Neural Comput & Applic (2019) 31:1995–2014
DOI 10.1007/s00521-015-1923-y

Monarch butterfly optimization

Gai-Ge Wang[1,2,3] · **Suash Deb**[4] · **Zhihua Cui**[5]

Received: 27 February 2015 / Accepted: 5 May 2015 / Published online: 19 May 2015

Abstract In nature, the eastern North American monarch population is known for its southward migration during the late summer/autumn from the northern USA and southern Canada to Mexico, covering thousands of miles. By simplifying and idealizing the migration of monarch butterflies, a new kind of nature-inspired metaheuristic algorithm, called monarch butterfly optimization (MBO), a first of its kind, is proposed in this paper. In MBO, all the monarch butterfly individuals are located in two distinct lands, viz. southern Canada and the northern USA (Land 1) and Mexico (Land 2). Accordingly, the positions of the monarch butterflies are updated in two ways. Firstly, the offsprings are generated (position updating) by migration operator, which can be adjusted by the migration ratio. It is followed by tuning the positions for other butterflies by means of butterfly adjusting operator. In order to keep the population unchanged and minimize fitness evaluations, the sum of the newly generated butterflies in these two ways remains equal to the original population. In order to demonstrate the superior performance of the MBO algorithm, a comparative study with five other metaheuristic algorithms through thirty-eight benchmark problems is carried out. The results clearly exhibit the capability of the MBO method toward finding the enhanced function values on most of the benchmark problems with respect to the other five algorithms. Note that the source codes of the proposed MBO algorithm are publicly available at GitHub (https://github.com/ggw0122/Monarch-Butterfly-Optimization, C++/MATLAB) and MATLAB Central (http://www.mathworks.com/matlabcentral/fileexchange/50828-monarch-butterfly-optimization, MATLAB).

Keywords Evolutionary computation · Monarch butterfly optimization · Migration · Butterfly adjusting operator · Benchmark problems

✉ Gai-Ge Wang
gaigewang@163.com; gaigewang@gmail.com

Suash Deb
suashdeb@gmail.com

Zhihua Cui
cuizhihua@gmail.com

[1] School of Computer Science and Technology, Jiangsu Normal University, Xuzhou 221116, Jiangsu, China

[2] Institute of Algorithm and Big Data Analysis, Northeast Normal University, Changchun 130117, China

[3] School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China

[4] Cambridge Institute of Technology, Cambridge Village, Tatisilwai, Ranchi 835103, Jharkhand, India

[5] Complex System and Computational Intelligence Laboratory, Taiyuan University of Science and Technology, Taiyuan 030024, Shanxi, China

1 Introduction

In the areas of computer science, mathematics, control and decision making, a relatively new set of algorithms, called nature-inspired algorithms, has been proposed and used to address an array of complex optimization problems. Among various nature-inspired algorithms, swarm-based algorithms and evolutionary algorithms (EAs) are two of the most representative paradigms.

Swarm-based algorithms, also called swarm intelligence (SI) methods [1], are one of the most well-known paradigms in nature-inspired algorithms which have been widely used in various applications, such as scheduling,

directing orbits of chaotic systems [2], wind generator optimization [3] and fault diagnosis [4]. Swarm intelligence (SI) concerns the collective, emerging behavior of multiple, interacting agents who follow some simple rules [5]. Two of widely used SI are particle swarm optimization (PSO) [6–10] and ant colony optimization (ACO) [11, 12]. The idea of PSO [6] originated from the social behavior of bird flocking when searching for the food. The ants in nature are well capable of keeping the past paths in mind by pheromone. Inspired by this phenomenon, the ACO algorithm [11] is proposed by Dorigo et al. Recently, more superior SI algorithms have been proposed, such as artificial bee colony (ABC) [13, 14], cuckoo search (CS) [15–19], bat algorithm (BA) [20–22], grey wolf optimizer (GWO) [23, 24], ant lion optimizer (ALO) [25], firefly algorithm (FA) [26–29], chicken swarm optimization (CSO) [30] and krill herd (KH) [31–33]. These are inspired by the swarm behavior of honey bees, cuckoos, bats, grey wolves, chickens and krill, respectively.

By simplifying and idealizing the genetic evolution process, different kinds of EAs have been proposed and used in a wide range of applications. Genetic algorithm (GA) [34, 35], evolutionary programming (EP) [36, 37], genetic programming (GP) [38] and evolutionary strategy (ES) [39] are four of the most classical EAs among them. With the development of the evolutionary theory, some new methods have been proposed over the last decades that significantly improved the theory and search capacities of EAs. Differential evolution (DE) [40, 41] is a very efficient search algorithm that simulates the biological mechanisms of natural selection and mutation. The best-to-survive criteria are adopted in the above algorithms on a population of solutions. Stud genetic algorithm (SGA) [42, 43] is a special kind of GA that uses the best individual and the other randomly selected individuals at each generation for crossover operator. By incorporating the sole effect of predictor variable as well as the interactions between the variables into the GP, Gandomi and Alavi [44] proposed an improved version of GP algorithm, called multi-stage genetic programming (MSGP), for nonlinear system modeling. Recently, motivated by the natural biogeography, Simon has provided the mathematics of biogeography and accordingly proposed a new kind of EA: biogeography-based optimization (BBO) [45–49]. Inspired by the animal migration behavior, animal migration optimization (AMO) [50] is proposed and compared with other well-known heuristic search methods.

By simulating the migration behavior of the monarch butterflies in nature, a new kind of nature-inspired metaheuristic algorithm, called MBO, is proposed for continuous optimization problems in this paper. In MBO, all the monarch butterfly individuals are idealized and located in two lands only, viz. Southern Canada and the northern USA (Land 1) and Mexico (Land 2). Accordingly, the positions of the monarch butterflies are updated in two ways. At first, the offsprings are generated (position updating) by migration operator, which can be adjusted by the migration ratio. Subsequently, the positions of other butterflies are tuned by butterfly adjusting operator. In other words, the search direction of the monarch butterfly individuals in MBO algorithm is mainly determined by the migration operator and butterfly adjusting operator. Also, migration operator and butterfly adjusting operator can be implemented simultaneously. Therefore, the MBO method is ideally suited for parallel processing and well capable of making trade-off between intensification and diversification, a very important phenomenon in the field of metaheuristics. In order to demonstrate the performance of MBO method, it is compared with five other metaheuristic algorithms through thirty-eight benchmark problems. The results clearly show that the MBO method is able to find the better function values on most benchmark problems as compared to five other metaheuristic algorithms.

The goal of this paper is twofold. Firstly, the new optimization method called MBO is introduced. It is carried out by first studying the migration behavior of monarch butterflies and then generalizing it to formulate a general-purpose metaheuristic method. Secondly, a comparative study of the performance of MBO with respect to other population-based optimization methods is done. This has been addressed by looking at the commonalities and differences from an algorithmic point of view as well as by comparing their performances on an array of benchmark functions.

Section 2 reviews the migration behavior of monarch butterflies in nature, and Sect. 3 discusses how the migration behavior of monarch butterflies can be used to formulate a general-purpose search heuristic. Several simulation results comparing MBO with other optimization methods for general benchmark functions are presented in Sect. 4. Finally, Sect. 5 presents some concluding remarks along with scope for improvements and expansion of the present work.

2 Monarch butterfly and its migration behavior

As one of the most familiar North American butterflies, the monarch butterfly has an orange and black pattern that can be easily recognized [51]. It is a milkweed butterfly in the family Nymphalidae. Female and male monarchs have different wings that can be used to identify them.

The eastern North American monarch is known for its ability of migrating by flying thousands of miles from the USA and southern Canada to Mexico every summer. It

involves which flying over west of the Rocky Mountains to California. In order to overwinter, they move thousands of miles to Mexico. Southward movements commence in August and end at the first frost. However, during the spring, opposite things happen. The female ones lay eggs for generating offspring during these movements [52]. Recent research shows some butterflies perform Lévy flight when they migrate or move [53].

3 Monarch butterfly optimization

In order to make the migration behavior of monarch butterflies address various optimization problems, the migration behavior of monarch butterflies can be idealized into the following rules.

1. All the monarch butterflies are only located in Land 1 or Land 2. That is to say, monarch butterflies in Land 1 and Land 2 make up the whole monarch butterfly population.
2. Each child monarch butterfly individual is generated by migration operator from monarch butterfly in Land 1 or in Land 2.
3. In order to keep the population unchanged, an old monarch butterfly will pass away once a child is generated. In the MBO method, this can be performed by replacing its parent with newly generated one if it has better fitness as compared to its parent. On the other hand, the newly generated one is liable to be discarded if it does not exhibit better fitness with respect to its parent. Under this scenario, the parent is kept intact and undestroyed.
4. The monarch butterfly individuals with the best fitness moves automatically to the next generation, and they cannot be changed by any operators. This can guarantee that the quality or the effectiveness of the monarch butterfly population will never deteriorate with the increment of generations.

