BMOA: Binary Magnetic Optimization Algorithm

SeyedAli Mirjalili and Siti Zaiton Mohd Hashim

***Abstract*—Recently, the behavior of natural phenomena has become one the most popular sources for researchers in to design optimization algorithms. One of the recent heuristic optimization algorithms is Magnetic Optimization Algorithm (MOA) which has been inspired by magnetic field theory. It has been shown that this algorithm is useful for solving complex optimization problems. The original version of MOA has been introduced in order to solve the problems with continuous search space, while there are many problems owning discrete search spaces. In this paper, the binary version of MOA named BMOA is proposed. In order to investigate the performance of BMOA, four benchmark functions are employed, and a comparative study with Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) is provided. The results indicate that BMOA is capable of finding global minima more accurate and faster than PSO and GA.**

***Index Terms*—Optimization algorithm, evolutionary algorithm, genetic algorithm, particle swarm optimization, PSO.**

I. Introduction

Recently, many heuristic evolutionary optimization algorithms have been developed in order to solve complex computational problems. Some of them are Particle Swarm Optimization (PSO) [1] , Genetic Algorithm (GA) [2], Simulated Annealing (SA) [3], and Ant Colony Optimization (ACO) [4]. It has been proven that there is no algorithm which could perform general enough to solve all optimization problems [5]. In other words, some of these algorithms may possess better solutions for some specific problems better and worse for others.

Magnetic Optimization Algorithm (MOA) is a novel heuristic optimization algorithm, inspired from the principles of magnetic field theory [6]. It has been proven that this algorithm has a good performance in solving optimization problems. In this algorithm, the search is carried out by magnetic particles as search agents while they interact with each other based on electromagnetic force law. The original version of MOA has been designed in order to solve problems with continuous real search spaces (domains) [6],[7].

There are many optimization problems which have discrete binary search spaces. They need binary algorithms to be solved. In this paper, a binary version of MOA is introduced called BMOA in order to solve these kinds of problems. BMOA owns magnetic particles, travelling in discrete binary search spaces by changing the dimensions of their positions from "0" to "1" and vice versa.

The rest of the paper is organized as follow. Section 2 presents a brief introduction to Magnetic Optimization Algorithm. Section 3 discusses the basic principles of binary version of MOA. The experimental results are demonstrated in section 4. Finally, section 5 concludes the work and suggests some researches for future works.

Manuscript received April 19, 2012, revised May 23, 2012. This work was supported by a research grant from Ministry of Science, Technology and Innovation (MOSTI), Malaysia through E-Science Fund Scheme project number 01-01-06- SF0530, VOTE 79332.

Authors are with the Soft Computing Research Group, Faculty of Computer Science and Information Systems, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia (e-mail: ali.mirjalili@gmail.com; sitizaiton@utm.my).

II. The Magnetic Optimization Algorithm

Electromagnetic force is one of the four fundamental forces in the universe. This kind of force is available among electromagnetic particles which is directly proportional to the distance between them. In other words, this force is decreased by increasing distance between two magnetic particles. The electromagnetic force is the basic principle which MOA is inspired from [6].

In fact, MOA can be considered as a collection of search agents, magnetic particles, whose have magnetic fields and masses proportional to their values of fitness function. During generations, agents attract each other by the electromagnetic forces between them. The intensities of these forces are proportional to the magnetic fields and distances of agents.

The MOA has been mathematically modeled as follows [6]:

Suppose there is a system with N agents. The algorithm starts with randomly placing all agents in a search space. During all epochs, the electromagnetic force from u,v-th agent on i,j-th agent at a specific time t is defined as follow:

$$f^k_{(i,j),(u,v)}(t) = \frac{B_{u,v}(t)}{D\left(x^k_{i,j}(t), x^k_{u,v}(t)\right)}\left(x^k_{u,v}(t) - x^k_{i,j}(t)\right) \quad (1)$$

where $x^k_{ij}(t)$ is the k-th dimension of i,j-th agent at iteration t, $B_{u,v}(t)$ is the magnetic field of agent u,v at iteration t, $x^k_{i,j}(t)$ and $x^k_{u,v}(t)$ are k-th dimensions of i,j-th and u,v-th agents at iteration t, and D is the function for calculating distance between agents.

