# A Novel Multi-State Gravitational Search Algorithm for Discrete 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
Faculty of Manufacturing Engineering
Universiti Malaysia Pahang
26600 Pekan, Pahang, Malaysia
zmdyusof@ump.edu.my

***Abstract* — The binary-based algorithms including the binary gravitational search algorithm (BGSA) were designed to solve discrete optimization problems. Many improvements of the binary-based algorithms have been reported. In this paper, a variant of GSA called multi-state gravitational search algorithm (MSGSA) for discrete optimization problems is proposed. The MSGSA concept is based on a simplified mechanism of transition between two states. The performance of the MSGSA is empirically compared to the original BGSA based on six sets of selected benchmarks instances of traveling salesman problem (TSP). The results are statistically analyzed and show that the MSGSA has performed consistently in solving the discrete optimization problems.**

***Keywords - component; rule-based; multi-state; gravitational search algorithm; discrete combinatorial optimization problem; travelling salesman problem***

## I. INTRODUCTION

Combinatorial discrete optimization is one of the most active fields in computer science, artificial intelligence, operations research, bioinformatics, applied mathematics, and electronic commerce. Typically, the goal of a combinatorial optimization algorithm is to find the best possible ordering of objects satisfying certain conditions or constraints. Furthermore, algorithms for solving these problems can be separated as either complete or approximate [1]. Complete algorithms are assured to find an optimal solution in bounded time for every finite size instance of a combinatorial optimization problem [2-3]. However, for combinatorial optimization problems that are NP-hard [4], no polynomial time algorithm exists. Thus, for the worst case, complete methods might need exponential computational time and this makes the complete methods impractical for most real-world applications. Therefore, approximate algorithms to solve NP-hard combinatorial optimization problems have been gaining interest, as the algorithms lead to significant reduction of computation times, which by its nature sacrifices the guarantee of finding exact optimal solutions [1, 5].

In the class of approximate algorithms, two subclasses of algorithms may be distinguished: approximation algorithms and heuristics algorithms. Approximation algorithms provide provable solution quality and provable run-time bounds. At the other hand, heuristics find good solution on large-size problem instances. These algorithms are permitted to obtain acceptable performance at acceptable costs in a large variety of optimization problems. Heuristics may be classified into two families: specific heuristics and metaheuristics. Unlike specific heuristics, which is designed to solve a specific problem or instance, metaheuristics is designed to solve nearly most of optimization problems [1, 5]. Those metaheuristics includes, but is not restricted to, random Multi-start Local Search (MLS) [6-7], Genetic Algorithms (GA) [8-10], Simulated Annealing (SA) [11], Tabu Search (TS) [12], Ant Colony Optimization (ACO) [13], Artificial Immune System (AIS) [14], Particle Swarm Optimization (PSO) [15], and Variable Neighbourhood Search (VNS) [16].

In past few years, a stochastic population-based metaheuristic called Gravitational Search Algorithm (GSA) has been developed by Rashedi *et al.* in 2009 [17]. GSA is inspired by the Newton's law of universal gravitation where all objects attract to each other with a force of gravitational attraction. This force of gravitational attraction is directly dependent upon the masses of both objects and inversely proportional to the square of the distance that separates their centres. Since the gravitational force is directly proportional to the mass of both interacting objects, bigger objects attract other object with a greater gravitational force.

The conventional GSA was originally designed to solve problems in continuous-valued space. Later, a reworked of the conventional GSA known as binary gravitational search algorithm (BGSA) has been developed to allow GSA to operate in discrete binary variables [18]. Afterward, the standard BGSA is modified by applying the concepts and principles of quantum behaviour to improve the search capability with a fast convergence rate [19-24]. In this paper, a new variant of GSA called multi-state GSA (MSGSA) that represents each agent's vector as state is proposed to extend the application of GSA for solving discrete optimization problems. With regard to the MSGSA, all states (including

the current state) are initially considered as the candidate of the next state. A state is then selected randomly among the candidate states as the next state. As the iteration increases, by exploiting the velocity value of GSA, the number of the candidate of the next state is reduced. The optimization process reaches its convergence when the transition to a different state is not possible

## II. GRAVITATIONAL SEARCH ALGORITHM

The computation of GSA requires a set of $N$ agents, which are randomly positioned in the search space during the initialization. The position of agents, which are the candidate solutions to the problem are represented as follow:

$$X_i = (x_i(1),..., x_i(d)..., x_i(n)) \text{ for } i = 1,2,3,...N \quad (1)$$

where $x_i(d)$ presents the position of $i^{\text{th}}$ agent in the $d^{\text{th}}$ dimension, and $n$ is the space dimension.

