A Novel Monarch Butterfly Optimization with Greedy Strategy and Self-adaptive Crossover Operator

Gai-Ge Wang*
School of Computer Science and Technology
Jiangsu Normal University
Xuzhou, China
E-mail: gaigewang@163.com

Xinchao Zhao
School of Science
Beijing University of Posts and Telecommunications
Beijing, China
Email: xcmmrc@gmail.com

Suash Deb
Dept. of Computer Science & Engineering
Cambridge Institute of Technology
Ranchi, INDIA
Email: suashdeb@gmail.com

***Abstract*—Recently, inspired by migration of monarch butterflies in the Northern American, a new kind of metaheuristic algorithm, called monarch butterfly optimization (MBO), is proposed for solving global optimization problems. It has been experimentally shown that MBO outperforms five state-of-the-art metaheuristic algorithms on most benchmarks. However, the main disadvantage of MBO is that it has poorer Std values and worse mean fitness on certain benchmarks. In this paper, in order to overcome this shortcoming, a greedy strategy is incorporated into the migration operation, and this incorporated strategy can only accept the monarch butterfly individuals that have better fitness than their parents. In addition, a self-adaptive crossover (SAC) operator is incorporated into the butterfly adjusting operator, and this SAC operator can significantly improve the diversity of population at later run phase of the search. In butterfly adjusting operator, the greedy strategy is also used to select the fitter monarch butterfly individual, which can accelerate convergent speed. Accordingly, a new version of *M*onarch *B*utterfly *O*ptimization with *G*reedy strategy and self-adaptive *C*rossover operator (GCMBO) is proposed. Finally, the proposed GCMBO method is benchmarked by eighteen standard test functions. The results indicate that GCMBO method significantly outperforms the basic MBO method on almost all the test cases.**

Keywords- Monarch butterfly optimization; Migration; Butterfly adjusting operator; Greedy strategy; Crossover; Benchmark problems

I. INTRODUCTION

Optimization is to select a certain function value within a given domain. With the development of society, optimization is becoming more and more complicated so that they are difficult to be solved by the traditional methods. Modern metaheuristic algorithms [1-3] are used to solve these complicated optimization problems.

Since genetic algorithms (GAs) [4, 5] are proposed in the 1960s, various metaheuristic algorithms are put forward and used to successfully address many complicated engineering problems, such as scheduling [6], path planning [7], directing orbits of chaotic systems [8], task assignment problem [9, 10], feature selection [11], wind generator optimization [12], reliability problems [13, 14], knapsack problem [15], and fault diagnosis [16]. Among different kinds of metaheuristic algorithms, swarm intelligence (SI) methods [17] are one of the most representative paradigms. Recently, several excellent SI methods have been proposed , such as ant colony optimization (ACO) [18, 19], particle swarm optimization (PSO) [20-26], artificial bee colony (ABC) [27, 28], cuckoo search (CS) [29-34], bat algorithm (BA) [35-38], grey wolf optimizer (GWO) [39-41], ant lion optimizer (ALO) [42], wolf search algorithm (WSA) [43], multi-verse optimizer (MVO) [44], dragonfly algorithm (DA) [45], firefly algorithm (FA) [46-49], animal migration optimization (AMO) [50], krill herd (KH) [51-57], earthworm optimization algorithm (EWA) [58], and monarch butterfly optimization (MBO) [59].

Through the study of the migration of monarch butterfly, monarch butterfly optimization (MBO) [59], is recently proposed by Wang et al. In MBO method, the butterflies in Land 1 and Land 2 are respectively updated by migration operator and butterfly adjusting operator. These optimization process are repeated until certain stop condition is met. By comparing with five other algorithms on thirty-eight benchmark problems, the experiments show that MBO outperforms five other metaheuristic methods on most cases.

However, MBO has worse Std values and average fitness on some benchmarks [59]. In order to overcome this shortcoming, a new variant of MBO, called GCMBO, is proposed in this paper. In GCMBO, two optimization strategies are used to improve the basic MBO method. Firstly, an improved crossover operator, called self-adaptive crossover (SAC) operator, is incorporated into the butterfly adjusting operator with the aim of increasing the diversity of population at the later search phase. This SAC operator can also take fully use of the whole population information. Secondly, except SAC operator, a greedy strategy is incorporated into the migration operator and butterfly adjusting operator, and this incorporated greedy strategy can only accept the monarch butterfly individuals that have better fitness than their parents. While, all the updating monarch butterfly individuals are accepted in the basic MBO. Therefore, it is no doubt that the greedy strategy can significantly speed up convergence. In order to demonstrate the superiority of GCMBO method, an array of experiments are implemented on eighteen test cases. The results show that GCMBO is well capable of finding much fitter monarch butterfly individuals than the basic MBO on almost all the benchmarks.