The D is calculated as (2):

$$D\left(x^k_{i,j}(t), x^k_{u,v}(t)\right) = \frac{1}{m}\sum_{k=1}^{m}\frac{x^k_{i,j}(t) - x^k_{u,v}(t)}{u_k - l_k} \quad (2)$$

where u_k and l_k are the upper and lower bounds of the k-th dimension of the search space, respectively, and m is the dimension of the search space.

Note that two indices *(i,j)* are used to indentify two-dimension indexing cellular topology of original MOA as shown in Fig.1(a)[6].

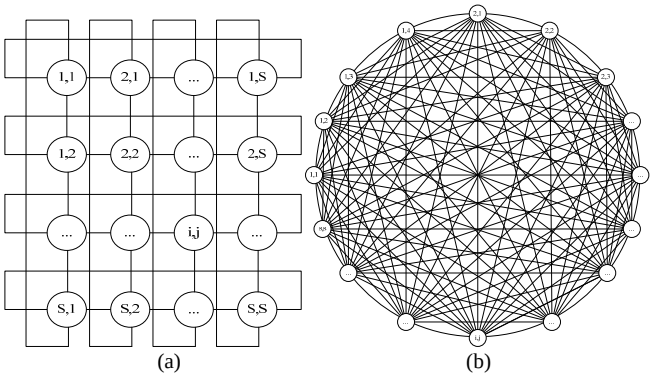

Fig. 1. Two interaction topologies: (a) cellular; (b) fully-connected

Magnetic field of *i*,*j*-th agent at the iteration *t* is calculated as (3):

$$B_{i,j} = \text{Fitness}_{i,j}(t) \tag{3}$$

where Fitness(t) might be any fitness function to solve.

In original MOA, the magnetic agents interact in a lattice (cellular topology). The employed lattice in [6] is shown in Fig.1(a). The topology defines which agents could affect each other by electromagnetic force.

An alternative to cellular topology is fully-connected topology. As it can be seen in Fig.1(b), all agents can interrelate. Fully-connected topology is utilized in this work due to some limitations of cellular topology like low and limited interaction between agents.

In a problem space with dimension *m*, the resultant force that acts on *i*,*j*-th agent is calculated as (4):

$$F_{i,j}^{m}(t) = \sum_{u,v\ in\ N_{ij}} f_{(i,j),(u,v)}^{m}(t) \tag{4}$$

where N_{ij} is the set of neighbors of the agent *i*,*j*.

According to the law of motion, the acceleration of an agent is proportional to the resultant force and inverse of its mass, so the acceleration of all agents should be calculated as (5):

$$a_{i,j}^{k}(t) = \frac{F_{i,j}^{k}(t)}{M_{i,j}(t)} \tag{5}$$

where $M_{i,j}(t)$ is the mass of *i*,*j*-th agent at iteration t.

Masses of agents are calculated as (6):

$$M_{i,j}(t) = \alpha + \rho \times B_{i,j}(t) \tag{6}$$

where α and ρ are constant values.

The velocity and position of agents are updated as follows:

$$v_{i,j}^{k}(t+1) = rand \times\ v_{i,j}^{k}(t) +\ a_{i,j}^{k}(t) \tag{7}$$

$$x_{i,j}^{k}(t+1) = x_{i,j}^{k}(t) +\ v_{i,j}^{k}(t+1) \tag{8}$$

where $rand$ is a random number in the interval [0,1].

In MOA, at first all agents are initialized with random values. Each agent is considered as a candidate solution. The following steps continuously run until meeting an end criterion. Magnetic fields and masses for all agents are defined using (3) and (6). After that, agents' total forces and accelerations are calculated as (4) and (5), respectively. Then, the velocities and positions are updated using (7) and (8).

To see how MOA is efficient, some remarks are noted:

- The quality of solution (fitness) is considered in the velocity updating procedure.
- In updating the agents' velocities, the velocity of the previous iteration is considered. This consideration makes agents capable of exploring new areas of a search space.
- An agent near a good solution has a high intense magnetic field. Higher intensity of magnetic field causes greater electromagnetic attraction force. Therefore, agents tend to move toward the best agent.
- Because masses of agents are a function of magnetic field, agents near good solutions become heavy. Hence, they move slowly and search the search space more locally.
- Original MOA uses cellular topology to define the interrelation of agents. Hence, a trapped agent in a local optimum could not attract the other agents to be trapped in the same local optimum.

