An Improved Multi-State Particle Swarm Optimization for Discrete Optimization Problems

Ismail Ibrahim, Zuwairie Ibrahim, Hamzah Ahmad, and Zulkifli Md. Yusof
Universiti Malaysia Pahang
26600 Pekan, Pahang, Malaysia
pee12001@stmail.ump.edu.my, zuwairie@ump.edu.my, hamzah@ump.edu.my, zmdyusof@ump.edu.my

Abstract—Particle swarm optimization (PSO) has been successfully applied to solve various optimization problems. Recently, a state-based algorithm called multi-state particle swarm optimization (MSPSO) has been proposed to solve discrete combinatorial optimization problems. The algorithm operates based on a simplified mechanism of transition between two states. However, the MSPSO algorithm has to deal with the production of infeasible solutions and hence, additional step to convert the infeasible solution to feasible solution is required. In this paper, the MSPSO is improved by introducing a strategy that directly produces feasible solutions. The performance of the improved multi-state particle swarm optimization (IMSPSO) is empirically evaluated based on a set of travelling salesman problems (TSPs). The experimental results showed the newly introduced approach is promising and consistently outperformed the binary PSO algorithm.

Keywords-component; discrete combinatorial optimization problem; multi-state; particle swarm optimization

I. INTRODUCTION

Particle swarm optimization (PSO) algorithm is a population-based stochastic optimization technique developed by Kennedy and Eberhart [1]. It mimics swarms behavior in performing their tasks like bird flocks and fishes to discover an optimal solution based on an objective function. The original PSO is predominately used to find solutions for continuous optimization problems. Later, a reworked of the original PSO algorithm known as binary particle swarm optimization (BPSO) algorithm has been developed to allow PSO algorithm to operate in discrete binary variables [2].

At present, a lot of proposals have been presented to improve the BPSO algorithm in terms of convergence speed [3-6], stagnation in optimum [7-10], computational time [5,11], and local exploration [12]. Previously, the authors have introduced a new variant of PSO for discrete optimization problems called multi-state particle swarm optimization (MSPSO) [13-14]. However, a limitation of the MSPSO algorithm is that infeasible solutions are frequently produced and additional step is required to convert infeasible solution to feasible solution.

In this paper, to address the limitation of MSPSO algorithm, a rule-based strategy is embedded in MSPSO algorithm. As a test, the improved MSPSO (IMSPSO) is applied to solve the travelling salesman problem (TSP). The results were promising and in several occasions, the IMSPSO algorithm able to obtain better solutions compared to the BPSO algorithm.

II. PARTICLE SWARM OPTIMIZATION

In the original PSO algorithm, an optimal solution is found by simulating social behaviour of bird flocking. PSO requires individuals or particles, which encode the possible solutions to the optimization problem using their positions. The particles can attain the solution effectively by using the common neighbouring information. Using this information, each particle compares its current position with the best position found by its neighbours so far. Pseudo-code of PSO algorithm is shown below:

- Generate the initial swarm;
- Evaluate the fitness for each particle;
- While stopping criteria is not satisfied Do
- Update pbest and gbest of the swarm;
- Update particles’ velocity and position;
- Evaluate fitness for each particle;
- Endwhile
- Output: Best solution found.

where pbest and gbest are defined as personal and global best of the particles.

Consider a function minimization problem. In initialization stage, I particles are randomly positioned in a search space and the particles are assigned with initial velocity. A particle’s position is represented as \(s_i(d)\) \((i = 1, 2, \ldots I; d = 1, 2, \ldots, D)\), which represent a solution. Fitness value for each particle is evaluated by calculating the objective functions with respect to \(s_i(k)\), where \(k\) represents the iteration number. Each particle updates its velocity and position as:

\[
v_i(k + 1, d) = \omega v_i(k, d) + c_1 r_1 pbest_i(k, d) - s_i(k, d) + c_2 r_2 gbest(k, d) - s_i(k, d)
\]

\[
s_i(k + 1, d) = s_i(k, d) + v_i(k + 1, d)
\]

where \(c_1\) and \(c_2\) are the cognitive and social coefficients, respectively, \(r_1\) and \(r_2\) are random number uniformly
distributed between 0 and 1, and $\omega$ is called inertia weight, which is used to control the impact of the previous history of velocities on the current velocity of each particle.

