Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-020-01702-y

ORIGINAL RESEARCH

# Improving monarch butterfly optimization through simulated annealing strategy

Dongfang Yang[1] · Xitong Wang[2] · Xin Tian[3] · Yonggang Zhang[2]

Received: 15 July 2019 / Accepted: 7 January 2020
© The Author(s) 2020

**Abstract**
Currently, a novel of meta-heuristic algorithm called monarch butterfly optimization (MBO) is presented for solving machine learning and continuous optimization problems. It has been proved experimentally that MBO is superior to artificial bee colony algorithm (ABC), ant colony optimization algorithm (ACO), Biogeography-based optimization (BBO), differential evolution algorithm (DE) and simple genetic algorithm (SGA) algorithms on most test functions. This paper presents a new version of MBO with simulated annealing (SA) strategy called SAMBO. The SA strategy is put in the migration operator and butterfly adjusting operator. So the newly proposed algorithm has two features: One is that the algorithm accepts all the butterfly individuals whose fitness are better than their parents. The other is that the algorithm randomly selects some individuals which are worse than their parents to disturbance the convergence of algorithm. In this way, the SAMBO algorithm can escape from local optima. Finally, the experiments are carried on 14 continuous nonlinear functions, and results show that SAMBO method exceeds the MBO algorithm on most test functions.

**Keywords** Monarch butterfly optimization · Simulated annealing · Benchmark problems · Neighborhood search

## 1 Introduction

Recently, more and more nature-inspired algorithms (Yang 2014) have been proposed and generally applied in numerous applications, such as path planning (Wang et al. 2012), machine learning (Zhou 2016), knapsack problem (Feng et al. 2015), fault diagnosis (Duan and Luo 2015; Gao et al. 2008) and directing orbits of chaotic system (Cui et al. 2013). Among all kinds of natural-inspired algorithms, clustering algorithms and evolutionary algorithms are the most representative ones.

Analysis of nature-inspired algorithms from the exploration and search space shows that they both have two major components: exploitation and exploration, also called reinforcement and diversification. Exploitation is mainly to obtain relevant information from the search space of neighborhood to generate new solution superior to the current. But in the process it always presents to attempt to find the local optimal solution. Therefore, the advantage of exploitation is that it usually has fast convergence. But its disadvantage is also obvious that it is easy to fall into the local optimum. Exploration is the opposite. Generally speaking, the search space of exploration is within the global scope, so the diversity of search space is better. Therefore exploration could generate new solutions far away from the existing ones. It is not easy for exploration to fall into the local optimal, so the possibility of finding the global optimal will be greatly improved. But it is slow to converge and has a lot of calculation. In order to make the best performance of the algorithm, it is important to balance exploitation and exploration.

SI was originally inspired by the collective behavior of social insects. It was first formally proposed by Dorigo et al. in his book "Swarm Intelligence: From Natural to Artificial Systems" in 1999. The following particle swarm optimization (PSO) (Zhao 2010; Kennedy and Eberhart 2002; Mirjalili et al. 2014), animal migration optimization (AMO) (Li et al. 2014), artificial fish swarm algorithm (Zainal et al.

✉ Yonggang Zhang
zhangyg@jlu.edu.cn

Dongfang Yang
yangdk18@mails.jlu.edu.cn

1 College of Software, Jilin University, Changchun 130012, China

2 College of Computer Science and Technology, Jilin University, Changchun 130012, China

3 Second Hospital of Jilin University, Changchun 130041, China

2015) and monarch butterfly optimization (MBO) (Wang et al. 2015) belong to this category.

Neighborhood search (NS) (Ahuja et al. 2002) is an incomplete and perturbed search method. This approach starts with the candidate solution, iteratively looking for a better alternative in its neighborhood. The neighborhood is a basic concept in optimization, which in metric space is a sphere centered on one point. When seeking an extreme value for a continuous function, you can implement the approximation of function extreme by seeking the direction of independent variable's change to accomplish iteration of location in the neighborhood of current location. Neighborhood search is very practical. Firstly, it requires a certain amount of solutions, so it needs less memory space. Secondly, it usually show nice performance, and could find solutions faster than large complete approaches. Finally, it can be applied to combinational optimization problem through redefining the concept of neighborhood in discrete space.

Neighborhood search have been applied triumphantly in many combinatorial optimization problems, such as bin-packing (Ceschia et al. 2013) and scheduling (Ceschia and Schaerf 2011). Several popular NS methods appear, such as hill climbing (HC) (Sha et al. 2012), steepest descent (SD) (Meza 2010), simulating annealing (SA) (Kirkpatrick et al. 1983), and tabu search (TS) (Glover 1998; Glover and Marti 2006).

Inspired by the migration behavior of monarch butterfly individuals in the nature, Wang et al. put forward monarch butterfly optimization (MBO) (Wang et al. 2015) recently. The algorithm is obtained by simplifying and idealizing the migration behavior of monarch butterfly. The algorithm divides the butterfly population into two parts, one for exploitation and the other or exploration. In the process of iteration, migration operator and butterfly adjusting operator are respectively used to generation the new population. MBO is easy to operate, and the operation of two operators is theoretically parallel.

In order to escape local optima to get better fitness and accept moves which reduce the solution quality, MBO adopts simulated annealing. This paper presents an improved version of MBO, called SAMBO. In SAMBO, simulated annealing is involved into the migration operator and butterfly adjusting operator. If the new generated individuals' fitness outperforms their parents', the algorithm will accept all of them as the new generation. Otherwise, the algorithm receives it with certain probability. While basic MBO accepts all the new generated individuals. Obviously, this new algorithm can speed up convergence and augment the diversity of individual in the later search process. Towards proving the advantage of SAMBO, a series of experiences are performed on 14 test instances. The results indicate that SAMBO perform better than basic MBO in finding the optimal solution on almost all the benchmarks.

Section 2 reviews MBO algorithm and simulating annealing. Section 3 discusses how to incorporate the simulating annealing strategy into the migration operator and butterfly adjusting operator. Section 4 shows the comparison of MBO and SAMBO through testing on 14 benchmark problems. Finally, Sect. 5 outlines the summary of algorithm and the plan for the future work.