The rest of paper is structured as follows. Section 2 reviews the main process of the basic MBO method, and Section 3 discusses how the greedy strategy and self-adaptive crossover operator can be used to improve the performance of MBO. Subsequently, GCMBO is fully investigated on 20-D and 40-D benchmarks in Section 4. Section 5 presents some concluding remarks and suggestions for further work.

978-1-4673-9819-0/15 $31.00 © 2015 IEEE
DOI 10.1109/ISCMI.2015.19

II. MONARCH BUTTERFLY OPTIMIZATION

The migration of monarch butterfly is simplified into the following rules in the basic MBO [59].

1) The monarch butterflies in Land 1 and Land 2 are composed of the whole population.

2) Each offspring is only generated by the butterfly in Land 1 or in Land 2.

3) The size of the butterfly population keep unchanged during the optimization process.

4) Certain number of the fittest monarch butterfly individuals are not updated by migration operator or butterfly adjusting operator [59].

A. Migration operator

According to the time when monarch butterflies stay at Land 1 and Land 2, their numbers in Land 1 and Land 2 can be considered as ceil(p*NP) (NP_1, Subpopulation 1) and NP-NP_1 (NP_2, Subpopulation 2), respectively. Here, ceil(x) rounds x to the nearest integer greater than or equal to x; NP is the population size; p is the ratio of monarch butterflies in Land 1. This migration process can be formulated below [59].

$$x_{i,k}^{t+1} = x_{r_1,k}^{t} \tag{1}$$

where $x_{i,k}^{t+1}$ indicates the kth element of x_i at generation t+1. Similarly, $x_{r_1,k}^{t}$ indicates the kth element of x_{r_1}. t is the current generation number. Butterfly r_1 is randomly selected from Subpopulation 1. When $r \le p$, $x_{i,k}^{t+1}$ is generated by Eq. (1). Here, r can be calculated as

$$r = rand * peri \tag{2}$$

$peri$ indicates migration period and is set to 1.2 in the basic MBO method [59]. $rand$ is a random number. If $r > p$, $x_{r_1,k}^{t}$ is generated by

$$x_{i,k}^{t+1} = x_{r_2,k}^{t} \tag{3}$$

where $x_{r_2,k}^{t}$ indicates the kth element of x_{r_2}, and butterfly r_2 is randomly selected from Subpopulation 2.

B. Butterfly adjusting operator

For all the elements in butterfly j, if $rand \le p$, it can be updated as [59]

$$x_{j,k}^{t+1} = x_{best,k}^{t} \tag{4}$$

where $x_{j,k}^{t+1}$ indicates the kth element of x_j at generation t+1. Similarly, $x_{best,k}^{t}$ indicates the kth element of the fittest butterfly x_{best}. If $rand > p$, it can be updated as

$$x_{j,k}^{t+1} = x_{r_3,k}^{t} \tag{5}$$

where $x_{r_3,k}^{t}$ indicates the kth element of x_{r_3}. Here, $r_3 \in \{1, 2, \ldots, NP_2\}$.

Under this condition, if $rand > BAR$, it can be further updated as follows [59].

$$x_{j,k}^{t+1} = x_{j,k}^{t+1} + \alpha \times (dx_k - 0.5) \tag{6}$$

where BAR indicates butterfly adjusting rate. dx is the walk step of butterfly j that can be calculated by Lévy flight [59].

$$dx = Levy(x_j^t) \tag{7}$$

In Eq. (6), α is the weighting factor as shown in Eq. (8).

$$\alpha = S_{max} / t^2 \tag{8}$$

where $S_{\max}$ is max walk step.

III. GCMBO METHOD

MBO is a new robust metaheuristic algorithm, and it has shown his performance on benchmark evaluation. However, MBO may fail to reach the optimal performance on average and Std values on certain test cases [59]. In this paper, two strategies are incorporated into the basic MBO method with the aim of improving the performance of MBO. In the following, the mainframe of GCMBO method is given.

