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

Ismail Ibrahim, Zuwairie Ibrahim, Hamzah Ahmad
Faculty of Electrical and Electronic Engineering
Universiti Malaysia Pahang
26600 Pekan, Pahang, Malaysia
pee12001@std.ump.edu.my, zuwairie@ump.edu.my,
hamzah@ump.edu.my

Zulkifli Md. Yusof[2]
Faculty of Manufacturing Engineering
Universiti Malaysia Pahang
26600 Pekan, Pahang, Malaysia
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 are statistically analyzed and show that the IMSPSO 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 [3-12], stagnation in local optimum [13-18], computational time [5, 19, 20], exploration [9, 21, 22], and premature convergence [23]. Previously, the authors have introduced a new variant of PSO for discrete optimization problems called multi-state particle swarm optimization (MSPSO) [24, 25]. 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;
Set each particle's *pbest* and *gbest*;
While stopping criteria is not satisfied **Do**
  Update particles' velocity and position;
  Re-evaluate fitness for each particle;
  Update *pbest* and *gbest* of the swarm;
Endwhile
Output: Best solution found.

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

The pseudo-code of PSO algorithm can be described as follows:

1. 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, … I; d = 1, 2, …, D), which represent a solution.

2. The fitness of each particle is evaluated by calculating the objective functions with respect to $s_i(k)$, where k represents the iteration number.
3. The best previous position of each particle called *pbest* is set. The best particle among all the particles in the group called *gbest* is then also set.
4. Each particle updates its velocity and position as:

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

$$s_i(k+1,d) = s_i(k,d) + v_i(k+1,d) \quad (2)$$

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 ω is called inertia weight, which is used to control the impact of the previous history of velocities on the current velocity of each particle.
5. The fitness of each particle is re-evaluated by calculating the objective functions with respect to $s_i(k)$.
6. The *pbest* is then updated by a more optimal, and the *gbest* is also updated by the most optimal *pbest* of all particles.
7. Finally, the optimum solution is found when the stopping condition is met.

III. Multi-State Particle Swarm Optimization

To solve 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 $pbest_i(k,d)$, and $s_i(k,d)$ 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:

$$v_i(k+1,d) = \omega v_i(k,d) + c_1 r_1 C(pbest_i(k,d), s_i(k,d)) + c_2 r_2 C(gbest(k,d), s_i(k,d)) \quad (3)$$

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

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

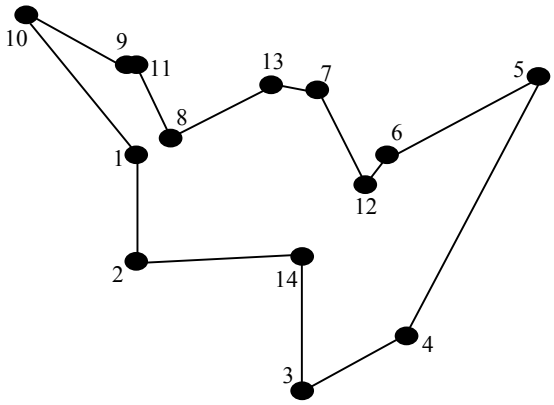

Figure 1. An example of Travelling Salesman Problem (TSP). The name of this benchmark instance is Burma14.

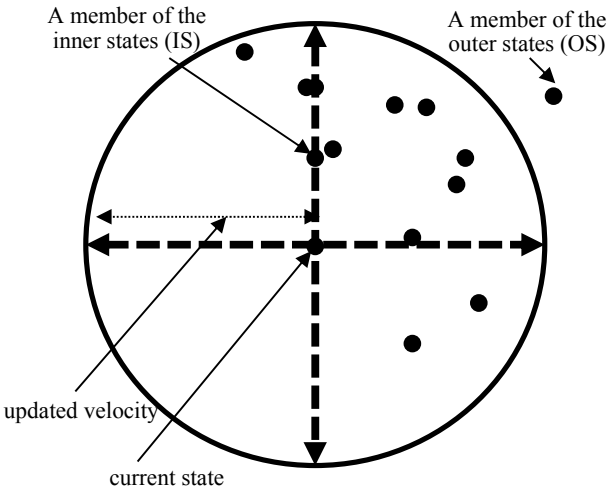

Figure 2. The illustration of the multi-state representation in MSPSO algorithm for Burma14 benchmark instance of TSP. Each dimension of each particle applies the same representation.

