Multi-State Particle Swarm Optimization for Discrete Combinatorial Optimization Problem

Ismail Ibrahim[1], Hamzah Ahmad[1], Zuwairie Ibrahim[1], Mohd Falfazli Mat Jusoh[1], Zulkifli Md. Yusof[2]
[1]Faculty of Electrical and Electronic Engineering
[2]Faculty of Manufacturing Engineering
Universiti Malaysia Pahang
26600 Pekan, Pahang, Malaysia
ismail_xyz@yahoo.com, hamzah@ump.edu.my,
zuwairie@ump.edu.my, mfalfazli@ump.edu.my,
zmdyusof@fke.utm.my

Sophan Wahyudi Nawawi, Kamal Khalil, Muhammad Arif Abdul Rahim
Faculty of Electrical Engineering
Universiti Teknologi Malaysia
81310 UTM Skudai, Johor, Malaysia
sophan@fke.utm.my, kamal@fke.utm.my,
arif@fke.utm.my

***Abstract*—The binary-based algorithms including the binary particle swarm optimization (BPSO) algorithm are proposed to solve discrete optimization problems. Many works have focused on the improvement of the binary-based algorithms. Yet, none of these works have been represented in states. In this paper, by implementing the representation of state in particle swarm optimization (PSO), a variant of PSO called multi-state particle swarm optimization (MSPSO) algorithm is proposed. The proposed algorithm works based on a simplified mechanism of transition between two states. The performance of MSPSO algorithm is emperically compared to BPSO and other two binary-based algorithms on six sets of selected benchmarks instances of traveling salesman problem (TSP). The experimental results showed that the newly introduced approach manage to obtain comparable results, compared to other algorithms in consideration.**

Keywords - particle swarm optimization, multi-state, discrete combinatorial optimization problems

I. INTRODUCTION

Swarm intelligence (SI) is one of artificial intelligence (AI) discipline formed using intelligent multi-particle systems which inspired from the behaviour of insects and animal such as ants, bees, and fish. Swarm intelligence methods have been widely used to solve optimization problems including train scheduling, timetabling, shape optimization, telecommunication network design, and problems from computational biology [1].

Particle swarm optimization (PSO) algorithm is a population-based stochastic optimization technique developed by Kennedy and Eberhart [2]. 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. Due to many optimization problems are defined in the discrete space, research on extending the PSO to solve discrete combinatorial optimization problems (COPs) has become an attractive subject in recent years.

Later, a reworked of the original PSO algorithm known as binary particle swarm optimization (BPSO) algorithm based on the binary coding scheme has been developed to allow PSO algorithm to operate in discrete binary variables [3]. At present, a lot of proposals have been presented to improve the BPSO algorithm in term of convergence speed [4-7], stagnation in local optimum [7-14], complexity of representation of particle, and expensive computational time [7], [15], and local exploration [16].

To the best of our knowledge, no research works have been reported on the use of state representation to solve discrete optimization problems. This paper is the extention of our previous work called multi-state PSO [17] that represents each particle's vector as a state and the velocity as radius of a circle subjected to the current state. As a test, this algorithm was applied to the Traveling Salesman Problem (TSP). The results were promising and in several occasions the proposed algorithm managed to obtain better performance compared with the original BPSO.

The rest of the paper is organized as follows. Section 2 explains some information on optimization problems. In Section 3, we explain the original particle swarm optimization (PSO). In Section 4, the concept of multi-state PSO (MSPSO) is elaborated. In Section 5, implementation of the proposed MSPSO to solve Travelling Salesman Problem (TSP) is described. Experimental results and discussions are presented in Section 6. Finally, in Section 7 conclusions of the research are given.

II. OPTIMIZATION PROBLEM

Generally, any optimization problem P can be described as a triple (Z, Ω, f), where:

The preliminary version of this paper has been presented at the 2012 Fourth International Conference on Computational Intelligence, Modelling and Simulation

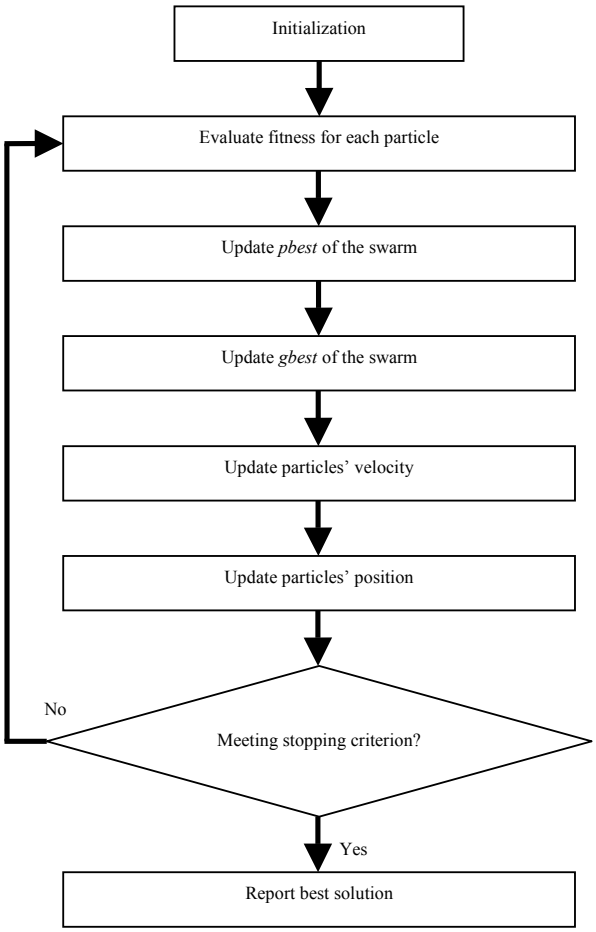

Fig. 1 : General principle of PSO.

- Z is the search space defined over a finite set of decision variables $X = \{x_1,...,x_n\}$. Discrete domain variables $E_1,...,E_n$ are used when dealing with discrete optimization or combinatorial optimization problem. Concurrently, continuous domain variables are used in solving continuous optimization problem. Mixed variable problems also exist.
- Ω is a set of constraints among variables.
- $f : Z \to I\Re^+$ is the objective function that appoints a positive cost value to each element or solution of Z.