III. MULTI-STATE PARTICLE SWARM OPTIMIZATION

For solving discrete optimization problems, each particle’s vector or dimension in the MSPSO algorithm is represented as state. To elaborate the multi-state representation, Burma14 benchmark instance of TSP, as shown in Fig. 1, is used as an example.

All the cities in Burma14 can be represented as a collective of states, as presented in Fig. 2, in which the small black circle represents the states. A centroid of the circle shows the current state. Radius of the circle represents velocity value possessed by the current state. These three elements occur in each dimension for each particle. The exploitation of particle’s velocity and a mechanism of state transition in the MSPSO algorithm take places after $pbest$ and $gbest$ of all particles are updated.

The velocity calculation in the MSPSO algorithm is different compared with the original PSO algorithm due to $pbest(k,d)$, and $s(k,d)$ in the form of state. With regard to the PSO algorithm, a particle has three movement components; the inertia, cognitive, and social component. The effect of the first, second, and third component are the particle bias to follow in its own way, to go back to its best previous position, and to go towards the global best particle, respectively. However, in the MSPSO algorithm, the velocity value is the summation of previous velocity, cost function subjected to particle’s best position and current particle’s position multiplies with a cognitive coefficient and a uniform random value, and cost function subjected to global best particle and current particle’s position multiplies with a social coefficient and a uniform random value. The velocity equation is derived as follow:

$$
\begin{align*}
    v_i(k+1,d) &= \omega v_i(k,d) + c_1 r_1 C(pbest_i(k,d)), \\
    s_i(k,d) &= c_2 r_2 C(gbest_i(k,d),s_i(k,d)) 
\end{align*}
$$

Cost can be defined as the distance TSP or in general, the cost between two states is a positive number given by $C(s_i(k,d),s_j(k,d))$.

In the MSPSO algorithm, once the velocity is updated, the process of updating current to next state for each dimension of each particle is executed. Let the current state be a centroid and the updated velocity value as the radius of the circle. Any state that is located in the area of the circle is defined as a member of inner states (IS) group. A next state is then selected randomly among the member of IS group using (4). Given a set of $j$ IS members $I_i(k,d) = (I_1(k,d),..,I_j(k,d)))$.

$$
    s_i(k+1,d) = \text{random} (I_1((t,d),..,I_j((t,d))) 
$$

In order to update state in the MSPSO algorithm, a random function is applied. Equation (4) may lead to the existence of repeated state in an updated solution. Let consider a solution of a particle at a particular iteration consisting 14-dimensional vector $\{s_5, s_3, s_4, s_{11}, s_2, s_8, s_9, s_{13}, s_{12}, s_{10}, s_1, s_4, s_6, s_7\}$. Note that this solution has no repeated state. This solution is then subjected to dimension-by-dimension updates. After updating each state in the solution, the updated solution could be a 14-dimensional vector $\{s_4, s_7, s_8, s_{11}, s_{13}, s_5, s_2, s_1, s_4, s_6, s_9, s_{10}, s_3\}$, as illustrated in Fig. 3. Obviously, the state in the 3rd, 5th, 6th, 8th, 9th, 12th, and 13th dimension occurs more than once, and hence, the solution is infeasible for the TSP. As an example, the updated state in the 5th and 9th dimension of the updated solution is $x_{13}$, which are identical.

![Figure 1](image1.png)

**Figure 1.** An example of Travelling Salesman Problem (TSP). The name of this benchmark instance is Burma14.

![Figure 2](image2.png)

**Figure 2.** The illustration of the multi-state representation in MSPSO algorithm for Burma14 benchmark instance of TSP. Each dimension of each particle applies the same representation.
IV. THE IMPROVED MSPSO

In this section, the improved MSPSO (IMSPSO) is introduced. The IMSPSO algorithm practices similar general principal of MSPSO algorithm but with incorporating a feasible-based strategy to the mechanism of state transition. The feasible-based strategy fundamentally removes the limitation of MSPSO algorithm in which all solutions generated are feasible. To operate this strategy, the information regarding all members of inner states (IS) and outer states (OS) should be known. In addition, we introduce a new group called the selected states (SS) consisting all states that have been selected as the next state from these two groups IS and OS. Given a set of _h_ selected states (SS) \( T_i(k,d) = (T_i(k,d), \ldots, T_h(k,d)) \) All members of the selected states (SS) group should be identified because all members of this group invalid to be selected as the next state. This happens because these states have been selected in previous selections.