A. Updating migration operator

Similarly, the number of butterflies in Land 1 and Land 2 are ceil(p*NP) (NP_1, Subpopulation 1) and NP-NP_1 (NP_2, Subpopulation 2), respectively. The $x_{i,k}^{t+1}$ in Subpopulation 1 is generated according to Eqs. (1)-(3) [59]. In the basic MBO method, all the newly-generated butterflies are accepted, and pass into the next generation [59], while a greedy strategy is used to only accept the butterfly individuals that have better fitness in GCMBO method. This greedy strategy can be formulated as follows.

$$x_{i,new}^{t+1} = \begin{cases} x_i^{t+1}, & f(x_i^{t+1}) < f(x_i^t) \\ x_i^t, & else \end{cases} \tag{9}$$

where $x_{i,new}^{t+1}$ is a newly-generated butterfly individual for the next generation. $f(x_i^{t+1})$ and $f(x_i^t)$ are the fitness of butterfly x_i^{t+1} and x_i^t, respectively. In our current work, suppose all the problems are minimal problems.

Based on the analyses above, the updated migration operator in GCMBO method can be given in Figure 1.

for i= 1 to NP_1 (for all butterflies in Subpopulation 1) **do**
  for k=1 to D (all the elements in ith butterfly) **do**
    Generate $x_{i,k}^{t+1}$ by Eq. (1) and Eq. (3).
  end for k
  Generate $x_{i,new}^{t+1}$ by greedy strategy as Eq. (9).
end for i

Figure 1. Pseudo code of updated migration operator

B. Updating butterfly adjusting operator

For all the butterfly individuals in Subpopulation 2, they are updated as shown in Eqs. (4)-(8) [59]. After that, the self-

adaptive crossover operator and greedy strategy are incorporated into the optimization process.

With the aim of clear representation, the generated butterfly individual by Eqs. (4)-(8) is called x_{j1}^{t+1}.

In order to take full use of the information of butterfly population, a updated version of crossover operator is incorporated into the butterfly adjusting operator, which can be given as

$$x_{j2}^{t+1}=x_{j1}^{t+1}\times(1-Cr)+x_j^t\times Cr \qquad (10)$$

where x_{j2}^{t+1} is another newly-generated butterfly individual by using x_{j1}^{t+1} and x_j^t, and Cr is crossover rate. In the present work, a self-adaptive strategy is used to adjust Cr, which can be shown as follows.

$$Cr=0.8+0.2\times\frac{f(x_j^t)-f(x_{best})}{f(x_{worst})-f(x_{best})} \qquad (11)$$

where $f(x_j^t)$ is the fitness of butterfly j in Subpopulation 2; x_{best} and x_{worst} are the best and worst butterfly individual in butterfly population, and their fitness are respectively $f(x_{best})$ and $f(x_{worst})$. From Eq. (11), Cr is in [0.2, 0.8].

After that, the new butterfly individual is determined by using greedy strategy, and it can be shown as Eq. (12).

$$x_{j,new}^{t+1}=\begin{cases}x_{j1}^{t+1}, & f(x_{j1}^{t+1})<f(x_{j2}^{t+1})\\ x_{j2}^{t+1}, & f(x_{j2}^{t+1})<f(x_{j1}^{t+1})\end{cases} \qquad (12)$$

where $f(x_{j1}^{t+1})$ and $f(x_{j2}^{t+1})$ are fitness of the butterfly x_{j1}^{t+1} and x_{j2}^{t+1}, respectively. $x_{j,new}^{t+1}$ is newly-generated butterfly for the next generation.

So, the updated butterfly adjusting operator can be described in Figure 2.

for j= 1 to NP_2 (for all butterflies in Subpopulation 2) **do**
  Calculate the walk step dx by Eq. (7);
  Calculate the weighting factor by Eq. (8);
  for k=1 to D (all the elements in jth butterfly) **do**
    Generate $x_{j1,k}^{t+1}$ by Eq. (4) and Eq. (5).
  end for k
  Generate x_{j2}^{t+1} by performing SAR operator as Eq. (10).
  Generate $x_{j,new}^{t+1}$ according to greedy strategy as Eq. (12).
end for j

Figure 2. Pseudo code of updated butterfly adjusting operator

Based on the above analyses, the mainframe of GCMBO method can be provided in Figure 3.

