A Modified Gravitational Search Algorithm for Discrete Optimization Problem

Shahdan Sudin, Sophan Wahyudi Nawawi, Amar Faiz Zainal Abidin, Muhammad Arif Abdul Rahim, Kamal Khalil
Faculty of Electrical Engineering
Universiti Teknologi Malaysia
81310 Skudai, Johor
shahdan@fke.utm.my, sophan@fke.utm.my,
amarfaiz@fke.utm.my, arif@fke.utm.my,
kamal@fke.utm.my

Zuwairie Ibrahim[1], Zulkifli Md Yusof[2]
[1]Faculty of Electrical and Electronic Engineering
[2]Faculty of Manufacturing
Universiti Malaysia Pahang
26600 Pekan, Malaysia
zuwairie@ump.edu.my, zmdyusof@ump.edu.my

***Abstract*—This paper presents a modified Gravitational Search Algorithm (GSA) called Discrete Gravitational Search Algorithm (DGSA) for discrete optimization problems. In DGSA, an agent's position is updated based on its direction and velocity. Both the direction and velocity determine the candidates of integer values for the position update of an agent and then the selection is done randomly. Unimodal test functions are used to evaluate the performance of the proposed DGSA. The experimental result shows that the FDGSA able to find better solutions and converges faster compared to the Binary Gravitational Search Algorithm.**

Keywords - gravitational search algorithm, discrete optimization problem.

I. INTRODUCTION

Gravitational Search Algorithm (GSA) has been originally proposed by Rashedi *et al.* in 2009 [1]. The idea of GSA came from the Newtonian laws of gravitation and motion where all objects move as a result of attraction with each other by gravitational forces. Objects with heavier mass have stronger attraction and move slower than the objects with relatively smaller mass. Results in [1] showed that GSA performed considerably better compared to Particle Swarm Optimization (PSO) [2] and Central Force Optimization (CFO) [3].

A variant of GSA algorithm, which is called Binary GSA (BGSA), for solving discrete optimization problems has been proposed Rashedi *et al.* in 2010 [4]. BGSA employs a probability function to update a string of binary bits based on the absolute velocity value. Higher velocity indicates higher probability of a bit 0 to change to 1 and vice versa. In this study, another variant of GSA algorithm, which is called Fast Discrete GSA (FDGSA) algorithm, is proposed. Based on the proposed FDGSA, a discrete optimization problem is modeled by integer values, instead of a string of binary bits. The integer values, which correspond to the agent's position in a search space, is updated based on its direction and velocity. The information related to the direction and velocity of an agent is used to determine the candidates of integer values for the position update. After that, the selection is done randomly.

A preliminary version of this paper was presented at the 2012 Fourth International Conference on Computational Intelligence, Modelling and Simulation

II. THE GRAVITATIONAL SEARCH ALGORITHM (GSA)

A. GSA for Continuous-valued Search Problem

Assume each agent's position in the searching space can be represented by

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

where N presents the number of agents and the position of ith agents in the dth dimension can be represented as x_i^d. The mass of each agent is calculated for updating each ith agent with reference to the equality of gravitational and inertia mass assumption as follows:

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \quad (2)$$

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \quad (3)$$

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

$$best(t) = \min_{j \in \{1,\ldots,N\}} fit_j(t) \quad (4)$$

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

Otherwise, Eq. (6) and Eq. (7) are used to define the $best(t)$ and $worst(t)$ for maximization problem.

$$best(t) = \max_{j \in \{1,\ldots,N\}} fit_j(t) \quad (6)$$

$$worst(t) = \min_{j \in \{1,\ldots,N\}} fit_j(t) \quad (7)$$

The fitness value, $fit_i(t)$, affects the mass value of ith agents, which corresponds to the position of the particle in a search space.