For the IS and the OS group, let us consider a set of \( IS \) \( I_j(k,d) = (I_j(k,d), \ldots, I_{h_j}(k,d)) \) and a set of \( OS \) \( O_j(k,d) = (O_j(k,d), \ldots, O_{h_j}(k,d)) \). Based on the current state and the updated velocity of the current state, a next state can be selected as in (5):

\[
 s_{i}(k+1,d) = \begin{cases} 
 \text{random}(Valid_{I_i}(k,d)) & \text{if } Valid_{I_i}(k,d) \neq \emptyset \\
 \text{random}(Valid_{O_i}(k,d)) & \text{if } Valid_{I_i}(k,d) = \emptyset 
\end{cases}
\]  

(5)

where \( Valid_{I_i}(k,d) = (I_i(k,d) - (I_i(k,d) \cap T_i(k,d))) \), \( Valid_{O_i}(k,d) = (O_i(k,d) - (O_i(k,d) \cap T_i(k,d))) \), and \( \emptyset \) is empty set.

The next state is randomly chosen from the \( Valid_{I_i} \) group. If remaining members of the \( Valid_{I_i} \) group do not exist, the next state is then randomly chosen from the \( Valid_{O_i} \) group. This process is applied to each dimension of each particle. This process ends when all dimensions in all particles have been updated.

It is been observed that the introduction of the SS group and the selection of the next state either from any member of the \( Valid_{I_i} \) group or the \( Valid_{O_i} \) group has successfully produced feasible solutions for each particle because this strategy embeds a rule which is “each state can be chosen only once as the next state in a solution”.

The IMSPSO algorithm is presented in the following pseudo-code:

Input: \( I \) as the number of particles, \( D \) as the maximum dimensions, \( s_{i}(k,d)(i=1,2,\ldots,I; d=1,2,\ldots,D) \) as the current position of each particle, and \( v_{i}(k+1,d)(i=1,2,\ldots,I; d=1,2,\ldots,D) \) as current velocity of each particle.

\[ \text{particle} = 1; \]
\[ \text{Repeat} \]
\[ \text{Initialize member of the SS group;} \]
\[ \text{Dimension} = 1; \]
\[ \text{Repeat} \]
\[ \text{Generate a circle based on the current state and updated velocity to determine the location of all members of the IS group and the OS group;} \]
\[ \text{Remove all members of the SS group from the IS group;} \]
\[ \text{If there is any member of the IS group still exists} \]
\[ \text{Then} \]
\[ \text{Randomly choose any state from Valid}_{I_i} \text{ group as the next state;} \]
\[ \text{Randomly choose any state from Valid}_{O_i} \text{ group as the next state;} \]
\[ \text{Add the next state chosen as a new member of the SS group;} \]
\[ \text{dimension} ++; \]
\[ \text{Until dimension} = D; \]
\[ \text{particle} ++; \]
\[ \text{Until particle} = I; \]
\[ \text{Output: Feasible solutions generated.} \]

V. EXPERIMENTS

In this study, six sets of TSP benchmark instances taken from TSPlib (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/) are considered, namely:

\( a) \) Burma14: A TSP problem of 14 cities. The optimal route length is 3323.
\( b) \) Ulysses16: A TSP problem of 16 cities. The optimal route length is 6859.
\( c) \) Ulysses22: A TSP problem of 22 cities. The optimal route length is 7013.
\( d) \) Bays29: A TSP problem of 29 cities. The optimal route length is 2020.
\( e) \) Eil51: A TSP problem of 51 cities. The optimal route length is 426.
\( f) \) Berlin52: A TSP problem of 52 cities. The optimal route length is 7542.