Fig. 1 portrays the principle of GSA. Initially, all agents are assigned with velocity $v_i(t,d)$ that is equal to zero where $t$ represents the iteration number. Next, the fitness of agent $i$ at $t$, $fit_i(t)$ for each agent is evaluated with respect to $x_i(t,d)$. The gravitational constant $G(t)$ is then updated as in (2):

$$G(t) = G_0 e^{-\beta \frac{t}{T}} \quad (2)$$

where $T$ is the number of maximum iteration, $G_0$ and $\beta$ are constant values. The gravitational constant is a decreasing function of time where it is valued to $G_0$ at the beginning and it is exponentially decreased towards zero as the iteration increases to control the search accuracy.

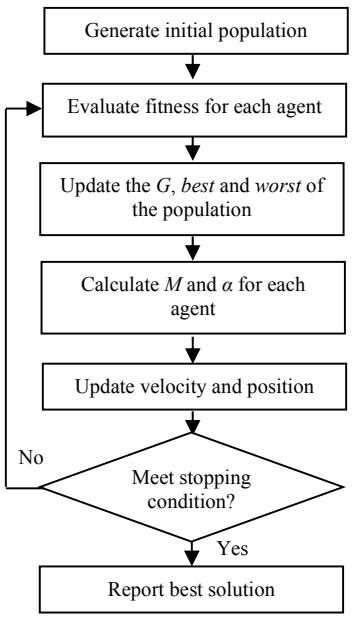


Figure 1. General principle of GSA.

Next, $best(t)$ and $worst(t)$ are calculated. For minimization problem, the definition of $best(t)$ and $worst(t)$ are given as follows:

$$best(t) = min_{j\in\{1,....,N\}} fit_j(t) \quad (3)$$

$$worst(t) = max_{j\in\{1,....,N\}} fit_j(t) \quad (4)$$

For a maximization problem, (3) and (4) are changed to (5) and (6), respectively.

$$best(t) = max_{j\in\{1,....,N\}} fit_j(t) \quad (5)$$

$$worst(t) = min_{j\in\{1,....,N\}} fit_j(t) \quad (6)$$

The gravitational and inertial mass are then updated using (7) and (8):

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \quad (7)$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \quad (8)$$

where $M_i(t)$ is the inertial mass of $i^{\text{th}}$ agent. The acceleration, $\alpha$, of mass $i$ at $t$ in the $d^{\text{th}}$ dimension is calculated as follows:

$$\alpha_i(t,d) = \frac{F_i(t,d)}{M_i(t)} \quad (9)$$

where the force acting $F_i^d(t)$ is calculated as follows:

$$F_i(t,d) = \sum_{j=1, j\neq i}^{N} rand_j F_{ij}(t,d) \quad (10)$$

$$F_{ij}(t,d) = G(t)\frac{M_i(t)}{R_{ij}(t)+\varepsilon}(x_j(t,d) - x_i(t,d)) \quad (11)$$

where $\varepsilon$ is a small constant, $R_{ij}(t)$ is the Euclidean distance between agent $i$ and $j$, and $rand_j$ is a random number uniformly distributed between 0 and 1. The velocity and position of agent's are then updated using (12) and (13), respectively as given by:

$$v_i(t+1,d) = rand_i \times v_i(t,d) + \alpha_i(t,d) \quad (12)$$

$$x_i(t+1,d) = x_i(t,d) + v_i(t+1,d) \quad (13)$$

where $rand_i$ is random number uniformly distributed between 0 and 1. 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.

## III. MSGSA

The MSGSA algorithm, as shown in Fig. 2, is explained in this section. The MSGSA practices similar general principle of original GSA with a few modifications in updating the velocity and position of each agent and formulating the calculation of force for each agent.