is then selected randomly among the member of IS group using (4). Given a set of j IS members $I_i(k,d) = (I_{i_1}((k,d),...,I_{i_j}(k,d)))$.

$$s_i(k+1,d) = random\,(I_{i_1}((t,d),...,I_{i_j}(t,d)) \qquad (4)$$

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 $\{s_5, s_3, s_{14}, s_{11}, s_2, s_8, s_9, s_{13}, s_{12}, s_{10}, s_1, s_4, s_6, s_7\}$. 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 {s4, s7, s8, s11, s13, s8, s5, s12, s13, s10, s1, s8, s12, s9}, 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 x13, which are identical.

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 introduction 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) $T_i(k,d) = (T_{i_1}(k,d),...,T_{i_h}(k,d))$. 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. The introduction of the SS group does not cause complexity problem in the improved MSPSO.

For the IS and the OS group, let us consider a set of j IS $I_i(k,d) = (I_{i_1}(k,d),...,I_{i_j}(k,d))$ and a set of l OS $O_i(k,d) = (O_{i_1}(k,d),...,O_{i_l}(k,d))$. Based on the current state and the updated velocity of the current state, a next state can be selected as in (5):

$$s_i(k+1,d) = \begin{cases} random\,(Valid\_I_i(k,d)) \\ \text{if } Valid\_I_i(k,d) \neq \emptyset \\ \\ random\,(Valid\_O_i(k,d)) \\ \text{if } Valid\_I_i(k,d) = \emptyset \end{cases} \qquad (5)$$

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.

particle = 1;
Repeat
  Initialize member of the SS group;
  Dimension = 1;
  Repeat
    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;
    Remove all members of the SS group from the IS group;
    If there is any member of the IS group still exists **Then**
      Randomly choose any state from $Valid\_I_i$ group as the next state;
    Randomly choose any state from $Valid\_O_i$ group as the next state;
    Add the next state chosen as a new member of the SS group;
    dimension ++;
  Until dimension == D;
  particle ++;
Until particle == I;
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_dist*) for traveling between the two cities is formulated as in Equation (6):

$$euc\_dist_{e_i,e_j} = \sqrt{(u_1 - u_2)^2 + (q_1 - q_2)^2} \quad (6)$$

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:

$$lat_1 = \pi \frac{(floor(u_1) + 5(x_1 - floor(u_1))/3)}{180} \quad (7)$$

$$lat_2 = \pi \frac{(floor(u_2) + 5(u_2 - floor(u_2))/3)}{180} \quad (8)$$

$$long_1 = \pi \frac{(floor(q_1) + 5(q_1 - floor(q_1))/3)}{180} \quad (9)$$

$$long_2 = \pi \frac{(floor(q_2) + 5(q_2 - floor(q_2))/3)}{180} \quad (10)$$

The geographical distance (*geog_dist*) for traveling between the two cities is then denoted as:

$$geog\_dist_{e_i,e_j} = floor((radian \times a\cos(0.5 \times ((1+g_1) \times g_2 - (1-g_3) \times g_3)) + 1)) \quad (11)$$

where *floor* changes a value to be the largest integer smaller than the value, the value of radian is 6378.3888, *a*cos is the inverse of the cosine function, the value of π is 3.141592, g_1 is cos($long_1$ - $long_2$), g_2 is cos(lat_1 - lat_2), and g_3 is 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 ω 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 Ulysses29, the BPSO converges slower compared with the IMSPSO. The pattern of converge for this condition is illustrated in Fig. 5.

