Improved Brain Storm Optimization Algorithms for Optimization Problems

by

Yang Yu

A dissertation

submitted to the Graduate School of Innovative Life Science

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Engineering

University of Toyama

Gofuku 3190, Toyama-shi, Toyama 930-8555 Japan

2019

(Submitted December 16, 2019)

Acknowledgements

I would like to show gratitude to the people following for their selfless help during my research and study period. I could not gain so many fulfillments and accomplish my thesis without their help and concern.

To Professor Zheng Tang, who brought me into a brilliant and gorgeous world of artificial intelligence, by contacting this advanced subject, I expanded my vision and enriched my knowledge.

To Associate Professor Shangce Gao, who guides me into particular researches and teach me without reservation during my whole master period.

To all my research mates of the Intelligent Information System Research Lab in University of Toyama, they gave me help and concern that making me very comfortable.

In particular, I would like to thank my parents, I appreciate all they have done for me in my research period.

I would like to thank all the members of my family, for their unconditional love, support, and encouragement through this process, through all my study process.

Abstract

Brain storm optimization (BSO) is a swarm intelligence optimization algorithm which is proven to have practical values in various fields. During these years, many modifications have been facilitated to effectively improve BSO's search performance. So far, these modifications focus on improving the solution quality by applying different clustering methods and learning strategies, in which the population diversity is often neglected. However, in recent studies, BSO still suffers from sticking into stagnation during exploitation phase, and lacks of flexibility and variety which makes a poor search efficiency and robustness of BSO. Besides, the population diversity of BSO often falls into lower levels and deteriorates the solutions' quality. Population diversity plays a more significant role in designing optimization algorithm. A population that maintains its diversity in a high level can easily obtain better solutions than the one with low level of diversity. Therefore, in this thesis, I propose several improved brain storm optimization algorithms and testify them on optimization problems. These algorithms are introduced as follows.

(1) A novel method which incorporates BSO with chaotic local search (CLS) has the purpose of alleviating this situation. Chaos has properties of randomicity and ergodicity. These properties ensure CLS can explore every state of the search space if the search time duration is long enough. The incorporation of CLS can make BSO break the stagnation and keep the population's diversity simultaneously, thus realizing a better balance between exploration and exploitation. Twelve chaotic maps are randomly selected for increasing the diversity of the search mechanism. Experimental and statistical results based on 25 benchmark functions demonstrate the superiority of the proposed method.

(2) To alleviate the problem that BSO lacks of flexibility and variety, an adaptive step length structure together with a success memory selection strategy are proposed to be incorporated into BSO. This proposed method, adaptive step length based on memory selection BSO, namely ASBSO, applies multiple step lengths to modify the generation process of new solutions, thus supplying a flexible search according to corresponding problems and convergent periods. The novel memory mechanism which is capable of evaluating and storing the degree of improvements of solutions is used to determine the selection possibility of step lengths. A set of 57 benchmark functions are used to test ASBSO's search ability, and four real-world problems are adopted to show its application value. All these test results indicate the remarkable improvement in solution quality, scalability and robustness of ASBSO.

(3) This paper proposes a control method that evaluates the population diversity of BSO to improve its performance. Two diversity measures, which are known as distance-based diversity and fitness-based diversity, are implemented to realize the adaptation of algorithm parameters. The new algorithm is called multiple diversity-driven BSO (MDBSO). Its performance is verified by CEC2017 benchmark function suit and a neuron model training task. The results demonstrate the effectiveness and efficiency of MDBSO.

The thesis is organized as follows: Chapter 1 gives a brief introduction about the Swarm Intelligence (SI) algorithms. Chapter 2 briefly introduces the brain storm optimization algorithm (BSO). Chapter 3 gives one of the modifications of BSO called CBSO, which uses chaotic local search method to enhance the exploitation ability of BSO. Then, in Chapter 4, an improved brain storm optimization with flexible search length and memory-based selection (ASBSO) is introduced. In Chapter 5, I propose a multiple diversity-driven brain storm optimization algorithm with adaptive parameters (MDBSO). Finally, Chapter 6 gives some general conclusions of this thesis and also points some valuable research trends.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 A Brief History of Swarm Intelligence

In recent years, various swarm intelligence (SI) algorithms have been proposed for solving diverse optimization problems. The main property of this kind of algorithms is that they mimic the social behaviors of nature creatures. As far as we know, it is full of wisdom and intelligence when animals are hunting, foraging and navigating in nature. Survival instincts drive them to improve search ability for creating more suitable living environment. Their behaviors gradually arouse great interests among researchers in the field of artificial intelligence [1]. Particle swarm optimization (PSO) which is one of the most popular SI algorithms is modeled based on the social behaviors of flocks of birds and schools of fish [2]. It supposes that a swarm of particles fly randomly in a multidimensional search space. Each of them represents a candidate solution for the optimization task. Their trajectories change according to the best position of the individual and the global best position of the whole population. Particles can effectively search for better solutions by taking advantage of this mechanism [3, 4, 5, 6, 7, 8, 9, 10, 11].

In addition to PSO, more and more SI optimization algorithms progressively spring into our view. Ant colony optimization (ACO) [12], fireworks algorithm (FA) [13], gravitational search algorithm (GSA) [14], artificial bee colony algorithm (ABC) [15] and brain storm optimization (BSO) [16] are some powerful optimization algorithms. These SI algorithms can be roughly divided into three categories according to the

types of behaviors they take inspiration from [17, 18, 19].

The first category is called bio-inspired. Classical algorithms in this category such as ACO and ABC emulate the foraging behaviors of ant colony and bee colony, respectively. In ACO, individuals utilize a special chemical substance called pheromone to mark their search trajectory. The trajectory with more pheromone is considered as a preferred path to the global optimum, and further attracts other individuals [11]. ABC simulates the organizational structure of bees to categorize individuals into three groups: employed artificial bees, onlookers and scouts. The employed artificial bees represent candidate solutions and the onlookers are responsible for sharing the information of employed bees. After these steps, scouts are sent to diverse search area for discovering new solutions. This sophisticated idea of giving different functions to individuals makes the search procedure of ABC efficient and effective [15].

The second category can be named as physics-inspired. The algorithms belong to this category such as FA and GSA straightly take inspiration from physical phenomena or laws. For examples, the explosion processes of fireworks are utilized to design the search mechanism of FA, in which the distribution of individuals is analogized by the sparks in firework explosion. In GSA, the law of gravity is used to depict the relationship among individuals in search space. They are attracted by each other and the gravitational force is directly proportional to their fitness and inversely proportional to the square of the distance between them. The performance of GSA in different kinds of problems implies its powerful search ability [20, 21, 22].