### *A. Velocity and position updates in MSGSA*

Each agent's vector in the MSGSA is represented by a state, which is neither a continuous nor discrete value. To elaborate this state representation, the Burma14 benchmark instance of the Travelling Salesman Problem (TSP) is used as an example, as illustrated in Fig. 3. All cities in the Burma14 benchmark instance can be represented as a collective of states, in which the states are represented by small black circles, as presented in Fig. 4. A centroid of the circle shows the current state, and the radius of the circle represents the next velocity of the current state. These three elements occur in each dimension for each agent. The updating velocity and position in form of state in the MSGSA are performed after the inertial mass $M$ and acceleration $\alpha$ are calculated.

In the MSGSA once the velocity is updated, the process of updating the current state to the next state for each dimension of each agent is executed. The current state is defined as a centroid and the updated velocity as a radius, thus creating a circle. Any state that is located in the area of the circle is defined as a member of the inner states (IS) group. Any state that is located outside of the circle is then defined as a member of the outer states (OS) group. A next state is then selected randomly among the member of IS group using (14). Given a set of $j$ IS members is $I_i(t,d) = (I_{i_1}(t,d),...,I_{i_j}(t,d))$.

$$x_i(t+1,d) = random(I_{i_1}((t,d),...,I_{i_j}(t,d)) \quad (14)$$

### *B. Force formulation in MSGSA*

A subtraction operation, as presented by (11), is executed to calculate the difference of the vector value between the positions of two agents $x_j(t,d)$ and $x_i(t,d)$ for each vector and iteration, resulting in a numerical value. However, in the MSGSA, each vector's position of each agent is represented as a state. Because a state is not associated with any value, the subtraction operation in (11) cannot be used to find the difference between two positions. To accommodate the calculation of force, $F_{ij}(t,d)$ in the MSGSA, a cost function $C(.)$ is introduced and incorporated into the force formulation as follows:

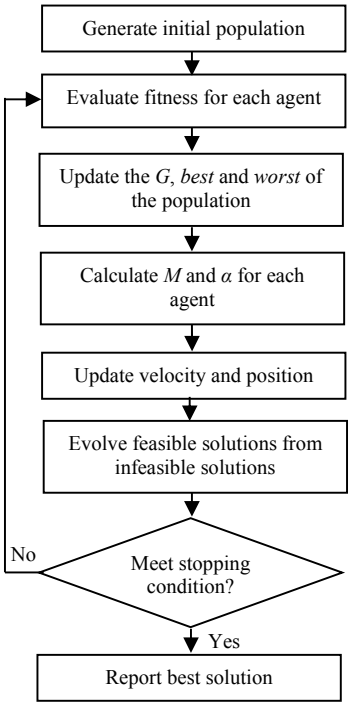


Figure 2. General principle of MSGSA.

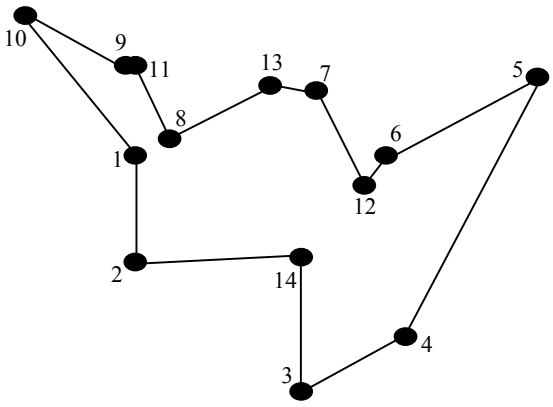


Figure 3. An example of Travelling Salesman Problem (TSP). The name of this benchmark instance is Burma14.

$$F_{ij}(t,d) = G(t)\frac{M_i(t)}{R_{ij}(t)+\varepsilon}\mathrm{C}(x_j(t,d),x_i(t,d)) \quad (15)$$

Cost can be defined as distance and time in the TSP and assembly sequence planning (ASP) problem, respectively [25, 26]. Hence, the cost between two states is a positive number given by $\mathrm{C}(x_j(t,d),x_i(t,d))$. In this force

formulation, $R_{ij}$ is the difference of fitness between agent $i$ and $j$.