The above-mentioned remarks make MOA powerful enough to solve wide range optimization problems [6]. However, the original MOA is a continuous algorithm which is not capable of solving binary problems.

III. Binary Magnetic Optimization Algorithm

A binary search space could be considered as a hypercube. The agents of a binary optimization algorithm might only shift to nearer and farther corners of the hypercube by flipping various numbers of bits [8]. Hence, for designing binary version of MOA, some basic concepts such as velocity and position updating process should be modified.

In the original MOA, agents could move around the search space because of having position vectors with continuous real domain. Consequently, the concept of position updating can be easily implemented for agents adding velocities to positions using (8). However, the meaning of position updating is different in a discrete binary space. In binary space, due to dealing with only two numbers ("0" and "1"), the position updating process cannot be done using (8). Therefore, we have to find a way to use velocities to change agents' positions from "0" to "1" or vice versa. In other words, we have to find a link between velocity and position, as well as revise (8).

Basically, in discrete binary space, the position updating means a swithcing between "0" and "1" values. This switching should be done based on velocities of agents. The question here is that how the concept of velocity in real space should be employed in order to update positions in binary space. According to [8] and [9], the idea is to change position of an agent with the probability of its velocity. In order to do this, a transfer function is necessary to map the velocities values to probability values for updating the positions.

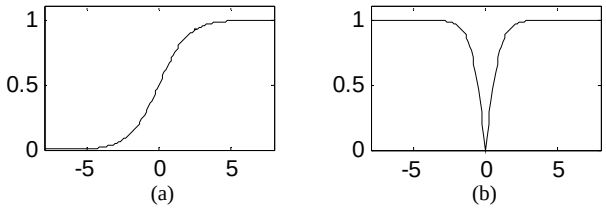

Fig. 2. Two transfer functions: (a) sigmoid; (b) tangent hyperbolic

As mentioned above, transfer functions define the probability of changing position vector's elements from "0" to "1" and vice versa. Transfer functions force agents to move in a binary space. According to [9], some concepts should be taken into account for selecting a transfer function in order to map velocity values to probability values.

The transfer function should be able to provide a high probability of changing the position for a large absolute value of the velocity. It should also present a small probability of changing the position for a small absolute value of the velocity. Moreover, the range of a transfer function should be bounded in the interval [0,1] and increased with the increasing of velocity. The functions that have been used in [8] and [9] are presented as (9) and (10). These functions are also depicted in Fig.2(a) and (b).

$$S\left(v_{i,j}^{k}(t)\right) = \frac{1}{1+e^{-v_{i,j}^{k}(t)}} \tag{9}$$

$$S\left(v_{i,j}^{k}(t)\right) = \left|tanh\left(v_{i,j}^{k}(t)\right)\right| \tag{10}$$

We use (10) in order to map velocities of BMOA algorithm's agents to probabilities of flipping their position vectors' elements. After calculating the probabilities, the agents update their positions based on the presented rules in (11).

$$\begin{aligned} &\text{If } rand < S\left(v_{i,j}^{k}(t+1)\right) \\ &\text{then } x_{i,j}^{k}(t+1) = complement\left(x_{i,j}^{k}(t)\right) \\ &\text{else } x_{i,j}^{k}(t+1) = x_{i,j}^{k}(t) \end{aligned} \tag{11}$$

According to [8], to achieve a good convergence rate, the velocity should be limited. So, the maximum velocity for BMOA in this work is set to 6.

The general steps of BMOA are as follows:

a) All agents are initialized with random values
b) Repeat steps c-g until end condition is met
c) For all agents, magnetic fields and masses are defined using (3) and (6)
d) Agents' total forces and accelerations are calculated as (4) and (5) based on interaction topology
e) Velocities of agents are updated using (7)
f) Probabilities for changing elements of position vectors are calculated as (10)
g) Update the elements of position vectors based on the rules in (11).

IV. Experimental Results and Discussions

"In order to evaluate the performance of BMOA, it is applied to 4 standard benchmark functions [10], and the results are compared with BPSO and GA. Table I lists down these benchmark functions and the range of their search space. Fig.3(a), Fig.3(b), Fig.3(c), and Fig.3(d) illustrate them, Spherical, Rastrigin, Rosenbrock, and Griewank functions, respectively. Furthermore, function's dimension is set to 5 (m=5). To represent each continuous variable, 15 bits are used. It should be noticed that one bit is reserved for the sign of each functions' dimension. Therefore, the dimension of agents are 75 (Dim=m×15).