The next subsections will present a snapshot of the migration operator and butterfly adjusting operator.

3.1 Migration operator

As mentioned in Sect. 2, monarch butterflies migrate from Land 1 to Land 2 during the month of April and from Land 2 to Land 1 during the month of September. By simplifying and idealizing the migration process, it can be summarized that the monarch butterflies stay at Land 1 from April to August (5 months) and Land 2 from September to March (7 months). Therefore, the number of monarch butterflies in Land 1 and Land 2 is ceil(p * NP) (NP_1) and NP–NP_1 (NP_2), respectively. Here, ceil(x) rounds x to the nearest integer greater than or equal to x; NP is the total number of the population; p is the ratio of monarch butterflies in Land 1. In consideration of clear expression, monarch butterflies in Land 1 and Land 2 are called Subpopulation 1 and Subpopulation 2, respectively. This migration process can be expressed as follows.

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

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

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

peri indicates migration period and is set to 1.2 in our work (12 months a year). rand is a random number drawn from uniform distribution. On the contrast, if $r > p$, the element k in the newly generated monarch butterfly 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} that is the newly generated position of the monarch butterfly r_2. Monarch butterfly r_2 is randomly selected from Subpopulation 2.

Through the above analyses, it can be seen that the MBO method can balance the direction of migration operator by adjusting the ratio p. If p is big, more elements from monarch butterflies in Land 1 will be selected. This indicates that the Subpopulation 1 plays a more important role in newly generated monarch butterfly. If p is small, more elements from monarch butterflies in Land 2 will be selected. This indicates Subpopulation 2 plays a more important role in newly generated monarch butterfly. In the current work, p is set to 5/12 as per migration period. Accordingly, the migration operator can be represented in Algorithm 1.

Algorithm 1 Migration operator

Begin
  for i= 1 to NP_1 (for all monarch butterflies in Subpopulation 1) **do**
    for k=1 to D (all the elements in ith monarch butterfly) **do**
      Randomly generate a number *rand* by uniform distribution;
      r=*rand***peri*;
      if $r \leq p$ **then**
        Randomly select a monarch butterfly in Subpopulation 1 (say r_1);
        Generate the kth element of the x_i^{t+1} as Eq. (1).
      else
        Randomly select a monarch butterfly in Subpopulation 2 (say r_2);
        Generate the kth element of the x_i^{t+1} as Eq. (3).
      end if
    end for k
  end for i
End.

3.2 Butterfly adjusting operator

Except migration operator, the positions of the monarch butterflies can also be updated by the following butterfly adjusting operator. The process of butterfly adjusting operator can be simply described as follows. For all the elements in monarch butterfly j, if a randomly generated number rand is smaller than or equal to p, it can be updated as

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

where $x_{j,k}^{t+1}$ indicates the kth element of x_j at generation $t+1$ that presents the position of the monarch butterfly j. Similarly, $x_{\text{best},k}^{t}$ indicates the kth element of x_{best} that is the best monarch butterfly in Land 1 and Land 2. t is current generation number. On the contrast, if rand is bigger than 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} that is randomly selected in Land 2. Here, $r_3 \in \{1, 2, \ldots, NP_2\}$. Under this condition, if rand $>$ BAR, it can be further updated as follows.

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

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

$$\mathrm{d}x = \mathrm{Levy}\left(x_j^t\right) \tag{7}$$

In Eq. (6), α is the weighting factor that is given as Eq. (8).

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

where $S_{\max}$ is max walk step that a monarch butterfly individual can move in one step, and t is the current generation. The bigger α, signifying long step of search, increases the influence of dx on $x_{j,k}^{t+1}$ and encourages the process of exploration, while the smaller α, indicating short step of search, decreases the influence of dx on $x_{j,k}^{t+1}$ and encourages the process of exploitation. Accordingly, the description of butterfly adjusting operator can be given in Algorithm 2.

Algorithm 2 Butterfly adjusting operator

Begin

for j= 1 to NP_2 (for all monarch 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 monarch butterfly) **do**

Randomly generate a number *rand* by uniform distribution;

if *rand* $\leq$ p **then**

Generate the kth element of the x_j^{t+1} as Eq. (4).

else

Randomly select a monarch butterfly in Subpopulation 2 (say r_3);

Generate the kth element of the x_j^{t+1} as Eq. (5).

if *rand* > *BAR* **then**

$$x_{j,k}^{t+1} = x_{j,k}^{t+1} + \omega \times (dx_k - 0.5);$$

end if

end if

end for k

end for j

End.

3.3 Schematic presentation of MBO algorithm

By idealizing the migration behavior of the monarch butterfly individuals, MBO method can be formed, and its schematic description can be given as shown in Algorithm 3. A brief presentation of the MBO algorithm is shown in Fig. 1.

According to Algorithm 3, firstly, all the parameters are initialized followed by the generation of initial population and evaluation of the same by means of its fitness function. Subsequently, the positions of all monarch butterflies are updated step by step until certain conditions are satisfied. It should be mentioned that, in order to make the population fixed and reduce fitness evaluations, the number of monarch butterflies, generated by migration operator and butterfly adjusting operator, are NP_1 and NP_2, respectively.

Algorithm 3 Monarch Butterfly Optimization algorithm

Begin

Step 1: Initialization. Set the generation counter $t = 1$; initialize the population P of NP monarch butterfly individuals randomly; set the maximum generation $MaxGen$, monarch butterfly number NP_1 in Land 1 and monarch butterfly number NP_2 in Land 2, max step S_{Max}, butterfly adjusting rate BAR, migration period $peri$, and the migration ratio p.

Step 2: Fitness evaluation. Evaluate each monarch butterfly according to its position.

Step 3: While the best solution is not found **or** $t < MaxGen$ **do**

Sort all the monarch butterfly individuals according to their fitness.

Divide monarch butterfly individuals into two subpopulations (Land 1 and Land 2);

for i= 1 to NP_1 (for all monarch butterflies in Subpopulation 1) **do**

Generate new Subpopulation 1 according to Algorithm 1.

end for i

for j= 1 to NP_2 (for all monarch butterflies in Subpopulation 2) **do**

Generate new Subpopulation 2 according to Algorithm 2.

end for j

Combine the two newly-generated subpopulations into one whole population;

Evaluate the population according to the newly updated positions;

$t = t$+1.

Step 4: end while

Step 5: Output the best solution.

End.

4 Simulation results

In this section, the MBO method is evaluated from various aspects by using an array of experiments conducted in benchmark functions (see Table 1). More detailed descriptions of all the benchmarks can be referred as [45, 54, 55]. Note that the dimensions of functions F01–F25 and F26–F38 are twenty and two, respectively. That is, benchmarks F01–F25 and F26–F38 are the high-dimensional functions and low-dimensional functions, respectively.

In order to obtain fair results, all the implementations are conducted under the same conditions as shown in [56].

The same parameters for MBO method are set as follows: max step $S_{\max} = 1.0$, butterfly adjusting rate BAR = 5/12, migration period peri = 1.2, and the migration ratio $p = 5/12$. Note that, for high-dimensional functions (F01–F25), both population size NP and maximum generation $MaxGen$ are set to 50; while for low-dimensional functions (F26–F38), less population size NP and maximum generation $MaxGen$ are used in our experiments, and both of them are set to 30. Accordingly, for F01–F25, NP_1 and NP_2 are 21 and 29, respectively; while for F26–F38, NP_1 and NP_2 are 13 and 17, respectively.

Metaheuristic algorithms are based on certain distribution, and hence, 200 independent trials have been made with the aim of decreasing the influence of the randomness. In the following experiments, the optimal solution for each test problem is highlighted. For all the tables, the total number of functions in which the optimization method has the best performance is provided in the last row. It is worth mentioning that the scales which are used to normalize values in the tables are fully different with each other and hence values from different tables are not comparative. The detailed normalization process can be defined as follows:

Firstly, the real function values are presented by matrix $\mathbf{A}_{m\times n} = [A_1, A_2, \ldots, A_i, \ldots, A_m]^T$. A_i means the ith row, and a_{ij} is an item in $\mathbf{A}$, $1 \le i \le m$, $1 \le j \le n$. Here, $m = 38$ and $n = 6$ denote the number of benchmark functions and algorithms used in this paper, respectively.