## 2 Literature reviews

### 2.1 Monarch butterfly optimization

As one of the most familiar North American butterflies, the monarch butterfly has an orange and black pattern that can be easily recognized (Garber 2013). Male and female monarchs have different wings, which makes it easy to tell them apart. It is well known that every summer North American monarch will fly thousands of miles from the USA and southern Canada to Mexico to complete its migration, it involves flying over the Rocky Mountains in the west to California. In order to overwinter, they travel thousands of miles to Mexico. The southward movement occurs in August and ends with the first frost. Then, in the spring, they moved back from the south. Females lay eggs during these migrations to produce offspring. Recent studies have shown that some monarchs show levy flight as they migrate or move. By simulating the migration behavior of Monarch butterflies in nature, a new nature-inspired algorithm has been proposed, which is called monarch butterfly optimization (MBO). The Monarch Butterfly Optimization algorithm mainly solves continuous Optimization problems. In monarch optimization, all individual monarch butterflies are idealized and distributed only in two locations, namely, southern Canada, northern United States (Land 1) and Mexico (Land 2). Therefore, the location of monarch butterflies can be updated in two ways. First, all descendants are generated by the migration operator, which can be adjusted by the migration ratio. Then, the position of the other butterflies is adjusted by the butterfly adjustment operator. In other words, the search direction of all butterfly individuals in the monarch butterfly optimization algorithm is mainly determined by migration operator and butterfly adjustment operator. The migration operator and the butterfly adjustment operator can be implemented simultaneously. Therefore, the monarch butterfly optimization algorithm can be processed in parallel theoretically, and can balance intensification and diversification well, which is a very important point in the field of metaheuristic algorithm.

To solve optimization problems, the migration of monarch butterfly could be reduced to the following regulations in MBO (Wang et al. 2015).

1. The butterflies which are on Land1 and Land2 make up the whole species group.
2. Every progeny is just produced by the individuals in Land 1 or Land 2.
3. The scale of the butterfly species group remains the same.
4. There is elitist strategy in MBO that the individual with the best fitness is preserved.

### 2.1.1 Migration operator

By simplifying monarch butterflies' migration process, the number of butterfly individuals in Land1 and Land2 can be recorded as $ceil(p * NP)(NP1)$ and $NP - NP1(NP2)$ respectively, according to the time in Land 1 and Land 2. Here, $ceil(x)$ rounds $x$ to the nearest integer greater than or equal to $x$; $NP$ represents the total number of butterfly individuals; $p$ presents the ratio of butterfly individuals in Land 1.This migration process can be marked out below:

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

where $x_{i,k}^{t+1}$ indicates $k$th element of $x_i$ at $t+1$ generation. Alike, $x_{r_1,k}^{t}$ indicates the $k$th element of $x_{r_1}$ at t generation. Monarch butterfly $r1$ is chosen from Land1 randomly. While $r \le p, x_{i,k}^{t+1}$ is updated by Eq. (1). $r$ is computed as Eq. (2)

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

where rand is a random number gained from uniform distribution. $peri$ is set to 1.2 that presents the migration period in basic MBO. When $r > p, x_{i,k}^{t+1}$ is updated by Eq. (3)

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

where $x_{r_2,k}^{t}$ indicates the $k$th element of $x_{r_2}$ at $t$ generation. Monarch butterfly $r2$ is randomly chosen from Land2.

### 2.1.2 Butterfly adjusting operator

For all the elements of monarch butterfly $j$, if $rand \le p$, it is generated as Eq. (4):

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

in the same way where $x_{j,k}^{t+1}$ indicates the $k$th element of $x_j$ at $t+1$ generation, and $x_{best,k}^{t}$ indicates the $k$th element of $x_{best}$ which owns the best fitness in the whole species group. If $rand > p$, it could be generated as Eq. (5):

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

where $x_{r_3,k}^{t}$ indicates the $k$th element of $x_{r_3}$ . Butterfly $r_3$ is randomly chosen from Land 2.

With this addition,if $rand > BAR$, it is generated as Eq. (6) (Wang et al. 2015):

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

where $BAR$ presents butterfly adjusting rate. $d_x$ indicates the walk step of butterfly $j$ which could be computed through Lévy flight as shown in Eq. (7). $\alpha$ indicates the weighting factor as follow in Eq. (8):

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

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

where $S_{max}$ indicates the max walk step.

Although MBO could obtain the better solution, it has a poor performance on finding the average value. So we propose a new method that could escape from local optimal to get the better value. And the part of Sect. 5 gives the result of experimental proof.

## 2.2 Simulated annealing

Simulated annealing (SA) was first proposed by Metropolis in 1953, and used into combinatorial optimization problems by Kirkpatrick in 1983. SA mimics the annealing process of solid matter as the optimization process, where the energy function is expressed in form of the objective function. The simulated annealing strategy can search for the system state with the lowest energy, and select the adjacent state of raised energy with certain probability along with the decrease of temperature.This makes SA completely different from the greedy strategy, and the ability to escape from the local optima to be greatly enhanced.

One of the limitations of neighborhood search methods is that only local knowledge, the neighborhood $N(s)$ of the solution $s$, is thought over at each step. So these neighborhood search methods are simple to fall into the local optimum. To avoid such, $SA$ accepts a worsening move $m \in N(s)$ at certain probability which depends on $\Delta_f(s,m)$. The bigger is the loss of solution's quality attracted by move, the less likely move is chosen. If the move gets a better function value, it must be selected. The probability of electing a move $m$ is calculated as Eq. (9) (Urli 2015):

$$P(m|s, temp) = \begin{cases} e^{-\Delta_f(s,m)/temp} & if \quad \Delta_f(s,m) > 0 \\ 1 \quad otherwise & \end{cases} \tag{9}$$

To command the frequency of accepting worse moves, a parameter $temp$ (temperature) is presented and initialized to $t_0$ which could be calculated heuristically or adjusted for the special problems. As the time passes, $temp$ can be updated continually in terms of a cooling schedule. Then,a random parameter $r'$ is a random number in (0, 1). If

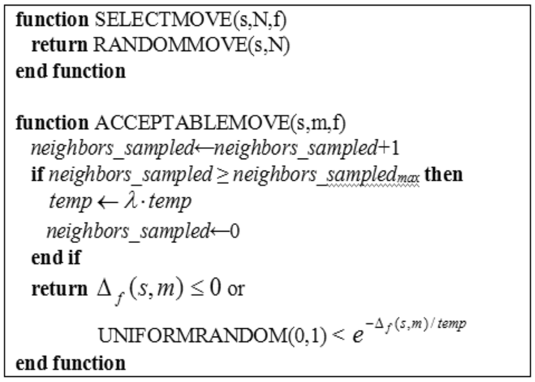
$r' < P(m|s, temp)$, the move m is adopted. If not, the move is refused. Due to influence of *temp*, SA will adjust the value of *temp* as time goes. At the start of the search, many worse moves are adopted when *temp* is large. As *temp* declines, the algorithm accepts less and less moves, and implements standard hill climbing (Sun et al. 2018b) in the last stage of search.