The last category is called sociology-inspired. The major property of the algorithms in this category is that they are inspired by human social behaviors. BSO is very notable among SI algorithms and has already achieved great success in various applications [23]. Its operations of generating new individuals adopt the brainstorming process in human social behaviors. In reality, a group of people should be called together to figure out a solution when we encounter problems that can not be solved alone. This brainstorming process needs repetitive discussions and debates. BSO is enlightened by this feature and obtains an elaborated search process. At the rudimentary stage of optimization, individuals are divided into multiple clusters, then each

cluster selects the best individual as the center. BSO has four independent individual generation methods and the selections of corresponding method are depending on three preset parameters p_1, p_2 and p_3. p_1 decides the usage of one or two clusters. In the condition of using one cluster, p_2 is adopted to choose the center or one random individual in the selected cluster. Otherwise, when two clusters are selected, p_3 determines the adoption of two centers or two random individuals. Being beneficial from this sophisticated selection mechanism, BSO can avoid sticking into local optima and outperform other optimization algorithms when dealing with multimodal problems [23]. However, the inherent feature of BSO that can not maintain good diversity reduces its robustness and deteriorates the performance of solving different problems. In the meanwhile, the parameter adjustment is very important in designing algorithms but it generally costs much time to find an acceptable parameter set. Therefore, more and more researchers prefer making parameters adaptive or self-adaptive to enhance the robustness and performance of algorithms [24, 25, 26, 27, 28].

Chapter 2

Brain Storm Optimization (BSO)

2.1 Description of BSO

BSO is an SI algorithm which mimics the human brainstorming in social behaviors. Algorithm 3 gives its optimization procedure. The main difference between BSO and other SI algorithms is that BSO divides the population into n clusters and the individual with the best fitness in each cluster is selected as the center. Then, $X_{selected}$ is selected to generate new individuals according to the process controlled by p_1, p_2 and p_3. If a random number is smaller than p_1 (= 0.8 [16]), one cluster will be selected. Otherwise, two clusters are applied to generate new individuals. In the condition of using one cluster, p_2 (= 0.4 [16]) decides the usage of the center or one random individual in the selected cluster. In addition, p_3 makes the centers and random individuals in two selected clusters have equal chance to participate in generating new individuals. Besides, BSO has another parameter p_0 to control the operation that replaces one cluster center by a randomly generated individual to avoid premature convergence. Finally, the population is updated based on the elite survival rule, i.e., the old individual will be replaced by the generated individual when the old one's fitness is worse. The mutation operator of BSO is shown as:

$$X_{generated} = X_{selected} + \xi \cdot N(0,1) \tag{2.1}$$

Algorithm 1: Flowchart of BSO.

```
Randomly generate a population with N individuals;
Calculate the fitness of each individual;
while maximum number of function evaluations is not reached do
    Use k-means to divide N individuals into n clusters;
    Choose the best individual in each cluster as the center;
    if random(0,1) < p_0 = 0.2 then
        replace one cluster center by a randomly generated individual
    end
    if random(0,1) < p_1 = 0.8 then
        select one cluster;
        if random(0,1) < p_2 = 0.4 then
            choose the cluster center as X_selected
        else
            randomly choose an individual in the cluster as X_selected
        end
    else
        randomly select two clusters;
        if random(0,1) < p_3 = 0.5 then
            choose the combination of two centers as X_selected
        else
            choose the combination of two randomly selected individuals in
             two clusters as X_selected
        end
    end
    Generate new individual by adding step length generated by Eqs. (2.1)
     and (2.2) to X_selected;
    if the new individual is better than the old one then
        replace the old individual
    end
end
```

where $X_{selected}$ and $X_{generated}$ are the selected and newly generated individuals, respectively. $N(0,1)$ is the Gaussian distribution with mean 0 and variance 1. ξ is a search step length which is calculated by Eq. (2.2).

$$\xi = logsig((0.5 * iteration_{max} - t)/k) \cdot rand \tag{2.2}$$

where $logsig()$ means a logarithmic sigmoid transfer function, and its interval is $(0,1)$. $iteration_{max}$ and t are the maximum iteration and current iteration count, respectively. k ($= 20$ [16]) is a scale factor to control the slope of $logsig()$ function.

Chapter 3

CBSO: A Memetic Brain Storm Optimization with Chaotic Local Search

3.1 Introduction

Compared to deterministic algorithms, meta-heuristic algorithms have much faster development speed in the last few decades. Deterministic algorithms usually start with a similar initial point and get same answers during iterations. This behavior easily results in local optimal entrapment. In contrast meta-heuristic algorithms apply stochastic operators to get start with a random initialization which can avoid local solutions [29]. Besides the stochasticity, meta-heuristic algorithms also have other properties such as they are approximate, usually non-deterministic and not problem-specific. These properties ensure that meta-heuristic algorithms can freely develop and have various classifications. According to the properties, they can be classified as follows.

(1) Local search vs. Global search

(2) Single-solution vs. Population-based

(3) Hybridization and memetic algorithms

(4) Parallel meta-heuristics

(5) Nature-inspired meta-heuristics

Global search and local search in the first classification usually play a role in controlling the balance between exploration and exploitation [30]. The research of adjusting the balance is very important for the modification of meta-heuristic algorithms. In recent years, many attempts have been done and the effect is remarkable, among which the memetic algorithm (MA) has achieved great successes[27]. MA is a combination of evolutionary algorithms with local search. The core concept of MAs is that implementing local search with global optimization algorithm to improve the quality of individuals. Memes refers to the local search strategies such as constructive methods and local refinement. Chaotic local search is an effective strategy that can accelerate convergence speed and improve solution quality in the exploitation phase [31, 32, 33]. Chaos has some properties such as ergodicity and randomicity [34]. Its unpredictable dynamic mechanism endows the meta-heuristic algorithms with the ability of avoiding trapping into local optimal solutions. It is the motivation that BSO equipped with chaotic local search to enhance its search ability and break stagnation in the exploitation phase.

There are many methods to implement chaos into meta-heuristic algorithms. Besides the chaotic search mechanism, replacing the random variables is one main category. In [31], it uses sequences generated by chaotic maps (SGCMs) to replace the ones generated by random number generators (RNGs) in evolutionary algorithms. The experiments demonstrated that using chaotic maps instead of RNGs could get much better performance. Using chaotic sequences in population initialization, selection procedure, crossover and mutation [35, 36] operations can influence the whole optimization procedure and make complex effects. On the other hand, CLS is usually applied to the current global best agent and generates a new one. If the new agent is better than the current best, then replaces it to enter the next iteration. Gao [21] used both SGCMs and CLS into gravitational search algorithm (GSA) and the results showed that CLS was better than SGCMs.

The implementations of CLS have been greatly developed in recent years. In

[37], the mechanism of CLS is static which is exhibited in the generation of chaotic sequence and adjustment of search radius. In [38], the initialization of chaotic sequences has been randomized. Some variations are introduced in search radius and partial selection of the current global best agent.

In this paper, we propose a new incorporation method to combine BSO with CLS, namely CBSO in order to balance the exploration and exploitation of the search. Twenty-five benchmark functions are used to testify the performance of CBSO. Four algorithms are compared and non-parametric statistical analyses suggest that CBSO performs better than the compared algorithms. Additionally, population diversity and time complexity are also discussed to further give some insights into the efficiency of CBSO. The proposed method can make the new agent generated by CLS spread away from the best agent in each cluster once the whole population stick into stagnation during optimization procedure.

The paper is organized as follows. Section 3.2 gives a brief description of traditional BSO. Section 5.2 introduces 12 different chaotic maps and their distinct properties. The combination of chaotic local search and BSO will be explained in Section 5.3. Section 5.4 gives the experimental result and the implementation of statistical analysis. A discussion regarding the computational time complexity and population diversity is put forward in Section 3.5.

3.2 Chaotic Maps

Chaotic maps are used for generating chaotic sequences which are implemented into CLS. We adopt twelve well-known one-dimensional chaotic maps.