Secondly, the minimum of the ith row b_i is calculated according to the Eq. (9).

$$b_i = \min(A_i) \tag{9}$$

Thirdly, the normalized results $\mathbf{C}_{m\times n}$ can be obtained as shown in Eq. (10).

$$c_{ij} = \frac{a_{ij}}{b_i} \tag{10}$$

The values in the following tables are c_{ij} that are normalized by the above method. As an example, if matrix $\mathbf{A}$ is defined as

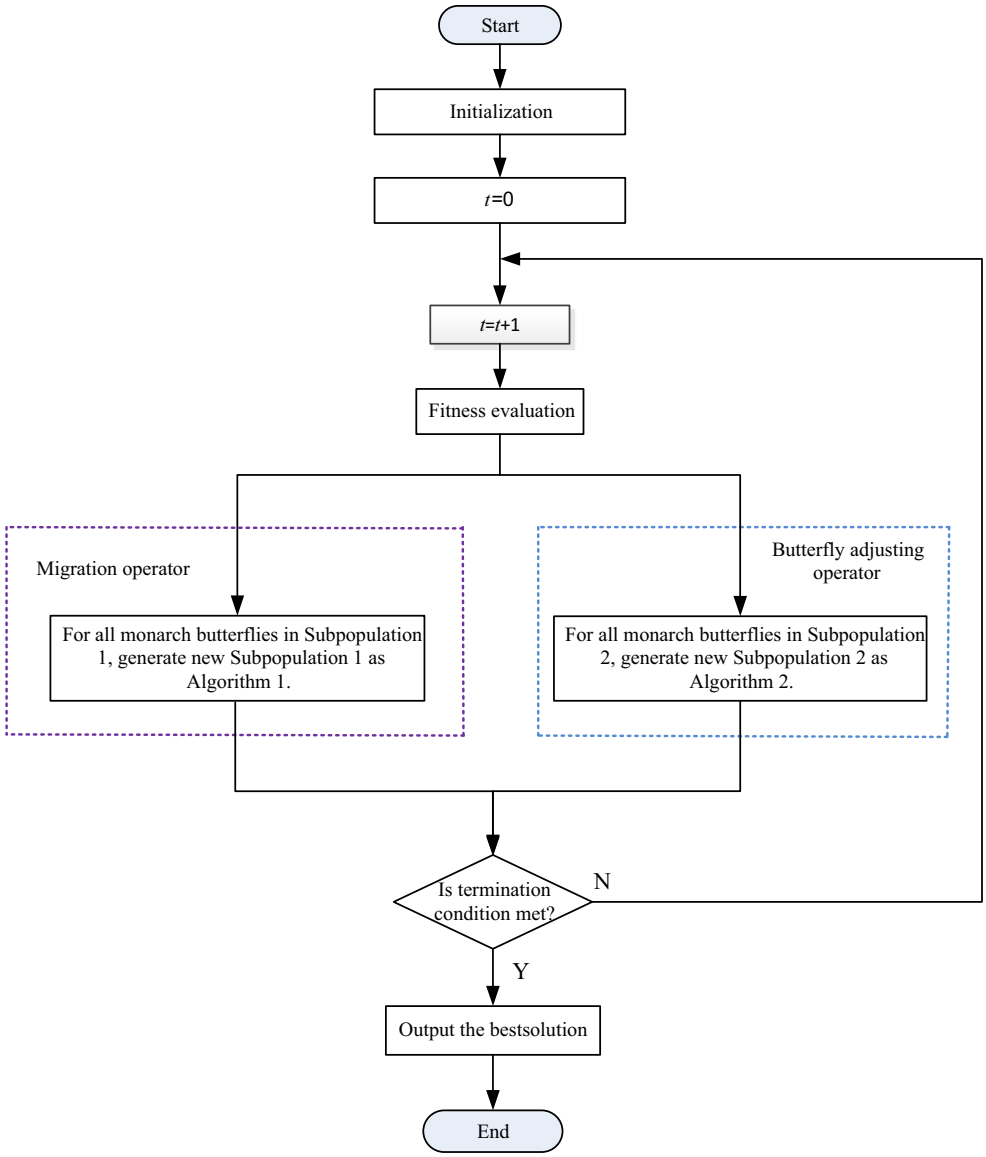

Fig. 1 Schematic flowchart of MBO method

$$\mathbf{A} = \begin{bmatrix} 2 & 5 & 4 & 3 \\ 3 & 4 & 9 & 8 \\ 5 & 4 & 8 & 7 \end{bmatrix} \tag{11}$$

After normalized, the matrix **C** is defined as

$$\mathbf{C} = \begin{bmatrix} 1.00 & 2.50 & 2.00 & 1.50 \\ 1.00 & 1.33 & 3.00 & 2.67 \\ 1.25 & 1.00 & 2.00 & 1.75 \end{bmatrix} \tag{12}$$

4.1 Comparisons of the MBO method with other methods

The performance of MBO was compared with five optimization methods (ABC [13], ACO [11], BBO [45], DE [40] and SGA [42]) on thirty-eight optimization problems. For the parameters used in the other methods, their settings can be referred as [45, 57]. The results obtained by the six algorithms on an array of benchmarks are recorded in Table 2.

From Table 2, it can be seen that, on average, MBO has the fifth performance on six out of thirty-eight benchmarks. SGA, ABC, BBO and DE have better performance than MBO method, and they rank 1, 2, 3 and 4, respectively.

For the best solutions, Table 2 shows that MBO is well capable of finding the optimal solutions on thirty-four out of thirty-eight benchmarks. DE and ABC can search for the best solutions on twelve and eleven out of thirty-eight benchmarks. Generally, ABC, ACO, DE and SGA have identical performance with respect to each other and this is

Table 1 Benchmark functions

No.	Name	No.	Name
F01	Ackley	F20	Schwefel 2.21
F02	Alpine	F21	Sphere
F03	Brown	F22	Step
F04	Dixon and price	F23	Sum function
F05	Fletcher-Powell	F24	Zakharov
F06	Griewank	F25	Wavy1
F07	Holzman 2 function	F26	Beale
F08	Levy	F27	Bohachevsky #1
F09	Pathological function	F28	Bohachevsky #2
F10	Penalty #1	F29	Bohachevsky #3
F11	Penalty #2	F30	Booth
F12	Perm	F31	Branin
F13	Powel	F32	Easom
F14	Quartic with noise	F33	Foxholes
F15	Rastrigin	F34	Freudenstein-Roth
F16	Rosenbrock	F35	Goldstein-price
F17	Schwefel 2.26	F36	Hump
F18	Schwefel 1.2	F37	Matyas
F19	Schwefel 2.22	F38	Shubert

more evident for ACO and SGA. By carefully looking at the Table 2, we found that, for twenty-five high-dimensional complicated functions, MBO has absolute advantage over the other five methods. For thirteen low-dimensional functions, all the methods can solve them well under the given conditions.

For the worst function values shown in Table 2, the first two algorithms are ABC and BBO, and they perform the best on nineteen and fifteen out of thirty-eight benchmarks, respectively. MBO and ACO are well capable of finding the best solutions on thirteen functions, which are nothing but inferior to the above two methods.

From Table 2, for low-dimensional benchmarks ($D = 2$), the performance of the six methods has little difference. In particular, ABC and DE are the two best methods when dealing with 2D functions. For high-dimensional benchmarks ($D = 20$), especially for the best average performance, MBO is the best method at searching for the optimal function values.

In addition, to demonstrate and prove the superiority of the MBO method, convergence trajectories of six methods are also illustrated in the current work. Full details are beyond the scope of this manuscript, and hence, only the illustrations of some of the most representative benchmarks will be provided, as shown in Figs. 2, 3, 4, 5, 6, 7 and 8.

From Fig. 2, there is no ambiguity in concluding that even though all the methods commence the process of optimization at the beginning of same fitness, MBO can find the satisfactory function values faster and the function values found by MBO are always smaller than other five methods during the whole optimization process.

For this test problem, it is obvious that, though BBO and SGA perform well and have the similar performance, both of them are outperformed by MBO from the beginning of the optimization process. Looking carefully at generations 1–5 from Fig. 3, MBO has a faster convergence than other methods.

As shown in Fig. 4, it is obvious that the convergent trajectory of MBO is far away from others. This indicates that MBO significantly outperforms other five methods for Pathological function. Furthermore, DE and SGA rank 2 and 3 among six methods.

For this case, though MBO and SGA have similar final fitness value, the convergent trajectory of MBO is always below SGA. This implies that the fitness found by MBO is better than SGA during the convergent process.

For this case as shown in Fig. 6, SGA and BBO have similar convergent trajectories and final fitness values, and both of them are only inferior to MBO method. MBO method has better fitness from generation 1 to generation 50.

For this case, the convergent trajectory of MBO is far below other methods, and this indicates MBO has found much better function value than other five comparative methods. Furthermore, for other five methods, SGA and ACO are only worse than MBO and rank 2 and 3, respectively.

From the above analyses about the Figs. 2, 3, 4, 5, 6 and 7, we can arrive at a conclusion that MBO algorithm significantly outperforms the other five comparative algorithms. On most of the occasions, SGA and BBO have better performance as compared to the other methods, but worse only than that of MBO.