The goal of an optimization algorithm is to find a solution $s \in Z$ such that $f(z) \le f(z')$, $\forall z' \in Z$ for the case of minimization of the objective function, or $f(z) \ge f(z')$, $\forall z' \in Z$ for the case of maximization of the objective function. In optimizing several objective functions at the same time, multi-objective optimization (MOO) algorithm is used.

In the context of the combinatorial optimization problem (COP) [18], the set of all possible feasible assignments is formulated as in Eq. (1).

$$Z = \{s = \{(x_1, q_1),...,(x_n, q_n)\} \mid q_i \in E_i\} \quad (1)$$

Since S satisfied all the constraints, each set can be seen as a candidate solution. One of the most common constraints is that a value cannot be repeated in a solution. For example, two or more variables may not have same value.

In addressing optimization problems, many algorithms have been proposed, varying both on the formulation of the search space, and on the algorithm used to search the space. In combinatorial optimization (CO), these algorithms can be separated as either complete or approximate algorithms. Complete algorithms are assured to find an optimal solution in bounded time for every finite size instance of a CO problem [19, 20]. However, for CO problems that are *NP*-hard [21], no polynomial time algorithm exists. Therefore, for the worst case, complete methods might need exponential computational time. Whereas complete methods lead to the existence of expensive computational times for practical purposes, approximate methods as in AI-based algorithms lead to significant reduction of computational times, which by its nature sacrifices the guarantee of finding optimal solutions. Other advantages derived from the approximate methods, such as the implementation is easier than classical gradient-based techniques, and gradient information is no more required.

III. PARTICLE SWARM OPTIMIZATION

In the original PSO algorithm, an optimal or good enough solution is found by simulating social behavior of bird flocking. The PSO algorithm consist of a group of individuals named particles which encode the possible solutions to the optimization problem using their positions. The group can attain the solution effectively by using the common information of the group and the information owned by the particle itself, which each particle share its current position to neighboring particles. Using this information, each particle compares its current position with the best position found by its neighbors so far.

Fig. 1 portrays the flow chart of the PSO algorithm. Consider the following minimization problem: there are *I*-particles flying around in a *D*-dimensional search space, where their position, $s_i(d)$ (i = 1,2,…,I; d = 1,2,…,D), represent the possible solutions. In the initialization stage, all particles are randomly positioned in the search space. All particles are then assigned with random velocity $v_i(k,d)$, where k represents the iteration number. Next, the objective fitness $F_i(k)$, for each particle is evaluated by calculating the objective functions with respect to $s_i(k)$. Each particle's best position is $pbest_i(k)$ then initialized to its current position. The global best among the all $pbest_i(k)$ is called $gbest(k)$, is chosen as the swarm's best position, as in Eq. (2),

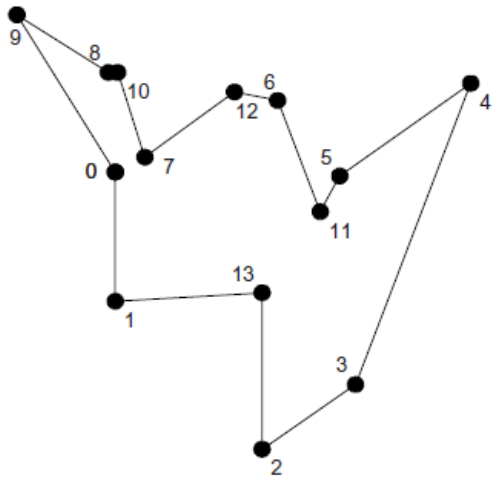

Fig. 2 : An example of Travelling Salesman Problem (TSP). The name of this benchmark instance is Burma14.

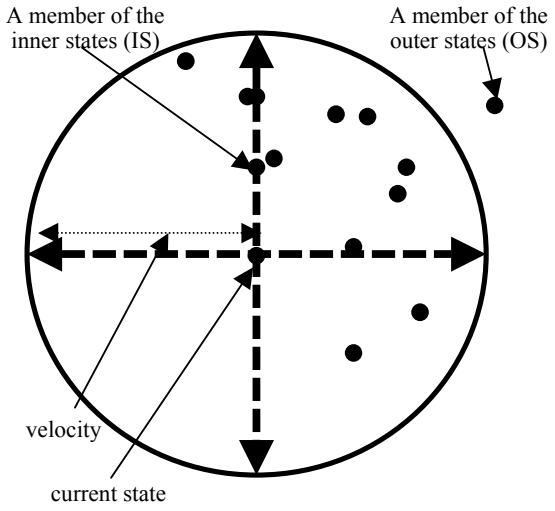

Fig. 3 : The illustration of the multi-state representation in the MSPSO for Burma14 benchmark instance. This representation applies to each dimension of each particle.

$$gbest = \left\{ pbest_i \in S \mid f(pbest_i) = \min f(\forall\ pbest_i \in S) \right\} \quad (2)$$

where S is the swarm of particles. Subsequently, the algorithm iterates until the stopping condition is met, either the maximum number of iteration is reached or a particular amount of error is obtained.

Each iteration updates each particle's velocity and position using Eq. (3) and Eq. (4), respectively.

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

$$s_i(k+1,d) = s_i(k,d) + v_i(k+1,d) \quad (4)$$

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 used to control the impact of the previous history of velocities on the current velocity of each particle. After updating the velocity and position, $F_i(k)$ for each particle is calculated again. The $pbest_i(k)$ is then updated by a more optimal, obtained either from the new position of the i^{th} particle or $pbest_i(k)$. The $gbest(k)$ is also updated by the most optimal $pbest_i(k)$ of all the particles, as denoted in (2). Finally, the optimum solution of the problem represented by $gbest(k)$ is yielded when the stopping condition is met.

IV. The Proposed Multi-State Particle Swarm Optimization (MSPSO)

The proposed multi-state Particle Swarm Optimization (MSPSO) for solving discrete optimization problems is explained in this section. MSPSO practices similar general principal of original PSO. However, a few modifications have been made on MSPSO in term of exploitation of particle's velocity and a mechanism of state transition.