The difference between the solutions obtained by these two algorithms is also analysed using boxplots, as shown in Fig. 9, 10, 11, 12, 13, and 14. In these figures, each rectangle represents one of the six benchmark. Inside each rectangle, boxplots representing the distribution of the best solution value for the IMSPSO algorithm and the BPSO algorithm are drawn. In each boxplot, the minimum and maximum values are the lowest and highest lines, the upper and lower ends of the box are the upper and lower quartiles, a line within the box shows the median, and the isolated points are the outliers of the distribution.

Each boxplot offers the information about the quality and the performance of the IMSPSO algorithm and the BPSO algorithm. The size of the box shows the magnitude of the variance of the results; thus a smaller box presents a consistent performance of the parameters. In some occasions, the results obtained by these two algorithms contain outliers. The outliers are valid solutions as the outliers are genuine

outliers that are produced without clerical errors and with an actual measurement from the experiments. These out-of-norm observations are caused by the stochastic behaviour of the algorithms. Because the TSP is a minimization problem, a lower boxplot is desirable as it indicates better quality of the solutions found. It can be observed from Fig. 9, 10, 11, 12, 13, and 14 that the IMSPSO gives good performance on the six sets of TSP benchmark instances; the IMSPSO is promising.

It seems that the solutions produced by the IMSPSO algorithm and the BPSO algorithm are not normally distributed. In this study, the Wilcoxon Signed Rank test is chosen and then presented in Table 4. The null hypothesis is that the quality results of the IMSPSO algorithm and the BPSO algorithm are identical. In Table 4, the term of "diff" is the difference of mean value for 50 runs between the IMSPSO algorithm and the BPSO algorithm.

The test statistic for the Wilcoxon Signed Rank Test is W, defined as the smallest of W^+ and W^- which are the sums of the positive and negative ranks, respectively. The level of significance is set at 5% ($p < 0.05$) and sample size is 6. The critical value for two-sided test is 1 and the decision rule is as follows: Reject the null hypothesis if $W \leq 1$.

The test statistic can be obtained using following equation

$$W = \min(W+, W-) \tag{12}$$

where min is a minimum function which is used to find the smaller value between W^+ and W^-. With regard to the Wilcoxon Signed Rank Test, the test statistic is W = 0, where $W^+ = 0$ and $W^- = [2 + 4 + 6 + 3 + 1 + 5] = 21$. Therefore, the null hypothesis can be rejected because $0 < 1$. The result is statistically significant at $p < 0.05$, where the IMSPSO algorithm performs significantly better than the BPSO algorithm.

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. In future, the performance of the IMSPSO algorithm will be investigated to solve other discrete combinatorial optimization problems such as VLSI routing, DNA sequence design, airport gate allocation, and PCB routing.

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

[1] J. Kennedy, and R. Eberhart, "Particle Swarm Optimization," Proceeding of IEEE International Conference on Neural Networks, Dec. 1995, pp. 1942-1948, doi:10.1109/ICNN.1995.488968.

[2] J. Kennedy, and R. C. Eberhart, "A Discrete Binary Version of the Particle Swarm Algorithm," IEEE International Conference on Computational Cybernetics and Simulation, Oct. 1997, pp. 4104-4108, doi:10.1109/ICSMC.1997.637339.

[3] Y-W. Jeong, J-B. Park, S-H. Jang, and K. Y. Lee, "A New Quantum-Inspired Binary PSO for Thermal Unit Commitment Problems,"International Conference on Intelligence on Intelligence System Applications, Nov. 2009, pp. 1-6, doi:10.1109/ISAP.2009.5352869.

[4] S. Lee, S. Soak, S. Oh, W. Pedrycz, and M. Jeon, "Modified Binary Particle Swarm Optimization," Progress in Natural Science, vol. 18, Sep. 2008, pp. 1161-1166, doi:10.1016/j.pnsc.2008.03.018.

[5] M. I. Menhas, M. Fei, L. Wang, and X. Fu, "A Novel Hybrid Binary PSO Algorithm," Advances in Swarm Intelligence, Lecture Notes in Computer Science, vol. 6728, pp. 93-100, doi:10.1007/978-3-642-21515-5_12.S.