4.2 Comparisons with other optimization methods by using *t* test

Based on the final search results of 200 independent trials on thirty-eight functions, Table 3 presents the t values on thirty-eight functions of the two-tailed test with the 5 % level of significance between the MBO and other optimization methods. In the table, the value of t with 398 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test. Boldface indicates that MBO performs significantly better than the comparative algorithm. For the last three rows, the "Better", "Equal", and "Worse" represent that MBO is better than, equal to, and worse than certain comparative method for this case. Here, we take the comparison between the MBO and the ACO for instance. The MBO method performs better and worse than ACO on twenty and seven functions, respectively. What's more, the MBO method performs similarly with ACO method on 11

Table 2 continued

	ABC	ACO	BBO	DE	MBO	SGA
F25						
Mean	2.33	1.82	**1.00**	2.49	1.18	1.05
Best	1.7E5	1.1E5	6.6E4	1.9E5	**1.00**	5.5E4
Worst	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F26						
Mean	1.01	1.03	1.07	**1.00**	1.44	1.14
Best	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Worst	**1.00**	**1.00**	**1.00**	**1.00**	1.80	**1.00**
F27						
Mean	**1.00**	2.27	5.91	1.13	69.49	2.01
Best	**1.00**	1.13	1.30	**1.00**	**1.00**	**1.00**
Worst	**1.00**	1.63	3.55	1.15	293.62	2.05
F28						
Mean	**1.00**	1.99	6.25	1.10	58.76	1.45
Best	**1.00**	**1.00**	1.24	**1.00**	**1.00**	**1.00**
Worst	**1.00**	3.20	8.77	1.11	1.46	1.42
F29						
Mean	**1.00**	1.93	7.06	1.07	83.07	1.54
Best	**1.00**	1.04	1.04	**1.00**	**1.00**	**1.00**
Worst	**1.00**	3.44	3.78	1.09	4.43	1.40
F30						
Mean	1.03	1.03	1.23	**1.00**	2.08	1.03
Best	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Worst	1.02	1.05	1.36	**1.00**	1.98	1.98
F31						
Mean	**1.00**	1.04	1.09	1.06	1.46	1.05
Best	**1.00**	1.01	1.01	**1.00**	**1.00**	1.01
Worst	**1.00**	1.13	1.02	1.07	1.23	2.77
F32						
Mean	**1.00**	1.07	1.13	1.09	1.04	1.05
Best	1.5E3	2.0E3	1.2E4	3.2E3	**1.00**	91.68
Worst	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F33						
Mean	**1.00**	**1.00**	1.01	**1.00**	5.80	**1.00**
Best	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Worst	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F34						
Mean	2.82	2.50	9.35	2.30	15.43	**1.00**
Best	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Worst	**1.00**	**1.00**	3.55	66.68	3.55	3.55
F35						
Mean	1.43	1.09	1.58	**1.00**	4.74	2.33
Best	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Worst	**1.00**	**1.00**	1.97	1.02	6.13	**1.00**
F36						
Mean	**1.00**	1.01	1.02	**1.00**	1.19	1.01
Best	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Worst	**1.00**	**1.00**	**1.00**	**1.00**	303.21	303.21

Table 2 continued

	ABC	ACO	BBO	DE	MBO	SGA
F37						
Mean	1.01	1.01	1.02	**1.00**	1.02	1.01
Best	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Worst	1.01	**1.00**	**1.00**	**1.00**	1.08	2.05
F38						
Mean	**1.00**	63.93	34.42	31.31	82.54	2.76
Best	2.54	452.25	452.25	237.31	**1.00**	452.25
Worst	**1.00**	21.89	**1.00**	2.81	**1.00**	**1.00**
Total						
Mean	8	1	7	7	6	13
Best	11	10	8	12	34	10
Worst	19	13	15	10	13	8

The best value obtained by each method is indicated by bold

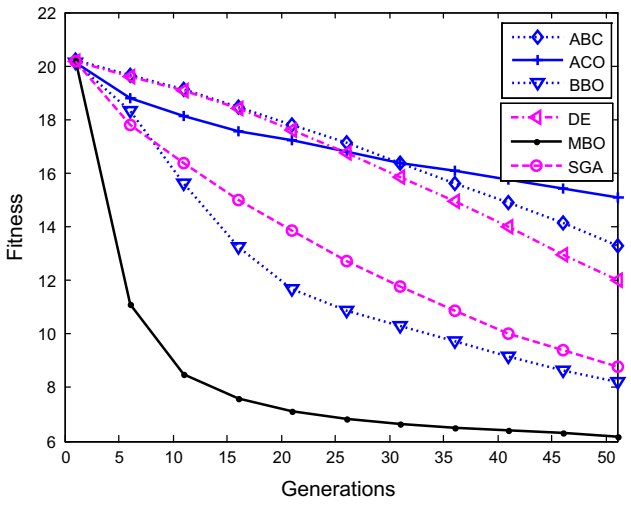

Fig. 2 Comparison of six methods for the F01 Ackley function with 50 generations

functions out of thirty-eight functions. In other words, we can say, the MBO performs better than or equally to ACO for most cases. Furthermore, for ABC, the number of "Better", "Equal", and "Worse" are 22, 4 and 12, respectively. This shows that MBO has better performance than ABC on most functions. Therefore, MBO is not worse than ABC for twenty-six cases. Although the MBO performed equally to, or slightly poorer than the comparative methods on some functions, Table 3 indicates that it outperforms the other five methods for most of the functions.

4.3 The study of the number of fitness evaluations

In order to further investigate the performance of MBO method, fitness evaluations are also studied from the following two respects.

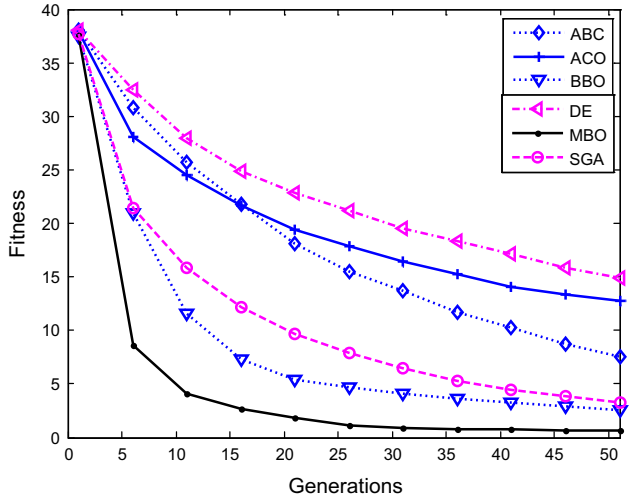

Fig. 3 Comparison of six methods for the F02 Alpine function with 50 generations

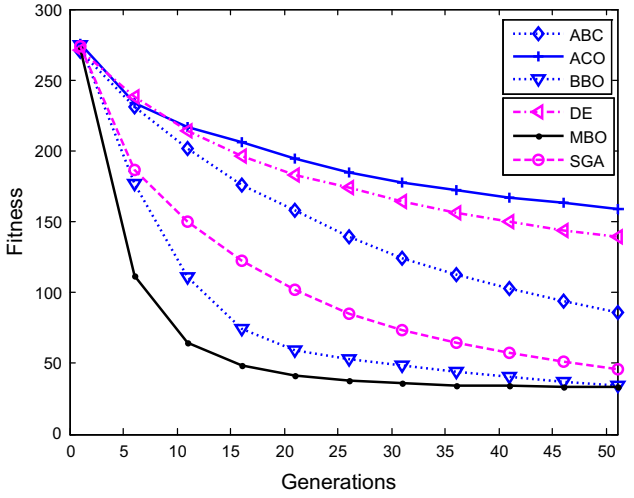

Fig. 5 Comparison of six methods for the F15 Rastrigin function with 50 generations

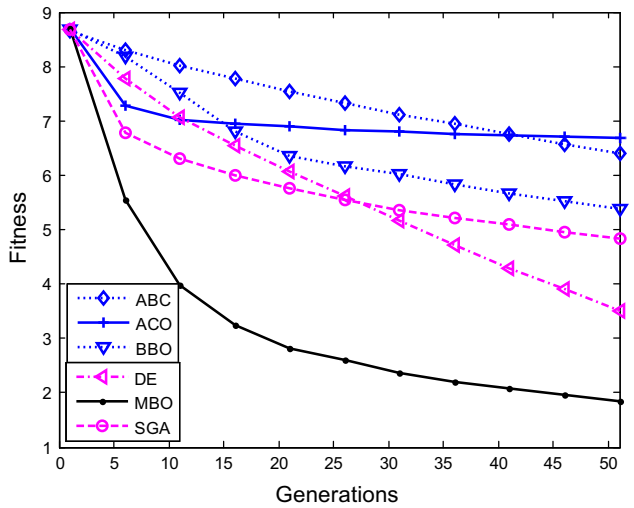

Fig. 4 Comparison of six methods for the F09 Pathological function with 50 generations