Each particle's vector or dimension in MSPSO is represented as state; neither continuous nor discrete value. To elaborate the multi-state representation, Burma14 benchmark instance of Travelling Salesman Problem (TSP) as shown in Fig. 2 is used. All the cities in Burma14 can be represented as a collective of states, as presented in Fig. 3, in which the states are represented by the small black circle. 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 MSPSO take places after *pbest* and *gbest* of all particles are updated.

The calculation of velocity value in MSPSO is different compared with the original PSO due to the multi-state representation does not allow numerical calculation between $pbest_i(k,d)$ and $s_i(k,d)$, and also $gbest(k,d)$ and $\mathbf{s}_i(k,d)$ in the cognitive and social component of velocity calculation, respectively. Referring 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 MSPSO, 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. To accommodate the calculation of velocity in MSPSO, a *cost* (.) function is introduced and incorporated as follows:

$$v_i(k+1,d) = \omega v_i(k,d) + c_1 r_1 C(pbest_i(k,d), s_i(k,d)) \quad (5)$$

$$+ c_2 r_2 C(gbest(k,d), s_i(k,d))$$

The cost can be defined as the distance and time in Traveling Salesman Problem (TSP) and Assembly Sequence Planning (ASP), respectively [22-23]. Hence, the cost between two states is a positive number given by $C\left(s_j(t,d), s_i(t,d)\right)$.

In MSPSO, once the velocity is updated, the process of updating position or called state to obtain the next state for each dimension of each particle is executed. Given the current state as a centroid and the updated velocity value as a radius, a circle is built. Any state that is located in the area of the circle is defined as a member of inner states (IS). Any state that is located outside of the circle is defined as a member of outer states (OS). Given a set of j IS members $M_i(t,d) = \left(M_{i_1}(t,d),...,M_{i_j}(t,d)\right)$. A next state is then selected randomly among the IS using Eq. (6).

$$s_i(t+1,d) = random\left(M_{i_1}(t,d),...,M_{i_j}(t,d)\right) \quad (6)$$

V. Solving Symmetric Travelling Salesman Problem with MSPSO

In TSP, a salesman finding the shortest tour in which all cities are visited such that no city is visited more than once and the salesman returns to the initial visited city at the end of the tour. An instance of TSP is called symmetric if for all pairs i and j, the distance of two adjacent cities, c_i and c_j, is equivalent to the distance of two adjacent cities, c_j and c_i. Otherwise, it is called asymmetric.

The TSP is belongs to the class of NP-hard combinatorial optimization problems. The TSP has become a standard case study for any new proposed optimization algorithm, such as in Honey Bees Mating Optimization [24] and Parallel Immune [25]. In fact, various problems including path-finding, routing, and scheduling can be modeled as a TSP [26].

A. Limitation of MSPSO

In order to update state in MSPSO, a random function in (6) is applied. The equation may lead to the existence of repeated state in an updated solution. 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. Let say the updated solution is a 14-dimensional vector $\{s_4, s_7, s_8, s_{11}, s_{13}, s_8, s_5, s_{12}, s_{13}, s_{10}, s_1, s_8, s_{12}, s_9\}$ as illustrated in Fig. 4. Obviously, the state in the 3rd, 5th, 6th, 8th, 9th, 12th, and 13th dimension occurs more than once, making the updated solution infeasible for the TSP. In particular, the updated state in the 5th and 9th dimension of the updated solution is x_{13}, which are identical.

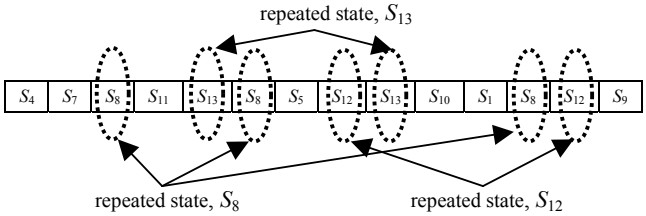

Fig. 4 : An example of a solution with repeated states.

B. Producing a Feasible Solution from an Infeasible Solution

In order to overcome the limitation of MSPSO in solving TSP, an additional procedure is added after updating velocity and state to obtain a feasible solution from an infeasible solution. Fig. 5 illustrates the principle of this procedure. Initially, a solution of a particle is read. Also, a blank solution of n dimensions is created and an archive is initialized. The archive is then sorted in natural order. The solution that has been read is then checked, whether the solution is feasible or not. If the solution is feasible, the process is stopped. Otherwise, a feasible solution must be generated from the infeasible solution. To generate the feasible solution, the current dimension d of the solution with repeated state is then checked whether it has exceeded the maximum number of dimension n or not. If the maximum number of dimension is not exceeded, the state in the current dimension d of the solution with repeated state is read. Otherwise, the process is stopped. If the state in the current dimension d of the solution with repeated state is read, the state is checked whether the state still exists in the archive. If the state still exists in the archive, the state is put in the current dimension d of the solution with unrepeated state. Otherwise, a state is randomly chosen from the archive at first and the state is then put in the current dimension d of the solution with unrepeated state. Next, the state chosen is removed from the archive. Finally, a feasible solution with unrepeated state is produced when all dimensions in the solution with repeated state has been checked. This procedure is applied to the solution of each particle.

VI. Experiment, Results, and Discussion

This section presents comparisons between the proposed MSPSO algorithm and some works in the literature, namely the original binary PSO (BPSO) [2], the improved binary PSO1 (IBPSO1)[15], and the improved binary PSO2 (IBPSO2)[14], based on six sets of TSP benchmark instances taken from TSPlib (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/), namely:

- Burma14 – A small size TSP problem of 14 cities. The optimal route length is 3323.
- Ulysses16 – A small size TSP problem of 16 cities. The optimal route length is 6859.
- Ulysses22 – A medium size TSP problem of 22 cities. The optimal route length is 7013.