[6] A. Modiri, and K. Kiasaleh, "Modification of Real-Number and Binary PSO Algorithms for Accelerated Convergence," IEEE Transactions on Antennas and Propagation, vol. 59, Jan. 2011, pp. 214-224, doi:10.1109/TAP.2010.2090460.

[7] P. Li, D. Xu, Z. Zhou, W-J. Lee, and B. Zhao, "Stochastic Optimal Operation of Microgrid Based on Chaotic Binary Particle Swarm Optimization," IEEE Transactions on Smart Grid, vol. 7, Jan. 2016, pp. 66-73, doi:10.1109/TSG.2015.2431072.

[8] S. A. MirHassani, S. Raeisi, and A. Rahmani, "Quantum Binary Particle Swarm Optimization-Based Algorithm for Solving a Class of Bi-level Competitive Facility Location Problems," Optimization Methods and Software, vol. 30, Aug. 2015, pp. 756-768, doi:10.1080/10556788.2014.973875.

[9] Z. Behesti, S. M. Shamsuddin, and S. Hasan, "Memetic Binary Particle Swarm Optimization for Discrete Optimization Problems," Information Sciences, vol. 299, Apr. 2015, pp. 58-84, doi:10.1016/j.ins.2014.12.016.

[10] M. R. Senouci, D. Bouguettouche, F. Souilah, and A. Mellouk, "Static Wireless Sensor Networks Deployment using an Improved Binary PSO," International Journal of Communication Systems, vol. 29, Mar. 2016, pp. 1026-1041, doi:10.1002/dac.3040.

[11] K. Suresh, and N. Kumarappan, "Hybrid Improved Binary Particle Swarm Optimization Approach for Generation Maintenance Scheduling Problem," Swarm and Evolution Computation, vol. 9, Apr. 2013, pp. 69-89, doi:10.1016/j.swevo.2012.11.003.

[12] K. Fan, W. You, Y. Li, "An Effective Modified Binary Particle Swarm Optimization (mBPSO) Algorithm for Multi-Objective Resource Allocation Problem (MORAP)," Applied Mathematics and Computation, vol. 221, Sep. 2013, pp. 257-267, doi:10.1016/j.amc.2013.06.039.

[13] F. Afshinmanesh, A. Marandi, A. Rahimi-Kian, "A Novel Binary Particle Swarm Optimization Mehod Using Artificial Immune System," The International Conference on Computer as a Tool (EUROCON 2005), Nov. 2005, pp. 217-220, doi:10.1109/EURCON.2005.1629899.

[14] L-Y. Chuang, H-W. Chang, C-J. Tu, and C-H. Yang, "Improved Binary PSO for Feature Selection using Gene Expression Data," Journal Computational Biology and Chemistry, vol. 32, Feb. 2008, pp. 29-38, doi:10.1016/j.compbiolchem.2007.09.005.

[15] L-Y. Chuang, C-J. Hsiao, and C-H. Yang, "An Improved Binary Particle Swarm Optimization with Complementary Distribution

Strategy for Feature Selection," 2009 International Conference on Machine Learning and Computing, 2011, pp. 244-248.

[16] L-Y Chuang, S-W. Tsai, C-H. Yang, "Improved Binary Particle Swarm Optimization using Catfish Effect for Feature Selection," Expert System with Applications, vol. 38, Sep. 2011, pp. 12699-12707, doi:10.1016/j.eswa.2011.04.057.

[17] A. M. Othman, A. A. El-Fergany, and A. Y. Abdelaziz, "Optimal Reconfiguration Comprising Voltage Stability Aspect using Enhanced Binary Particle Swarm Optimization, " Electric Power Components and Systems, Aug. 2015, pp. 1656-1666, doi:10.1080/15325008.2015.1041623.

[18] S. Chakraborty, T. Ito, T. Senjyu, and A. Y. Saber, "Unit Commitment Strategy of Thermal Generators by using Advanced Fuzzy Controlled Binary Particle Swarm Optimization Algorithm," International Journal of Electrical Power and Energy Systems, vol. 43, Dec. 2012, pp. 1072-1080, doi:10.1016/j.ijepes.2012.06.014.