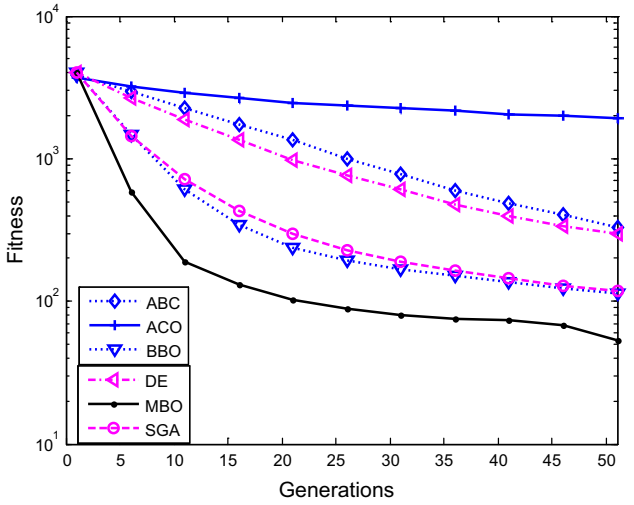

Fig. 6 Comparison of six methods for the F16 Rosenbrock function with 50 generations

4.3.1 Fixed fitness

In this subsection, we are analyzing the number of fitness evaluations on twenty-five high-dimensional functions for each method to the fixed fitness that are set to opt + 1. opt indicates the theoretical optimal solution for each function. The maximum generation is set to 1000, and the maximum of fitness evaluations is therefore 50,000. In other words, a method will return 50,000 if it cannot find the required fitness, or return the fitness evaluations if it can find opt + 1 under given conditions (see Table 4). Table 4 indicates that, for fourteen functions, MBO can successfully find the required fitness by using the least fitness evaluations. SGA is only inferior to MBO method and can successfully search for opt + 1 by using least CPU resources on the ten functions. DE and BBO rank 3 and 4, and are well capable of finding the required opt + 1 on eight and five functions, respectively. Both ABC and ACO are able to find the required opt + 1 on four functions. In particular, for F02 (Alpine), F08 (Levy), F09 (Pathological function), F19 (Schwefel 2.22) and F21 (Sphere), MBO is well capable of finding the required function value by only using 1680, 1135, 3235, 2420 and 1520 fitness evaluations that are far less than the other five comparative methods. It should be mentioned that, for F05 (Fletcher-Powell function), F06 (Griewank), F12 (Perm) and F18 (Schwefel 1.2), all the methods fail to find the required opt + 1 within 500,000 function evaluations. We have made great effort to

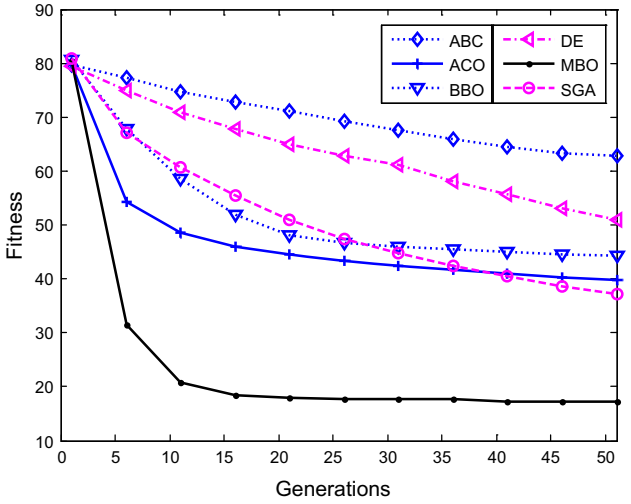

Fig. 7 Comparison of six methods for the F20 Schwefel 2.21 function with 50 generations

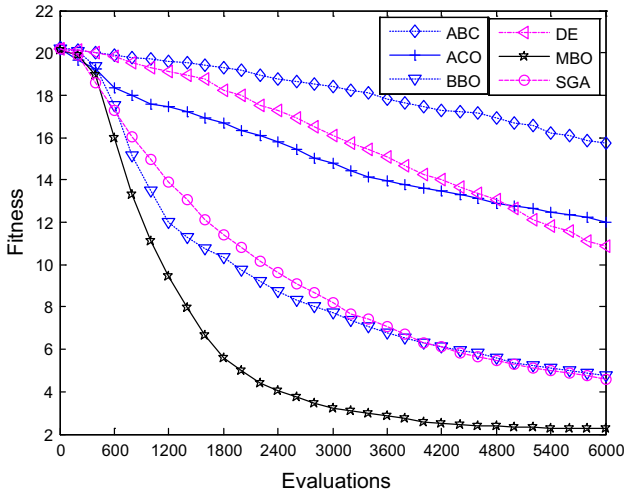

Fig. 8 Convergent history for the F01 Ackley function with 6000 evaluations

find the required values by increasing population size and maximum generations. However, finally, no one method can find the required opt + 1 under given conditions. To sum up, for most benchmarks, MBO has the strong ability of finding the required function values by using the least fitness evaluations.

4.3.2 Fixed fitness evaluations

In this section, the fitness is investigated on twenty-five high-dimensional functions for each method by using the fixed number of fitness evaluations. The fixed fitness evaluation is set to 6000. The average and best results are recorded in Table 5 after 6000 fitness evaluations. From Table 5, it can be seen that, on average, SGA can find the most satisfactory fitness on thirteen functions by using fixed fitness evaluations (6000). MBO is only inferior to SGA and find the best fitness on nine functions after 6000 evaluations. For the best performance as shown in Table 5,

Table 3 Comparisons between MBO and other methods at $\alpha = 0.05$ on a two-tailed t tests

	ABC	ACO	BBO	DE	SGA
F01	**14.67**	**18.19**	**4.18**	**12.13**	**5.34**
F02	**31.91**	**42.91**	**10.61**	**58.56**	**13.55**
F03	**3.30**	**5.22**	−0.54	−0.13	−0.75
F04	**2.88**	**6.42**	−2.68	−1.14	−2.93
F05	−18.14	**16.35**	−28.89	−10.51	−27.46
F06	**12.87**	−0.51	−3.00	**4.05**	−3.22
F07	**3.25**	**6.72**	−2.67	−0.63	−3.07
F08	**12.73**	**18.77**	0.34	**16.31**	−0.80
F09	**39.70**	**42.85**	**30.18**	**13.44**	**24.70**
F10	**9.52**	**7.01**	0.04	**8.27**	−1.87
F11	**8.28**	**8.24**	−1.27	1.72	−1.77
F12	−2.23	−3.10	**4.60**	−5.16	−4.75
F13	**2.85**	**21.29**	−1.76	**13.25**	−3.15
F14	**2.92**	**3.21**	−2.72	−0.55	−3.07
F15	**13.87**	**31.34**	0.27	**27.77**	**3.31**
F16	**13.59**	**42.65**	**3.23**	**12.44**	**3.43**
F17	**8.96**	**2.25**	−4.83	**14.98**	−3.76
F18	**2.92**	1.90	−3.36	**6.34**	0.24
F19	**7.85**	**29.35**	−1.14	**10.70**	1.30
F20	**31.38**	**15.85**	**17.71**	**24.92**	**12.26**
F21	**9.63**	**23.83**	−0.73	**3.44**	−0.52
F22	**6.88**	0.02	−3.50	0.66	−3.89
F23	**6.50**	**16.69**	−2.85	0.23	−2.99
F24	−1.27	1.87	−4.67	0.50	−1.95
F25	**9.06**	**5.00**	−1.41	**10.31**	−0.99
F26	−5.72	−5.30	−4.72	−5.79	−3.46
F27	−4.36	−4.27	−4.03	−4.35	−4.29
F28	−5.04	−4.95	−4.57	−5.03	−5.00
F29	−3.74	−3.69	−3.45	−3.73	−3.71
F30	−4.29	−4.29	−3.41	−4.40	−4.28
F31	−4.38	−3.97	−3.41	−3.73	−3.93
F32	−0.64	0.65	**2.14**	0.99	0.15
F33	−1.55	−1.55	−1.55	−1.55	−1.55
F34	−3.86	−4.24	−1.70	−4.03	−4.82
F35	−4.54	−5.27	−4.48	−5.42	−2.95
F36	−3.18	−3.07	−2.83	−3.17	−3.05
F37	−1.61	−2.62	−0.73	−3.52	−1.85
F38	−5.99	−1.07	−3.19	−3.58	−5.85
Better	22	20	7	15	6
Equal	4	7	12	10	12
Worse	12	11	19	13	20