Step 1: Initialization. Set the generation counter t=1; initialize the population P of NP butterflies; set the maximum generation $MaxGen$, butterfly number NP_1 in Land 1 and butterfly number NP_2 in Land 2.
Step 2: Fitness evaluation. Evaluate butterfly population.
Step 3: While $t<MaxGen$ **do**
  Sort the butterfly population.
  Divide all butterflies into two subpopulations (Land 1 and Land 2).
  for i=1 to NP_1 (for all butterflies in Subpopulation 1) **do**
    Generate $x_{i,new}^{t+1}$ by updated migration operator as Figure 1.
  end for i
  for j= 1 to NP_2 (for all butterflies in Subpopulation 2) **do**
    Generate $x_{j,new}^{t+1}$ by updated butterfly adjusting operator as Figure 2.
  end for j
  Evaluate the butterfly population.
  $t = t$+1.
Step 4: end while

Figure 3. Pseudo code of GCMBO algorithm

IV. SIMULATION RESULTS

In this section, the superiority of GCMBO is fully investigated through benchmark evaluations from various respects. In the present work, eighteen high-dimensional benchmarks are used as shown in Table I. In addition, all the experiments are performed under the same conditions [60, 61].

TABLE I. BENCHMARK FUNCTIONS

No.	*Name*	*No.*	*Name*
F01	Ackley	F10	Penalty #1
F02	Alpine	F11	Penalty #2
F03	Brown	F12	Perm
F04	Dixon & Price	F13	Powell
F05	Fletcher-Powell	F14	Quartic *with noise*
F06	Griewank	F15	Rastrigin
F07	Holzman 2 function	F16	Rosenbrock
F08	Levy	F17	Schwefel 2.26
F09	Pathological function	F18	Schwefel 1.2

For MBO and GCMBO, their parameters are set as follows: max step S_{max}=1.0, butterfly adjusting rate BAR=5/12,

migration period *peri*=1.2, the migration ratio *p*=5/12, and population size *NP*=50. Therefore, the number of butterflies in Land 1 and Land 2, i.e., NP_1 and NP_2, are 21 and 29, respectively. Fifty independent runs are performed with the aim of getting the most representative results. In the following experiments, the optimal solution for each test problem is highlighted.

A. D=20

Here, the dimension of test function is set to 20. The function evaluations (FEs) is considered as stop condition, and it is set to 8000. The average results achieved by MBO and GCMBO are recorded in Table II. In Table II, "11.43±6.43" indicates that the average and Std value are 11.43 and 6.43, respectively.

From Table II, it can be observed that, GCMBO significantly outperforms MBO on almost all the benchmarks (in fact, except F12 Perm function). GCMBO has better average function values and smaller Std values than MBO. The superiority of GCMBO on functions F01-F04 can also be shown in Figure 4.

B. D=40

Here, the dimension of test function is set to 40. The function evaluations (FEs) is considered as stop condition, and it is set to 16000. The average results achieved by MBO and GCMBO are recorded in Table III.

From Table III, it can be observed that, GCMBO significantly outperforms MBO on all the benchmarks. GCMBO has far better average function values and smaller Std values than MBO. This indicates that, GCMBO performs more stably and is more suitable for engineering optimization. In addition, the superiority of GCMBO on functions F01-F04 can also be shown in Figure 5.