(1) Logistic map: logistic map is known as a classic chaotic map that appears in nonlinear dynamics of biological population, and it is expressed as follows:

$$z_{k+1} = \mu z_k(1 - z_k) \tag{3.1}$$

where z_k is the kth chaotic number. $z_k \in (0, 1)$ under the conditions that the initial

$z_0 \in (0,1)$ and $z_0 \notin \{0, 0.25, 0.5, 0.75, 1.0\}$. In our experiment, we set μ=4 and the initial value z_0=0.152.

(2) PWLCM map: PWLCM map with ergodic property is given by the following equation:

$$z_{k+1} = \begin{cases} z_k/p, & z_k \in (0,p) \\ (1-z_k)(1-p), & z_k \in [p,1) \end{cases} \tag{3.2}$$

p is set to 0.7 and z_0=0.002.

(3) Singer map: this chaotic function is formulated as:

$$z_{k+1} = \mu(7.86z_k - 23.31z_k^2 + 28.75z_k^3 - 13.302875z_k^4) \tag{3.3}$$

Singer map exhibits chaotic behaviors when the parameter μ is set between 0.9 and 1.08. We set μ=1.073 and z_0=0.152.

(4) Sine map: the sine map is a unimodal map and can be formulated as:

$$z_{k+1} = \frac{a}{4} sin(\pi z_k) \tag{3.4}$$

where $a \in (0,4]$, and $z \in (0,1)$. We set $a = 4$ and $z_0 = 0.152$ in this experiment.

(5) Gaussian map: the following equation define Gaussian map:

$$z_{k+1} = \begin{cases} 0, & z_k = 0 \\ (\mu/z_k) mod(1), & z_k \neq 0 \end{cases} \tag{3.5}$$

where $\mu = 1$ and $z_0 = 0.152$.

(6) Tent map: tent map, which is similar to logistic map, displays some specific chaotic effects. It can be defined by the following equation:

$$z_{k+1} = \begin{cases} z_k/\beta, & 0 < z_k \leq \beta \\ (1-z_k)/(1-\beta), & \beta < z_k \leq 1 \end{cases} \tag{3.6}$$

we set $\beta = 0.4$ and $z_0 = 0.152$.

(7) Bernoulli map: this chaotic function is formulated as below:

$$z_{k+1} = \begin{cases} z_k/(1-\lambda), & 0 < z_k \leq 1-\lambda \\ (z_k - 1 + \lambda)/\lambda, & 1-\lambda < z_k < 1 \end{cases} \tag{3.7}$$

we set $\lambda = 0.4$ and $z_0 = 0.152$.

(8) Chebyshev map: the family of Chebyshev map is defined as follows:

$$z_{k+1} = \cos(\phi \cos^{-1} z_k) \tag{3.8}$$

where the parameter ϕ is set to be 5 and the initial value z_0=0.152.

(9) Circle map: the circle map is represented by the following equation:

$$z_{k+1} = z_k + a - \frac{b}{2\pi}\sin(2\pi z_k) mod(1) \tag{3.9}$$

When $a = 0.5$ and $b = 2.2$, it can generate chaotic sequence in (0,1). We also set $z_0 = 0.152$ in the experiment.

(10) Cubic map: it is one of the most commonly used maps in generating chaotic sequence in various applications. It can be formally defined by:

$$z_{k+1} = \rho z_k (1 - z_k^2) \tag{3.10}$$

we set $\rho = 2.59$ and $z_0 = 0.242$.

(11) Sinusoidal map: this iterator is given below:

$$z_{k+1} = a z_k^2 \sin(\pi z_k) \tag{3.11}$$

where $a = 2.3$ and we set initial value $z_0 = 0.74$.

(12) ICMIC map: this map has infinite fixed points, and can be defined using:

$$z_{k+1} = \sin(a/z_k) \tag{3.12}$$

where $a \in (0, \infty)$ is an adjustable parameter, and we set $a = 70$ in our experiment. It should be pointed out that ICMIC generates sequence in $(-1, 0) \cup (0, 1)$ and if the value locates in negative interval, we take its absolute value. Fig. 3.1 exhibits the histogram distribution graph of 12 chaotic maps with 10^5 iterations. From these figures, it is obvious that different chaotic maps give different distributions. For example, Fig. 3.1 (c), the singer map has much higher possibility to generate values in interval $[7.7, 7.9]$. While the sine map favors both ends between 0 to 1. It should be emphasized that all these chaotic maps locate in the interval $(0, 1)$ except the sinusoidal map, it can only value from 0.48 to 0.92. This distinctive feature makes sinusoidal map provide a different search range from others in chaotic local search.

According to these distinct dynamic properties, we adopt a simple random selection strategy to perform the chaotic local search. It means all chaotic maps will have equal probability to be selected into chaotic local search during the whole optimization procedure. By doing so, the algorithm complexity will decrease while the diversity of chaotic local search dynamic will increase.

3.3 Chaotic Local Search and CBSO

Local search is a search strategy that aims to improve solution quality of individuals. Most applications execute local search in a narrow range in which a better solution can be promised. CLS method uses chaotic sequences that locate in the interval $(0, 1)$ as a variable to adjust the search range. Based on this, the equation of CLS is denoted in Eq. (3.13).