[19] G. Pampara, N. Franken, A.P. Engelbrrecht, "Combining Particle Swarm Optimisation with Angle Modulation to Solve Binary Problems," IEEE Congress on Evolutionary Computation, Sep. 2005, pp. 89-96, doi:10.1109/CEC.2005.

[20] M. S. Mohamad, S. Omatu, S. Deris, M. Yoshioka, A. Abdullah, and Z. Ibrahim, "An Enhancement of Binary Particle Swarm Optimization for Gene Selection in Classifying Cancer Classes," Algorithms for Molecular Biology, vol. 8, Apr. 2013, pp. 1-11, doi:10.1186/1748-7188-8-15.

[21] J. Liu, and X. Fan, "The Analysis and Improvement of Binary Particle Swarm Optimization,"The International Conference on Computational Intelligence and Security, Dec. 2009, pp. 254-258, doi:10.1109/CIS.2009.261.

[22] J. C. Bansai, and K. Deep, "A Modified Binary Particle Swarm Optimization for Knapsack Problems," Applied Mathematics and Computation, vol. 218, Jul. 2012, pp. 11042-11061, doi:10.1016/j.amc.2012.05.001.

[23] S. M. Vieira, L. F. Mendonca, G. J. Farinha, and J. M. C. Sousa, "Modified Binary PSO for Feature Selection using SVM Applied to Mortality Prediction of Septic Patients," Applied Soft Computing, vol. 13, Aug. 2013, pp. 3494-3504, doi:10.1016/j.asoc.2013.03.021.

[24] I. Ibrahim, Z. M. Yusof, S. W. Nawawi, M. A. A. Rahim, K. Khalil, H. Ahmad, and Z. Ibrahim, "A Novel Multi-state Particle Swarm Optimization for Discrete Combinatorial Optimization Problems," The International Conference on Computational Intelligence, Modelling and Simulation, Sept. 2012, pp. 18-23, doi:10.1109/CIMSim.2012.46.

[25] I. Ibrahim, H. Ahmad, Z. Ibrahim, M. F. Mat Jusoh, Z. Md. Yusof, S. W. Nawawi, K. Khalil, and M. A. Abdul Rahim, "Multi-State Particle Swarm Optimization for Discrete Combinatorial Optimization Problem", International Journal of Simulation - Systems, Science and Technology, vol. 15, no. 1, Feb. 2014, pp. 15-25.

TABLE I. PARAMETERS SETTING FOR THE IMSPSO ALGORITHM AND THE BPSO ALGORITHM

Parameter	Algorithm	
	IMSPSO	***BPSO***
Number of trials	50	50
Number of iterations	10000	10000
Number of particles	30	30
Cognitive and social coefficient, c_1 and c_2	2	2
Range of inertia weight, ω	0.9-0.4	0.9-0.4
Random, r_1 and r_2	[0,1]	[0,1]

TABLE II. NUMERICAL RESULTS FOR THE IMSPSO ALGORITHM

Benchmark instance	IMSPSO			
	Minimum	***Mean***	***Maximum***	***Standard deviation***
Burma14	3475	3712.66	4004	121.41
Ulysses16	7011	7765.40	8087	210.28
Ulysses22	9457	9677	9834	154.81
Bays29	3646	3912.42	4107	110.99
Eil51	1143	1219.22	1258	25.82
Berlin52	19980	21686.10	22360	497.20

TABLE III. NUMERICAL RESULTS FOR THE BPSO ALGORITHM

Benchmark instance	BPSO			
	Minimum	***Mean***	***Maximum***	***Standard deviation***
Burma14	3527	3739.88	3829	101.69
Ulysses16	7572	7828.26	8203	163.38
Ulysses22	9565	10084.10	10499	367.60
Bays29	3747	3952.52	4114	96.25
Eil51	1218	1241.80	1264	16.38
Berlin52	21124	21853.20	22393	364.29