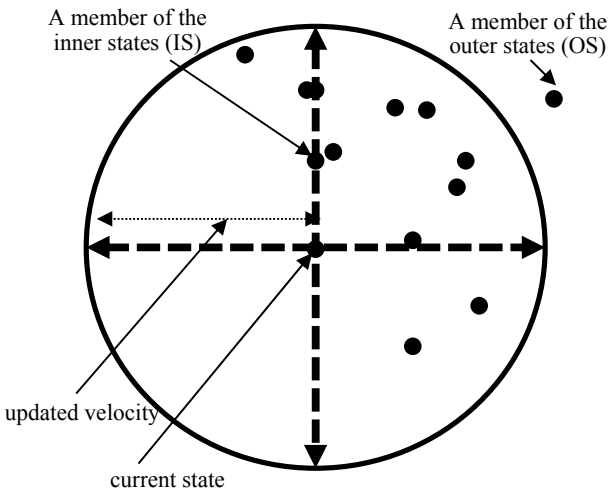


Figure 4. Illustration of the multi-state representation in MSPSO algorithm for Burma14 benchmark instance of TSP. Each dimension of each agent applies the same representation.

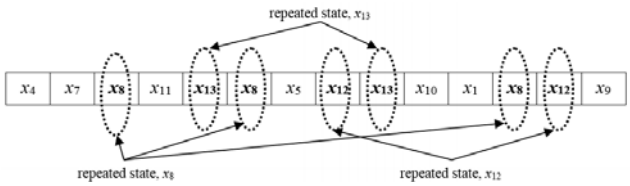


Figure 5. Example of a solution with repeated states.

## IV. Solving Symmetric TSP using MSGSA

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, $e_i$ and $e_j$, is equivalent to the distance of two adjacent cities, $e_j$ and $e_i$. Otherwise, it is called asymmetric. The TSP is belongs to the class of NP-hard combinatorial optimization problems. The MSGSA eminently suitable for solving the TSP due to the constraint of this problem, which is one city or state can only be visited or used once. This means the repetitive states in solutions can be prevented. Thus, feasible solutions can be produced.

### A. Limitation of the MSGSA

Equation (14) may lead to the existence of repeated state in an updated solution. Let consider a solution of an agent at a particular iteration consisting 14-dimensional vector $\{x_5, x_3, x_{14}, x_{11}, x_2, x_8, x_9, x_{13}, x_{12}, x_{10}, x_1, x_4, x_6, x_7\}$. 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 $\{x_4, x_7, x_8, x_{11}, x_{13}, x_8, x_5, x_{12}, x_{13}, x_{10}, x_1, x_8, x_{12}, x_9\}$, as illustrated in Fig. 5. 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.

### B. Evolve an infeasible solution to a feasible solution

To overcome the limitation of the MSGSA in solving the TSP, an additional procedure is introduced after velocity and position (state) are updated. The introduction of this procedure is crucial in order to evolve a feasible solution from an infeasible solution. Fig. 6 illustrates the principle of this procedure. Initially, a solution of an agent 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. It is worth mentioning that this procedure is applied to the solution of each agent.

## V. Results and Discussions

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

*Burma14*: A TSP problem of 14 cities. The optimal route length is 3323.

*Ulysses16*: A TSP problem of 16 cities. The optimal route length is 6859.

*Ulysses22*: A TSP problem of 22 cities. The optimal route length is 7013.

*Bays29*: A TSP problem of 29 cities. The optimal route length is 2020.

*Eil51*: A TSP problem of 51 cities. The optimal route length is 426.

*Berlin52*: A TSP problem of 52 cities. The optimal route length is 7542.