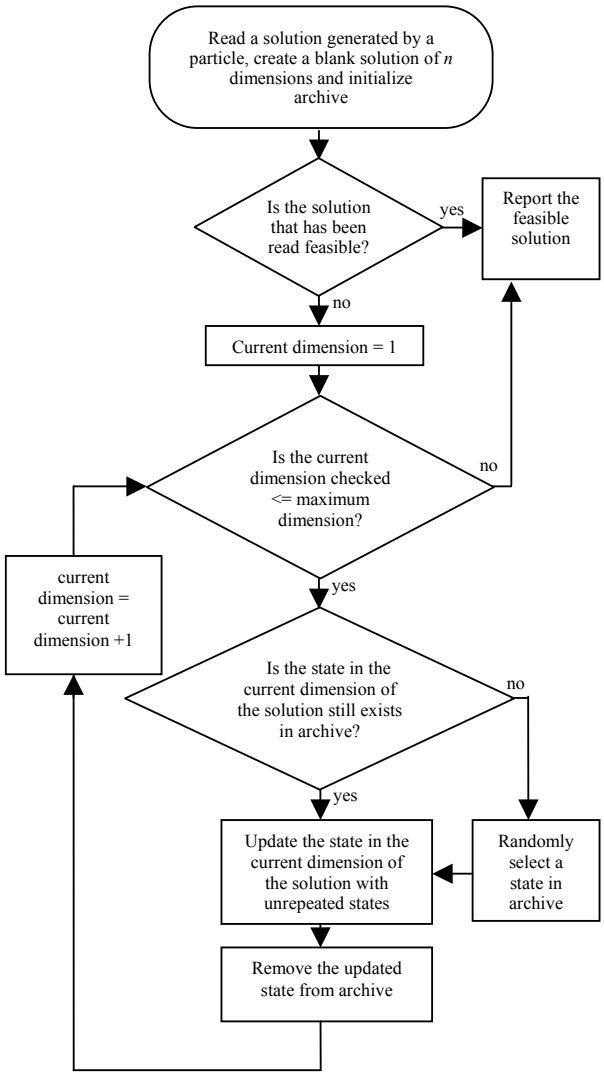

Fig. 5 : The procedure of producing feasible solution from infeasible solution.

TABLE I. NUMERICAL RESULTS OF THE MSPSO, BPSO, IBPSO1, AND IBPSO2 ON EACH BENCHMARK INSTANCE

Parameter	Algorithm			
	MSPSO	***BPSO***	***IBPSO1***	***IBPSO2***
Number of trial	50	50	50	50
Number of iteration	10000	10000	10000	10000
Number of particle	30	30	30	30
c_1 and c_2	2	2	2	1.49618
r_1 and r_2	[0,1]	[0,1]	[0.1]	[0.1]
ω initial	0.9	0.9	1.2	0.729844
ω final	0.4	0.4	0.4	0.729844

- Bays29 – A medium size TSP problem of 29 cities. The optimal route length is 2020.
- Eil51 – The second biggest TSP problem of 51 cities. The optimal route length is 426.
- Berlin52 – The biggest TSP problem of 52 cities. The optimal route length is 7542.

Due to sensitivity of algorithmic parameters, c_1, c_2, and ω for the original binary PSO (BPSO) [2], the improved binary PSO1 (IBPSO1) [15], and the improved binary PSO2 (IBPSO2) [14] were chosen according to their reported values. Meanwhile MSPSO used similar c_1, c_2, and ω value as in the original BPSO. In the initialization stage, all particles are randomly positioned in the search space. The particles are then assigned with velocity that is equal to zero. 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 average (mean), minimum (min) and maximum (max) of fitness values of 50 solutions, and the standard deviation (SD) are recorded. The quality of results of MSPSO against BPSO, IBPSO1, and IBPSO2 for 50 trials for each benchmark instance is presented in Table 2, 3, 4, 5, 6, and 7, respectively. Based on the results given in Table 2, MSPSO outperformed IBPSO2 in obtaining the quality of results. In Table 3, MSPSO has been outperformed by three other binary-based algorithms in obtaining the quality of results. In Table 4, MSPSO outperformed BPSO and IBPSO2 in obtaining the quality of results. In Table 5, MSPSO outperformed BPSO and IBPSO1 in obtaining the quality of results. In Table 6, MSPSO outperformed BPSO in obtaining the quality of results. In Table 7, MSPSO has been outperformed by all three algorithms in obtaining the quality of results.

A comparison between the performances, regarding the frequency of time for MSPSO performed better, similar, and worst against BPSO, IBPSO1, and IBPSO2 for each benchmark instance is shown in Table 8, 9, 10, 11, 12, and 13. 50 trials are evaluated for this purpose. In Table 8, the solutions obtained by the MSPSO algorithm are 30, 15, and 32 times better than the solutions presented by the BPSO algorithm, the IBPSO1 algorithm, and the IBPSO2 algorithm, respectively. In Table 8, the solutions obtained by the MSPSO are 20, 10, and 26 times better than the solutions presented by the BPSO algorithm, the IBPSO1 algorithm, and the IBPSO2 algorithm, respectively. In Table 9, the solutions obtained by the MSPSO are 19, 1, and 25 times better than the solutions presented by the BPSO algorithm, the IBPSO1 algorithm, and the IBPSO2 algorithm, respectively. In Table 10, the solutions obtained by the MSPSO algorithm are 30, 15, and 32 times better than the solutions presented by the BPSO algorithm, the IBPSO1 algorithm, and the IBPSO2 algorithm, respectively. In Table 8, the solutions obtained by the MSPSO algorithm are 20, 18, and 39 times better than the solutions presented by the

BPSO algorithm, the IBPSO1 algorithm, and the IBPSO2 algorithm, respectively. In Table 11, the solutions obtained by the MSPSO are 28, 35, and 22 times better than the solutions presented by the BPSO algorithm, the IBPSO1 algorithm, and the IBPSO2 algorithm, respectively. In Table 12, the solutions obtained by the MSPSO are 33, 17, and 20 times better than the solutions presented by the BPSO algorithm, the IBPSO1 algorithm, and the IBPSO2 algorithm, respectively. In Table 13, the solutions obtained by the MSPSO are 18, 21, and 24 times better than the solutions presented by the BPSO algorithm, the IBPSO1 algorithm, and the IBPSO2 algorithm, respectively. Fig. 6, 7, 8, 9, 10, and 11 present the convergence pattern of the best solution found by MSPSO, BPSO, IBPSO1, and IBPSO2 for each benchmark instance.