The usual method to reduce the value of *temp* is to employ a geometric cooling schedule, i.e.,to choose a cooling rate parameter $\lambda (0 < \lambda < 1)$, and to generate the new temperature as Eq. (10):

$$temp = \lambda \cdot temp \tag{10}$$

after lots of moves' $neighbors\_sampled_{max}$ have been measured for acceptance. The main component of simulated annealing is shown as Fig. 1.

Customarily when $temp < t_{min}$ (set by experimenter), the search quits. But we control the stop condition through the iteration generation of monarch butterfly in algorithm.



```
function SELECTMOVE(s,N,f)
  return RANDOMMOVE(s,N)
end function

function ACCEPTABLEMOVE(s,m,f)
  neighbors_sampled←neighbors_sampled+1
  if neighbors_sampled ≥ neighbors_sampled_max then
    temp ← λ·temp
    neighbors_sampled←0
  end if
  return $\Delta_f(s,m) \le 0$ or
      UNIFORMRANDOM(0,1) < $e^{-\Delta_f(s,m)/temp}$
end function
```

**Fig. 1** Components of simulated annealing

## 3 Research methodology

MBO is a newly proposed nature-inspired meta-heuristic algorithm. Though it has revealed its advantage on some benchmark problems, it may fail to obtain the optimal property on average value. In this paper, the basic MBO algorithm incorporates simulated annealing to enhance the property of algorithm. The main structure of SAMBO method is given below.

The butterfly individuals in Land1 and in Land2 are updated according to Eqs. (1)–(3) and Eqs. (4)–(8), respectively. If the fitness $f(x_i^{t+1})$ of generated individual $x_i^{t+1}$ is less than the fitness $f(x_i^t)$, it will be accepted. Otherwise $P(m|s, temp)$ can be formulated by Eq. (9). If random number $r'$ is less than $P(m|s, temp)$,the generated individuals could also be accepted. Besides, the algorithm will refuse new generated individuals. This is the difference between simulated annealing and greedy strategy.

**Algorithm 1** Pseudo code of renewed migration operator

```
for i=1 to NP1 (all the individuals in Land 1) do
    for k=1 to dim ( all the elements of ith butterfly) do
        Generate $x_{i,k}^{t+1}$ by Eqs.(1) and Eq.(3).
    end fork
    Evaluate the fitness $f(x_i^{t+1})$ of $x_i^{t+1}$
    if $f(x_i^{t+1}) < x_i^t$ then
        $x_{i,new}^{t+1} = x_i^{t+1}$
    end if
    Generate a random number $r'$ in (0,1).
    Calculate $P(m|s, temp)$ as Eq.(9)
    if $r' < P(m|s, temp)$ then
        $x_{i,new}^{t+1} = x_i^{t+1}$
    else
        $x_{i,new}^{t+1} = x_i^t$
    end if
end fori
```

**Algorithm 2** Pseudo code of renewed butterfly adjusting operator

```
1: for j=1 to NP2 ( all the individuals in Land 2) do
2:     Calculate the walk step dx by Eq.(7);
3:     Calculate the weighting factor by Eq.(8);
4:     for k=1 to dim ( all the elements of jth butterfly) do
5:         Generate $x_{j,k}^{t+1}$ by Eqs.(4)-Eq.(6).
6:     end fork
7:     Evaluate the fitness $f(x_j^{t+1})$ of $x_j^{t+1}$
8:     if $f(x_j^{t+1}) < x_j^t$ then
9:         $x_{j,new}^{t+1} = x_j^{t+1}$
10:    end if
11:    Generate a random number $r'$ in (0,1).
12:    Calculate $P(m|s, temp)$ as Eq.(9)
13:    if $r' < P(m|s, temp)$ then
14:        $x_{j,new}^{t+1} = x_j^{t+1}$
15:    else
16:        $x_{j,new}^{t+1} = x_j^t$
17:    end if
18: end forj
```

**Algorithm 3** SAMBO

```
1: Initialize the population NP of all the monarch butterflies,initial temperature $t_0$ and cooling rate $\lambda$ .Set the generation counter $t$=1, the maximum generation $MaxGen$,the population $NP1$ of butterflies in Land 1,and the population $NP2$ of butterflies in Land 2
2: Fitness evaluation: Evaluate each individual
3: while $t < MaxGen$ do
4:     Sort the butterfly population according to their evaluation. Divide all the butterfly individuals into Land 1 and Land 2
5:     for $i = 1$ to $NP_1$ do
6:         Generate $x_{i,new}^{t+1}$ as algorithm 1
7:     end fori
8:     for $j = 1$ to $NP_2$ do
9:         Generate $x_{j,new}^{t+1}$ as algorithm 2
10:    end forj
11:    Combine new updated Land1 with Land2 to generate a new population
12:    Evaluate each butterfly individual
13:    $temp = temp * \lambda$
14:    $t = t + 1$
15: end while
```

On the basis of above analysis, the updated migration operator and updated butterfly adjusting operator are shown in algorithm 1. and algorithm 2. The main frame of SAMBO could be described in algorithm 3. The algorithm flow chart is as follows:

It is notable when the simulated annealing (SA) joins. If SA occurs too early, it is prejudicial for convergence of algorithm. If it is cited late, the algorithm has already fallen into local optima so that the simulated annealing has no effect. And initial temperature and cooling rate of algorithm have much influence on final function value. So there are three parameters to be adjusted (Fig. 2).

## 4 Experiment

### 4.1 Dataset descriptions and parameters setting

In this section, toward investigating the advantage of SAMBO, fourteen high-dimensional benchmark functions which are shown in Table 1 are adopted to evaluate the algorithm improved above. Among all the benchmark functions, some are multimodal, which means that they have multiple local minima, some are nonseparable, which means that they cannot be written as a sum of functions of individual variables, some are regular, which means they are analytical (differentiable) at each point of their domain. Each of the functions in this study has 20 independent variables. The functions are summarized in Table 1. More information about these functions, including their domains, can be found in Back (1996), Yao et al. (1999), and Cai and Wang (2006). In addition, all the tests are implemented under the same conditions: MATLAB R2014b on Win7 64-bit Intel i7-3770 processor, 8 GB RAM.

The property of SAMBO was compared with MBO, probability-based incremental learning (PBIL), particle swarm optimization (PSO). The same parameters of MBO method and SAMBO method are set as follows: max step $S_{max} = 1.0$, butterfly adjusting rate $BAR = 5/12$, migration period $peri = 1.2$, the migration ratio $p = 5/12$, population size $NP = 50$, and maximum generation $MaxGen = 50$. Accordingly, the monarch butterfly individuals' number in Land1 and Land2 are 21 and 29, respectively. For other algorithms' parameters, the settings could be referred as (Dan 2008).