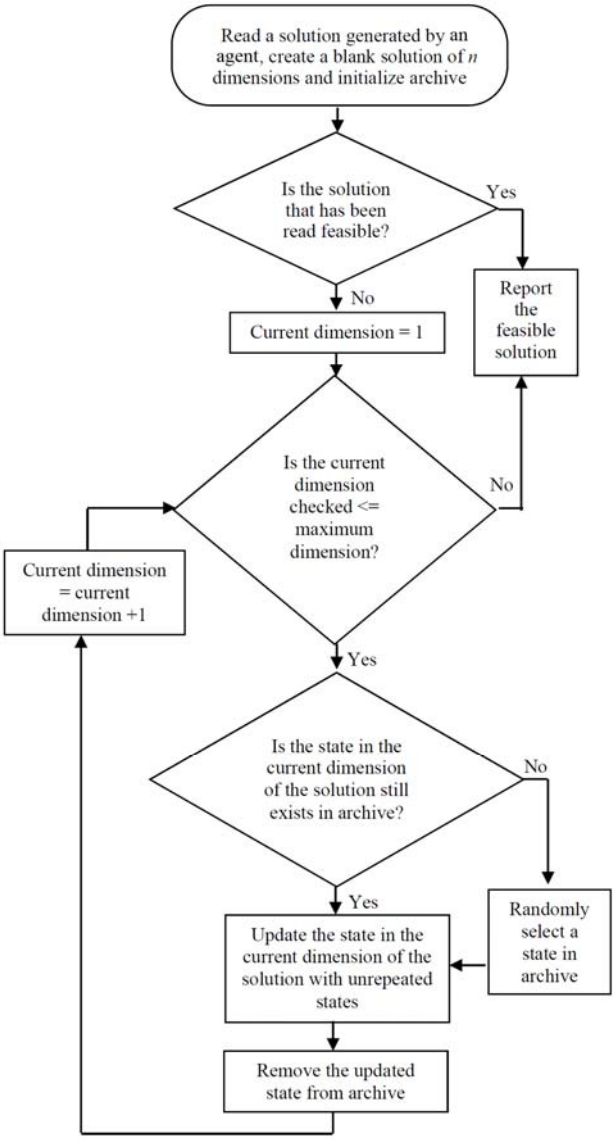


Figure 6. Procedure of evolving an infeasible solution to feasible solution.

To demonstrate the relative efficiency of the MSGSA, the performance of MSGSA and the BGSA [19] is compared. In experiments, $G_o$ is set to100, $\beta$ is set to 20, the total number of iterations, $T$, is set to 10000, and 30 number of agents is used. In the initialization stage, all agents are randomly positioned in the search space. All agents are then assigned with velocity $v_i(t,d)$ that is equal to zero.

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 the MSGSA and the BGSA for 50 trials is presented in Table 1 and Table 2, respectively. Based on the results given in Table 1 and Table 2, the MSGSA outperformed the BGSA in obtaining the quality of results as the MSGSA successfully produces smaller values of the mean for each benchmark instance.

The difference between the solutions obtained by these two algorithms is also analysed using boxplots, as shown in Fig. 7, 8, 9, 10, 11, and 12. 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 MSGSA and the BGSA 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 MSGSA and the BGSA. 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. It can be noticed from Fig. 7, 8, 9, 10, 11, and 12 that the MSGSA is performed consistently based on the size of the box, compared to the BGSA. 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. 7, 8, 9, 10, 11, and 12 that the MSGSA gives good performance on the six sets of TSP benchmark instances; the MSGSA is promising.

It seems that the solutions produced by the MSGSA and the BGSA are not normally distributed. In this study, the Wilcoxon Signed Rank test is chosen and then presented in Table 3. The null hypothesis is that the quality results of the MSGSA and the BGSA are identical. In Table 3, the term of "diff" is the difference of mean value for 50 runs between the MSGSA and the BGSA.

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

$$W = \min(W+, W-) \tag{16}$$

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 $W^- = [3 + 4 + 5 + 2 + 1 + 6] = 21$. Therefore, the null hypothesis can be rejected because $0 < 1$.

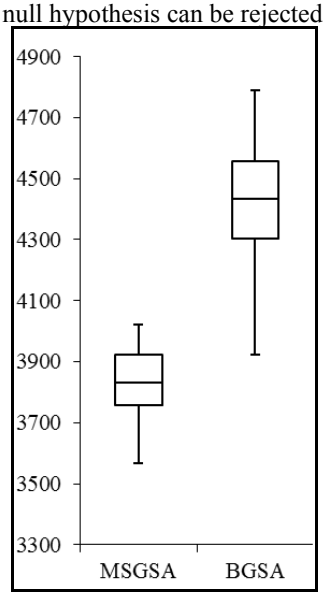


Figure 7. Boxplots of the MSGSAand the BGSAfor Burma14 benchmark instances.

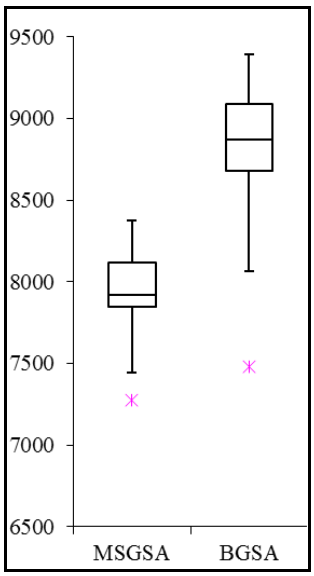


Figure 8. Boxplots of the MSGSAand the BGSAfor Ulysses16 benchmark instances.