There are three differences between MSPSO and BPSO in obtaining an updated solution. Firstly, when applying BPSO, the current solution and next solution are formed by a binary string. Meanwhile, in MSPSO, the current and next solutions are formed by a multi-state string. Secondly, in BPSO, based on Burma14 benchmark instance, the binary string used 56-dimensional vectors to represent its solution since four bits are required to represent a city. On the other hand, in MSPSO, the string just required 14-dimensional vectors, as many as the number of cities involved in the instance. Thirdly, for each dimension of current solution in BPSO, new velocity value is converted into a probability function to yield an updated position (next position). Meanwhile, in MSPSO, the updated position (next state) of each dimension is updated subjected to the new velocity value, which is used as radius of a circle. The centroid of the circle represents the current state and any element located in area of the circle can be selected as the next state.

Fig. 12, 13, and 14 present the characteristic of velocity of MSPSO for a dimension of a particle for a particular iteration. Similar characteristic can be observed to all dimensions of all particles. As mentioned, the initial velocity is equal to zero. For the first iteration, the updated velocity becomes bigger as presented in Fig. 12. As the iteration increases, the velocity becomes smaller, as illustrated in Fig. 13 and 14. At the end of iteration, the velocity becomes smaller and MSPSO converges. The exploitation of velocity in MSPSO reduces the number of candidate states as iteration increases and hence, a good balance between exploration and exploitation can be achieved. An example of the pattern of the velocity value of MSPSO over iteration based on one dimension for Burma14 benchmark instance is shown in Fig. 15. This figure proves that the velocity value decreases over iteration.

VII. CONCLUSION

In this study, multi-state Particle Swarm Optimization (MSPSO) has been proposed to solve discrete optimization problem, particularly in combinatorial optimization problem. In order to solve this problem, state-based representation is introduced, substituting binary-based representation proposed in the BPSO, the IBPSO1, and IBPSO2. To evaluate the performance of the proposed MSPSO and other algorithms, six of TSP benchmark instance, which is one of the most popular discrete optimization problem were used. For this problem, each algorithm was executed to find the shortest route. Experimental results obtained from the six database used showed that the MSPSO managed to obtain comparable results, compared to other algorithms in consideration. Beside, the representation based on state has made the particles just need fewer dimensions to be evolved for the TSP, compared to the binary-based algorithms in consideration.

ACKNOWLEDGMENT

This work is supported by Research University Grant (Q.J130000.2623.09J03) from Universiti Teknologi Malaysia and Fundamental Research Grant Scheme (FRGS) (RDU140132) from Universiti Malaysia Pahang. The first author is thankful to Universiti Malaysia Pahang (UMP) for granting him an opportunity to further his study in PhD degree.

REFERENCES

[1] J. Kennedy, and R. Eberhart, "Particle swarm optimization," Proceeding of IEEE International Conference on Neural Networks, Dec. 1995, pp.1942-1948, doi:10:10.1109/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] S. Lee, S. Soak, S. Oh, W. Pedrycz, M. Jeon," Modified Binary Particle Swarm Optimization," Progress in Natural Science, Sep. 2008, pp.1161-1166, doi:10.1016/j.pnsc.2008.03.018.

[4] 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

[5] 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.

[6] 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.

[7] 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.

[8] L. Wang and J. Yu, "Fault Feature Selection Based on Modified Binary PSO with Mutation and Its Application in Chemical Process Fault Diagnosis," Proceedings of the First international conference on Advances in Natural Computation (ICNC 05), 2005, pp. 832-840, doi:10.1007/11539902_102.

[9] L. Yuan and Z-d. Zhao, "A Modified Binary Particle Swarm Optimization Algorithm for Permutation Flow Shop Problem," Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Aug. 2007, pp. 902-907, doi: 10.1109/ICNLC.2007.4370270.

[10] 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.

REFERENCES CONTINUE AFTER TABLES AND FIGURES.

TABLE II. NUMERICAL RESULTS OF MSPSO AGAINST BPSO, IBPSO1, AND IBPSO2 ON BURMA14 BENCHMARK INSTANCE

Value	Algorithm			
	MSPSO	***BPSO***	***IBPSO1***	***IBPSO2***
Best	3411	3527	3359	3506
Worst	3955	3829	3809	3956
Average	3753.26	3739.88	3633.80	3771.46
Standard Deviation	144.00	101.69	134.68	108.27

TABLE III. NUMERICAL RESULTS OF MSPSO AGAINST BPSO, IBPSO1, AND IBPSO2 ON ULYSSES16 BENCHMARK INSTANCE

Value	Algorithm			
	MSPSO	***BPSO***	***IBPSO1***	***IBPSO2***
Best	7499	7572	6747	7374
Worst	8204	8203	7852	8283
Average	7913.66	7828.26	7213.18	7908.82
Standard Deviation	200.64	163.38	274.16	201.96

TABLE IV. NUMERICAL RESULTS OF MSPSO AGAINST BPSO, IBPSO1, AND IBPSO2 ON ULYSSES22 BENCHMARK INSTANCE

Value	Algorithm			
	MSPSO	***BPSO***	***IBPSO1***	***IBPSO2***
Best	9603	9565	8985	9167
Worst	10297	10499	10316	10669
Average	9907.80	10084.40	9674.48	10152.40
Standard Deviation	234.09	367.60	364.38	293.96

TABLE V. NUMERICAL RESULTS OF MSPSO AGAINST BPSO, IBPSO1, AND IBPSO2 ON BAYS29 BENCHMARK INSTANCE

Value	Algorithm			
	MSPSO	***BPSO***	***IBPSO1***	***IBPSO2***
Best	3669	3747	3881	3635
Worst	4126	4114	4091	4138
Average	3950.02	3952.52	4001	3942.02
Standard Deviation	99.78	96.25	74.57	109.02

TABLE VI. NUMERICAL RESULTS OF MSPSO AGAINST BPSO, IBPSO1, AND IBPSO2 ON EIL51 BENCHMARK INSTANCE

Value	Algorithm			
	MSPSO	***BPSO***	***IBPSO1***	***IBPSO2***
Best	1184	1218	1146	1137
Worst	1266	1264	1239	1266
Average	1226.64	1241.80	1204.60	1223.82
Standard Deviation	20.85	16.38	30.95	26.76