The best value obtained by each method is indicated by bold

Table 4 The number of fitness evaluations for different methods

	ABC	ACO	BBO	DE	MBO	SGA
F01	47,232	50,000	36,145	**18,090**	**19,085**	33,135
F02	25,480	50,000	5585	27,450	**1680**	4500
F03	16,922	50,000	3315	6880	1365	2740
F04	50,000	50,000	48,835	**27,460**	42,860	47,350
F05	**50,000**	**50,000**	**50,000**	**50,000**	**50,000**	**50,000**
F06	**50,000**	**50,000**	**50,000**	**50,000**	**50,000**	**50,000**
F07	21,787	50,000	11,130	13,990	7235	**6055**
F08	17,710	50,000	4405	12,540	**1135**	3850
F09	50,000	50,000	50,000	13,600	**3235**	50,000
F10	21,455	45,005	8630	17,490	35,340	**5435**
F11	27,020	50,000	16,700	18,890	35,515	**9440**
F12	**50,000**	**50,000**	**50,000**	**50,000**	**50,000**	**50,000**
F13	46,777	50,000	30,730	46,900	36,310	**22,670**
F14	7,875	2485	**830**	4230	860	940
F15	50,000	50,000	34,125	50,000	**26,420**	35,035
F16	50,000	50,000	50,000	50,000	**40,595**	50,000
F17	50,000	50,000	50,000	50,000	45,135	**44,690**
F18	**50,000**	**50,000**	**50,000**	**50,000**	**50,000**	**50,000**
F19	26,565	50,000	16,245	15,590	**2420**	18,125
F20	50,000	50,000	50,000	**46,810**	50,000	50,000
F21	16,152	50,000	3640	8550	**1520**	4015
F22	37,205	50,000	48,995	**19,920**	31,260	26,400
F23	27,230	50,000	22,345	14,830	18,560	**14,020**
F24	50,000	50,000	50,000	50,000	**45,230**	50,000
F25	48,440	50,000	50,000	29,150	**28,325**	50,000
Total	4	4	5	8	14	10

The best value obtained by each method is indicated by bold

MBO significantly outperforms other five compared methods, and is able to search for the optimal function values on twenty-three functions by using 6000 fitness evaluations. To sum up, for most benchmarks, MBO is well capable of finding the best function values by using the fixed fitness evaluations.

With the aim of further investigation of the fixed fitness emulations, we have recorded the fitness at intervals of fifty evaluations that are illustrated in Figs. 8, 9, 10, 11, 12 and 13.

From Fig. 8, it can be seen that, BBO and SGA have almost the same fitness when using the same fitness evaluations. MBO has better fitness than BBO and SGA when using the same fitness evaluations during the optimization process.

For this test problem, though MBO has the similar performance with BBO and SGA that perform well too at the beginning of the search, MBO outperforms them after 100 fitness evaluations. Finally, MBO has far better fitness than BBO and SGA.

Table 5 Mean function values with the fixed evaluations

	ABC	ACO	BBO	DE	MBO	SGA
F01						
Mean	7.02	5.35	2.11	4.84	**1.00**	2.04
Best	1.3E4	9.3E3	3.5E3	8.8E3	**1.00**	3.1E3
F02						
Mean	1.1E3	732.21	76.42	1.2E3	**1.00**	72.60
Best	1.4E5	8.6E4	5.9E3	1.3E5	**1.00**	4.5E3
F03						
Mean	1.7E3	566.72	20.46	116.53	**1.00**	10.42
Best	8.4E6	5.4E6	9.5E4	9.1E5	**1.00**	4.4E4
F04						
Mean	3.1E3	233.30	3.48	114.24	978.41	**1.00**
Best	9.8E4	6.5E3	54.42	4.6E3	**1.00**	32.11
F05						
Mean	7.34	16.02	**1.00**	6.17	2.60	1.19
Best	17.45	40.36	**1.00**	20.76	3.07	2.49
F06						
Mean	50.64	3.16	1.50	8.48	31.10	**1.00**
Best	50.57	2.40	1.45	8.53	**1.00**	1.39
F07						
Mean	1.2E4	802.84	7.85	599.97	811.66	**1.00**
Best	4.0E13	3.6E12	3.0E10	4.2E12	**1.00**	1.6E9
F08						
Mean	52.55	20.08	1.52	23.65	4.54	**1.00**
Best	1.6E8	4.8E7	3.8E6	6.7E7	**1.00**	3.3E6
F09						
Mean	18.80	17.50	11.08	7.95	**1.00**	10.16
Best	270.96	242.87	145.71	65.85	**1.00**	128.36
F10						
Mean	2.0E7	1.4E8	3.26	1.1E5	5.8E5	**1.00**
Best	1.8E14	1.0E8	4.1E8	4.6E10	**1.00**	1.3E8
F11						
Mean	1.2E7	4.2E7	21.31	2.0E5	1.8E4	**1.00**
Best	6.4E14	6.8E7	2.4E8	1.5E12	**1.00**	6.5E7
F12						
Mean	1.2E6	1.6E5	4.5E6	**1.00**	1.9E6	7.3E4
Best	6.5E5	2.7E9	2.7E9	**1.00**	1.0E6	2.7E9
F13						
Mean	70.58	100.47	2.04	60.12	28.39	**1.00**
Best	8.3E5	1.6E6	2.2E4	1.2E6	**1.00**	1.0E4
F14						
Mean	7.6E3	294.50	6.56	389.58	367.33	**1.00**
Best	6.3E12	2.7E11	3.6E9	7.4E11	**1.00**	3.4E8
F15						
Mean	10.93	11.90	1.50	12.06	**1.00**	2.10
Best	2.3E6	2.4E6	2.7E5	2.8E6	**1.00**	3.7E5
F16						
Mean	10.86	22.38	1.06	3.48	**1.00**	1.03
Best	2.2E5	4.2E5	1.5E4	6.3E4	**1.00**	1.3E4

Table 5 continued

	ABC	ACO	BBO	DE	MBO	SGA
F17						
Mean	13.17	3.34	1.34	13.01	7.82	**1.00**
Best	1.1E7	1.4E6	5.7E5	1.0E7	**1.00**	3.5E5
F18						
Mean	2.96	1.69	**1.00**	3.06	2.28	1.57
Best	1.9E3	609.84	570.97	2.1E3	**1.00**	738.67
F19						
Mean	10.41	14.27	1.12	6.08	**1.00**	1.42
Best	2.0E4	2.4E4	1.9E3	1.3E4	**1.00**	2.5E3
F20						
Mean	2.53	1.26	1.32	1.70	1.08	**1.00**
Best	108.74	33.67	39.73	69.79	**1.00**	23.21
F21						
Mean	3.0E6	1.7E6	5.3E4	4.2E5	**1.00**	3.6E4
Best	1.2E8	6.1E7	1.3E6	1.6E7	**1.00**	8.5E5
F22						
Mean	124.97	8.69	2.58	19.84	27.72	**1.00**
Best	2.0E19	1.6E18	2.9E17	3.5E18	**1.00**	1.6E17
F23						
Mean	86.30	59.00	1.74	11.94	6.44	**1.00**
Best	3.0E7	1.7E7	2.0E5	4.8E6	**1.00**	1.9E5
F24						
Mean	3.03	4.1E4	1.28	3.50	**1.00**	1.76
Best	6.9E4	3.2E4	1.9E4	8.6E4	**1.00**	2.2E4
F25						
Mean	8.23	2.61	1.10	5.00	2.74	**1.00**
Best	6.3E4	1.9E4	7.1E3	4.3E4	**1.00**	5.1E3
Total						
Mean	0	0	2	1	9	13
Best	0	0	1	1	23	0

The best value obtained by each method is indicated by bold

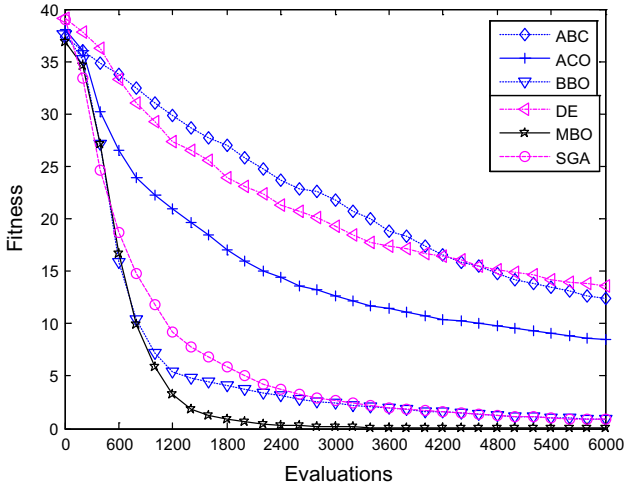

Fig. 9 Convergent history for the F02 Alpine function with 6000 evaluations

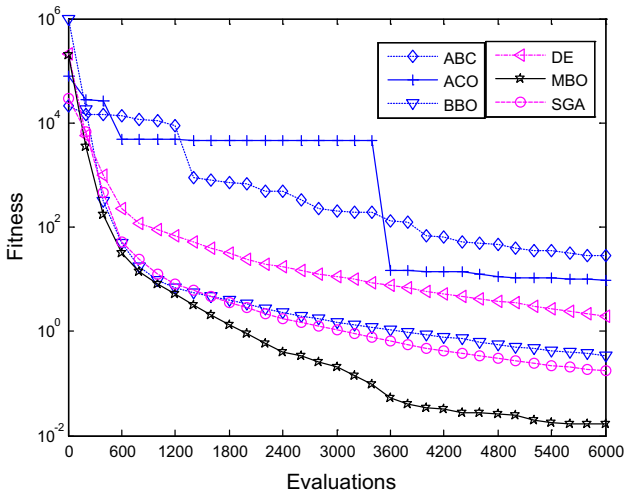

Fig. 10 Convergent history for the F03 Brown function with 6000 evaluations

As shown in Fig. 10, similarly with Figs. 8 and 9, though SGA has better fitness than BBO finally, both of them have the similar convergent trend with the same fitness evaluations. MBO has better fitness than other five methods, including BBO and SGA.