TABLE IV. WILCOXON SIGNED RANK TEST

Benchmark instance	IMSPSO	BPSO	Diff	Rank
Burma14	3712.66	3739.88	-27.22	2
Ulysses16	7765.40	7828.26	-62.86	4
Ulysses22	9677	10084.10	-407.1	6
Bays29	3912.42	3952.52	-40.1	3
Eil51	1219.22	1241.80	-22.58	1
Berlin52	21686.10	21853.20	-167.1	5

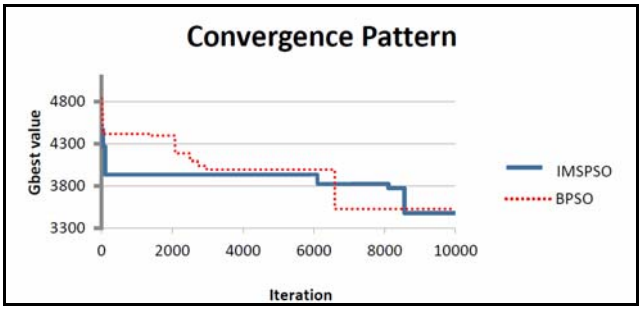

Figure 3. Comparison of the convergence pattern of the IMSPSO and the BPSO for Burma14 benchmark instance.

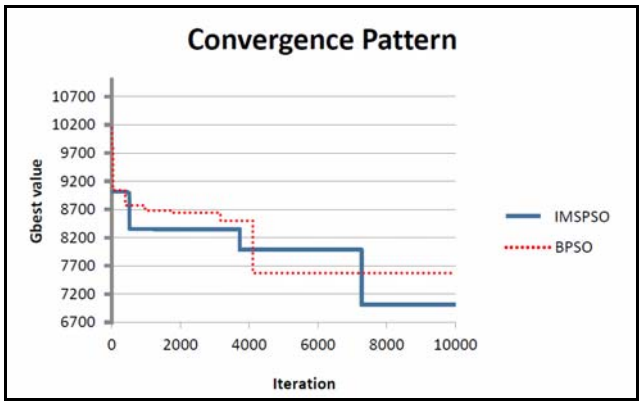

Figure 4. Comparison of the convergence pattern of the IMSPSO and the BPSO for Ulysses16 benchmark instance.

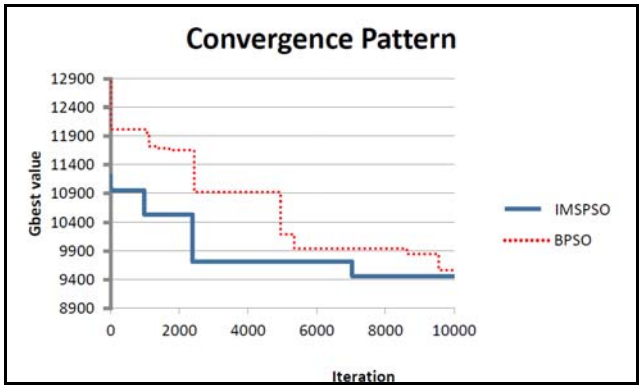

Figure 5. Comparison of the convergence pattern of the IMSPSO and the BPSO for Ulysses29 benchmark instance.

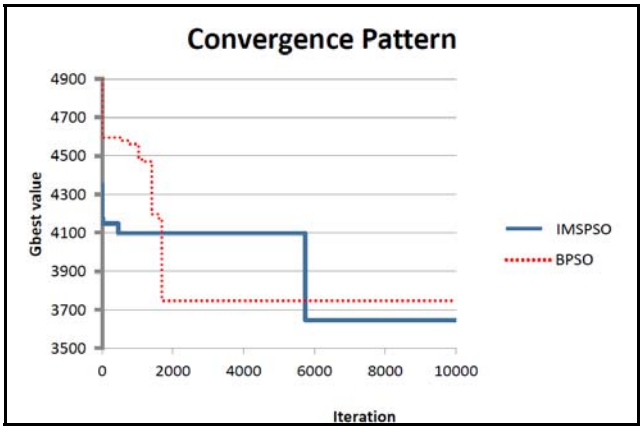

Figure 6. Comparison of the convergence pattern of the IMSPSO and the BPSO for Bays29 benchmark instance.