TABLE VII. NUMERICAL RESULTS OF MSPSO AGAINST BPSO, IBPSO1, AND IBPSO2 ON BERLIN52 BENCHMARK INSTANCE

Value	Algorithm			
	MSPSO	***BPSO***	***IBPSO1***	***IBPSO2***
Best	21261	21124	20717	20921
Worst	22606	22393	22548	22704
Average	22021.40	21853.20	21911.90	21884.90
Standard Deviation	352.81	364.29	442.48	489.23

TABLE VIII. THE NUMBER OF TIMES OF THE MSPSO ALGORITHM PERFORMS BETTER, SIMILAR, OR WORSE COMPARED TO THE BPSO, IBPSO1, AND IBPSO2 ALGORITHM ON BURMA14 BENCHMARK INSTANCE

Status	Algorithm		
	BPSO	***IBPSO1***	***IBPSO2***
Beter	20	10	26
Similar	0	0	1
Worse	30	40	23

TABLE IX. THE NUMBER OF TIMES OF THE MSPSO ALGORITHM PERFORMS BETTER, SIMILAR, OR WORSE COMPARED TO THE BPSO, IBPSO1, AND IBPSO2 ALGORITHM ON ULYSSES16 BENCHMARK INSTANCE

Status	Algorithm		
	BPSO	***IBPSO1***	***IBPSO2***
Beter	19	1	25
Similar	0	0	0
Worse	31	49	25

TABLE X. THE NUMBER OF TIMES OF THE MSPSO ALGORITHM PERFORMS BETTER, SIMILAR, OR WORSE COMPARED TO THE BPSO, IBPSO1, AND IBPSO2 ALGORITHM ON ULYSSES22 BENCHMARK INSTANCE

Status	Algorithm		
	BPSO	***IBPSO1***	***IBPSO2***
Beter	20	18	39
Similar	0	0	0
Worse	30	32	11

TABLE XI. THE NUMBER OF TIMES OF THE MSPSO ALGORITHM PERFORMS BETTER, SIMILAR, OR WORSE COMPARED TO THE BPSO, IBPSO1, AND IBPSO2 ALGORITHM ON BAYS29 BENCHMARK INSTANCE

Status	Algorithm		
	BPSO	***IBPSO1***	***IBPSO2***
Beter	28	35	22
Similar	0	0	1
Worse	22	15	27

TABLE XII. THE NUMBER OF TIMES OF THE MSPSO ALGORITHM PERFORMS BETTER, SIMILAR, OR WORSE COMPARED TO THE BPSO, IBPSO1, AND IBPSO2 ALGORITHM ON EIL51 BENCHMARK INSTANCE

Status	Algorithm		
	BPSO	***IBPSO1***	***IBPSO2***
Beter	33	17	20
Similar	1	1	3
Worse	16	32	27

TABLE XIII. THE NUMBER OF TIMES OF THE MSPSO ALGORITHM PERFORMS BETTER, SIMILAR, OR WORSE COMPARED TO THE BPSO, IBPSO1, AND IBPSO2 ALGORITHM ON BERLIN52 BENCHMARK INSTANCE

Status	Algorithm		
	BPSO	***IBPSO1***	***IBPSO2***
Beter	18	21	24
Similar	0	0	0
Worse	32	29	26

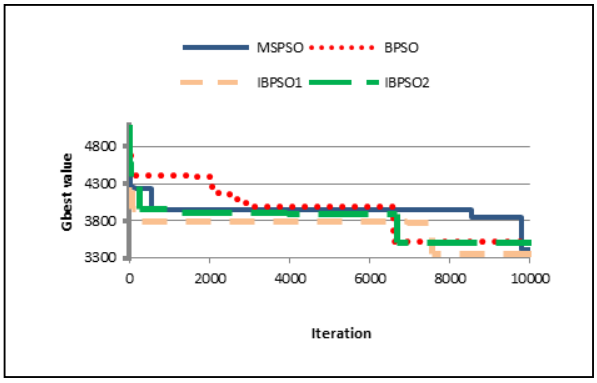

Fig. 6 : Comparison of the convergence pattern of the MSPSO, BPSO, IBPSO1, and IBPSO2 for Burma14 benchmark instance.

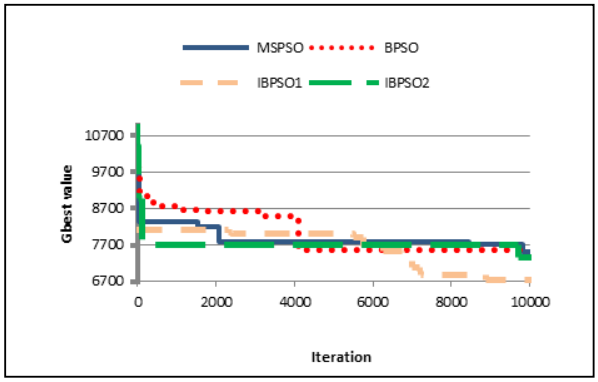

Fig. 7 : Comparison of the convergence pattern of the MSPSO, BPSO, IBPSO1, and IBPSO2 for Ulysses16 benchmark instance.

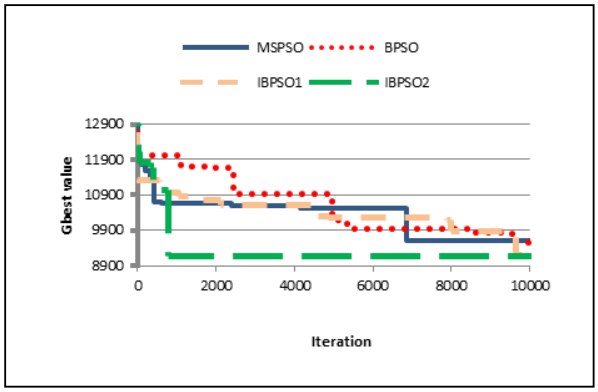

Fig. 8 : Comparison of the convergence pattern of the MSPSO, BPSO, IBPSO1, and IBPSO2 for Ulysses22 benchmark instance.