To further investigate the effect of topology on BMOA, BMOA1 is implemented based on fully-connected topology, while BMOA2 is based on the cellular topology.

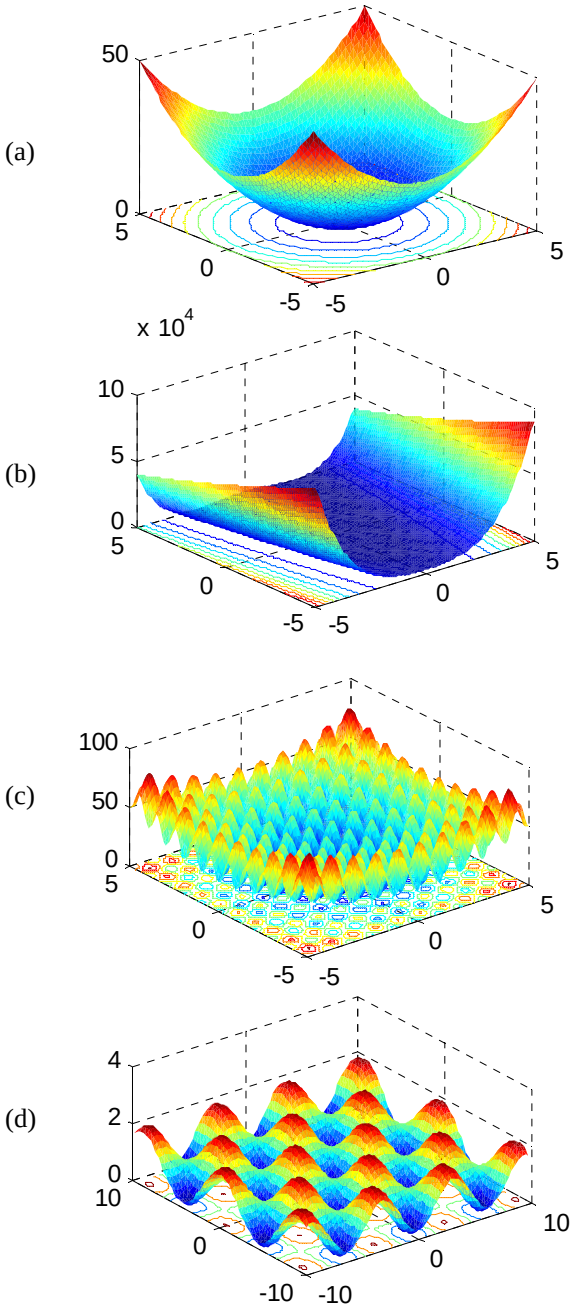

Fig. 3. The 2-D versions of benchmark functions: (a) F1; (b) F2; (c) F3; (d) F4

TABLE I: BENCHMARK FUNCTIONS

Function	Range
$F_1(x) = \sum_{i=1}^{n} x_i^2$	[-100,100]^m
$F_2(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-30,30]^m
$F_3(x) = \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$	[-5.12,5.12]^m
$F_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600,600]^m

As shown in Fig.1(a), in cellular topology, an agent could interact only with its neighbors. The number of these neighbors is limited in cellular topology. In this work we let agents to interact with 4 other agents. According to Fig.1(b), the fully-connected topology does not have any limitation, so all agents can interact together without any restriction.

In this paper, our objective is minimization. The global minimum values for all appeared functions in Table I are 0.

BMOA, BPSO, and GA have several parameters which should be initialized before running. Table II shows the initial values of basic parameters for these algorithms.

TABLE II: INITIAL PARAMETERS FOR BMOA, BPSO, AND GA

Algorithm	Parameter	Value
BMOA	Number of gents	30
	α	1
	ρ	5.5
	Max interations	500
	Max velocity	6
	Stoppig criteria	Max iteration
BPSO[11]	Number of articles	30
	C_1,C_2	2,2
	W	is decreased lineraly from 0.9 to 0.4
	Max interations	500
	Max velocity	6
	Stoppig criteria	Max interation
GA [12]	Number of individuals	30
	Selection	Roulette wheel
	Crossover(probability)	One-point (0.9)
	Mutation(probability)	Uniform (0.005)
	Max generation	500
	Stoppig criteria	Max interation

The experimental results are presented in Table III. The results are averaged over 30 runs and the best results are indicated in bold type.