The general principle of GSA is shown in Fig. 1. The optimization process started with positioning the agents randomly with random velocity values and initialization of gravitational constant. The next step is to evaluate the fitness, $fit_i(t)$, of each agent according to the objective function. Then, the gravitational constant, $G(t)$, is updated based on Eq. (8) due to the effect of decreasing gravity.

$$G(t) = G(t_o) \times \left(\frac{t_o}{t} \right)^{\beta}, \beta < 1 \tag{8}$$

Mass, M, for each agent is calculated using Eq. (2) and Eq. (3), and acceleration, α, is calculated using $\alpha_i^d(t) = F_i^d(t)$, where the force acting is calculated as follows:

$$F_i^d(t) = \sum_{j=1, j \neq i}^{N} rand_j F_{ij}^d(t) \tag{9}$$

$$F_{ij}^d(t) = G(t) \frac{M_{aj}(t)}{R_{ij}(t) + \varepsilon} \left(x_j^d(t) - x_i^d(t) \right) \tag{10}$$

where M_{aj} is the active gravitational mass related to agent j, ε is a small constant, R_{ij} is the distance between agent i and j and $rand_j$ is a uniform random variable in the interval [0,1]. Then, the velocity, v_d^i, and position, x_d^i, of ith agents is calculated as follows:

$$v_i^d(t+1) = rand_i \times v_i^d(t) + \alpha_i^d(t) \tag{11}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{12}$$

where $rand_i$ is a uniform random variable in the interval [0,1]. This updating process is repeated as long as the stopping criterion is not satisfied.

B. GSA for Discrete Search Problem

To solve discrete optimization using Binary Gravitational Search Algorithm (BGSA) [4], the position update is modified to accommodate with binary search space. Bit exchanged is based on Eq. (13), as for small $|v_i^d|$, the probability of changes of an agent's position must be near zero or *vice versa*.

$$S\left(v_i^d(t+1)\right) = \left|\tanh\left(v_i^d(t)\right)\right| \tag{13}$$

Moreover, the agent's velocity is normalized as follows:

$$\begin{aligned} &\text{if} \quad rand < S\left(v_i^d(t=1)\right) \\ &\text{then} \quad x_i^d(t+1) = \text{complement}\left(x_i^d(t)\right) \\ &\text{else} \quad x_i^d(t+1) = x_i^d(t) \end{aligned} \tag{14}$$

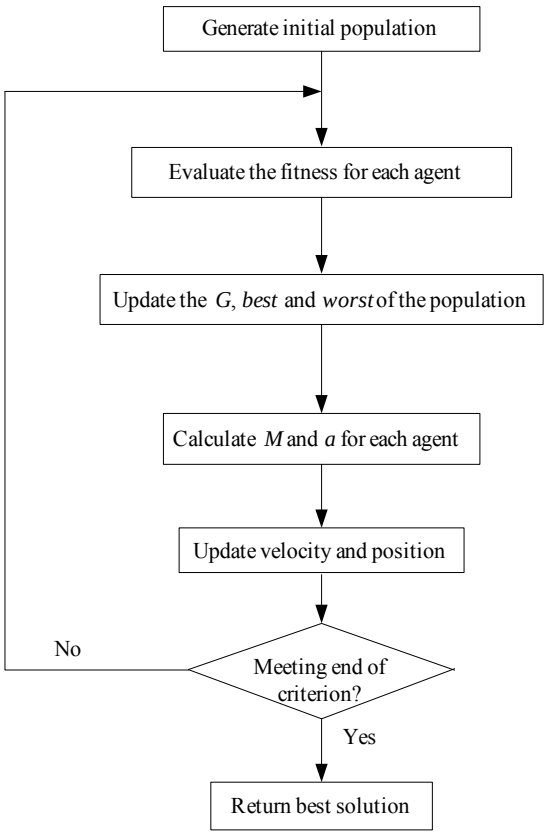

Fig. 1 : Flowchart of Gravitational Search Algorithm.

III. THE DISCRETE GRAVITATIONAL SEARCH ALGORITHM

In DGSA, a new approach is introduced to update agent's position. The direction and velocity of an agent will determine the candidates of the next position. Specifically, an agent moves in the positive direction if the velocity value is positive. Otherwise, the agent moves in the negative direction. The velocity of the agent determines the candidate of the next position and then, position update is done randomly based on the candidate of the next position.