In order to make the simulated annealing strategy play its greatest advantage, it is advisable to consider which generation is suitable to introduce the simulated annealing. In the simulated annealing process there are two parameters initial temperature and cooling rate. Different initial temperature and cooling rate will lead to disparate results, so all of three parameters influencing function value need to be tuned. Simulated annealing is introduced at 10, 15, 20 generation respectively in the experiments, the initial temperature is set to 200, 500 and 1000 respectively, and the cooling rate is set to 0.9, 0.95, 0.99, respectively. There are 27 forms to arrange combinations of three parameters. We run all combination forms on each function. In order to reduce the effect of random function in the experiment, each experiment runs two hundred times independently. The experimental result of SAMBO shown in tables is the best situation in the all experiments under all the different combinations of three parameters. The each value recorded in tables is the average value of 200 experiments, and the optimum of best and mean solution for each function is bold. By the way, in the paper proposed by MBO algorithm, the author has already compared MBO algorithm with ABC, ACO, BBO, DE and SGA algorithms, and concluded that MBO algorithm is better than these algorithms in most cases (see the table below for details). Therefore, in this paper, we do not repeat the comparison between SAMBO algorithm and these algorithms. So, in the experimental process, we only compared our proposed SAMBO algorithm with MBO, PBLB and PSO algorithms while the benchmark function dimension is 20, 40 and 80. The experimental results show that, in most cases, our proposed SAMBO algorithm is superior to other algorithms in terms of best solution time and average solution time. See Tables 2, 3, and 4 for details.

### 4.2 Experimental results and performance evaluation

#### 4.2.1 Experimental results with benchmark functions of 20 dimensions

The dimension of benchmark functions is set to 20. The number of iteration is taken as stop condition. It is set to 50. Table 2 shows the experiment results obtained by MBO, SAMBO, PBIL and PSO.

For the best and mean values shown in Table 2, there is no ambiguity in concluding that SAMBO could find the better value of most functions than other algorithms except the best value of F04, F06 and F11 and mean value of F13. It can be observed that even though the best value obtained by SAMBO is slightly better than those got by three other methods on most test functions, SAMBO significantly outperforms MBO, PBIL and PSO on finding the mean value of functions. Then there is some improvement on mean values for follow functions. For F04 and F05, the result that SAMBO obtains is smaller than those which found by MBO about 6 times and 7.7 times, respectively, about 140 times smaller while compared with PBIL and PSO. For F08 and F10, SAMBO's solution is smaller than that of MBOPBIL and PSO about 30 times. Compared with other algorithms on the F02 and F06, SAMBO has a little advantage. And SAMBO has a slight advantage over MBOPBIL and PSO on certain functions such as F01, F03, F07, F09 and F12. It's worth noting that MBO and PBIL have a terrible performance on the mean value of F11. For optimal solution,

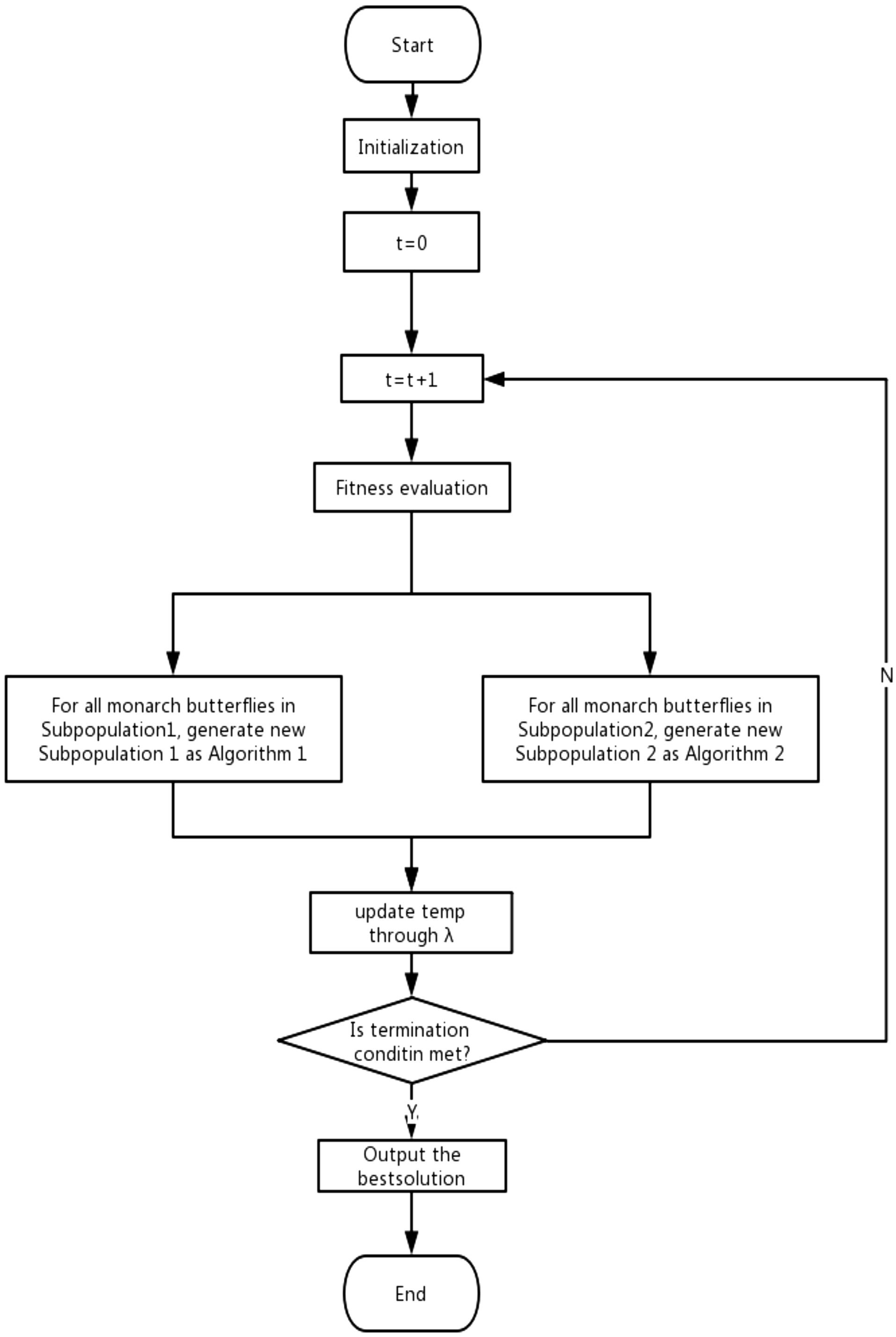

Start
Initialization
t=0
t=t+1
Fitness evaluation
For all monarch butterflies in Subpopulation1, generate new Subpopulation 1 as Algorithm 1
For all monarch butterflies in Subpopulation2, generate new Subpopulation 2 as Algorithm 2
update temp through λ
Is termination conditin met?
N
Y
Output the bestsolution
End

**Fig. 2** SAMBO algorithm flowchart

SAMBO has a good performance on most of all the test functions expect F04, F06 and F11.