TABLE III: MINIMIZATION RESULTS OF FOUR BENCHMARK FUNCTIONS OVER 30 INDEPENDENT RUNS

F		BMOA1	BMOA2	BPSO	GA
F1	*Std* [a]	11.6676	66.6773	**2.7657**	24.9445
	Ave [b]	**2.6803**	24.7982	5.2965	10.0705
	Med [c]	**0.065**	0.8535	4.6684	2.6534
F2	*Std*	**62.2039**	4093.7667	91.8494	26589.1496
	Ave	**30.3040**	1314.4107	130.7784	7562.9363
	Med	**12.8189**	299.264	39.7398	170.3704
F3	*Std*	**1.0925**	1.607	1.4467	8.3294
	Ave	**1.79**	2.6518	3.4206	6.9265
	Med	**1.9903**	2.4597	3.2541	4.522
F4	*Std*	**0.0725**	0.1825	0.1302	0.3223
	Ave	**0.0958**	0.2728	0.3873	0.7067
	Med	**0.0726**	0.224	0.387	0.7336

a. Indicates standard deviation of best so far solution over 30 runs in the last iteration
b. Indicates average best so far solution over 30 runs in the last iteration
c. Indicates median best so far solution over 30 runs in the last iteration

For the functions F1 and F2, BMOA1 reaches much accurate results than BMOA2 and GA. The BMOA1 also performs better than BPSO. The functions F1 and F2 belong to family of unimodal functions which are monotonous functions without any local solution. As shown in Fig.3(a) and Fig3(b), there is only one global solution for these kinds of functions. Hence, the results show BMOA2 has a good ability to exploit the global minimum. Moreover, Fig.4(a) and Fig.4(b) prove that BMOA1 possesses good convergence rate, too.

In addition, BMOA2 that uses cellular topology does not perform as good as BPSO and GA. This could due to several reasons. In limited interaction of agents of cellular topology, an agent near a good solution could not attract the other agents to move toward it. Another reason is related to the high-dimensional nature of binary optimizations problems. In binary problems, the problem dimensions are generally higher than the real ones (in this paper 75). So, the binary optimization problems are more complex than the real ones. This complexity needs different topologies. The results show fully-connected topology is a good topology in this field.