For this case, MBO has far better fitness than other methods with the same fitness evaluations. For other methods, DE, SGA and BBO rank 2, 3, and 4 among six methods, respectively. In addition, ACO and ABC have the similar convergent curves that indicate their fitness decrease little with the increment of fitness evaluations.

For this case as shown in Fig. 12, at first glance, the six methods can easily be divided into two groups: MBO, BBO, SGA and ABC, ACO, DE. It is quite clear that the first group has far better fitness than the second one with the same fitness evaluations. In more detail, MBO and ABC have the best fitness among three methods in group one and group two, respectively.

Similar to Rastrigin function as shown in Fig. 12, the six methods can be divided into two groups: MBO, BBO, SGA and DE, ABC, ACO. It is quite clear that the first group has far better fitness than the second one with the same fitness evaluations. Furthermore, MBO and DE have the best fitness among three methods in group one and group two, respectively.

From above analyses about the Figs. 8, 9, 10, 11, 12 and 13, we can infer that MBO algorithm is well capable of finding far better function values with the same fitness evaluations. For the other methods, in most cases, SGA and BBO have better fitness as compared to the rest with the

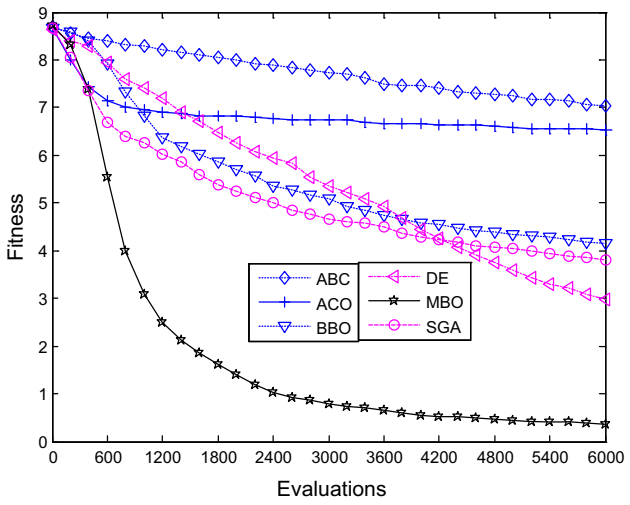

Fig. 11 Convergent history for the F09 Pathological function with 6000 evaluations

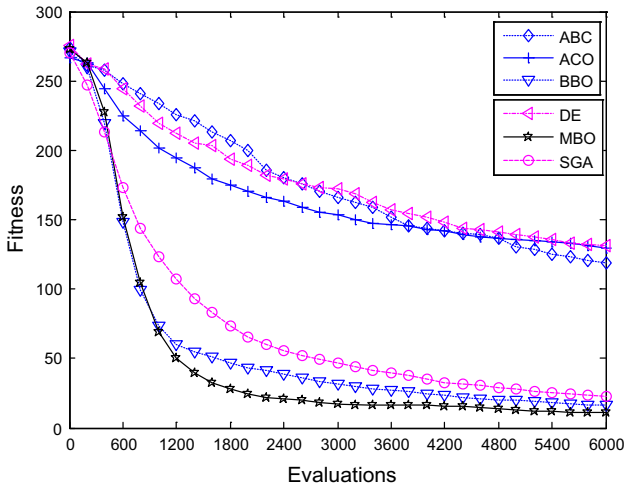

Fig. 12 Convergent history for the F15 Rastrigin function with 6000 evaluations

same fitness evaluations though they are worse as compared to that of MBO.

4.4 Comparisons of the MBO method with other methods on higher dimensions

In order to further investigate the performance of MBO method, it is further investigated on twenty-five higher-dimensional functions. Population size and maximum generations are set to 100. For other parameters used in this section, they are the same as the Sect. 4.1, except the dimension of test functions.

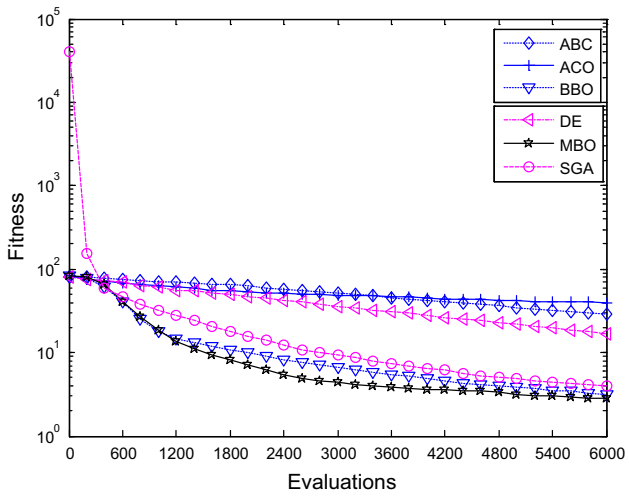

Fig. 13 Convergent history for the F19 Schwefel 2.22 function with 6000 evaluations

4.4.1 $D = 60$

Here, the dimension of test function is set to 60. The results obtained by six methods are recorded in Table 6.

From Table 6, it can be seen that, on average, MBO is well capable of finding the optimal solutions on twenty-two out of twenty-five benchmarks. ACO, BBO and DE can search for the best solutions on only one out of twenty-five benchmarks. Generally, ACO, BBO and DE have the similar performance each other.

For the best solutions, Table 6 indicates that MBO has the best performance on nineteen out of twenty-five benchmarks. BBO and DE have better performance than ABC, ACO and SGA methods, and they rank 2 and 3, respectively.

For the worst function values shown in Table 6, the first two algorithms are MBO and BBO, and they perform the best on seventeen and twelve out of twenty-five benchmarks, respectively. SGA is well capable of finding the best solutions on eight functions, which are only inferior to the above two methods.

4.4.2 $D = 100$

Here, the dimension of test function is set to 100. The results obtained by six methods are recorded in Table 7. The notation "–" in Table 7 indicates that the function cannot be solved by the corresponding method.

From Table 7, for average values, MBO is well capable of finding the optimal solutions on eighteen out of twenty-five benchmarks. BBO and SGA can search for the best

Table 7 continued

	ABC	ACO	BBO	DE	MBO	SGA
F21						
Mean	18.32	36.28	1.45	17.64	**1.00**	5.06
Best	3.3E8	5.8E8	2.1E7	2.6E8	**1.00**	6.2E7
Worst	**1.00**	2.02	300.50	300.50	300.50	300.50
F22						
Mean	39.77	25.78	3.14	40.29	**1.00**	6.94
Best	4.9E20	2.5E20	3.7E19	5.0E20	**1.00**	6.4E19
Worst	13.07	10.39	**1.00**	926.54	926.54	926.54
F23						
Mean	69.36	75.70	5.21	54.62	**1.00**	16.70
Best	2.2E9	2.1E9	1.3E8	1.7E9	**1.00**	4.2E8
Worst	19.64	19.08	**1.00**	13.62	597.00	597.00
F24						
Mean	1.24	6.2E10	**1.00**	1.47	2.9E3	1.07
Best	9.98	**1.00**	8.06	11.04	1.37	7.70
Worst	1.22	1.1E11	**1.00**	1.22	2.19	393.64
F25						
Mean	8.08	7.61	1.89	9.94	**1.00**	3.86
Best	3.2E5	2.7E5	6.9E4	3.9E5	**1.00**	1.3E5
Worst	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Total						
Mean	0	0	5	0	18	1
Best	0	1	1	0	22	0
Worst	6	4	13	3	15	6

The best value obtained by each method is indicated by bold

solutions on five and one out of twenty-five benchmarks, respectively.

For the best solutions, Table 7 indicates that MBO is well capable of finding the optimal solutions on twenty-two out of twenty-five benchmarks. ACO and BBO can search for the best solutions on only one out of twenty-five benchmarks. Hence, it is safe to conclude that MBO has the absolute advantage over other five methods at this respect.

For the worst function values shown in Table 7, MBO is a little better than BBO that significantly outperforms the other four methods. ABC and SGA have the similar performance, and they can find the best solutions on six functions, which are only inferior to the above two methods.