#### 4.2.2 Experimental results with benchmark functions of 40 dimensions

The dimension of benchmark functions is set to 40. The number of iteration is taken as stop condition. It is set to 50. Table 3 shows the experiment results obtained by MBO, SAMBO, PBIL and PSO. It can be obviously observed that SAMBO outperforms three other algorithms on most test functions except F12 and the best value of F06.

SAMBO significantly has better average function values than MBO, especially F10 and F11. For F10, the mean value obtained by SAMBO is smaller than that of MBO PBIL and PSO about 144 times, 30 times and 35 times, respectively.

**Table 1** Benchmark functions

| Name | Multimodal? | Separable? | Regular | Range of each dimension |
|---|---|---|---|---|
| Ackley | Yes | No | Yes | ±30 |
| Fletcher–Powell | Yes | No | No | $\pm\pi$ |
| Griewank | Yes | No | Yes | ±600 |
| Penalty1 ♯1 | Yes | No | Yes | ±50 |
| Penalty2 ♯2 | Yes | No | Yes | ±50 |
| Quartic | No | Yes | Yes | ±1.28 |
| Rastrigin | Yes | Yes | Yes | ±5.12 |
| Rosenbrock | No | No | Yes | ±2.048 |
| Schwefel 1.2 | No | No | Yes | ±65.536 |
| Schwefel 2.21 | No | No | No | ±100 |
| Schwefel 2.22 | Yes | No | No | ±10 |
| Schwefel 2.26 | Yes | Yes | No | ±512 |
| Spher | No | Yes | Yes | ±5.12 |
| Step | No | Yes | No | ±200 |

Especially for the mean value of F11, SAMBO is much better than others about 27 orders of magnitude. And for F04, F05 and F08, the average values acquired by SAMBO are 4 times smaller than that of MBO. While compared with PBIL and PSO, the newly proposed method is better about 37 times, 28 times and 13 times respectively. Then SAMBO has a little advantage over MBO, PBIL and PSO on certain functions such as F01, F02, F03, F07, F09, F13 and F14. And for F06, PSO performs best. Although MBO finds better solutions on the best and mean value of F12, the results obtained by each algorithm are not much different.

### 4.2.3 Experimental results with benchmark functions of 80 dimensions

The dimension of benchmark functions is set to 80. The iteration generation is taken as stop condition and set to 50. This time we only choose five test functions (F02, F04, F05, F08 and F10), because the advantage of SAMBO could be seen clearly and shown in the convergent graphs for these functions. Table 4 shows the experiment results acquired by MBO, SAMBO, PBIL and PSO. And this paper primarily demonstrates that the MBO has been improved, so we just provide the convergence curves of MBO and SAMBO on F02, F04, F05, F08 and F10.

From Table 4, there is no ambiguity in concluding that SAMBO outperforms MBO, PBIL and PSO. And compared with MBO, the advantage of SAMBO on F02, F04, F05, F08 and F10 are revealed in Figs. 5, 6 and 7.

Figure 3 shows the convergence curve of SAMBO and MBO on F02.The curve of MBO tends to be flat quickly, while the curve of SAMBO is always going down. The performance of SAMBO is much greater than MBO.

Figure 4 demonstrates the convergent process of SAMBO and MBO on F04. Except the fitness obtained by MBO is better than that of SAMBO at the generation 6,the curve of SAMBO is lower than MBO's all the time. And their gap is getting bigger and bigger later. Finally, the two curves tend to be parallel, and the final fitness of SAMBO is much smaller than MBO's.

Figure 5 illustrates the convergent trajectory of SAMBO and MBO on F05. Although initial value of SAMBO is worse than MBO's initial value, the convergence of SAMBO happens earlier than MBO. Ultimately, the curve of SAMBO is much lower than MBO's curve.

**Table 2** Best and mean function values obtained by MBO and SAMBO ($D = 20$)

| | MBO | | SAMBO | | PBIL | | PSO | |
|---|---|---|---|---|---|---|---|---|
| | Best | Mean | Best | Mean | Best | Mean | Best | Mean |
| F01 | 11.7705 | 12.6519 | *10.6003* | *11.3043* | 19.2873 | 20.9663 | 16.0218 | 19.8198 |
| F14 | 0.3735e+06 | 1.1873e+06 | *2.2333e+05* | *2.3636e+05* | 5.8347e+05 | 2.2663e+06 | 5.2603e+05 | 2.2404e+06 |
| F03 | 96.1304 | 136.2024 | *76.3990* | *99.2212* | 234.5175 | 656.1139 | 83.2786 | 332.4701 |
| F04 | 2.1934e+07 | 4.9713e+07 | 7.8373e+06 | *8.0896e+06* | 9.2807e+07 | 1.1500e+09 | *6.0524e+06* | 1.1606e+09 |
| F05 | 0.6782e+08 | 1.4290e+08 | *1.7739e+07* | *1.8635e+07* | 2.2276e+08 | 1.9643e+09 | 2.8503e+07 | 1.9584e+09 |
| F06 | 10.5719 | 44.5388 | 7.1984 | *11.9005* | 18.7189 | 141.2131 | *3.4290* | 141.5007 |
| F07 | 88.2089 | 114.7303 | *74.4715* | *79.6291* | 221.3520 | 377.6148 | 165.1462 | 296.7657 |
| F08 | 747.4 | 2668.4 | *328.9859* | *360.3084* | 2.0265e+03 | 1.0981e+04 | 592.1046 | 1.1078e+04 |
| F09 | 3.5192e+03 | 4.2384e+03 | *2.8417e+03* | *2.8529e+03* | 5.3140e+03 | 8.4250e+03 | 5.1332e+03 | 8.1747e+03 |
| F10 | 0.1028e+05 | 4.9686e+05 | *7.2452e+03* | *7.4306e+03* | 1.2434e+04 | 2.5749e+05 | 8.4943e+03 | 2.9529e+05 |
| F11 | *24.9725* | 1.3684e+12 | 25.0488 | *34.6025* | 61.6260 | 2.0336e+14 | 46.2554 | 46.2554 |
| F12 | 35.3581 | 35.4068 | *29.8312* | *29.8628* | 66.9115 | 98.9670 | 53.0918 | 96.3481 |
| F13 | 23.2107 | *23.2107* | *21.9257* | 34.6831 | 68.7995 | 192.6548 | 24.0148 | 97.3720 |
| F14 | 1.0641e+04 | 1.5337e+04 | *7.2464e+03* | *7.3133e+03* | 2.6234e+04 | 7.3013e+04 | 9.2383e+03 | 3.6359e+04 |