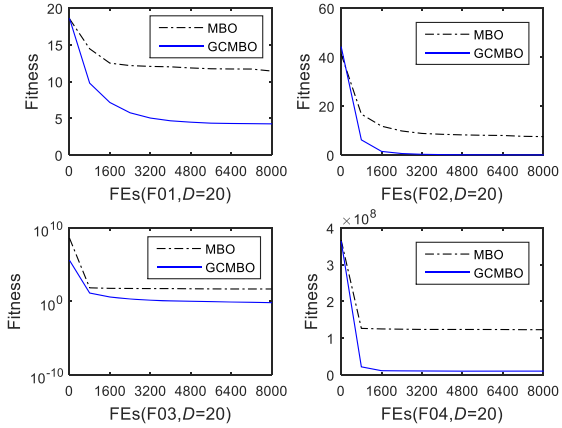

FIGURE 4. CONVERGENT PROCESS OF MBO AND GCMBO ON F01-F04 WITH *D*=20

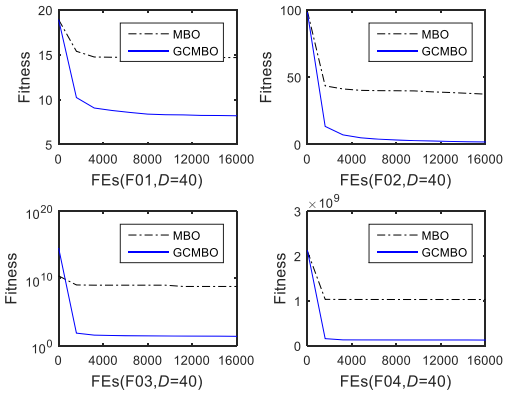

FIGURE 5. CONVERGENT PROCESS OF MBO AND GCMBO ON F01-F04 WITH *D*=40

TABLE II. MEAN FUNCTION VALUES OBTAINED BY MBO AND GCMBO METHODS (*D*=20)

	MBO	GCMBO		MBO	GCMBO
F01	11.43±6.43	**4.24±4.62**	F10	3.2E7±6.5E7	**3.1E5±1.2E6**
F02	7.51±10.37	**0.03±0.20**	F11	7.9E7±1.3E8	**1.1E6±5.6E6**
F03	48.58±102.82	**0.66±3.06**	F12	**5.9E50±1.1E51**	1.5E51±2.0E51
F04	1.2E8±1.0E8	**1.0E7±1.0E7**	F13	2.1E3±1.9E3	**435.79±562.68**
F05	3.0E5±1.1E5	**1.2E5±5.3E4**	F14	36.86±35.96	**0.09±0.31**
F06	93.72±94.71	**20.74±21.69**	F15	41.18±36.19	**7.71±8.49**
F07	6.2E4±5.9E4	**1.9E3±3.6E3**	F16	969.30±1.7E3	**69.97±116.50**
F08	20.58±33.27	**2.11±5.00**	F17	3.0E3±1.9E3	**1.0E3±1.0E3**
F09	1.62±0.94	**0.79±0.77**	F18	2.5E4±1.5E4	**1.1E4±8.5E3**

TABLE III. MEAN FUNCTION VALUES OBTAINED BY MBO AND GCMBO METHODS (D=40)

	MBO	GCMBO		MBO	GCMBO
F01	14.68±5.63	**8.19±5.25**	F10	2.9E8±3.3E8	**1.7E6±4.0E6**
F02	37.36±35.50	**1.76±4.53**	F11	5.3E8±6.3E8	**1.1E7±3.0E7**
F03	6.2E8±2.4E9	**23.87±53.71**	F12	1.4E127±3.4E127	**1.1E127±2.3E127**
F04	1.0E9±8.5E8	**1.2E8±1.3E8**	F13	7.3E3±7.9E3	**1.7E3±2.1E3**
F05	2.7E6±7.2E5	**8.4E5±2.2E5**	F14	272.95±231.23	**11.60±19.85**
F06	341.06±287.07	**65.66±71.85**	F15	162.21±113.05	**32.20±25.02**
F07	5.1E5±3.9E5	**4.7E4±5.1E4**	F16	7.3E3±8.0E3	**298.88±449.49**
F08	103.80±118.35	**12.24±18.15**	F17	8.4E3±3.8E3	**3.9E3±2.5E3**
F09	5.73±3.08	**1.97±1.67**	F18	1.2E5±6.9E4	**4.9E4±3.5E4**

V. CONCLUSION

Recently, MBO is a new kind of metaheuristic algorithm for solving global optimization problems. Though MBO outperforms five state-of-the-art metaheuristic algorithms on most benchmarks, it has poorer Std values and worse mean fitness on certain benchmarks. With the aim of overcoming this shortcoming and improve the performance of MBO method, a new version of MBO with greedy strategy and self-adaptive crossover operator (GCMBO) is proposed. In GCMBO, a greedy strategy is incorporated into the migration operator and butterfly adjusting operator. In MBO method, all the updated monarch butterfly individuals are accepted, while only the individuals that have fitter property than their parents are accepted in GCMBO through greedy strategy. This will surely significantly accelerate the convergent speed. In addition, a SAC operator is incorporated into the butterfly adjusting operator, and this SAC operator can add the diversity of population at later run phase of the search and make the butterfly population not be trapped into local optimum. Finally, the proposed GCMBO method is benchmarked by eighteen standard test functions. Though GCMBO requires more FEs at one generation, the results indicate that GCMBO method significantly outperforms the basic MBO method on almost all the test cases.