$$X^j_{new_c} = X^j + (z(c, k) - a) \cdot steplength \tag{3.13}$$

where

- X^j is the selected individual in Eq. (2.1),

- $X^j_{new_c}$ is a new individual generated by CLS,

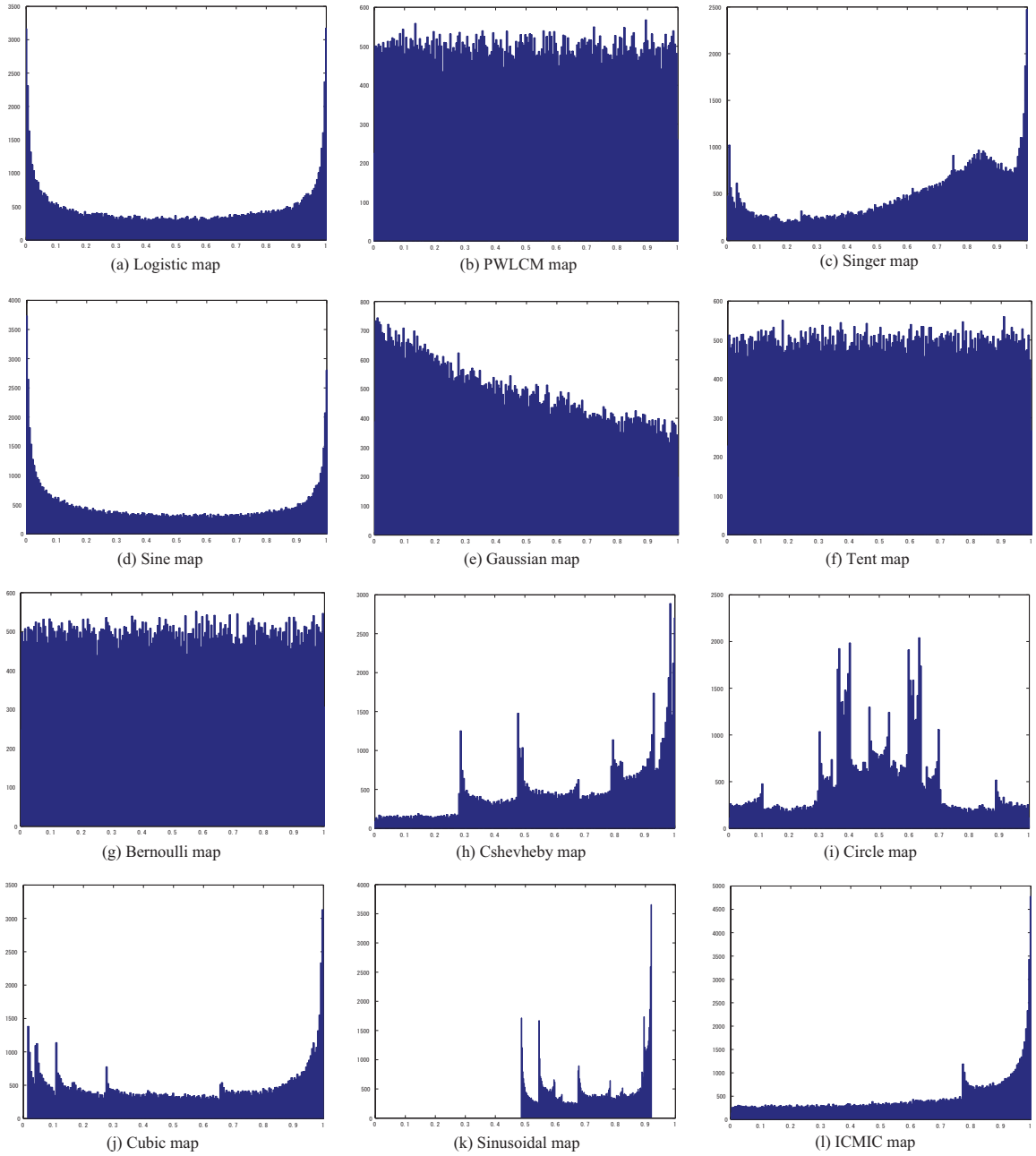

Figure 3.1: Histogram Distribution Graph of 12 Chaotic Maps with 10^5 Iterations.

- $z(c, k)$ is the chaotic variable randomly selected from chaotic sequences. $c \in [1, 12]$ denotes the chaotic map we used among all twelve adopted maps, and k is iteration number,

- $a = 0.3$ is an adjusted parameter,

- *steplength* is a vector whose length is the distance between X^j and the center of mth cluster which the individual to be updated belongs to. The direction is from the center pointing to X^j as defined in Eq. (3.14).

$$steplength = X^j - center(m) \tag{3.14}$$

Fig. 4.1 is a diagram regarding the Eq. (3.14). It is worth noting that the value of a is particularly set to 0.3 which makes $(z(c, k) - a) \in (-0.3, 0.7)$. After the calculation of Eq. (3.13), X^j may have a higher possibility to run away from the center and we call this a diffusion, while a lower possibility to approach the center which is called contraction. In other related researches such as [21], a often equals to 0.5 and $(z(c, k) - a)$ belongs to a balanced range in $(-0.5, 0.5)$. It is a common set and the algorithm will search in an equilibrium area by doing this. However in CBSO, we aim to break the stagnation and improve population diversity, thus an unbalanced search area is expected to accomplish our assumption.

By combining BSO with CLS, chaotic BSO (CBSO) is proposed. It should be noted that CLS is implemented only when BSO optimization procedure comes into a stagnation. The criteria is the best fitness of population stays the same for 50 iterations and then Eq. (3.13) will replace Eq. (2.1) to generate new individuals. Stagnation means the individuals are too concentrated around the centers, resulting in poor population diversity and make individuals lose search motivations. Therefore,

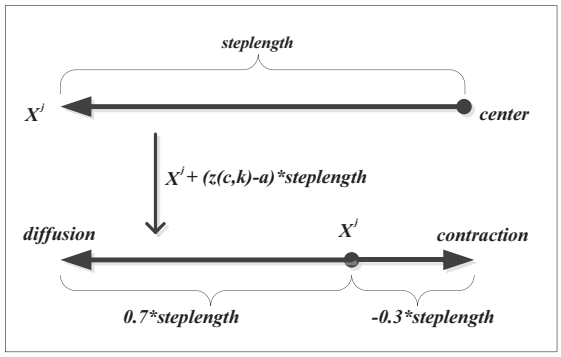

Figure 3.2: Diagram of Eq. (3.13) in a 2-D space.

it is necessary to let individuals spread out and break current position's balance. Algorithm 4 exhibits the execution of CBSO.

3.4 Experiments and Statistical Test

CEC'05 [39] is introduced to compare the performance between BSO and CBSO on optimization problems. There are 25 shifted or rotated benchmark problems which include unimodal functions and multimodal functions with plentiful local minima. These functions can test out the search capability of each algorithms objectively. We conduct two experiments in this study. The first experiment exhibits the comparison between BSO and CBSO. The second one compares CBSO with the differential evolution (DE) [40], multiple chaos embedded gravitational search algorithm (CGSA-P) [22] and whale optimization algorithm (WOA) [41]. Two non-parametric statistical test methods, i.e., Wilcoxon test and Friedman test [42], are implemented to detect the existence of significant difference among algorithms in two experiments respectively. In these experiment contrasts, population number N is 100 and dimension of problems D is 30. Each problem will be run for 30 times and maximum number of function evaluation (FES) is set to $D * 5000$. All the experiments are implemented on a PC with 3.10GHz Intel(R) Core(TM) i5-4440 CPU and 8GB of RAM using MATLAB R2013b.

Algorithm 2: Pseudo code of CBSO.

```
Randomly generate a population with N individuals;
Calculate the fitness of each individual;
while termination not reached do
    Cluster N individuals into M clusters using k − means method;
    Choose the best individual in each cluster as the center;
    if random(0, 1) < p = 0.2 then
        replace one cluster center by a randomly generated one
    end
    if random(0, 1) < p = 0.8 then
        select one cluster with a probability;
        if random(0, 1) < 0.4 then
            choose the cluster center as X^j
        else
            choose a random selected individual in the cluster as X^j
        end
    else
        randomly select two clusters;
        if random(0, 1) < 0.5 then
            choose the combination of two centers as X^j
        else
            choose the combination of two random selected individuals in two
              clusters as X^j
        end
    end
    if the criterion (the best fitness of population stays the same for 50
      iterations) satisfied then
        Generate new individual by adding random value to the selected X^j
          by using Eq. (3.13) and calculate the new individual
    else
        Generate new individual by adding random value to the selected X^j
          by using Eq. (2.1) and calculate the new individual
    end
    if new individual is better than old one then
        replace the old individual
    end
end
```

3.4.1 Performance between BSO and CBSO

First, we compare the performance between BSO and CBSO on CEC'05 benchmark functions. From Table 3.1, an intuitive assessment can be drawn that CBSO gets better solutions than BSO on most problems in terms of mean values. On a few problems, BSO gets a draw with CBSO while only on F10 it is worse than CBSO. These experimental results show the stability of the search ability of CBSO. A non-parametric statistical test named Wilcoxon rank-sum test for multiproblem is conduced to identify the differences between BSO and CBSO and testify this stability for precision and clarity. The procedure of Wilcoxon rank-sum test for detecting significant differences between mean values of two samples can be shown as follows:

1. Calculate the differences D_i between two algorithms on each problem.

2. These differences D_i will be ranked by absolute values.

3. R^+ is the sum rank for the problems in which the CBSO performs better than BSO and R^- is for the opposite. If $D_i = 0$, the rank of it will be divided evenly among the sums.
   $R^+ = \sum_{D_i>0} rank(D_i) + 0.5 \sum_{D_i=0} rank(D_i)$
   $R^- = \sum_{D_i<0} rank(D_i) + 0.5 \sum_{D_i=0} rank(D_i)$

4. $T = min(R^+, R^-)$, according the value of T to calculate p-value and judge to reject the null hypothesis of quality of mean values.