We must point out, for F12 Perm function, its formulation can be given as

$$f(\vec{x}) = \sum_{k=1}^{D} \left[\sum_{i=1}^{D} \left(i^k + 0.5\right) \left(\left(\frac{x_i}{i}\right)^k - 1 \right) \right]^2 \tag{13}$$

Its function values are related to dimension, and it is too large to be represented by MATLAB. In our experiment, it can be up to 10E4000. Therefore, the six methods cannot solve this function in our experiment.

From Tables 6 and 7, it can be seen, for high-dimensional functions, MBO has the absolute advantage over other five methods. This indicates that MBO can solve more complicated problem more efficiently and effectively than the other five comparative algorithms.

5 Discussion and conclusion

By simulating the migration behavior of the monarch butterflies in nature, a new kind of nature-inspired metaheuristic algorithm, called MBO, is presented for continuous optimization problems in this paper. In MBO, all the monarch butterfly individuals are idealized and only located in two lands: southern Canada and the northern USA (Land 1) and Mexico (Land 2). Accordingly, the positions of the monarch butterflies are updated in two ways. Firstly, the offsprings are generated (position updating) by migration operator, which can be adjusted by the migration ratio. And then, for other butterflies, their positions are tuned by means of butterfly adjusting operator. In other words, the search direction of the monarch butterfly individuals in MBO algorithm are mainly determined by the migration operator and butterfly adjusting operator. With the aim of showing the performance of MBO method, it is compared with five other metaheuristic algorithms through thirty-eight benchmark problems. The results show that the MBO method is able to find the better function values on most benchmark problems than five other metaheuristic algorithms.

In addition, MBO algorithm is simple and has no complicated calculation and operators. This makes the implementation of MBO algorithm easy and fast.

Despite various advantages of the MBO method, the following points should be clarified and focused on in the future research.

Firstly, it is well known that the parameters used in a metaheuristic method have great influence on its performance. In the present work, we do little effort to fine-tune the parameters used in MBO method. The best parameter settings will be selected through theoretical analyses or empirical experiments.

Secondly, computational requirements are of vital importance for any metaheuristic method. It is imperative to improve the search speed by analyzing the MBO method.

Thirdly, we use only thirty-eight benchmark functions to test our proposed MBO method. In future, more benchmark problems, especially real-world applications, should be used for effective implementation of the MBO method, such as image segmentation, constrained optimization, knapsack problem, scheduling, dynamic optimization, antenna and microwave design problems, and water, geotechnical and transport engineering.

Fourthly, in the current work, the characteristics of the migration behavior (essentially migration operator and butterfly adjusting operator) are idealized to form the MBO method. In future, more characteristics, such as swarm, defense against predators, and human interactions, can be idealized and simplified to be added to the MBO method.

Fifthly, as discussed in Sect. 4, MBO method has the absolute advantage over other five methods on best performance. However, for the average performance, MBO is not the best one among six methods. Furthermore, bad average performance must lead to bad standard deviation (SD). Efforts should be made to improve the average performance by updating the search process.

Sixthly, as shown in Sect. 4, MBO method has the absolute advantage over other five methods when dealing with high-dimensional functions. However, for the low-dimensional functions, MBO performs equally to or worse than other five methods. We plan to investigate this and endeavor to find out the reasons and thereby address this disadvantage in our future research.

And last, in the current work, the performance of MBO method is experimentally tested only using benchmark problems. The convergence of MBO method will be analyzed theoretically by dynamic systems and Markov chain. This can ensure stable implementation of MBO method.

Acknowledgments This work was supported by Research Fund for the Doctoral Program of Jiangsu Normal University (No. 13XLR041).

References

1. Cui Z, Gao X (2012) Theory and applications of swarm intelligence. Neural Comput Appl 21(2):205–206
2. Cui Z, Fan S, Zeng J, Shi Z (2013) APOA with parabola model for directing orbits of chaotic systems. Int J Bio-Inspired Comput 5(1):67–72
3. Gao XZ, Wang X, Jokinen T, Ovaska SJ, Arkkio A, Zenger K (2012) A hybrid PBIL-based harmony search method. Neural Comput Appl 21(5):1071–1083. doi:10.1007/s00521-011-0675-6
4. Gao XZ, Ovaska SJ, Wang X, Chow MY (2007) A neural networks-based negative selection algorithm in fault diagnosis. Neural Comput Appl 17(1):91–98. doi:10.1007/s00521-007-0092-z
5. Fister Jr I, Yang X-S, Fister I, Brest J, Fister D (2013) A brief review of nature-inspired algorithms for optimization. arXiv:1307.4186
6. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Paper presented at the proceeding of the IEEE international conference on neural networks, Perth, Australia, 27 November-1 December
7. Ram G, Mandal D, Kar R, Ghoshal SP (2014) Optimal design of non–uniform circular antenna arrays using PSO with wavelet mutation. Int J Bio-Inspired Comput 6(6):424–433
8. Mirjalili S, Wang G-G, Coelho LdS (2014) Binary optimization using hybrid particle swarm optimization and gravitational search algorithm. Neural Comput Appl 25(6):1423–1435. doi:10.1007/s00521-014-1629-6
9. Wang G-G, Gandomi AH, Alavi AH, Deb S (2015) A hybrid method based on krill herd and quantum-behaved particle swarm optimization. Neural Comput Appl. doi:10.1007/s00521-015-1914-z
10. Wang G-G, Gandomi AH, Yang X-S, Alavi AH (2014) A novel improved accelerated particle swarm optimization algorithm for global numerical optimization. Eng Comput 31(7):1198–1220. doi:10.1108/EC-10-2012-0232
11. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B Cybern 26(1):29–41. doi:10.1109/3477.484436
12. Krynicki K, Jaen J, Mocholí JA (2014) Ant colony optimisation for resource searching in dynamic peer-to-peer grids. Int J Bio-Inspired Comput 6(3):153–165. doi:10.1504/IJBIC.2014.062634
13. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 39(3):459–471. doi:10.1007/s10898-007-9149-x
14. Li X, Yin M (2012) Self-adaptive constrained artificial bee colony for constrained numerical optimization. Neural Comput Appl 24(3–4):723–734. doi:10.1007/s00521-012-1285-7
15. Yang XS, Deb S Cuckoo search via Lévy flights. In: Abraham A, Carvalho A, Herrera F, Pai V (eds) Proceeding of world congress on nature & biologically inspired computing (NaBIC 2009), Coimbatore, December 2009. IEEE Publications, USA, pp 210–214
16. Ouaarab A, Ahiod B, Yang X-S (2014) Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput Appl 24(7–8):1659–1669. doi:10.1007/s00521-013-1402-2
17. Yang X-S, Deb S (2013) Cuckoo search: recent advances and applications. Neural Comput Appl 24(1):169–174. doi:10.1007/s00521-013-1367-1
18. Li X, Wang J, Yin M (2013) Enhancing the performance of cuckoo search algorithm using orthogonal learning method. Neural Comput Appl 24(6):1233–1247. doi:10.1007/s00521-013-1354-6
19. Wang G-G, Gandomi AH, Zhao X, Chu HCE (2014) Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. Soft Comput. doi:10.1007/s00500-014-1502-7
20. Yang XS (2010) Nature-inspired metaheuristic algorithms, 2nd edn. Luniver Press, Frome
21. Mirjalili S, Mirjalili SM, Yang X-S (2013) Binary bat algorithm. Neural Comput Appl 25(3–4):663–681. doi:10.1007/s00521-013-1525-5
22. Fister Jr I, Fong S, Brest J, Fister I Towards the self-adaptation in the bat algorithm. In: Proceedings of the 13th IASTED international conference on artificial intelligence and applications, 2014. doi:10.2316/P.2014.816-011
23. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. doi:10.1016/j.advengsoft.2013.12.007
24. Saremi S, Mirjalili SZ, Mirjalili SM (2014) Evolutionary population dynamics and grey wolf optimizer. Neural Comput Appl. doi:10.1007/s00521-014-1806-7
25. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98. doi:10.1016/j.advengsoft.2015.01.010
26. Gandomi AH, Yang X-S, Alavi AH (2011) Mixed variable structural optimization using firefly algorithm. Comput Struct 89(23–24):2325–2336. doi:10.1016/j.compstruc.2011.08.002
27. Yang XS (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2(2):78–84. doi:10.1504/IJBIC.2010.032124
28. Wang G-G, Guo L, Duan H, Wang H (2014) A new improved firefly algorithm for global numerical optimization. J Comput Theor Nanos 11(2):477–485. doi:10.1166/jctn.2014.3383
29. Guo L, Wang G-G, Wang H, Wang D (2013) An effective hybrid firefly algorithm with harmony search for global numerical optimization. Sci World J 2013:1–10. doi:10.1155/2013/125625