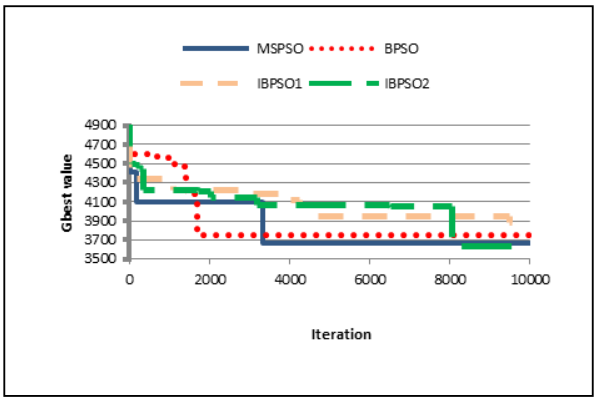

Fig. 9 : Comparison of the convergence pattern of the MSPSO, BPSO, IBPSO1, and IBPSO2 for Bays29 benchmark instance.

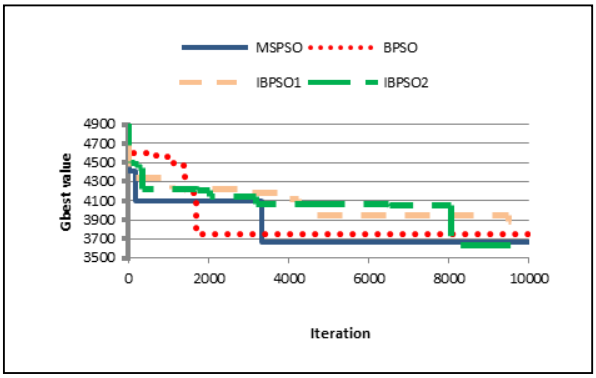

Fig. 10 : Comparison of the convergence pattern of the MSPSO, BPSO, IBPSO1, and IBPSO2 for Eil51 benchmark instance.

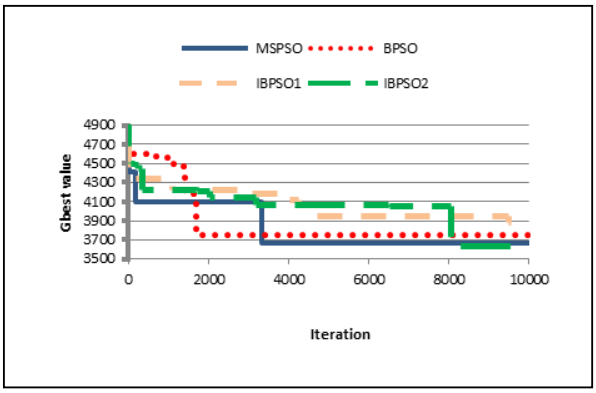

Fig. 11 : Comparison of the convergence pattern of the MSPSO, BPSO, IBPSO1, and IBPSO2 for Berlin52 benchmark instance.

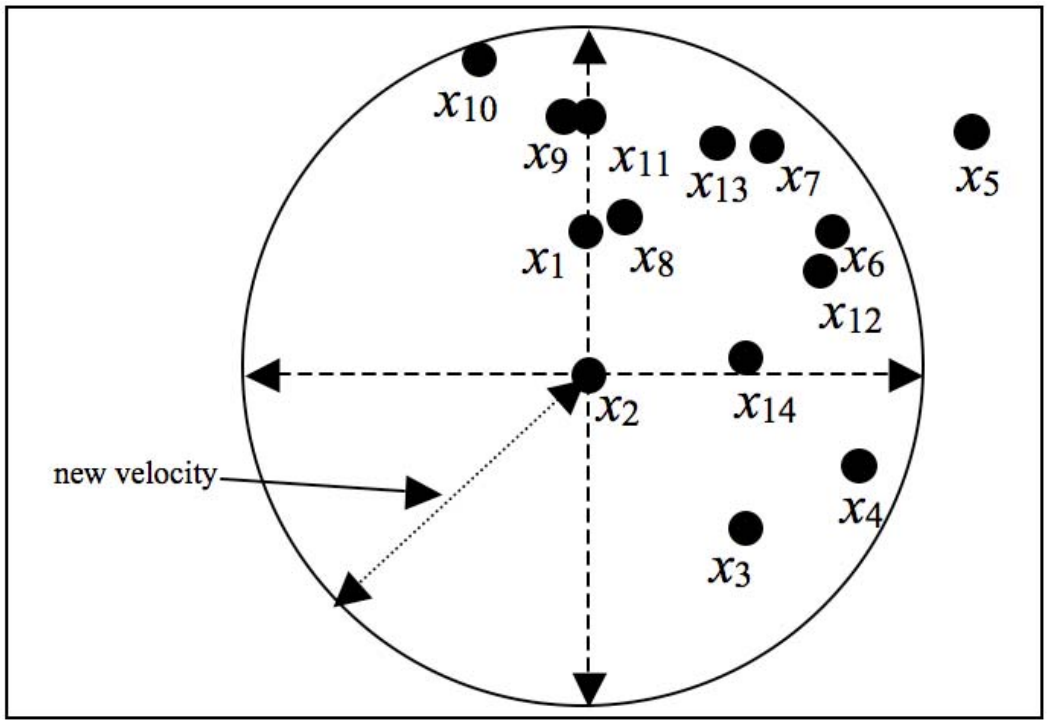

Fig. 12 : The characteristic of velocity update of MSPSO in a search space at t^{th} iteration.