By considering each two cities connected in Eil51 and Berlin52 benchmark instances, the first city and second city are defined by two points \((u_1, q_1)\) and \((u_2, q_2)\), respectively in the Euclidean plane. The Euclidean distance \( (euc \text{ dist}) \) for traveling between the two cities is formulated as in Equation (6):

\[ euc \text{ dist} = \sqrt{(u_2 - u_1)^2 + (q_2 - q_1)^2} \]
\[ euc_{\text{dist}_{i,j}} = \sqrt{(u_i - u_j)^2 + (q_i - q_j)^2} \]  \hspace{1cm} (6)

Meanwhile, for Burma14, Ulysses16, Ulysses22, and Bays29 benchmark instances, the first and city second city are also firstly defined by two points \((u_1, q_1)\) and \((u_2, q_2)\). The coordinate of the two cities are then converted to latitude and longitude format in which the new representation of the two cities are \((lat_1, long_1)\) and \((lat_2, long_2)\), converted using:

\[
l_{\text{lat}}_1 = \pi \left( \left\lfloor \frac{floor(u_1) + 5x_1 - floor(u_1)}{3} \right\rfloor \right) \frac{180}{\pi}
\]
(7)

\[
l_{\text{lat}}_2 = \pi \left( \left\lfloor \frac{floor(u_2) + 5x_2 - floor(u_2)}{3} \right\rfloor \right) \frac{180}{\pi}
\]
(8)

\[
l_{\text{long}}_1 = \pi \left( \left\lfloor \frac{floor(q_1) + 5x_1 - floor(q_1)}{3} \right\rfloor \right) \frac{180}{\pi}
\]
(9)

\[
l_{\text{long}}_2 = \pi \left( \left\lfloor \frac{floor(q_2) + 5x_2 - floor(q_2)}{3} \right\rfloor \right) \frac{180}{\pi}
\]
(10)

The geographical distance \((\text{geog\_dist})\) for traveling between the two cities is then denoted as:

\[
\text{geog\_dist}_{i,j} = \left\lfloor \text{radian} \times acos(0.5 \times ((1 + g_1) \times g_2 \times (1 - g_1) \times g_1) + 1)) \right\rfloor
\]
(11)

where \(\text{floor}\) changes a value to be the largest integer smaller than the value, the value of radian is \(6378.3888\), \(acos\) is the inverse of the cosine function, the value of \(\pi\) is \(3.141592\), \(g_1 = \cos(long_1 \cdot long_2)\), \(g_2 = \cos(lat_1 \cdot lat_2)\), and \(g_3 = \cos(lat_1 + lat_2)\).

VI. RESULTS AND DISCUSSIONS

This section presents comparisons between the IMSPSO algorithm and the binary PSO (BPSO) algorithm [2]. Due to sensitivity of algorithmic parameters, \(c_1, c_2,\) and \(\omega\) for the BPSO algorithm were chosen according to their reported values. The parameters and their respective value are listed in Table 1.

The quality of results is measured based on the objective values of the best solutions found by each algorithm on each TSP benchmark instance. Since the number of independent trials on each TSP benchmark instance is 50, the quality of results is determined based on the fitness values of 50 solutions. The mean, minimum, and maximum of fitness values of 50 solutions, and the standard deviation are recorded. The best result found from these 50 independent trials for each benchmark instance is selected and then portrayed in Fig. 3, 4, 5, 6, 7, and 8.

Table 2 and 3 show the quality of results of the IMSPSO and the BPSO for 50 trials on the six sets of the TSP benchmark instance. The IMSPSO yields smaller values of the mean for each benchmark instance compared with the BPSO, thus verifying that the IMSPSO produces higher quality of solutions. With regard to the pattern of convergence, the IMSPSO converges slower compared with the BPSO for Burma14, Ulysses16, Bays29, Eil51, and Berlin52 as presented in Fig. 3, 4, 6, 7, and 8. Meanwhile, for Ulysses22, the BPSO converges slower compared with the IMSPSO. The pattern of convergence for this condition is illustrated in Fig. 5.