In Table 3.2, R^+ of CBSO versus BSO is 258.0 and R^- is only 42.0, and p-value is calculated according to $T = 42.0$. It is obvious that the null hypothesis can be rejected under whether significance level $\alpha = 0.05$ or $\alpha = 0.1$, indicating that CBSO outperforms BSO statistically. This result adequately proved the efficiency of CBSO and realize the aim of using CLS to improve solution quality .

Table 3.1: Experimental results of CEC'05 benchmark functions (F1-F25) using BSO and CBSO.

Fun.	Algorithm	Mean	Std	Best	Worst
F1	BSO	-4.50E+02	3.50E-14	-4.50E+02	-4.50E+02
	CBSO	-4.50E+02	3.17E-14	-4.50E+02	-4.50E+02
F2	BSO	-4.48E+02	9.36E-01	-4.49E+02	-4.46E+02
	CBSO	-4.48E+02	9.91E-01	**-4.50E+02**	-4.46E+02
F3	BSO	2.04E+06	7.23E+05	9.67E+05	4.99E+06
	CBSO	**1.78E+06**	6.82E+05	**5.53E+05**	3.22E+06
F4	BSO	2.78E+04	8.05E+03	1.32E+04	4.71E+04
	CBSO	**2.05E+04**	6.24E+03	**8.35E+03**	3.03E+04
F5	BSO	4.70E+03	1.22E+03	2.95E+03	7.98E+03
	CBSO	**4.15E+03**	7.56E+02	**2.91E+03**	6.07E+03
F6	BSO	1.26E+03	9.48E+02	5.12E+02	4.29E+03
	CBSO	**9.48E+02**	3.73E+02	**4.19E+02**	1.56E+03
F7	BSO	6.25E+03	3.25E+02	5.59E+03	6.92E+03
	CBSO	**6.05E+03**	3.28E+02	**5.10E+03**	6.65E+03
F8	BSO	-1.20E+02	9.90E-02	-1.20E+02	-1.19E+02
	CBSO	-1.20E+02	6.22E-02	-1.20E+02	-1.20E+02
F9	BSO	-2.86E+02	1.27E+01	**-3.09E+02**	-2.64E+02
	CBSO	-2.86E+02	9.58E+00	-3.03E+02	-2.68E+02
F10	BSO	**-2.93E+02**	8.79E+00	**-3.07E+02**	-2.72E+02
	CBSO	-2.91E+02	1.05E+01	-3.06E+02	-2.58E+02
F11	BSO	1.10E+02	2.51E+00	1.04E+02	1.13E+02
	CBSO	**1.09E+02**	2.70E+00	1.04E+02	1.14E+02
F12	BSO	2.84E+04	1.99E+04	3.05E+03	1.04E+05
	CBSO	**2.43E+04**	1.68E+04	**2.51E+03**	6.65E+04
F13	BSO	-1.26E+02	1.05E+00	-1.28E+02	-1.24E+02
	CBSO	-1.26E+02	9.25E-01	-1.28E+02	-1.24E+02
F14	BSO	-2.87E+02	3.78E-01	-2.88E+02	-2.86E+02
	CBSO	-2.87E+02	3.61E-01	-2.88E+02	-2.86E+02
F15	BSO	5.43E+02	7.94E+01	**3.38E+02**	6.24E+02
	CBSO	**5.15E+02**	6.63E+01	3.67E+02	6.22E+02
F16	BSO	2.87E+02	1.34E+02	1.69E+02	6.20E+02
	CBSO	**2.63E+02**	1.42E+02	**1.60E+02**	6.20E+02
F17	BSO	3.10E+02	1.57E+02	1.72E+02	6.75E+02
	CBSO	**2.87E+02**	1.30E+02	**1.69E+02**	5.66E+02
F18	BSO	9.17E+02	1.36E+00	9.14E+02	9.19E+02
	CBSO	**9.16E+02**	1.20E+00	9.14E+02	9.19E+02
F19	BSO	9.16E+02	1.07E+00	9.14E+02	9.19E+02
	CBSO	9.16E+02	1.28E+00	9.14E+02	9.19E+02
F20	BSO	9.16E+02	1.36E+00	9.14E+02	9.19E+02
	CBSO	9.16E+02	1.17E+00	9.14E+02	9.18E+02
F21	BSO	9.27E+02	1.37E+02	8.60E+02	1.26E+03
	CBSO	**8.87E+02**	1.01E+02	8.60E+02	1.26E+03
F22	BSO	1.21E+03	1.99E+01	**1.17E+03**	1.25E+03
	CBSO	1.22E+03	1.76E+01	1.18E+03	1.25E+03
F23	BSO	9.48E+02	1.38E+02	8.94E+02	1.30E+03
	CBSO	**8.95E+02**	1.40E+00	8.94E+02	9.00E+02
F24	BSO	4.67E+02	6.23E+00	4.60E+02	4.75E+02
	CBSO	**4.60E+02**	1.67E-12	4.60E+02	4.60E+02
F25	BSO	1.88E+03	4.44E+00	1.87E+03	1.89E+03
	CBSO	1.88E+03	3.39E+00	1.87E+03	1.89E+03

Table 3.2: Results obtained by the Wilcoxon test for algorithm CBSO.

	R^+	R^-	p-value	$\alpha = 0.05$	$\alpha = 0.1$
CBSO vs. BSO	258.0	42.0	0.001864	YES	YES