The result is statistically significant at $p < 0.05$, where the MSGSA performs significantly better than the BGSA.

## VI. Conclusions and Future Work

Gravitational Search Algorithm is a relatively new heuristic search algorithm that uses the theory of Newtonian physics. Meanwhile, its searcher agents are the collection of masses. In this study, a variant of GSA called the MSGSA that uses states to represent solutions has been proposed to solve discrete combinatorial optimization problems, particularly in the TSP. To evaluate the performance of the proposed MSGSA, six sets of TSP benchmark instances are used and the performance of the proposed MSGSA is evaluated and then compared with the existing BGSA. Both the proposed MSGSA and the BGSA are executed to find the shortest route. Experimental results obtained from the six instances used show that the proposed MSGSA consistently outperformed the BGSA. In future, the application of the proposed MSGSA to discrete combinatorial optimization problems such as VLSI routing, DNA sequence design, and airport gate allocation, PCB routing will be investigated.

## Acknowledgement

This work is financially supported by the Ministry of Education Malaysia through the Fundamental Research Grant Scheme (FRGS) VOT RDU140114 granted to Universiti Malaysia Pahang.

## References

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

[2] C. H. Papadimitriou and K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity. Upper Saddle River, NJ: Prentice-Hall, 1982.

[3] G. L. Nemhauser and A. L. Wolsey, Integer and Combinatorial Optimization. NY: John Wiley & Sons, 1988.

[4] M. R. Garey and D. S. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness. NY: W. H. Freeman & Co, 1979.

[5] E. G. Talbi, Metaheuristics: From Design to Implementation. NJ: John Wiley & Sons, 2009.

[6] S. Reiter and D.B. Rice, "Discrete Optimizing Solution Procedures for Linear and Nonlinear Integer Programming Problems," Management Science, vol. 12, Jul. 1966, pp. 829-850, doi:10.1287/mnsc.12.11.829.

[7] S. Reiter and G. Sherman, "Discrete Optimizing," Journal of the Society for Industrial and Applied Mathematics, vol. 13, Sep. 1965, pp. 864-889, doi: 10.1137/0113056.

[8] J. H. Holland, "Outline for a Logical Theory of Adaptive Systems," Journal of the ACM, vol. 9, Jul. 1962, pp. 297-314, doi: 10.1145/321127.321128.

[9] J. H. Holland, Adaptation in Natural and Artificial Systems. MA: MIT Press Cambridge, 1992.

[10] Y. Wen, H. Xu and J. Yang, "A Heuristic-Based Hybrid Genetic-Variable Neighborhood Search Algorithm for Task Scheduling in Heterogeneous Multiprocessor System," International Journal of Information Sciences, vol. 181, Feb. 2011, pp. 567-581, doi: 10.1016/j.ins.2010.10.001.

[11] A. Kalashnikov and V. Kostenko, "A Parallel Algorithm of Simulated Annealing for Multiprocessor Scheduling," International Journal of Computer and Systems Sciences, vol. 47, Jul. 2008, pp. 455‑463, doi: 10.1134/S1064230708030155.

[12] S. Porto and C. Ribeiro, "A Tabu Search Approach to Task Scheduling on Heterogeneous Processors under Precedence Constraints," International Journal of High Speed Computing, vol. 7, Mar. 1995, pp. 45‑72, doi: 10.1142/S012905339500004X.

[13] F.F. Boctor, J. Renaud, A. Ruiz and S. Tremblay, "Ant Colony Optimization for Mapping and Scheduling in Heterogeneous

Multiprocessor Systems," Proc. IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS 08), IEEE Press, Jul. 2008, pp. 142-149, doi: 10.1109/ICSAMOS.2008.4664857.

