Particle Swarm Optimization
Containing Multiple Swarms

Masaki SUGIMOTO
(Tokushima University)
Taku HARAGUCHI
(Tokushima University)
Haruna MATSUSHITA
(Tokushima University)
Yoshifumi NISHIO
(Tokushima University)

1. Introduction
The particle swarm optimization (PSO) is a popular optimization technique for the solution of objective function and is an evolutionary algorithm to simulate the movement of flocks birds. Due to the simple concept, such as easy implementation and quick convergence, PSO has gained much attentions and wide applications in different fields. However, if a particle is trapped into a local optimum, the bust-out is difficult.

This study proposes Multiple PSO (MPSO). The feature of MPSO is that the swarm of MPSO is not one but multiple. The swarms share an information of the best value in each group. Except the swarm including the best particle in whole swarms, all the particles are repositioned to escape the local optimum. We investigate behaviors of MPSO and confirm its efficiency.

2. Multiple PSO (MPSO) Algorithm
The most important feature of MPSO is that MPSO has K swarms, i.e. multiple particle groups: Swarm_k (k = 1, 2, . . . , K). Each swarm contains N/K particles whose position X_ki = (x_ki1, x_ki2, . . . , x_kiD) (i = 1, 2, . . . , N/K) are located randomly. The velocity vector V_ki = [V_min, V_max] for each particle k_i belonging to Swarm_k is assigned randomly.

[MPSO1](Initialization) Let t = 0 and t_R = 0, i.e., t is the simulation step and t_R is the time step for the reposition. The quality of a particle position depends on a cost function F(X_ki). We find the personal best pbest_ki, which is the best cost of the particle i in Swarm_k, and the global best gbest_k, which is the best value of pbest_ki in Swarm_k. The whole global best gbest_w, which is the best value in whole particles in all swarms, is also found:

\[ gbest_k = \min(pbest_k), \quad gbest_w = \min(\{gbest_k\}) \quad (1) \]

The swarm including the particle with gbest_w is called the best swarm. P_k = \{p_k1, p_k2, . . . , p_kD\}, P_w = \{p_w1, p_w2, . . . , p_wD\} represent the position of particle k_i in Swarm_k, and the position of the particle with gbest_w, respectively.

[MPSO2] Compare the current cost F(X_k(t)) of each particle with its best cost so far: if F(X_k(t)) < pbest_ki then pbest_ki = F(X_ki(t)) and P_k = X_ki(t). Similarly, gbest_k and gbest_w are evaluated according to Eq. (1). Therefore, P_k and P_w are also evaluated.

[MPSO3](Update) The velocity vector V_ki for each particle and each particle position X_ki are updated:

\[ v_{ki}(t+1) = w v_{ki}(t) + c_1 r_1 \{p_{ki}(t) - x_{ki}(t)\} + c_2 r_2 \{p_{w}(t) - x_{ki}(t)\} + c_3 r_3 \{g_{best}(t) - x_{ki}(t)\} \]
\[ x_{ki}(t+1) = x_{ki}(t) + v_{ki}(t+1) \]

where d = 1, 2, . . . , D, r_1, r_2 and r_3 are random variables distributed uniformly in [0, 1], w is an inertia weight, and c_1, c_2, and c_3 are positive acceleration coefficients.

[MPSO4] If t_R = T/K, we perform [MPSO5], if not, we perform [MPSO6]. Thus, we perform [MPSO5] every time when T/K simulation steps are performed. T is the maximum number of the simulation.

[MPSO5](Reposition) We reposition the particle positions X_ki of all the Swarms_k except the best swarm at random and reassign their velocities V_ki at random. We reset t_R = 0.

[MPSO6] Let t = t + 1 and t_R = t_R + 1. Go back to [MPSO2], and repeat until t = T.

3. Numerical Experiments
In order to confirm the performance of MPSO algorithm, we apply MPSO to the Rastrigin Function:

\[ F(X_ki) = DA + \sum_{d=1}^{D}(x_{ki}^2 - A \cos(2\pi x_{ki}d)) \quad (3) \]

where -5.12 \leq x_{ki} \leq 5.12, D = 30, and A = 5. We set parameters as follows: w = 0.5, c_1 = 2.0, c_2 = 1.7, c_3 = 0.3, T = 2000. MPSO has K = 4 swarms, and each swarm contains 10 particles, i.e. \( N/K = 40 \).

We carry out the simulation 100 times and the results of MPSO, the standard PSO and PSO*, which is the standard PSO with the reposition process, are shown in Table 1. We can see that the result of MPSO is the best value in three algorithms. In MPSO, all the particles share the information of the best value in each swarm, and the particles except the best swarm are repositioned. As the result, if the particle is trapped into local optimum, the particle easily escapes from the local optimum. Therefore, we can confirm that MPSO algorithm is the most effective.

<table>
<thead>
<tr>
<th></th>
<th>PSO</th>
<th>PSO*</th>
<th>MPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG</td>
<td>35.94</td>
<td>22.91</td>
<td>13.83</td>
</tr>
<tr>
<td>MIN</td>
<td>11.87</td>
<td>10.75</td>
<td>6.92</td>
</tr>
<tr>
<td>MAX</td>
<td>39.50</td>
<td>45.52</td>
<td>27.20</td>
</tr>
</tbody>
</table>

4. Conclusions
In this study, we have proposed the new PSO algorithm, Multiple PSO (MPSO). We have investigated its behaviors with the simulation and have confirmed the efficiency.

Reference