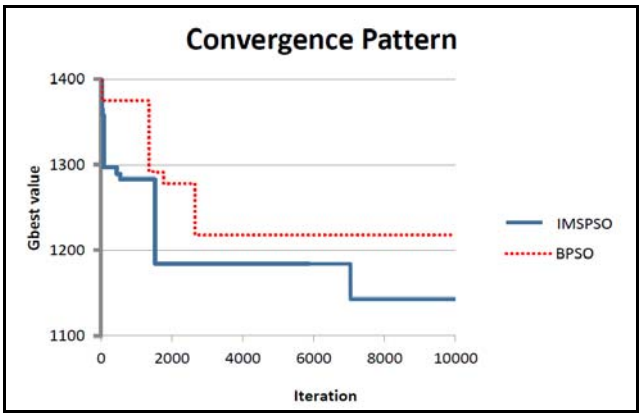

Figure 7. Comparison of the convergence pattern of the IMSPSO and the BPSO for Eil51 benchmark instance.

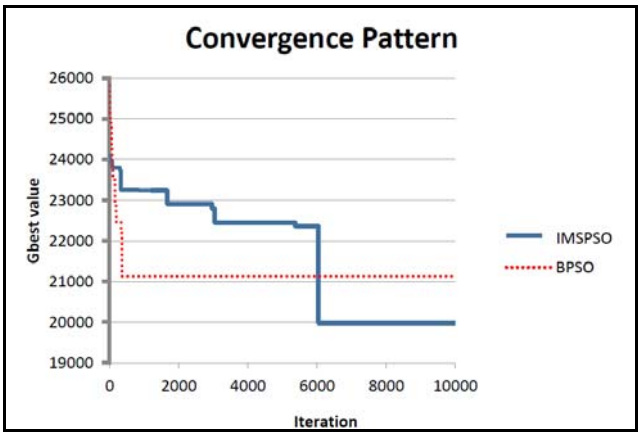

Figure 8. Comparison of the convergence pattern of the IMSPSO and the BPSO for Berlin52 benchmark instance.

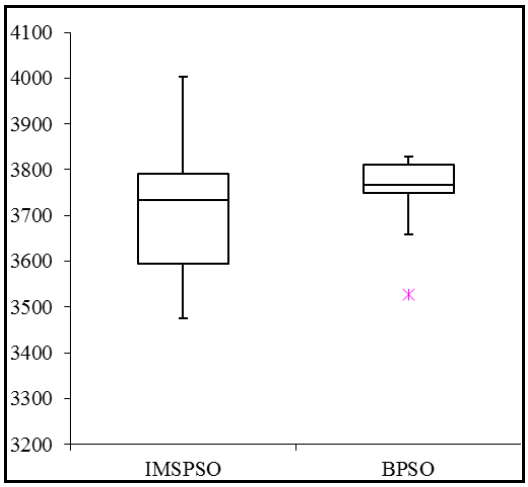

Figure 9. Boxplots of the IMSPSO algorithm and the BPSO algorithm for Burma14 benchmark instances.

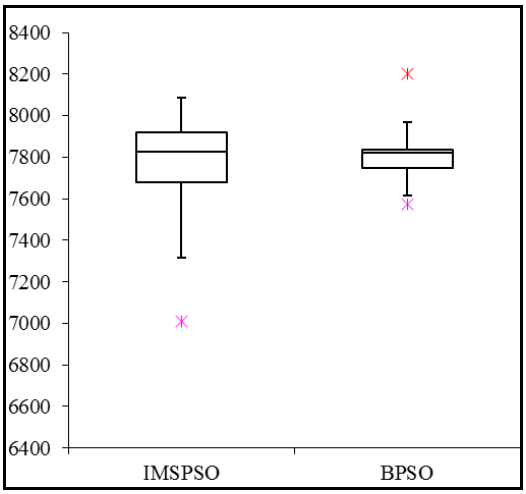

Figure 10. Boxplots of the IMSPSO algorithm and the BPSO algorithm for Ulysses16 benchmark instances.

