Multi-Speed Particle Swarm Optimization

BUTHAINAH S. AL-KAZEMI
Kuwait University
Department of Computer Engineering
P.O. Box 5969 Safat, 13060
KUWAIT
http://www.faculty.eng.kuniv.edu.kw/staff/bsnak.jsp

Abstract: This paper presents a modified version of the Particle Swarm Optimization (PSO). In the new version, particles are allowed to pick from among many positions to where it will move in the next generation. The new version outperforms the PSO algorithm on many benchmarks problems.

Key-Words: PSO, MSPSO, Evolutionary Computation, Convergence.

1. Introduction

The Particle Swarm Optimization (PSO) algorithm has been successful in solving a number of continuous optimization problems 0. Unlike other evolutionary algorithms, each particle (solution) in PSO moves in the search space by constantly updating its velocity vector based on the best solutions found so far by that particle as well as others in the population (swarm) [1] [3]. Many authors add some features to the original PSO to produce a hybrid algorithm that is more efficient [4][5].

This paper presents a modified version of PSO, in which each particle is allowed to pick the best velocity among many that will move it to a better position. This new version will be compared with the original PSO. In the future the algorithm will be compared with the modified versions of PSO.

In section 2, we explain a modified version of the original PSO algorithm which is used as the original PSO by all users. Section 3 describes the MSPSO algorithm. Section 4 contains details of the experiments, and Section 5 describes the results obtained on benchmark problems.

2. PSO Algorithm

The PSO algorithm evolves a population of particles called swarm. Each particle updates its current velocity and position in the search space using historical information regarding its own previous best position as well as the best position discovered by all other particles or neighbouring particles. As both local and global serach were invloved, inertia weights was added by Shi and Eberhart [6] to control those searches. As particles represent solution, the size of a particle depends on the problem. Each particle needs to calculate its new velocity for each of its dimension, then this velocity is used to move the particle to a new position. Figure 1 shows the PSO algorithm.

\[
\begin{align*}
V_{i,a}(t + 1) &= w \cdot V_{i,a}(t) + C_1 \cdot (G_i(t) - X_{i,a}(t)) + C_2 \cdot (U_{i,a}(t) - X_{i,a}(t)) \\
X_{i,a}(t + 1) &= X_{i,a}(t) + V_{i,a}(t + 1)
\end{align*}
\]

Where \(C_1\) and \(C_2\) are random numbers, \(G_i\) is the best particle found so far (by all particles) in dimension \(i\),

Figure 1: Particle Swarm Optimization algorithm
and \( l_{i,n} \) is the best position discovered so far by particle \( n \) in dimension \( i \). Velocity magnitude are clipped to a predetermined maximum value, \( V_{\text{max}} \).

3. MSPSO

The MSPSO algorithm use the same equation for the velocity and the position as the original PSO. The difference is in the way the velocity being updated. Instead of having only one value that is used directly to update the velocity, number of velocities (\( m \)) are produced. Then, those velocities are used to calculate (\( m \)) new positions. The new positions are compared together to find the best position that a particle can move to inorder to reach the optima. By best position we mean the position that can move closer to the optimum. This is done by evaluating the new \( m \) positions and pick the one with the best fitness.

In addition, in the original PSO, the new fitness is calculated after updating the position in all the dimension. In the MSPSO, the fitness is updated after each dimension change. Figure 2 shows the MSPSO algorithm.

![MSPSO algorithm](image)

4. Experiments

The modified PSO and the MSPSO were tested on four different benchmark problems described below. Both algorithms have the same parameters settings: population size = 10, \( V_{\text{max}} = 5 \), and the maximum generation = 200. For MSPSO, the new parameter \( m = 10 \). That is, in each dimension, 10 new velocities are being calculated and the best one of them is chosen.

The following functions were used as test functions, each of which was to be minimized:

1. Generalized Sphere Function:
   \[
   f(x) = \sum_{i=1}^{n} x_i^2
   \]
   Where \( x \) is an \( n \)-dimensional real-valued vector and \( x_i \) is the \( i \)th element of that vector.