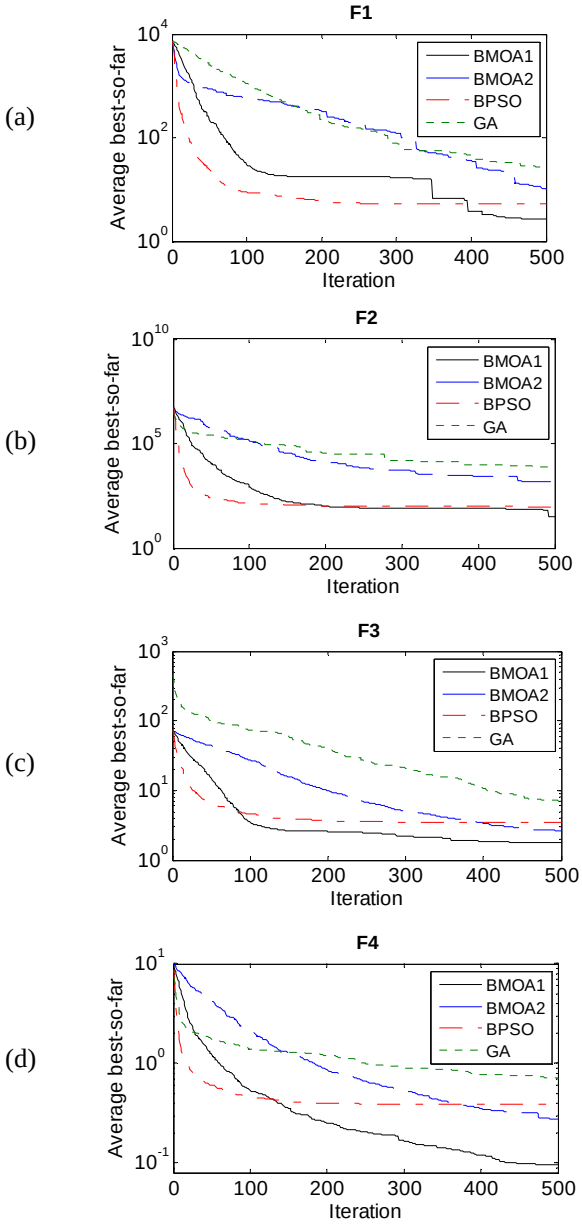

Fig. 4. Convergence curves of the algorithms on (a) F1; (b) F2; (c) F3; (d) F4

For functions F3 and F4, both BMOA1 and BMOA2 outperform the other algorithms. Meanwhile, BMOA1 shows the best result. Functions F3 and F4 are multimodal functions that have many local solutions in comparison with unimodal functions. Hence, it can be said that both versions of BMOA have good ability to avoid local minima. Fig.4(c) and Fig.4(d) illustrate that BMOA1 uses good convergence speed in multimodal functions, too.

To summarize, results prove BMOA1 and BMOA2 have better performance than BPSO and GA. In addition, BMOA1 performs better than BMOA2. It can be concluded that the employed fully-connected topology is more suitable than cellular topology for the BMOA.

V. Conclusion

In this paper, a binary version of MOA called BMOA is introduced utilizing the original MOA. Two versions of BMOA with different topologies are developed. In order to justify the performance of both versions, four benchmark functions are employed, and the results are compared with BPSO and GA. The results prove that BMOA with fully-connected topology has merit among heuristic optimization algorithms in binary search spaces.

For future studies, it is recommended to use other topologies in the population-based algorithms such as lbest topology PSO. It is also suggested to apply BMOA in real optimization problems to evaluate the efficiencies of BMOA in solving real world problems like travelling salesman problem.

References

[1] J Kennedy and RC. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE international conference on neural networks*, vol. 4, 1995, pp. 1942–1948.

[2] JH. Holland, *Adaptation in natural and artificial systems*. Ann Arbor, Michigan: The University of Michigan Press, 1975.

[3] S. Kirkpatrick, C. D. Gelati, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[4] M. Dorigo, V. Maniezzo, and A. Colorni, "The Ant system: optimization by a colony of cooperating agents," *IEEE Trans Syst Man Cybern B*, vol. 26, no. 1, pp. 29-41, 1996.

[5] D. H. Wolpert and W.G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67 - 82, 1997.

[6] M. H. Tayarani-N and M.R. Akbarzadeh-T, "Magnetic Optimization Algorithms a new synthesis," in *IEEE Congress on Evolutionary Computation*, 2008, pp. 2659 - 2664.

[7] M. H. Tayarani-N and M.R. Akbarzadeh-T, "Magnetic Optimization Algorithms a new synthesis," in *IEEE Congress on Evolutionary Computation*, 2008, pp. 2659 - 2664.

[8] J. Kennedy and R.C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *IEEE International Conference on Computational Cybernetics and Simulation*, vol. 5, 1997, pp. 4104 - 4108.

[9] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "BGSA: binary gravitational search algorithm," *Natural Computing*, vol. 9, no. 3, pp. 727-745, 2009.

[10] X. Yao, Yong Liu, and Guangming Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82 - 102, 1999.

[11] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proc. Congress on Evolutionary Computation 2001*, Seoul, Korea, 2001.

[12] DE Goldberg, *Genetic algorithms in search, optimization, and machine learning*.: Addison-Wesley, Reading, MA, 1989.

SeyedAli Mirjalili was born in Yazd, Iran on April 21, 1986. In 2008, he received B.Sc degree in Software Engineering from Yazd University, Yazd, Iran. He obtained M.Sc degree in Computer Science from Universiti Teknologi Malaysia (UTM), Johor Bahru, Malaysia, in 2011.

He is currently a Ph.D student at School of Information and Communication Technology, Griffith University, Nathan, Queensland, Australia. His research interests include Multi-objective Optimization, Robust Optimization, Evolutionary Algorithms, and Artificial Neural Networks.

Siti Zaiton Mohd Hashim obtained her BSc in Computer Science in 1990, MSc in Computer Science in 1997, and PhD (Soft Computing in Control) in 2005 from University of Hartford, USA, University of Bradford, UK, and Sheffield University, UK consecutively. Her research interests are Soft Computing and Intelligent Systems. She is currently the head of Office of Postgraduate Studies, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia. She is now actively doing research on RFID middleware and soft computing fundamental research.