-L-PSO catches up with a better fitness of 1.2823 E+01. However, PSO2-SA-DYSS has improved its results at this stage with a fitness of 3.98 E+00 to exceed that of DMS-L-PSO while it is still inferior to that of G-CMA-ES.

Moving further to FES = 100,000, it appears that PSO2-SA-DYSS becomes saturated and is unable to improve, whereas both DMS-L-PSO and G-CMA-ES continue to improve to reach better results.

Applying this same track for F1, F3 and F6, we can conclude that with lesser FES, PSO2-SA-DYSS always produce better results than DMS-L-PSO. However, when processing is allowed to grow considerably, DMS-L-PSO starts to beat PSO2-SA-DYSS. Unlike DMS-L-PSO, G-CMA-ES presented better performance than PSO2-SA-DYSS even at early stages of the run, except for F3 and F10.

Success rate results in table 4 and table 5 shows that PSO2-SA-DYSS achieves high success rate for functions 1 and 3. Function 1 results are better than those of function 3 because of the rotation in function 3 makes it more difficult. For functions 6 and 10, both multi-modals, PSO2-SA-DYSS achieves lower success rates than DMS-L-PSO.

After a comparison to DMS-L-PSO and G-CMA-ES results, we can see that PSO2-SA-DYSS needs further improvements in the local search ability.

IV. CONCLUSION AND FUTURE WORK

A new PSO algorithm that uses smaller independent PSO swarms for local search has been introduced. The proposed algorithm (PSO2) basic form has been tested and upgraded with several modifications producing a number of PSO2 variants (PSO2-RS, PSO2-SA, PSO2-SA-DYSS).

Different combination of velocity update equations and topologies were examined for PSO2. Results showed that constriction coefficient with gbest model presented best results. This model is considered PSO2 basic form.

Three variants were derived from PSO2 basic form. The first variant (PSO-RS) did not result in significant improvements. The second variant (PSO2-SA) improved the performance especially for the more complex problems with higher probability of local optima. Finally, in the third variant (PSO2-SA-DYSS) PSO2-SA was enhanced by dynamic adjustments to the swarms’ sizes. These dynamic adjustments change the algorithm behavior to be more exploitative at the end of the run. Dynamic adjustments improved PSO2 performance for multi-modal problems.

The algorithm still seems to fall into premature convergence as shown in the results in table 3. This finding is most probably due to using gbest topology which tends to converge quickly. More experiments will be conducted to examine the use of dynamic topologies with PSO2 in both the main swarm and local search swarms. PSO2 is to be tested on a more complex, real life, problems such as stereo matching problem. Results are to be published in a subsequent article.
REFERENCES


### Table 3. Best Functions Error Values Achieved When FES=1e+2, FES=1e+3, FES=1e+5

<table>
<thead>
<tr>
<th>FES</th>
<th>Function</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F6</th>
<th>F7</th>
<th>F8</th>
<th>F9</th>
<th>F10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>1.88E-03</td>
<td>5.58E+02</td>
<td>9.23E+00</td>
<td>1.07E+06</td>
<td>4.83E+06</td>
<td>1.09E-06</td>
<td>1.12E+01</td>
<td>3.5452E+06</td>
<td>2.57E+03</td>
</tr>
<tr>
<td>7th</td>
<td>7.56E+03</td>
<td>9.83E+02</td>
<td>1.87E+01</td>
<td>5.73E+06</td>
<td>1.13E+07</td>
<td>3.09E-06</td>
<td>1.27E+02</td>
<td>1.8916E+07</td>
<td>1.12E+04</td>
</tr>
<tr>
<td>13th - Median</td>
<td>1.34E-02</td>
<td>9.8257E+02</td>
<td>2.87E+01</td>
<td>1.14E+07</td>
<td>1.5739E+07</td>
<td>4.26E-06</td>
<td>1.91E+03</td>
<td>3.0379E+07</td>
<td>5.01E+04</td>
</tr>
<tr>
<td>19th</td>
<td>2.46E-02</td>
<td>1.5922E+03</td>
<td>3.90E+01</td>
<td>2.39E+07</td>
<td>2.5972E+07</td>
<td>7.81E-06</td>
<td>3.15E+03</td>
<td>5.0488E+07</td>
<td>1.63E+05</td>
</tr>
<tr>
<td>25th</td>
<td>4.69E-02</td>
<td>1.8506E+03</td>
<td>1.18E+02</td>
<td>5.67E+07</td>
<td>6.0234E+07</td>
<td>3.20E-05</td>
<td>9.89E+05</td>
<td>8.0001E+07</td>
<td>8.15E+05</td>
</tr>
<tr>
<td>Mean</td>
<td>1.70E-02</td>
<td>1.2977E+03</td>
<td>3.64E+01</td>
<td>1.68E+07</td>
<td>2.0606E+07</td>
<td>7.12E-06</td>
<td>4.30E+04</td>
<td>3.3504E+07</td>
<td>1.84E+05</td>
</tr>
<tr>
<td>Std</td>
<td>1.20E-02</td>
<td>3.7805E+02</td>
<td>2.79E+01</td>
<td>1.59E+07</td>
<td>1.3475E+07</td>
<td>7.46E-06</td>
<td>1.97E+05</td>
<td>2.0497E+07</td>
<td>2.77E+05</td>
</tr>
</tbody>
</table>

### Table 4. PSO2-SA-DYSS Success Rate

<table>
<thead>
<tr>
<th>Function</th>
<th>Successful Runs</th>
<th>Success Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>25</td>
<td>100 %</td>
</tr>
<tr>
<td>F3</td>
<td>24</td>
<td>96 %</td>
</tr>
<tr>
<td>F6</td>
<td>13</td>
<td>52 %</td>
</tr>
<tr>
<td>F10</td>
<td>0</td>
<td>0 %</td>
</tr>
</tbody>
</table>

### Table 5. DMS-L-PSO Success Rate

<table>
<thead>
<tr>
<th>Function</th>
<th>Successful Runs</th>
<th>Success Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>25</td>
<td>100 %</td>
</tr>
<tr>
<td>F3</td>
<td>25</td>
<td>100 %</td>
</tr>
<tr>
<td>F6</td>
<td>25</td>
<td>100 %</td>
</tr>
<tr>
<td>F10</td>
<td>0</td>
<td>0 %</td>
</tr>
</tbody>
</table>