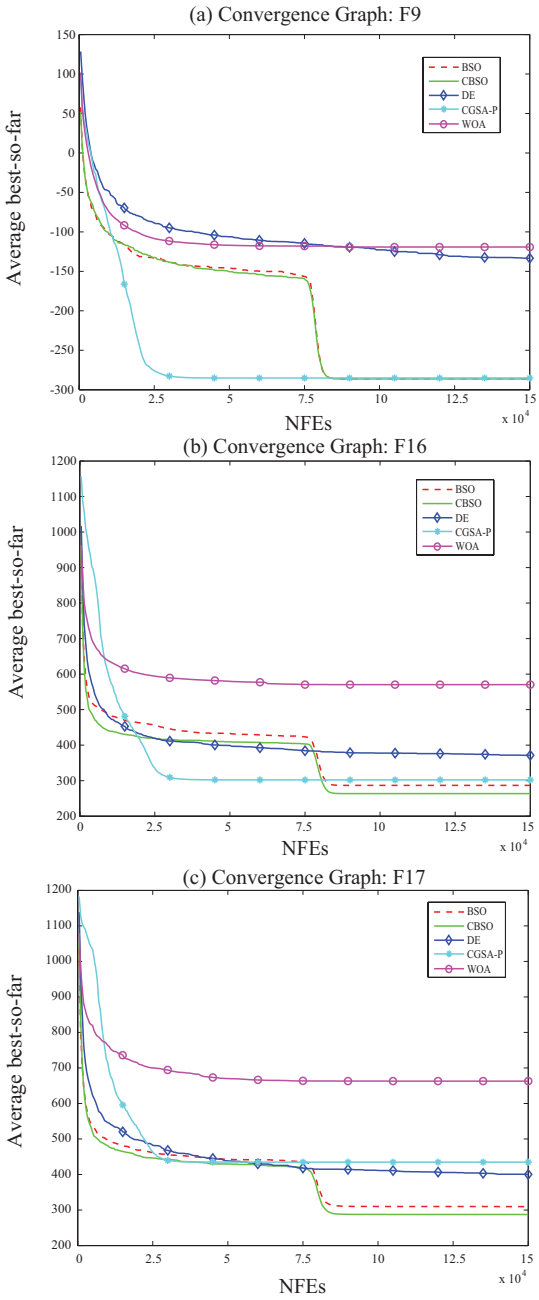

Figure 3.3: Convergence Graph of F9, F16 & F17.

3.4.2 CBSO Compared with Other Algorithms

In second experiment, CBSO is compared with some effective optimization algorithms proposed in the last few years to further discuss the competitiveness of CBSO. DE is a

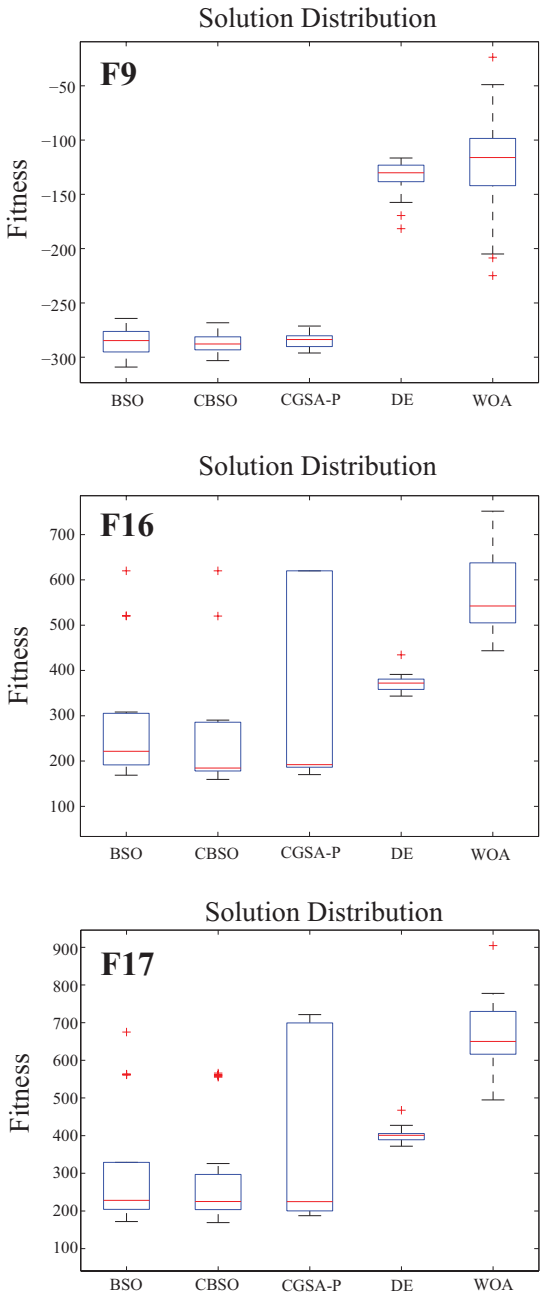

Figure 3.4: Box and Whisker Diagram of F9, F16 & F17.

very effective and powerful evolution-based optimization algorithm which has shown its splendid search ability in many scientific and engineering applications [40]. CGSA-P is a newly proposed algorithm which combines gravitational search algorithm [22] with chaotic local search. Three strategies of implementing chaos into local search

have been proposed and the parallelly embedding strategy is the most effective one. WOA proposed in 2016 mimics the behavior of humpback whales, and it is also an effective optimization algorithm [41]. In DE, we use the most efficient parameter set $F = 0.9$, $CR = 0.9$ suggested in [43]. In these experiment contrasts, population number N is 100 and dimension of problems D is 30. Each problem will be run for 30 times and maximum number of function evaluation (FES) is set to $D * 5000$.

Table 3.3 shows the experiment results and another non-parametric statistical test; Friedman test, is employed to detect the significance difference over a multiple comparison. In Friedman test, the null hypothesis assumes that mean values are the same among these algorithms and the alternative hypothesis negates the null hypothesis. Friedman test ranks the algorithms in each problem and then calculates the average rank that each algorithm obtained to estimate its performance. The lower the rank is, the better the algorithm performs. Table 3.4 shows the rank of each algorithm and we can see that CBSO gets the lowest value 1.4, which is the best rank in this multiple comparison. We can also observe the adjusted p-value obtained through the application of post hoc procedures in Table 3.4.

It should be emphasized that the multiple comparison test could cause probability error which may lead to a Type I error [44]. For example, a multiple comparison including n algorithms and level of significance α is 0.05. Then each single comparison will have a probability of $(1 - \alpha)$ not to make Type I error, and for the whole comparison, the probability is $(1 - \alpha)^{(n-1)}$. On the other hand, when $n = 10$, the reliability for a multiple comparison is only about 60% and it is apparent unacceptable [45]. Post hoc procedures are used to adjust p-values for avoiding Type I error. The Bonferroni-Dunn procedure, the Holm procedure and the Hochberg procedure are commonly used methods for adjusting p-values and the description of these procedures can be found in [45]. It also should be noticed that post hoc methods are conservative and adjusted values are usually higher than real ones. In Table 3.4, regardless the unadjusted p-value or the adjusted p-value proves the significant difference of CBSO comparing with DE, CGSA-P and WOA at the significance level of 0.05. Combining the comparison with BSO in Table 3.2, we can draw a conclusion

that CBSO performs better than all these algorithms.

Table 3.3: Experimental results of CEC'05 (F1-F25) using CBSO, DE, CGSA-P and WOA.