TABLE I. NUMERICAL RESULTS FOR THE MSGSA

| Benchmark instance | MSGSA | | | |
|---|---|---|---|---|
| | *Minimum* | *Mean* | *Maximum* | *Standard deviation* |
| Burma14 | 3567.00 | 3827.00 | 4023.00 | 112.01 |
| Ulysses16 | 7278.00 | 7938.58 | 8376.00 | 219.29 |
| Ulysses22 | 9411.00 | 10351.62 | 10859.00 | 317.05 |
| Bays29 | 3673.00 | 3993.46 | 4168.00 | 104.58 |
| Eil51 | 1175.00 | 1229.50 | 1278.00 | 24.23 |
| Berlin52 | 20859.00 | 21993.80 | 22848.00 | 451.47 |

TABLE II. NUMERICAL RESULTS FOR THE BGSA

| Benchmark instance | BGSA | | | |
|---|---|---|---|---|
| | *Minimum* | *Mean* | *Maximum* | *Standard deviation* |
| Burma14 | 3893.00 | 4437.00 | 4857.00 | 192.39 |
| Ulysses16 | 7481.00 | 9355.00 | 8800.20 | 389.83 |
| Ulysses22 | 10099.00 | 11835.00 | 11157.12 | 389.31 |
| Bays29 | 3819.00 | 4483.00 | 4235.85 | 108.70 |
| Eil51 | 1191.00 | 1326.00 | 1269.16 | 22.86 |
| Berlin52 | 21533.00 | 24056.00 | 22830.96 | 522.24 |

TABLE III. WILCOXON SIGNED RANK TEST

| Benchmark instance | MSGSA | BGSA | Diff | Rank |
|---|---|---|---|---|
| Burma14 | 3827.00 | 4437.00 | -610 | 3 |
| Ulysses16 | 7938.58 | 9355.00 | -1416.42 | 4 |
| Ulysses22 | 10351.62 | 11835.00 | -1483.38 | 5 |
| Bays29 | 3993.46 | 4483.00 | -489.54 | 2 |
| Eil51 | 1229.50 | 1326.00 | -96.5 | 1 |
| Berlin52 | 21993.80 | 24056.00 | -2062.2 | 6 |

[14] H. Yu, "Optimizing Task Schedules using an Artificial Immune System Approach," Proc. ACM Annual Conference on Genetic and Evolutionary Computation (GECCO 08), ACM, Jul. 2008, pp. 151-158, doi: 10.1145/1389095.1389116.

[15] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," Proc. IEEE International Conference on Neural Networks (ICNN 95), IEEE Press, Nov/Dec. 1995, pp. 1942-1948, doi: 10.1109/ICNN.1995.488968.

[16] N. Mladenović and P. Hansen, "Variable Neighborhood Search," Computers & Operations Research, vol. 24, Nov. 1997, pp. 1097-1100, doi: 10.1016/S0305-0548(97)00031-2.

[17] E. Rashedi, H. Nezamabadi-pour and S. Saryazdi, "GSA: a Gravitational Search Algorithm," Information Sciences, vol. 179, Jun. 2009, pp. 2232-2248, doi: 10.1016/j.ins.2009.03.004.

[18] E. Rashedi, H. Nezamabadi-pour and S. Saryazdi, "BGSA: Binary Gravitational Search Algorithm," Natural Computing, vol. 9, Sep. 2010, pp. 727-745, doi: 10.1007/s11047-009-9175-3.

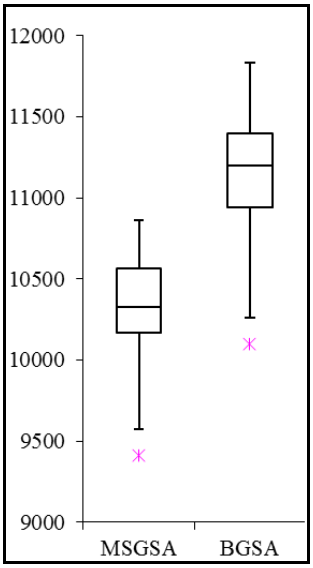


Figure 9. Boxplots of the MSGSAand the BGSAfor Ulysses22 benchmark instances.

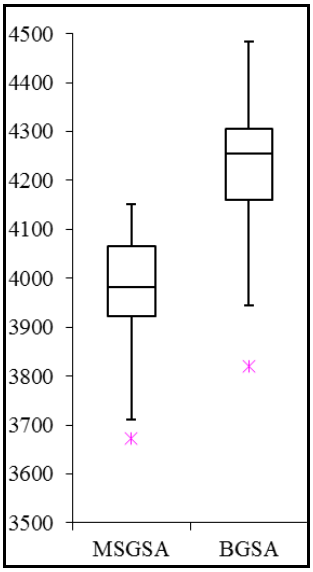


Figure 10. Boxplots of the MSGSAand the BGSAfor Bays29 benchmark instances.