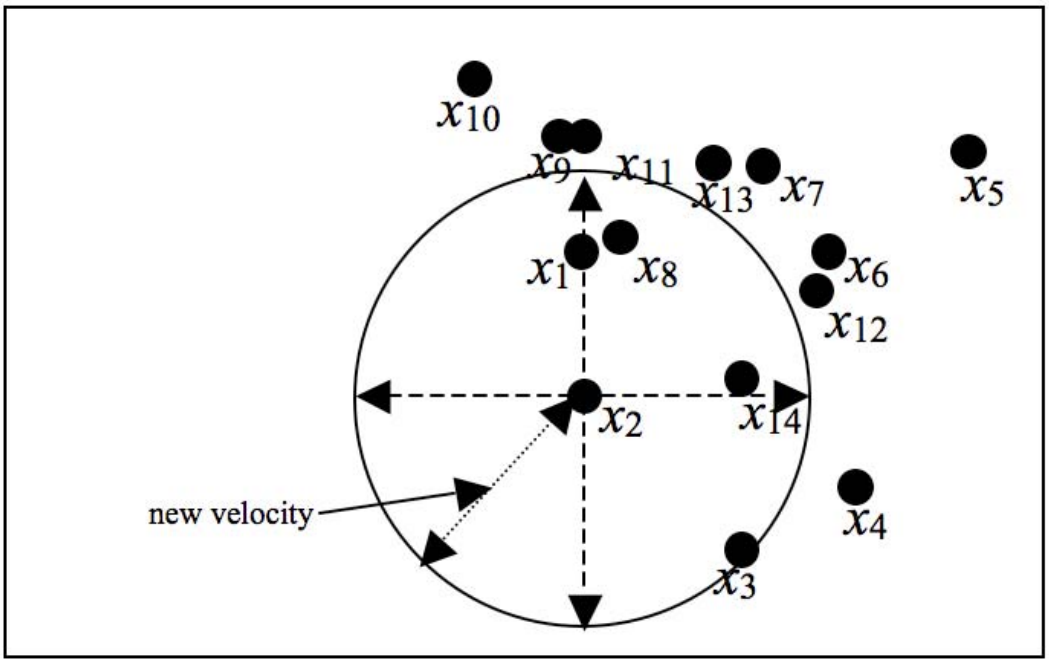

Fig. 13 : The characteristic of velocity update of MSPSO in a search space at $t{+}{+}^{th}$ iteration.

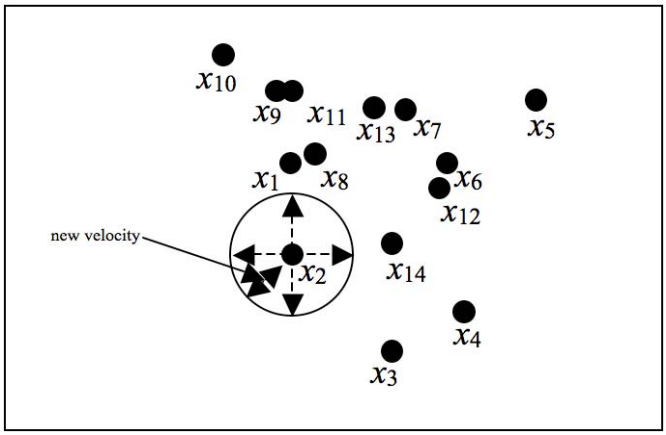

Fig. 14 : The characteristic of velocity update of MSPSO in a search space at near the end of maximum iteration.

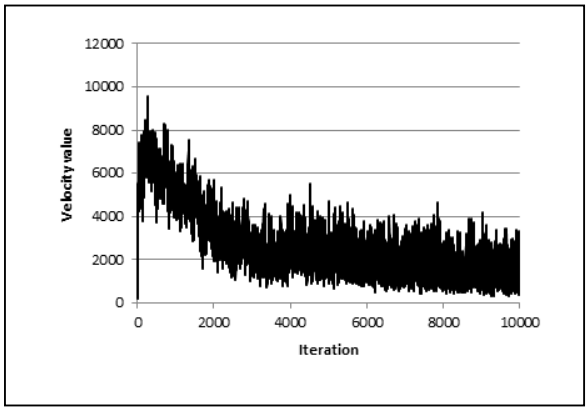

Fig. 14 : The pattern of velocity in the search space of MSPSO over 10000 iterations for each dimension for Burma14 benchmark instance.

[11] X. Yuan, H. Nie, A. Su, L. Wang, and Y. Yuan, "An Improved Binary Particle Swarm Optimization for Unit Commitment Problem," Expert System Application, vol. 36, May. 2009, pp. 8049-8055, doi:10.1016/j.eswa.2008.10.047.

[12] 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.

[13] 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.

[14] 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.2005.1554671.

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

[16] C. Blum and X. Li, "Swarm Intelligence in Optimization," Natural Computing, Springer Berlin Heidelberg, pp. 43-85, doi: 10.1007/978-3-540-74089-6_2 .

[17] 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," 2012 Fourth International Conference on Computational Intelligence, 2012,pp. 18-23, doi: 10.1109/CIMSim.2012.46.

[18] C. Blum and A. Roli, "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," ACM Computing Surveys(CSUR), vol. 35, Sep 2003, pp. 268–308, doi: 10.1145/937503.937505.

[19] C. H. Papadimitriou and K. Steiglitz, "Combinatorial Optimization—Algorithms and Complexity," Dover Publications, Inc., New York, NY, 1982.

[20] G. L. Nemhauser and A. L. Wolsey. "Integer and Combinatorial Optimization," John Wiley & Sons, New York, 1988.

[21] I. Ibrahim, Z. M. Yusof, S. W. Nawawi, M. A. A Rahim, K. Khalil, H. Ahmad, Z. Ibrahim, "A Novel Multi-state Particle Swarm Optimization for Discrete Combinatorial Optimization Problems," 2012 Fourth International Conference on Computational

Intelligence, Modelling and Simulation (CIMSiM), Sep 2012, pp.18,23, doi: 10.1109/CIMSim.2012.46.

[22] M. R. Garey and D. S. Johnson. "Computers and Intractability: a Guide to Theory of NP-Completeness," W. H. Freeman, 1979.

[23] Y. Marinakis, M. Marinaki, and G. Dounias, "Honey Bees Mating Optimization Algorithm for the Euclidean Traveling Salesman Problem," Information Sciences, vol. 181, pp. 4684-4698, 2011.

[24] J. Zhao, Q. Liu, W. Wang, Z. Mei, and P. Shi, "A Parallel Immune Algorithm for Traveling Saleman Problem and its Application on Cold Rolling Scheduling," Information Sciences, vol. 181, pp. 1212-1223, 2011.

[25] O. Ilaya, and C. Bil, "A Particle Swarm Optimisation Approach to Graph Permutations," in Proceeding of IEEE International Conference on Information, Decision, and Control, pp. 366-371, 2007.