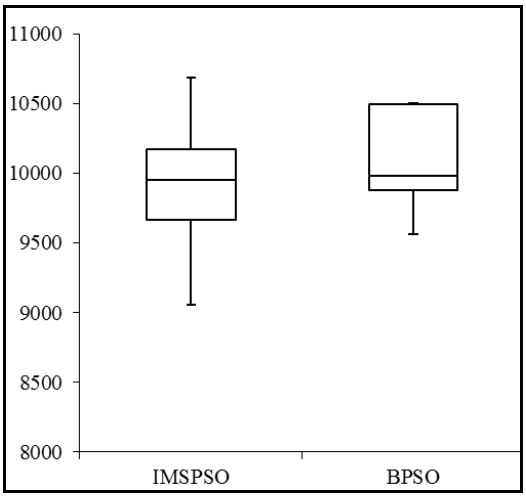

Figure 11. Boxplots of the IMSPSO algorithm and the BPSO algorithm for Ulysses22 benchmark instances.

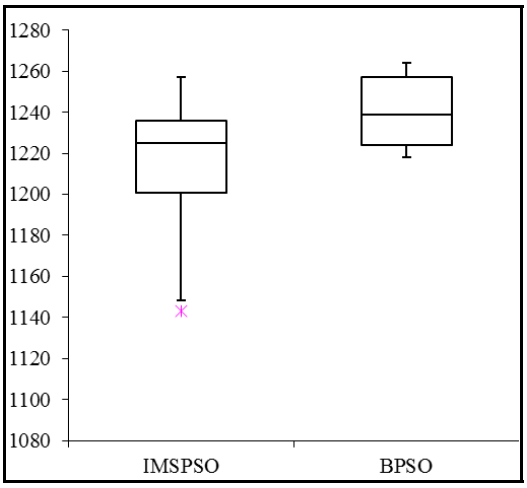

Figure 12. Boxplots of the IMSPSO algorithm and the BPSO algorithm for Bays29 benchmark instances.

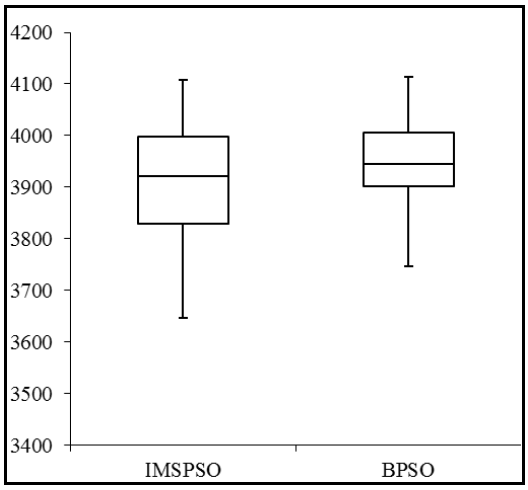

Figure 13. Boxplots of the IMSPSO algorithm and the BPSO algorithm for Eil51 benchmark instances.

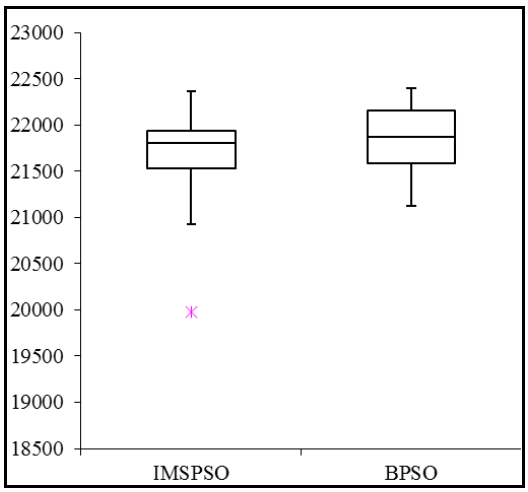

Figure 14. Boxplots of the IMSPSO algorithm and the BPSO algorithm for Berlin52 benchmark instances.