Consider an ith agent, which current position in dth dimension, $x_i^d(t) = 100$, as shown in Fig. 2. Let say the velocity, $v_i^d(t+1) = 60$. The candidates of the next position, $x_i^d(t+1)$ are the integer values 100, 101, 102, ….., and 160. Hence, the next position will be an integer, randomly chosen between 100 and 160.

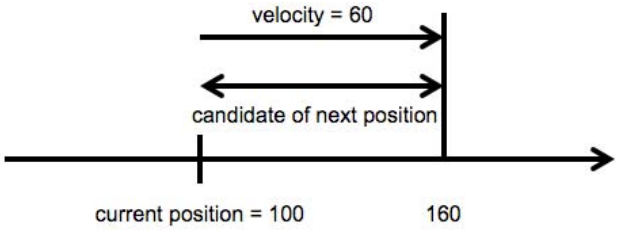

Fig. 2 : Identification of the candidates of the next position for the case $\left(v_i^d\left(t+1\right)\right)>0$.

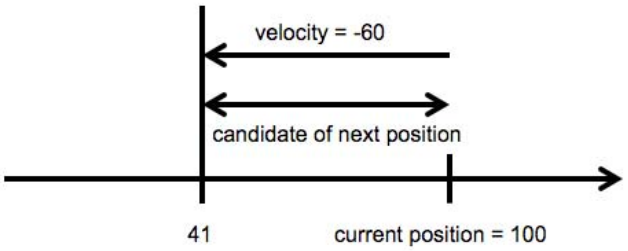

Fig. 3 : Identification of the candidates of the next position for the case $\left(v_i^d\left(t+1\right)\right)<0$.

Similarly, if the velocity is negative, for example, $v_i^d\left(t+1\right)=-60$, as shown in Fig. 3. The candidates of the next position, $x_i^d\left(t+1\right)$ are the integer values 100, 99, 98, ….., and 41. The next position will be an integer, randomly chosen between 41 and 100.

Based on these new concepts, while the velocity update is still the same, the position update can be formulated as:

$$\text{if } \left(v_i^d\left(t+1\right)\right)>0$$
$$x_i^d\left(t+1\right)=$$
$$random\left\{x_i^d\left(t\right),x_i^d\left(t\right)+1,....,x_i^d\left(t\right)+v_i^d\left(t+1\right)\right\} \quad (15)$$

$$\text{if } \left(v_i^d\left(t+1\right)\right)<0$$
$$x_i^d\left(t+1\right)=$$
$$random\left\{x_i^d\left(t\right),x_i^d\left(t\right)-1,....,x_i^d\left(t\right)-v_i^d\left(t+1\right)\right\} \quad (16)$$

Eq. (15) is used if the velocity is positive and Eq. (16) is used is the velocity is negative.

IV. Experiment, Result, and Discussion

Five benchmark functions used in this study, which are taken from [4], are shown in Table 1. These functions are to be minimized subjected to the integer value of x_i or x_j. Note that m is the dimension of the functions and $S \subseteq R^m$.

TABLE I. Test Functions

Test function	S
$F_1(x)=\sum_{i=1}^{m} x_i^2$	$[-100, 100]^m$
$F_2(x)=\sum_{i=1}^{m}\lvert x_i\rvert+\prod_{i=1}^{m}\lvert x_i\rvert$	$[-100, 100]^m$
$F_3(x)=\sum_{i=1}^{m}\left(\sum_{j=1}^{i} x_j\right)^2$	$[-100, 100]^m$
$F_4(x)=\max_i\left\{\lvert x_i\rvert, 1\le i\le m\right\}$	$[-100, 100]^m$
$F_5(x)=\sum_{i=1}^{m-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	$[-100, 100]^m$

TABLE II. Experimental Setup

	DGSA	BGSA
Number of agents, N	50	50
Number of dimensions, m	5	5
Gravitational constant, $G(t_0)$	100	100
Epsilon, ε	10	10
Number of iteration	500	500

Table 2 shows the experimental setup for both BGSA and DGSA. However, for the implementation of BGSA, eight bits binary representation is used to represent discrete values, where the 7th bit is used to indicate the sign of a number. The velocity is limited to $\lvert v_i^d\rvert < v_{max}$ where $v_{max} = 6$ and the distance between agents is calculated based on the Hamming distance.