In future, the following points should be clarified and focused on. On one hand, we use only eighteen benchmarks to test our proposed GCMBO method. More benchmark should be used for testing GCMBO. On the other hand, here, GCMBO is experimentally tested only using benchmark problems. The convergence of GCMBO method will be analyzed theoretically by dynamic systems and Markov chain. This can ensure stable implementation of GCMBO method.

ACKNOWLEDGMENT

This work was supported by Jiangsu Province Science Foundation for Youths (No. BK20150239) and National Natural Science Foundation of China (No. 61503165 and No. 61375066).

REFERENCES

[1] X. S. Yang, and Z. Cui, "Bio-inspired computation: success and challenges of IJBIC," *International Journal of Bio-Inspired Computation,* vol. 6, no. 1, pp. 1-6, 2014.

[2] I. Fister, D. Strnad, X.-S. Yang, and I. Fister, Jr., "Adaptation and Hybridization in Nature-Inspired Algorithms," *Adaptation and Hybridization in Computational Intelligence*, Adaptation, Learning, and Optimization I. Fister and I. Fister Jr, eds., pp. 3-50: Springer International Publishing, 2015.

[3] I. Fister Jr, X.-S. Yang, I. Fister, J. Brest, and D. Fister, "A brief review of nature-inspired algorithms for optimization," *arXiv preprint arXiv:1307.4186*, 2013.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine learning*, Addison-Wesley, New York, 1998.

[5] X. Z. Gao, and S. J. Ovaska, "Genetic algorithm training of Elman neural network in motor fault detection," *Neural Computing & Applications,* vol. 11, no. 1, pp. 37-44, 2002.

[6] Y. Hu, M. Yin, and X. Li, "A novel objective function for job-shop scheduling problem with fuzzy processing time and fuzzy due date using differential evolution algorithm," *The International Journal of Advanced Manufacturing Technology,* vol. 56, no. 9, pp. 1125-1138, 2011.

[7] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, and M. Shao, "Path planning for uninhabited combat aerial vehicle using hybrid meta-heuristic DE/BBO algorithm," *Advanced Science, Engineering and Medicine,* vol. 4, no. 6, pp. 550-564, 2012.

[8] Z. Cui, S. Fan, J. Zeng, and Z. Shi, "APOA with parabola model for directing orbits of chaotic systems," *International Journal of Bio-Inspired Computation,* vol. 5, no. 1, pp. 67-72, 2013.

[9] D. Zou, H. Liu, L. Gao, and S. Li, "An improved differential evolution algorithm for the task assignment problem," *Engineering Applications of Artificial Intelligence,* vol. 24, no. 4, pp. 616-624, 2011.

[10] D. Zou, L. Gao, S. Li, J. Wu, and X. Wang, "A novel global harmony search algorithm for task assignment problem," *Journal of Systems and Software,* vol. 83, no. 10, pp. 1678-1688, 2010.

[11] X. Li, and M. Yin, "Multiobjective binary biogeography based optimization for feature selection using gene expression data," *IEEE Transactions on NanoBioscience,* vol. 12, no. 4, pp. 343-353, 2013.

[12] X. Z. Gao, X. Wang, T. Jokinen, S. J. Ovaska, A. Arkkio, and K. Zenger, "A hybrid PBIL-based harmony search method," *Neural Computing and Applications,* vol. 21, no. 5, pp. 1071-1083, 2012.

[13] D. Zou, L. Gao, J. Wu, S. Li, and Y. Li, "A novel global harmony search algorithm for reliability problems," *Computers & Industrial Engineering,* vol. 58, no. 2, pp. 307-316, 2010.

[14] D. Zou, L. Gao, S. Li, and J. Wu, "An effective global harmony search algorithm for reliability problems," *Expert Systems with Applications,* vol. 38, no. 4, pp. 4642-4648, 2011.

[15] D. Zou, L. Gao, S. Li, and J. Wu, "Solving 0-1 knapsack problem by a novel global harmony search algorithm," *Applied Soft Computing,* vol. 11, no. 2, pp. 1556-1564, 2011.