2. Generalized Rastrigin Function:
   \[
   f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)
   \]

3. De Jong Function F2:
   \[
   f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1-x_1)^2,
   \]
   Where \(-2.048 \leq X_i \leq 2.048\)

4. De Jong Function F5:
   \[
   f(x_1, x_2) = \frac{1}{k + \sum_{j=1}^{25} f_j^{-1}(x_1, x_2)}
   \]
   Where \( f_j(x_1, x_2) = c_j + \sum_{i=1}^{2} (x_i - a_{ij})^6 \),
   Where \(-65.536 \leq X_i \leq 65.536, k = 500,\)

5. Results

Table 1 lists the average results for both PSO and MSPSO for the computational problem of Circle and Rastrigin 2D. Table 2 displays average results for PSO and MSPSO algorithms for the DeJongF2 and DeJongF5 equations.
<table>
<thead>
<tr>
<th>Circle</th>
<th>Rastrigin 2D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gen</td>
<td>PSO</td>
</tr>
<tr>
<td>5</td>
<td>1117.86</td>
</tr>
<tr>
<td>10</td>
<td>582.766</td>
</tr>
<tr>
<td>15</td>
<td>37.7236</td>
</tr>
<tr>
<td>20</td>
<td>4.0610</td>
</tr>
</tbody>
</table>

Table 1 Average fitness for Circle, Rastrigin 2D over 25 Generations

<table>
<thead>
<tr>
<th>DeJong_F2</th>
<th>DeJong_F5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gen</td>
<td>PSO</td>
</tr>
<tr>
<td>5</td>
<td>7.4910</td>
</tr>
<tr>
<td>10</td>
<td>2.2978</td>
</tr>
<tr>
<td>15</td>
<td>1.9998</td>
</tr>
<tr>
<td>20</td>
<td>1.0349</td>
</tr>
<tr>
<td>25</td>
<td>0.7210</td>
</tr>
</tbody>
</table>

Table 2 Average fitness for DeJong F2 and DeJong F5 over 25 Generations

Figure 3, 4, 5, and 6 shows the behaviour of both PSO and MSPSO for functions: Circle, Rastrigin, DeJong F2, and DeJong F5, respectively.
Table 3 shows the result of applying PSO and MSPSO to Rastrigin function in three different dimensions: 10, 20, and 30. Figures 7, 8, and 9 shows the performance of algorithms in Rastrigin 10, 20, and 30 respectively.

<table>
<thead>
<tr>
<th>Function</th>
<th>PSO</th>
<th>MSPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rastrigin 10</td>
<td>Gen: 200</td>
<td>Gen: 43</td>
</tr>
<tr>
<td>Fitness</td>
<td>14.59906</td>
<td>0.000045</td>
</tr>
<tr>
<td>Rastrigin 20</td>
<td>Gen: 200</td>
<td>Gen: 60</td>
</tr>
<tr>
<td>Fitness</td>
<td>49.98067</td>
<td>0.497507</td>
</tr>
<tr>
<td>Rastrigin 30</td>
<td>Gen: 200</td>
<td>Gen: 70</td>
</tr>
<tr>
<td>Fitness</td>
<td>116.0156</td>
<td>0.994974</td>
</tr>
</tbody>
</table>

Table 3 Average fitness for Rastrigin 10, Rastrigin 20, Rastrigin 30 over 25 Generations

6. Problem Solution

As table 1 and 2 show, MSPSO was able to reach the optimum faster than PSO for all the test functions. In circle function, PSO was able to reach the optimum in mean generation 78, while MSPSO already reached the optimum in generation 21. In addition, for Rastrigin function, PSO did not reached the optimum with the experimental settings, while MSPSO reached the optimum in generation 2. For DeJong F2, PSO reached the optimum in generation 177 and for DeJong F5 it did not reached the optimum, while MSPSO reached the optimum in generation 25 for DeJong F2, and near the optimum in DeJong F5. Figures 3 to 6 clearly shows the convergence of MSPSO to the optimum faster then the PSO algorithm.

From table 3, it is observed that MSPSO was able to maintain the convergence to the optimum faster than PSO when the problem size increased. For Rastrigin 10, MSPSO was able to reach the optimum, while PSO failed even to come closer to the optimum. As the problem size increased, MSPSO was still able to reach near the optimum (according to the problem settings), but PSO slowed down and was unable to converge. Figures 7 to 9, clearly shows the performance of both MSPSO and PSO.

Not shown in the paper, more tests were made in Rastrigin 10, 20, and 30 by changing the population’s size and the maximum generation for both PSO and MSPSO. PSO still used many generations to reach near the optimum, while MSPSO reached the optimum with less number of
Conclusions

The results presented in the previous section show that MSPSO is a powerful algorithm that succeeds in solving the benchmark optimization problems using a small number of generation to reach the best fitness.

The MSPSO algorithm was faster in converging to the optimum comparing it to the modified version of PSO. The new metric, which was picking the best velocity among multiple speeds, made the convergence faster. In addition, updating the fitness in each dimension will increase the diversity, and thus escape from local and global optima.

Future Work

As the MSPSO overcomes the PSO algorithm, it will be tested to the other modified versions to find how good it is as compared to MSPSO algorithm.

References