Italic values represent the optimal solution

**Table 3** Best and mean function values obtained by MBO and SAMBO ($D$ = 40)

| | MBO | | SAMBO | | PBIL | | PSO | |
|---|---|---|---|---|---|---|---|---|
| | Best | Mean | Best | Mean | Best | Mean | Best | Mean |
| F01 | 14.8501 | 15.5245 | *13.6714* | *14.3519* | 20.1512 | 21.0346 | 17.6506 | 19.8970 |
| F02 | 2.9336e+06 | 5.7194e+06 | *1.5742e+06* | *1.7756e+06* | 3.9607e+06 | 9.4956e+06 | 3.7530e+06 | 9.5174e+06 |
| F03 | 324.6347 | 474.6744 | *257.0004* | *279.0702* | 677.2620 | 1.3855e+03 | 310.7108 | 746.2803 |
| F04 | 1.3332e+08 | 2.5998e+08 | *5.8350e+07* | *6.5413e+07* | 5.4517e+08 | 2.6694e+09 | 6.2306e+07 | 2.4049e+09 |
| F05 | 2.6344e+08 | 4.9240e+08 | *1.3252e+08* | *1.4393e+08* | 1.1153e+09 | 4.4753e+09 | 1.6530e+08 | 4.0011e+09 |
| F06 | 80.2079 | 264.8352 | 71.8131 | *80.5933* | 155.6223 | 607.5816 | *32.1506* | 567.5818 |
| F07 | 294.9648 | 384.7409 | *245.2640* | *257.0589* | 531.6699 | 765.9487 | 398.1688 | 584.8991 |
| F08 | 3.2633e+03 | 7.3208e+03 | *1.4958e+03* | *1.7963e+03* | 8.1280e+03 | 2.5370e+04 | 2.3835e+03 | 2.3577e+04 |
| F09 | 8.2927e+03 | 9.6962e+03 | *7.3367e+03* | *7.4462e+03* | 1.2465e+04 | 1.6836e+04 | 1.2602e+04 | 1.6579e+04 |
| F10 | 4.5868e+04 | 4.5824e+06 | *2.8705e+04* | *3.1838e+04* | 4.7609e+04 | 9.4444e+05 | 3.5841e+04 | 1.1186e+06 |
| F11 | 91.1913 | 1.1171e+28 | *75.6560* | *94.5078* | 146.1005 | 7.6706e+28 | 167.7829 | 4.0435e+29 |
| F12 | *36.6260* | *36.6846* | 36.6973 | 36.7500 | 84.5315 | 99.6902 | 88.9677 | 99.5984 |
| F13 | 101.3803 | 158.6964 | *82.0935* | *103.3361* | 202.6079 | 410.7044 | 87.1691 | 206.3962 |
| F14 | 3.3558e+04 | 4.9196e+04 | *2.1735e+04* | *2.2891e+04* | 7.5390e+04 | 1.5335e+05 | 3.4414e+04 | 8.3307e+04 |

Italic values represent the optimal solution

**Table 4** Best and mean function values obtained by MBO and SAMBO ($D$ = 80)

| | MBO | | SAMBO | | PBIL | | PSO | |
|---|---|---|---|---|---|---|---|---|
| | Best | Mean | Best | Mean | Best | Mean | Best | Mean |
| F02 | 1.8289e+07 | 2.7423e+07 | *1.0179e+07* | *1.1553e+07* | 2.1691e+07 | 3.8800e+07 | 2.0702e+07 | 3.8692e+07 |
| F04 | 4.0641e+08 | 8.0260e+08 | *2.7744e+08* | *3.2124e+08* | 1.7532e+09 | 5.7534e+09 | 2.8269e+08 | 4.8917e+09 |
| F05 | 1.0348e+09 | 2.0543e+09 | *5.5375e+08* | *6.1619e+08* | 3.3302e+09 | 9.6167e+09 | 6.6903e+08 | 8.2376e+09 |
| F08 | 1.1140e+04 | 2.4538e+04 | *5.8067e+03* | *6.6961e+03* | 2.3444e+04 | 5.5400e+04 | 7.7114e+03 | 4.8950e+04 |
| F10 | 2.1304e+05 | 4.1116e+07 | *1.0909e+05* | *1.3203e+05* | 1.8536e+05 | 3.7114e+06 | 1.3994e+05 | 4.5329e+06 |

Italic values represent the optimal solution

Figure 6 reveals the convergent history of SAMBO and MBO on F08. Similarly with Fig. 5, though SAMBO has the worse initial value, SAMBO achieve the better fitness in the end.

Figure 7 displays the convergent track of SAMBO and MBO on F10. It could be observed visibly that the convergent amplitude of SAMBO is greater, and the final fitness found by SAMBO is much less than the value which MBO finds, despite SAMBO's fitness is much bigger than MBO's at start.

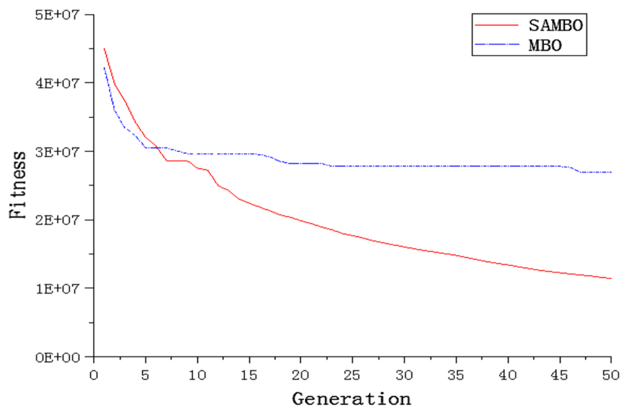


**Fig. 3** Convergence graph of SAMBO and MBO on F02 with D = 80

## 4.3 Experiment on real world dataset

Vehicle routing problem (VRP) (Elhassania et al. 2014) is one of the most challenging combinatorial optimization problems. This problem was defined more than 40 years ago and includes the design of an optimal path set for the fleet to serve a known set of customers. The practical application significance and its difficulty of solution are important in VRP. A company may have several warehouses for customers. If customers are clustered around these warehouses, the allocation problem can be modeled as several independent VRPs. However, if the distribution of customers and warehouses is mixed, this should be solved as a multi-depot vehicle routing problem (Renaud et al. 1996).