[16] H. Duan, and Q. Luo, "New progresses in swarm intelligence-based computation," *Int. J. Bio-Inspired Computation,* vol. 7, no. 1, pp. 26-35, 2015.

[17] Z. Cui, and X. Gao, "Theory and applications of swarm intelligence," *Neural Computing & Applications,* vol. 21, no. 2, pp. 205-206, 2012.

[18] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* vol. 26, no. 1, pp. 29-41, 1996.

[19] K. Krynicki, J. Jaen, and J. A. Mocholí, "Ant colony optimisation for resource searching in dynamic peer-to-peer grids," *International Journal of Bio-Inspired Computation,* vol. 6, no. 3, pp. 153-165, 2014.

[20] J. Kennedy, and R. Eberhart, "Particle swarm optimization," in Proceeding of the IEEE International Conference on Neural Networks, Perth, Australia, 1995, pp. 1942-1948.

[21] G. Ram, D. Mandal, R. Kar, and S. P. Ghoshal, "Optimal design of non-uniform circular antenna arrays using PSO with wavelet mutation," *International Journal of Bio-Inspired Computation,* vol. 6, no. 6, pp. 424-433, 2014.

[22] S. Mirjalili, G.-G. Wang, and L. d. S. Coelho, "Binary optimization using hybrid particle swarm optimization and gravitational search algorithm," *Neural Computing and Applications,* vol. 25, no. 6, pp. 1423-1435, 2014.

[23] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and S. Deb, "A hybrid method based on krill herd and quantum-behaved particle swarm optimization," *Neural Computing and Applications*, 2015.

[24] G.-G. Wang, A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "A novel improved accelerated particle swarm optimization algorithm for global numerical optimization," *Engineering Computations,* vol. 31, no. 7, pp. 1198-1220, 2014.

[25] X. Zhao, B. Song, P. Huang, Z. Wen, J. Weng, and Y. Fan, "An improved discrete immune optimization algorithm based on PSO for QoS-driven web service composition," *Applied Soft Computing,* vol. 12, no. 8, pp. 2208-2216, 2012.

[26] X. Zhao, "A perturbed particle swarm algorithm for numerical optimization," *Applied Soft Computing,* vol. 10, no. 1, pp. 119-124, 2010.

[27] D. Karaboga, and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization,* vol. 39, no. 3, pp. 459-471, 2007.

[28] X. Li, and M. Yin, "Self-adaptive constrained artificial bee colony for constrained numerical optimization," *Neural Computing and Applications,* vol. 24, no. 3-4, pp. 723-734, 2012.

[29] X. S. Yang, and S. Deb, "Cuckoo search via Lévy flights." pp. 210-214.

[30] A. Ouaarab, B. Ahiod, and X.-S. Yang, "Discrete cuckoo search algorithm for the travelling salesman problem," *Neural Computing and Applications,* vol. 24, no. 7-8, pp. 1659-1669, 2014.

[31] X.-S. Yang, and S. Deb, "Cuckoo search: recent advances and applications," *Neural Computing and Applications,* vol. 24, no. 1, pp. 169-174, 2013.

[32] X. Li, J. Wang, and M. Yin, "Enhancing the performance of cuckoo search algorithm using orthogonal learning method," *Neural Computing and Applications,* vol. 24, no. 6, pp. 1233-1247, 2013.

[33] G.-G. Wang, A. H. Gandomi, X. Zhao, and H. C. E. Chu, "Hybridizing harmony search algorithm with cuckoo search for global numerical optimization," *Soft Computing*, 2014.

[34] G.-G. Wang, S. Deb, A. H. Gandomi, Z. Zhang, and A. H. Alavi, "Chaotic cuckoo search," *Soft Computing*, 2015.

[35] X. S. Yang, *Nature-inspired metaheuristic algorithms*, 2nd ed., Luniver Press, Frome, 2010.

[36] S. Mirjalili, S. M. Mirjalili, and X.-S. Yang, "Binary bat algorithm," *Neural Computing and Applications,* vol. 25, no. 3-4, pp. 663-681, 2013.

[37] J.-W. Zhang, and G.-G. Wang, "Image matching using a bat algorithm with mutation," *Applied Mechanics and Materials,* vol. 203, no. 1, pp. 88-93, 2012.