[19] T. Chakraborti, A. Chatterjee, A. Halder, and A. Konar, "Automated Emotion Recognition Employing a Novel Modified Binary Quantum-Behaved Gravitational Search Algorithm with Differential Mutation," Expert Systems, vol. 32, Feb. 2015, pp. 522-530, doi:10.1111/exsy.12104.

[20] A. Mohamed, A. A. Ibrahim, and H. Shareef, "A Novel Quantum-Inspired Binary Gravitational Search Algorithm in Obtaining Optimal

Power Quality Monitor Placement," Journal of Applied Sciences, vol. 12, Jan. 2012, pp. 822-830, doi: 10.3923/jas.2012.822.830.

[21] X. Han, L. Quan, X. Y. Xiong, and B. Wu, "Facing the Classification of Binary Problems with a Hybrid System based on Quantum-Inspired Binary Gravitational Search Algorithm and K-NN Method,"

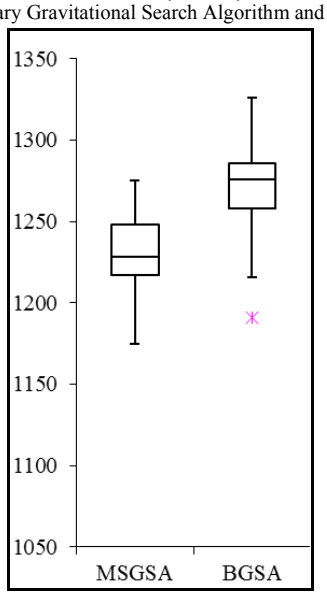


Figure 11. Boxplots of the MSGSAand the BGSAfor Eil51 benchmark instances.

Engineering Applications of Artificial Intelligence, vol. 26, Jul. 2013, pp. 2424-2430, doi: 10.1016/j.engappai.2013.05.011.

[22] B. Ji, X. Yuan, X. Li, Y. Huang, and W. Li, "Application of Quantum-Inspired Binary Gravitational Search Algorithm for Thermal Unit Commitment with Wind Power Integration," Energy Conversion and Management, vol. 87, Nov. 2014, pp. 589-598, doi:10.1016/j.enconman.2014.07.060.

[23] A. A. Ibrahim, A. Mohamed, and H. Shareef, "Optimal Power Quality Monitor Placement in Power Systems using an Adaptive Quantum-Inspired Binary Gravitational Search Algorithm," International Journal of Electrical Power and Energy Systems, vol. 57, May. 2014, pp. 404-413, doi:10.1016/j.ijepes.2013.12.019.

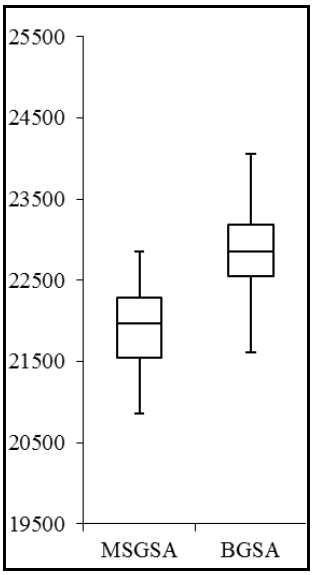


Figure 12. Boxplots of the MSGSAand the BGSAfor Berlin52 benchmark instances.

[24] H. Nezamabadi-Pour, "A Quantum-Inspired Gravitational Search Algorithm for Binary Encoded Optimization Problems," Engineering Applications of Artificial Intelligence, vol. 40, Apr. 2015, pp. 62-75, doi:10.1016/j.engappai.2015.01.002.

[25] A. Stentz, "Optimal and Efficient Path Planing for Partially-Known Environments," Proc. IEEE International Conference on Robotics and Automation (ROBOT 94), IEEE Press, May. 1994, pp. 3310-3317, doi: 10.1109/ROBOT.1994.351061.

[26] J. A. A. Mukred, Z. Ibrahim, I. Ibrahim, A. Adam, W. K. W. Ahmad, Z. M. Yusof and N. Mokhtar, "A Binary Particle Swarm Optimization Approach to Optimize Assembly Sequence Planning," Advanced Science Letters, vol. 13, Jun. 2012, pp. 732-738, doi: 10.1166/asl.2012.3879.