We use our proposed SAMBO algorithm to solve multi-depot vehicle routing problem, and in order to prove the

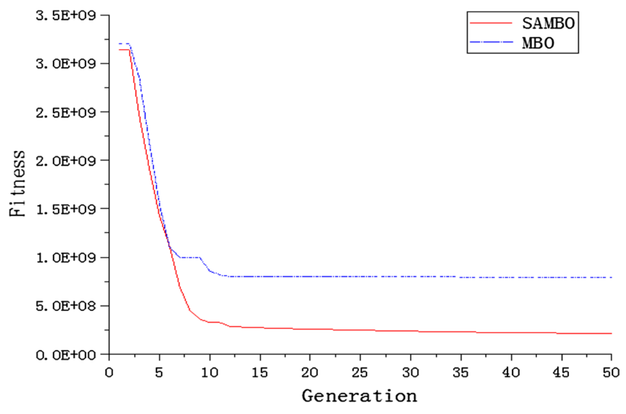


Fig. 4 Convergence graph of SAMBO and MBO on F04 with D = 80

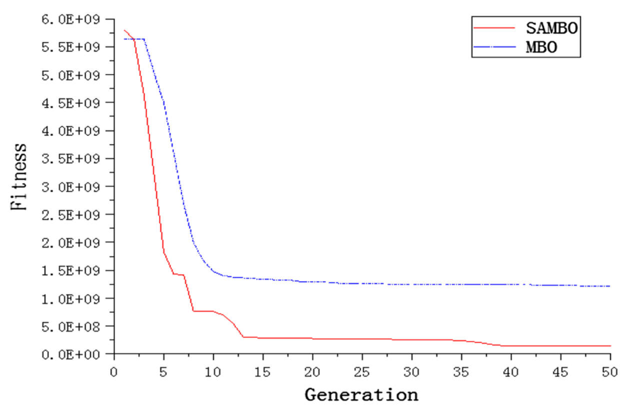


Fig. 5 Convergence graph of SAMBO and MBO on F05 with D = 80

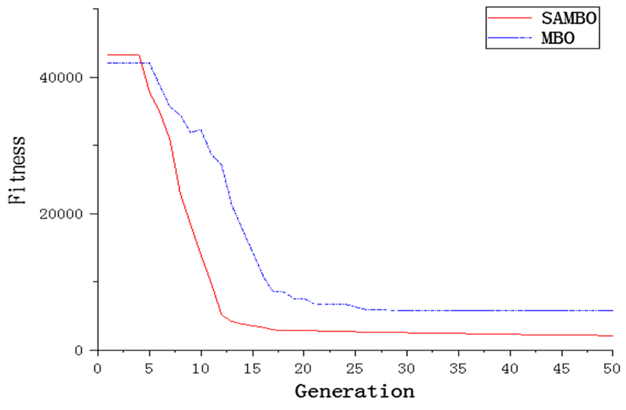


Fig. 6 Convergence graph of SAMBO and MBO on F08 with D = 80

efficiency of the new algorithm, we also use the traditional MBO algorithm and genetic algorithm to solve the same problem. In the experiment, the performance of MBO algorithm, genetic algorithm and SAMBO algorithm was tested by six unsolved multi-depot vehicle routing problems, Table 5 shows

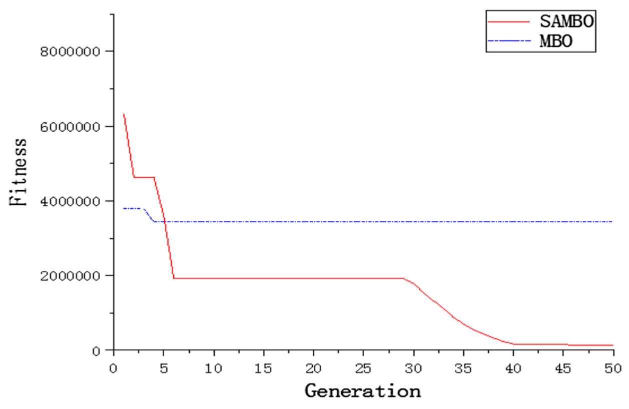


Fig. 7 Convergence graph of SAMBO and MBO on F10 with D = 80

Table 5 The specific information of the six unsolved MDVRP

| No. | Number of customer | Number of warehouses | Vehicle load |
|---|---|---|---|
| P01 | 50 | 4 | 80 |
| P02 | 75 | 5 | 140 |
| P03 | 100 | 3 | 100 |
| P04 | 100 | 4 | 100 |
| P05 | 249 | 4 | 310 |
| P06 | 249 | 5 | 310 |

the specific information of the six unsolved Multi-Depot Vehicle Routing Problems. In addition to the number of customers in the problem, the number of warehouses and the vehicle load, each instance's data also stores the coordinates of each customer's geographic location and the coordinates of the warehouse. Through the coordinates, the length of the path can be calculated and used as the cost value of the path. Table 6 shows the solutions found by three algorithms for six problems: cost is the total length of the path found, and the running time is the time required to run the algorithm once, with the unit of seconds (s). The italics represent the optimal solution.

The italics in the table clearly show that SAMBO found the best solution for all six problems. Compared with traditional genetic algorithm, both MBO and SAMBO perform better. On the first four problems (P01, P02, P03 and P04), the cost value of genetic algorithm is about 3 times of MBO and SAMBO, especially for P02, the cost value of genetic algorithm is 4 times of SAMBO. Moreover, the path found by SAMBO is indeed less expensive than that found by MBO, indicating that the algorithm has improved on the basic MBO algorithm, and with the increase of the size of the problem, the gap between solutions becomes larger and larger. For problems P05 and P06, the path cost found by the algorithm SAMBO is more than 8000 and 2000 less than that found by the genetic algorithm and the MBO

**Table 6** Three algorithms results for six problems

| | Genetic | | MBO | | SAMBO | |
|---|---|---|---|---|---|---|
| | Cost | Time (s) | Cost | Time (s) | Cost | Time (s) |
| P01 | 677.30 | 349.25 | 217.80 | 4.05 | *191.98* | 6.47 |
| P02 | 965.11 | 473.30 | 280.16 | 5.55 | *235.37* | 7.84 |
| P03 | 1.4649e+03 | 601.93 | 559.43 | 6.66 | *425.05* | 8.94 |
| P04 | 1.4107e+03 | 646.41 | 504.41 | 6.89 | *420.39* | 9.97 |
| P05 | 1.3538e+04 | 1639.54 | 7.2422e+03 | 17.44 | *5.3305e+03* | 17.79 |
| P06 | 1.3295e+04 | 1761.36 | 7.1036e+03 | 19.08 | *5.2335e+03* | 25.86 |

Italic values represent the optimal solution

algorithm respectively. This shows that as the size of the problem increases, the advantages of the SAMBO algorithm become more and more obvious. Through experiments, it is proved that SAMBO algorithm is effective in solving the multi-warehouse vehicle routing problem.