[38] G. Wang, and L. Guo, "A novel hybrid bat algorithm with harmony search for global numerical optimization," *Journal of Applied Mathematics,* vol. 2013, pp. 1-21, 2013.

[39] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software,* vol. 69, pp. 46-61, 2014.

[40] S. Saremi, S. Z. Mirjalili, and S. M. Mirjalili, "Evolutionary population dynamics and grey wolf optimizer," *Neural Computing and Applications*, 2014.

[41] S. Mirjalili, "How effective is the Grey Wolf optimizer in training multi-layer perceptrons," *Applied Intelligence,* vol. 43, no. 1, pp. 150-161, 2015.

[42] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software,* vol. 83, pp. 80-98, 2015.

[43] S. Fong, S. Deb, and X.-S. Yang, "A heuristic optimization method inspired by wolf preying behavior," *Neural Computing and Applications,* vol. 26, no. 7, pp. 1725-1738, 2015.

[44] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: a nature-inspired algorithm for global optimization," *Neural Computing and Applications*, 2015.

[45] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications*, 2015.

[46] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Mixed variable structural optimization using firefly algorithm," *Computers & Structures,* vol. 89, no. 23-24, pp. 2325-2336, 2011.

[47] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation,* vol. 2, no. 2, pp. 78-84, 2010.

[48] G.-G. Wang, L. Guo, H. Duan, and H. Wang, "A new improved firefly algorithm for global numerical optimization," *Journal of Computational and Theoretical Nanoscience,* vol. 11, no. 2, pp. 477-485, 2014.

[49] L. Guo, G.-G. Wang, H. Wang, and D. Wang, "An effective hybrid firefly algorithm with harmony search for global numerical optimization," *The Scientific World Journal,* vol. 2013, pp. 1-10, 2013.

[50] X. Li, J. Zhang, and M. Yin, "Animal migration optimization: an optimization algorithm inspired by animal migration behavior," *Neural Computing and Applications,* vol. 24, no. 7-8, pp. 1867-1877, 2014.

[51] A. H. Gandomi, and A. H. Alavi, "Krill herd: a new bio-inspired optimization algorithm," *Communications in Nonlinear Science and Numerical Simulation,* vol. 17, no. 12, pp. 4831-4845, 2012.

[52] G.-G. Wang, A. H. Gandomi, and A. H. Alavi, "A chaotic particle-swarm krill herd algorithm for global numerical optimization," *Kybernetes,* vol. 42, no. 6, pp. 962-978, 2013.

[53] A. H. Gandomi, S. Talatahari, F. Tadbiri, and A. H. Alavi, "Krill herd algorithm for optimum design of truss structures," *International Journal of Bio-Inspired Computation,* vol. 5, no. 5, pp. 281-288, 2013.

[54] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and G.-S. Hao, "Hybrid krill herd algorithm with differential evolution for global numerical optimization," *Neural Computing and Applications,* vol. 25, no. 2, pp. 297-308, 2014.

[55] G.-G. Wang, A. H. Gandomi, and A. H. Alavi, "An effective krill herd algorithm with migration operator in biogeography-based optimization," *Applied Mathematical Modelling,* vol. 38, no. 9-10, pp. 2454-2462, 2014.

[56] G.-G. Wang, L. Guo, A. H. Gandomi, G.-S. Hao, and H. Wang, "Chaotic krill herd algorithm," *Information Sciences,* vol. 274, pp. 17-34, 2014.

[57] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and S. Deb, "A Multi-Stage Krill Herd Algorithm for Global Numerical Optimization," *International Journal on Artificial Intelligence Tools*, 2015.

[58] G.-G. Wang, S. Deb, and L. d. S. Coelho, "Earthworm optimization algorithm: a bio-inspired metaheuristic algorithm for global optimization problems," *International Journal of Bio-Inspired Computation*, 2015.

[59] G.-G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Computing and Applications*, 2015.

[60] G. Wang, L. Guo, H. Wang, H. Duan, L. Liu, and J. Li, "Incorporating mutation scheme into krill herd algorithm for global numerical optimization," *Neural Computing and Applications,* vol. 24, no. 3-4, pp. 853-871, 2014.

[61] G.-G. Wang, A. H. Gandomi, and A. H. Alavi, "Stud krill herd algorithm," *Neurocomputing,* vol. 128, pp. 363-370, 2014.