### Table I. Parameters Setting for the IMSPSO Algorithm and the BPSO Algorithm

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Algorithm</th>
<th>IMSPSO</th>
<th>BPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of trials</td>
<td></td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Number of iterations</td>
<td></td>
<td>10000</td>
<td>10000</td>
</tr>
<tr>
<td>Number of particles</td>
<td></td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Cognitive and social coefficient, (c_1) and (c_2)</td>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Range of inertia weight, (\omega)</td>
<td></td>
<td>0.9-0.4</td>
<td>0.9-0.4</td>
</tr>
<tr>
<td>Random, (r_1) and (r_2)</td>
<td></td>
<td>[0.1]</td>
<td>[0.1]</td>
</tr>
</tbody>
</table>

### Table II. Numerical Results for the IMSPSO Algorithm

<table>
<thead>
<tr>
<th>Benchmark instance</th>
<th>IMSPSO</th>
<th>IMSPSO</th>
<th>IMSPSO</th>
<th>IMSPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimum</td>
<td>Mean</td>
<td>Maximum</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>Burma14</td>
<td>3475</td>
<td>3712.66</td>
<td>4004</td>
<td>121.41</td>
</tr>
<tr>
<td>Ulysses16</td>
<td>7011</td>
<td>7765.40</td>
<td>8087</td>
<td>210.28</td>
</tr>
<tr>
<td>Ulysses22</td>
<td>9457</td>
<td>9677</td>
<td>9834</td>
<td>154.81</td>
</tr>
<tr>
<td>Bays29</td>
<td>3646</td>
<td>3912.42</td>
<td>4107</td>
<td>110.99</td>
</tr>
<tr>
<td>Eil51</td>
<td>1143</td>
<td>1219.22</td>
<td>1258</td>
<td>25.82</td>
</tr>
<tr>
<td>Berlin52</td>
<td>19980</td>
<td>21686.10</td>
<td>22360</td>
<td>497.20</td>
</tr>
</tbody>
</table>

### Table III. Numerical Results for the BPSO Algorithm

<table>
<thead>
<tr>
<th>Benchmark instance</th>
<th>BPSO</th>
<th>BPSO</th>
<th>BPSO</th>
<th>BPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimum</td>
<td>Mean</td>
<td>Maximum</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>Burma14</td>
<td>3527</td>
<td>3739.88</td>
<td>3829</td>
<td>101.69</td>
</tr>
<tr>
<td>Ulysses16</td>
<td>7572</td>
<td>7828.26</td>
<td>8203</td>
<td>163.38</td>
</tr>
<tr>
<td>Ulysses22</td>
<td>9565</td>
<td>10084.10</td>
<td>10499</td>
<td>367.60</td>
</tr>
<tr>
<td>Bays29</td>
<td>3747</td>
<td>3952.52</td>
<td>4114</td>
<td>96.25</td>
</tr>
<tr>
<td>Eil51</td>
<td>1218</td>
<td>1241.80</td>
<td>1264</td>
<td>16.38</td>
</tr>
<tr>
<td>Berlin52</td>
<td>21124</td>
<td>21853.20</td>
<td>22393</td>
<td>364.29</td>
</tr>
</tbody>
</table>
Figure 3. Comparison of the convergence pattern of the IMSPSO and the BPSO for Burma14 benchmark instance.

Figure 4. Comparison of the convergence pattern of the IMSPSO and the BPSO for Ulysses16 benchmark instance.

Figure 5. Comparison of the convergence pattern of the IMSPSO and the BPSO for Ulysses29 benchmark instance.

Figure 6. Comparison of the convergence pattern of the IMSPSO and the BPSO for Bays29 benchmark instance.

Figure 7. Comparison of the convergence pattern of the IMSPSO and the BPSO for Eil51 benchmark instance.

Figure 8. Comparison of the convergence pattern of the IMSPSO and the BPSO for Berlin52 benchmark instance.
VII. CONCLUSIONS AND FUTURE WORK

In this study, the IMSPSO is proposed algorithm by introducing a strategy to avoid generation of unfeasible solutions. This strategy is operated to directly produce feasible solution for each particle in solving discrete combinatorial optimization problems, particularly in the TSP. To evaluate the performance of the IMSPSO algorithm and the BPSO, six sets of the TSP benchmark instances were used. For this problem, each algorithm was executed to find the shortest route. Experimental results showed that the IMSPSO algorithm consistently outperformed the BPSO in each TSP benchmark instances used.

ACKNOWLEDGEMENT

This work is financially supported by Ministry of Education Malaysia through the projects Fundamental Research Grant Scheme VOT RDU140132 granted by Universiti Malaysia Pahang.

REFERENCES