## 5 Conclusion and future work

In recent years, more and more swarm intelligence algorithms are proposed and applied in many different areas. MBO is a newly proposed meta-heuristic algorithm to solve continuous optimization problems and testified that it outperforms five other meta-heuristic algorithms on most of all the test problems. But the ability of MBO to find the average value is poor on certain test functions. So the simulated annealing strategy is incorporated into MBO algorithm to escape local optima. A novel version of MBO with simulated annealing called SAMBO is proposed. In MBO, all the updated individuals are accepted. While in SAMBO method, except the updated butterflies whose fitness is better than their parents are accepted, the algorithm could adopt the worsening new individuals with a certain probability when the algorithm introduces simulated annealing strategy. This can ensure that outstanding individuals are preserved and churn search space to seek the better solution. The experiment results demonstrate that SAMBO can find better solutions than MBOPBIL and PSO especially on average value on almost of all the benchmark problems (Sun et al. 2018b, a).

But there are limits. In future, there are three points to clarify that can facilitate the future research direction. Firstly, in the current work, experiments just proceed on 14 benchmark problems. More benchmarks should be used to evaluate SAMBO. Secondly, we could study a self-adapting way to solve how to tune three parameters (Shuai et al. 2017; Liu et al. 2017; Zheng et al. 2017). If the self-adapted way is worked out, it is convenient for us to use the SAMBO method to solve what we want to handle. Finally, SAMBO is just to solve single objective problems. We can ameliorate it to deal with multi-objective problems.

**Acknowledgements** This work is supported by the National Natural Science Foundation of China (61373052), the Project of Jilin Provincial Science and Technology Development (20170414004GH).

## References

Ahuja RK, Orlin JB, Punnen AP (2002) A survey of very large-scale neighborhood search techniques. Discr Appl Math 123(1):75–102

Back T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, Oxford

Cai Z, Wang Y (2006) A multiobjective optimization-based evolutionary algorithm for constrained optimization. IEEE Trans Evol Comput 10(6):658–675

Ceschia S, Schaerf A (2011) Local search and lower bounds for the patient admission scheduling problem. Comput Oper Res 38(10):1452–1463

Ceschia S, Schaerf A, Stützle T (2013) Local search techniques for a routing–packing problem. Comput Ind Eng 66(4):1138–1149

Cui Z, Fan S, Zeng J, Shi Z (2013) Apoa with parabola model for directing orbits of chaotic systems. Int J Bio-Inspir Comput 5(1):67–72

Dan S (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12(6):702–713

Duan H, Luo Q (2015) New progresses in swarm intelligence-based computation. Bio-Inspired Comput 7(1):26–35

Elhassania M, Jaouad B, Ahmed EA (2014) Solving the dynamic vehicle routing problem using genetic algorithms. In: 2014 International conference on logistics operations management, IEEE, pp 62–69

Feng Y, Wang G, Deb S, Lu M, Zhao XJ (2015) Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization. Neural Comput Appl 28:1619–1634

Gao XZ, Ovaska SJ, Wang X, Chow MY (2008) A neural networks-based negative selection algorithm in fault diagnosis. Neural Comput Appl 17(1):91–98

Garber SD (2013) The urban naturalist. Cour Corp 25(23):33–55

Glover F (1998) Tabu search – wellsprings and challenges. Eur J Oper Res 106(23):221–225

Glover F, Marti R (2006) Tabu Search. In: Alba E, Martí R (eds) Metaheuristic procedures for training neutral networks. Operations research/computer science interfaces series, vol 36. Springer, Boston, MA

Kennedy J, Eberhart R (2002) Particle swarm optimization. In: Icnn95-international conference on neural networks

Kirkpatrick S, Gellatt CD, Vecchi PM (1983) Optimisation by simulated annealing. Science 220(4598):671–680

Li X, Zhang J, Yin M (2014) Animal migration optimization: an optimization algorithm inspired by animal migration behavior. Neural Comput Appl 24(7–8):1867–1877

Liu S, Pan Z, Cheng X (2017) A novel fast fractal image compression method based on distance clustering in high dimensional sphere surface. Fractals Complex Geometr Pattern Scaling Nat Soc 25(23):1740004

Meza JC (2010) Steepest descent. Wiley Interdiscip Rev Comput Stat 2(6):719–722

Mirjalili S, Wang GG, Coelho LS (2014) Binary optimization using hybrid particle swarm optimization and gravitational search algorithm. Neural Comput Appl 25(6):1423–1435

Renaud J, Laporte G, Boctor FF (1996) A tabu search heuristic for the multi-depot vehicle routing problem. Comput Oper Res 23(3):229–235

Sha ZC, Huang ZT, Zhou YY, Wang FH (2012) Blind spreading sequence estimation based on hill-climbing algorithm. In: IEEE international conference on signal processing

Shuai L, Zheng P, Song H (2017) Digital image watermarking method based on dct and fractal encoding. IET Image Proc 11(10):815–821

Sun G, Liang L, Teng C, Feng X, Fei L (2018a) Network traffic classification based on transfer learning. Comput Electr Eng 69:S004579061732829X

Sun G, Teng C, Su Y, Li C (2018b) Internet traffic classification based on incremental support vector machines. Mob Netw Appl 23(14):1–8

Urli T (2015) Hybrid meta-heuristics for combinatorial optimization. Constraints 20(4):473–473

Wang G, Guo L, Hong D, Luo L, Wang H, Shao M (2012) Path planning for uninhabited combat aerial vehicle using hybrid metaheuristic de/bbo algorithm. Adv Sci 4(6):550–564

Wang GG, Deb S, Cui Z (2015) Monarch butterfly optimization. Neural Comput Appl 31:1–20

Yang XS (2014) Nature-inspired optimization algorithms, vol 1. Elsevier, Amsterdam, pp 77–87

Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102

Zainal N, Zain AM, Sharif S (2015) Overview of artificial fish swarm algorithm and its applications in industrial problems. Appl Mech Mater 815:253–257

Zhao X (2010) A perturbed particle swarm algorithm for numerical optimization. Appl Soft Comput J 10(1):119–124

Zheng P, Shuai L, Fu W (2017) A review of visual moving target tracking. Multimed Tools Appl 76(16):16989–17018

Zhou Z (2016) Machine learning. Qinghua University chu ban she 1(147):1–20