Examples of convergence curves are shown in Fig. 3 to Fig. 7. For all benchmark functions, the proposed DGSA converges significantly faster than the BGSA. The average speed of convergence over 40 runs is shown in Table 3. It turns out that while the BGSA requires 260 to 300 iteration to complete a run, the proposed DGSA only needs less than 100 iterations to get the solution.

The quality of the solutions found by both BGSA and DGSA are shown in Table 4. Both algorithms found the exact solutions for benchmark functions F_1 - F_4. More interestingly, even though the proposed DGSA able to complete a run significantly faster than the existing BGSA, the result based on benchmark function F_5 shows that, in average, the solution found by the DGSA is better than the BGSA. This is because the BGSA has found a local minima solution with the largest fitness value of 533.

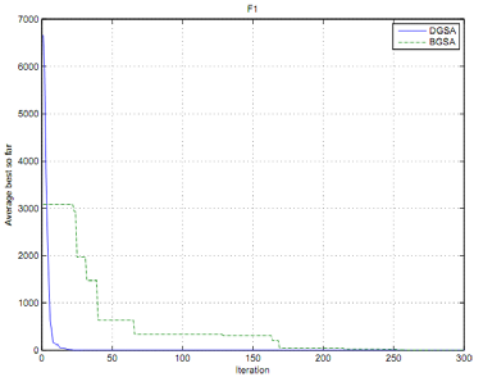

Fig. 3: Convergence curve for the case of F_1.

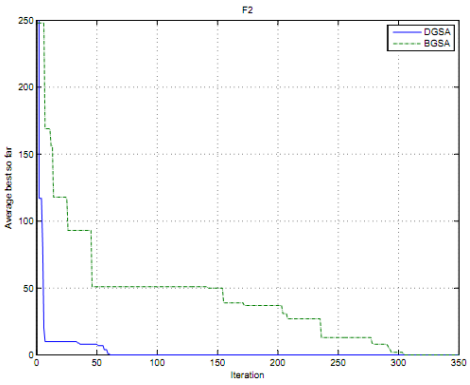

Fig. 4: Convergence curve for the case of F_2.

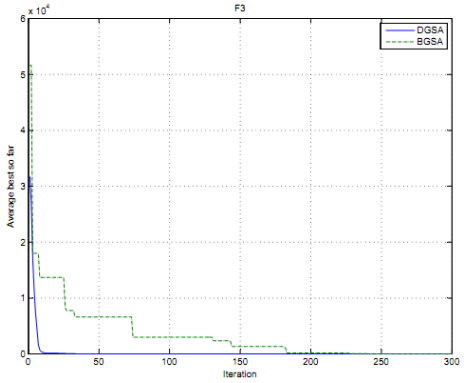

Fig. 5: Convergence curve for the case of F_3.

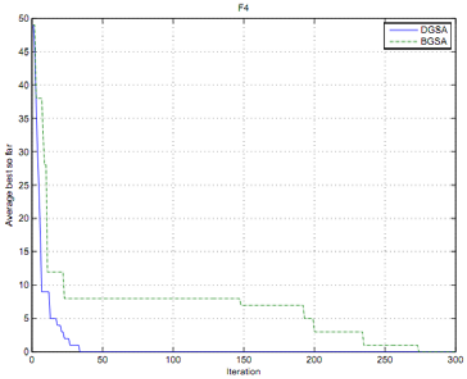

Fig. 6: Convergence curve for the case of F_4.