Algorithm	F1	F2	F3	F4	F5
CBSO	-4.50E+02 ± 3.17E-14	-4.48E+02 ± 9.91E-01	1.78E+06 ± 6.82E+05	2.05E+04 ± 6.24E+03	4.15E+03 ± 7.56E+02
DE	-4.42E+02 ± 3.37E+00	4.34E+03 ± 1.25E+03	3.98E+07 ± 9.47E+06	8.96E+03 ± 2.16E+03	3.25E+03 ± 5.88E+02
CGSA-P	-4.50E+02 ± 1.66E-01	1.64E+04 ±9.92E+02	1.62E+07 ±1.16E+07	5.34E+04 ±1.07E+04	1.87E+04 ±8.68E+02
WOA	-4.44E+02±9.86E+00	6.01E+04 ± 9.74E+03	3.67E+07 ± 1.30E+07	1.43E+05 ± 4.58E+04	1.78E+04 ± 3.54E+03
	F6	F7	F8	F9	F10
CBSO	9.48E+02 ± 3.73E+02	6.05E+03 ± 3.28E+02	-1.20E+02 ± 6.22E-02	-2.86E+02 ± 9.58E+00	-2.91E+02 ± 1.05E+01
DE	2.13E+04 ± 1.06E+04	4.52E+03 ± 6.63E-02	-1.19E+02 ± 7.23E-02	-1.33E+02 ± 1.50E+01	-1.05E+02 ± 1.29E+01
CGSA-P	2.23E+06 ±1.88E+06	6.31E+03 ±3.08E+02	-1.20E+02 ±6.73E-02	-2.85E+02 ±6.20E+00	-2.99E+02 ±4.69E+00
WOA	6.24E+04 ± 1.07E+05	4.57E+03 ± 5.41E+01	-1.19E+02 ± 9.80E-02	-1.19E+02 ± 4.58E+01	8.27E+01 ± 8.79E+01
	F11	F12	F13	F14	F15
CBSO	1.09E+02 ± 2.70E+00	2.43E+04 ± 1.68E+04	-1.26E+02 ± 9.25E-01	-2.87E+02 ± 3.61E-01	5.15E+02 ± 6.63E+01
DE	1.30E+02 ± 1.04E+00	3.52E+05 ± 7.71E+04	-1.09E+02 ± 1.19E+00	-2.86E+02 ± 1.66E-01	5.37E+02 ± 3.62E+01
CGSA-P	9.01E+01 ±3.14E-01	1.59E+04 ±1.24E+04	-1.21E+02 ±1.08E+00	-2.86E+02 ±1.73E-01	4.20E+02 ±4.37E-05
WOA	1.26E+02 ±2.13E+00	1.48E+05±9.46E+04	-1.09E+02 ± 5.17E+00	-2.87E+02±2.52E-01	7.81E+02 ± 2.07E+02
	F16	F17	F18	F19	F20
CBSO	2.63E+02 ± 1.42E+02	2.87E+02 ± 1.30E+02	9.16E+02 ± 1.20E+00	9.16E+02 ± 1.28E+00	9.16E+02 ± 1.17E+00
DE	3.72E+02 ± 1.76E+01	4.00E+02 ± 1.78E+01	9.17E+02 ± 2.49E-01	9.17E+02 ± 2.77E-01	9.17E+02 ± 2.43E-01
CGSA-P	3.02E+02 ±1.95E+02	4.35E+02 ±2.54E+02	9.26E+02 ±7.18E+01	9.48E+02 ±5.64E+01	9.55E+02 ±5.91E+01
WOA	5.70E+02 ± 8.48E+01	6.63E+02± 8.51E+01	1.05E+03 ± 9.62E+01	1.05E+03±9.21E+01	1.04E+03 ± 7.12E+01
	F21	F22	F23	F24	F25
CBSO	8.87E+02 ± 1.01E+02	1.22E+03 ± 1.76E+01	8.95E+02 ± 1.40E+00	4.60E+02 ± 1.67E-12	1.88E+03 ± 3.39E+00
DE	8.61E+02 ± 4.68E-01	1.28E+03 ± 1.25E+01	8.95E+02 ± 2.15E+00	4.63E+02 ± 7.84E-01	1.91E+03 ± 4.74E+00
CGSA-P	1.03E+03 ±2.91E+02	1.26E+03 ±1.15E+01	9.58E+02 ±1.78E+02	4.60E+02 ±1.27E-12	1.92E+03 ±7.87E+00
WOA	1.58E+03 ± 1.46E+02	1.57E+03 ± 8.93E+01	1.61E+03 ± 1.21E+02	1.58E+03 ± 1.24E+02	1.89E+03 ± 5.09E+00

Table 3.4: Adjusted p-values (FRIEDMAN).

Algorithm	Ranking	unadjusted p	p_{Bonf}	p_{Holm}	$p_{Hochberg}$	$\alpha = 0.05$
CBSO vs.	1.4					
WOA	3.48	0	0	0	0	YES
DE	2.58	0.001231	0.003693	0.002462	0.001796	YES
CGSA-P	2.54	0.001796	0.005388	0.002462	0.001796	YES

Fig. 3.3 and Fig. 3.4 can directly illuminate the experimental results obtained so far and give a distinct impression of the performance of CBSO. Fig. 3.3 (a) shows the convergence trend of each algorithm along with the number of function evaluation in F9. The vertical axis denotes the fitness of final best-so-far solutions. According to it, DE and WOA have slow convergence speed and poor solution quality from the beginning to the end. CGSA-P converges faster while CBSO finally gets the best solution. In Fig. 3.3 (c), CGSA-P loses its convergence speed and in the last is even worse than DE. CBSO has better solutions than BSO in the whole optimization procedure.

The CLS ensures CBSO can change its search mechanism once it falls into a stagnation. A stagnation means the individual generation method of conventional BSO couldn't find a better solution. In consideration of conventional mechanism preferring to explore the space near the cluster centers, our method intentionally generates new individual in the space away from the cluster centers in which better solutions are highly promised. The convergent performance comparison between BSO and CBSO clearly exhibits the effect of CLS which makes CBSO have a faster convergence speed than BSO.

Fig. 3.4 is the box and whisker diagram which can directly shows the properties of the algorithms. CBSO has the lowest medians and maximums in F9, F16 and F17. It also has a shorter interquartile range (IQR) which is the distance between the first quartile and the third quartile. A short IQR means a stable performance of search ability when repeating 30 times on the same benchmark problem. In particular, CGSA-P has acceptable solutions with shortest IQR and similar median to CBSO for F9. But CGSA-P performs well for only one problem and gets very unstable solutions for F16 and F17. The lower medians and shorter IQRs in all these functions suggest that CBSO has a stronger and more stable search ability in comparison with others. All these results verify the extraordinary performance of CBSO and the success of implementing CLS into BSO.

Although our method has some advantages such as the simplicity, better solution quality and can maintain the population diversity at a high level compared with the traditional BSO, it still needs to improve its convergence speed especially contrasting with CGSA-P in early convergent period. It will be a research emphasis for us to further enhance the search ability of CBSO to make it can obtain stable and fast convergence speed through the whole convergent period.