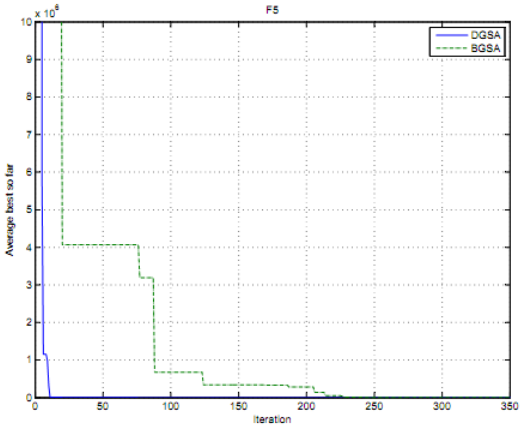

Fig. 7: Convergence curve for the case of F_5.

TABLE III. THE AVERAGE SPEED OF CONVERGENCE OVER 40 RUNS

Benchmark	Average completion (iteration)	
	DGSA	BGSA
F_1	30.3667	274.1667
F_2	61.9333	303.8667
F_3	57.5	262.8333
F_4	32.9333	273.4
F_5	76.2667	300.8

TABLE IV. COMPARISON OF QUALITY OF SOLUTION BETWEEN DGSA AND BGSA

Function	Statistical function	DGSA	BGSA
F_1	Mean	0	0
	Median	0	0
	Standard Deviation	0	0
F_2	Mean	0	0
	Median	0	0
	Standard Deviation	0	0
F_3	Mean	0	0
	Median	0	0
	Standard Deviation	0	0
F_4	Mean	0	0
	Median	0	0
	Standard Deviation	0	0
F_5	Mean	8.3667	21.2333
	Median	0	4
	Standard Deviation	38.3185	100.4506

TABLE V. FITNESS FOUND BY DGSA AND BGSA OVER 40 RUNS BASED ON BENCHMARK FUNCTION F_5

Fitness value	DGSA	BGSA
0	26 times	10 times
4	14 times	28 times
202	none	1 time
211	1 time	none
553	none	1 time

The benchmark function F_5 is more difficult to be solved compared to others. Both DGSA and BGSA algorithms often found the local minima solutions, with the fitness values 4, 202, 211, and 533. In addition, the results shown in Table 5 shows that the rate of the proposed DGSA to trap at local minima is lower than that of BGSA. Specifically, over 40 runs, the proposed DGSA has found the exact solution 26 times as opposed to BGSA, which only has 25% of the probability to find the exact solution.

V. CONCLUSION

The paper introduced a simple and efficient GSA algorithm, called Discrete GSA algorithm (DGSA), for discrete optimization problem. Instead of binary representation and update, as employed in the existing BGSA, integer values, which correspond to the agent's position in a search space, is updated. The integer value is updated randomly after the candidates of the integer values is determined based on its direction and velocity. The DGSA has been evaluated according to several criteria. It is found that the proposed DGSA is superior than the existing BGSA in terms of quality of solution found and speed of convergence. The proposed DGSA is also less likely to trap in a local minima.

ACKNOWLEDGMENT

This research is funded by the UTM-GUP Research Fund (Q.J130000.2623.09J03) and Fundamental Research Grant Scheme (FRGS) (4F374) from Universiti Teknologi Malaysia.

REFERENCES

[1] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A Gravitational Search Algorithm," Information Sciences, vol. 179, pp. 2232-2248, 2009.

[2] J. Kennedy, and R.C. Eberhart, "Particle Swarm Optimization," Proceedings of IEEE International Conference on Neural Networks, 1995, pp. 1942-1948.

[3] R.A. Formato, "Central Force Optimization: A New Nature Inspired Computational Framework for Multidimensional Search and Optimization," Studies in Computational Intelligence, vol. 129, pp. 221-238, 2008.

[4] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "BGSA: Binary Gravitational Search Algorithm," Natural Computing, vol. 9, pp. 727-745, 2010.