3.4.3 Comparison on Real-World Problems

Our proposed CBSO has great performance on benchmark problems, however, the motivation of improving algorithm is to make these algorithms can efficiently solve

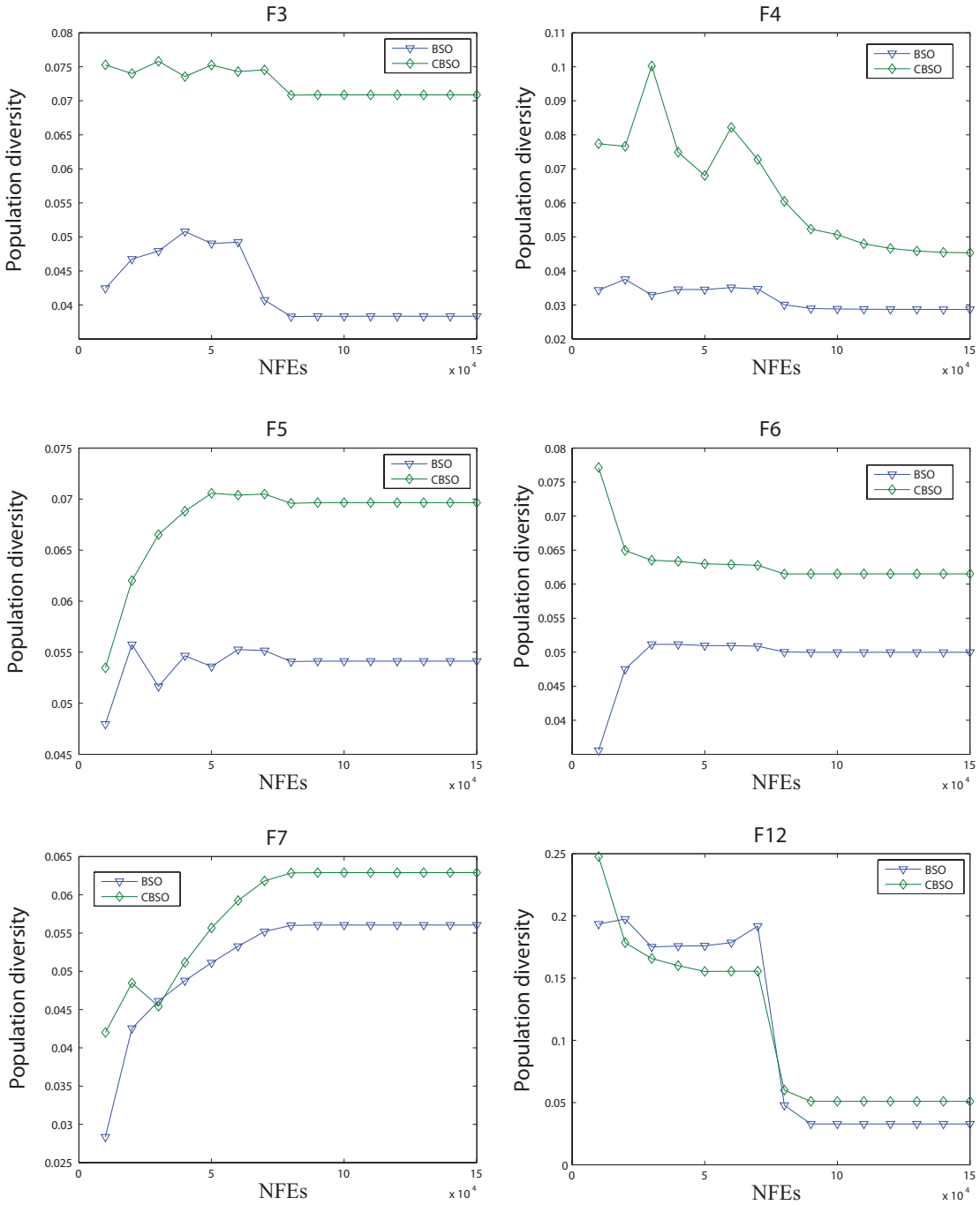

Figure 3.5: Population diversity of BSO and CBSO over $D * 5000$ FES on six benchmark functions.

real-world problems. It is far from enough for us that CBSO obtains good results only on benchmark problems. For further testing the performance of CBSO on real-world problems, the CEC2011 real world optimization problems [46] have been implemented.

Four representative problems are selected and the results of CBSO comparing with BSO, CGSA-P, DE and WOA are listed in Table 4.16 and the best results are highlighted. Each problem has been tested over 30 independent runs.

- RF1: Parameter Estimation for Frequency-Modulated (FM) Sound Waves
- RF2: Lennard-Jones Potential Problem
- RF4: Optimal Control of a Non-Linear Stirred Tank Reactor
- RF7: Transmission Network Expansion Planning (TNEP) problem

It is obvious that CBSO obtains the best results on all real-world problems. These results indicate the practical value of CBSO and realize our motivation that designing effective algorithm to solve real-world problems. In our future researches, more real-world problems will be applied to improve the practical ability of our proposed algorithms.

Table 3.5: Experimental results on real-world problems

	BSO	CBSO	CGSA-P	DE	WOA
RF1	1.40E+01±5.88E+00	**1.27E+01± 4.58E+00**	2.44E+01±1.51E+00	3.15E+01±2.01E+01	2.19E+01±5.00E+00
RF2	-2.04E+01± 2.32E+00	**-2.07E+01±2.62E+00**	-1.56E+01±3.65E+00	-4.24E+00±4.02E-01	-1.84E+01±4.89E+00
RF4	1.65E+01±2.32E+00	**1.46E+01 ±1.81E-01**	1.59E+01±2.02E+00	4.99E+01± 2.46E+01	1.47E+01 ±1.26E+00
RF7	8.81E-01±1.03E-01	**8.69E-01±9.51E-02**	8.80E-01±1.37E-01	3.31E+00±4.06E-01	1.92E+00±2.30E-01

3.5 Discussion

3.5.1 Population Diversity

Population diversity is an important reference standard in modifying optimization algorithms. A poor diversity means the population is too dense and it may cause a premature stagnation [47]. The aim of implementing CLS is to prevent poor diversity. We calculate the diversity to verify the effect of CLS. The definition can be found as follows:

$$d(X) = \frac{1}{N}\sum_{i=1}^{N} ||X_i - \bar{X}||/max_{1\leq i,j\leq N}||X_i - X_j||, \tag{3.15}$$

where $\bar{X}$ is defined as

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i. \tag{3.16}$$

All functions have been measured and the variation trend of population diversity of six typical ones can be observed in Fig. 3.5. We can see that CBSO can obtain a higher diversity level than BSO, not only in early search stage but also to the end of the whole optimization process. The implementation of CLS provides each individual a motivation of diffusion to spread out and search in a wider space. This operation provides more chances for individuals to keep distance between each other and explore better solutions through the whole search phase, which ensures a high population diversity level. In exploitation phase, CLS also has a probability of near 30% to provide an inward contraction to improve solution quality. By this mechanism, CLS enables BSO to keep a good population diversity well, thus improves its search ability and avoids a premature stagnation.

3.5.2 Computational Time Complexity

As the statistical results have shown the efficiency of CBSO, we should concern that whether CBSO increases the time complexity of the algorithm. In this section, we compare the time complexity of BSO and CBSO. The number of function evaluation is set to N*5000 and the time complexity in each procedure of BSO is described as follows:

(1) The time complexity of initialization is $O(N)$.

(2) Time complexity of evaluating the fitness of the population is $O(N)$.

(3) The K-means divides the population into k clusters, so the time complexity of this execution is $O(kN^2)$.

(4) Select one cluster center to be replaced by a randomly generated center. This step needs $O(N^2)$.

(5) The individual selection procedure costs $O(N^2)$.

(6) Time complexity of the generation of steplength is $O(N^2)$.

(7) New individual generation and fitness calculation costs $O(N^2)$ respectively.

The overall time complexity of BSO is

$$O(N)+O(N)+O(kN^2)+O(N^2)+2O(N^2)+2O(N^2) = 2O(N)+5O(N^2)+O(kN^2) \tag{3.17}$$

where N is the dimension. Thus the time complexity of BSO is $O(N^2)$.

The procedure of CBSO is shown as follows:

(1) The overall time complexity of initialization and the generation of chaotic sequences is $O(N) + O(1)$.

(2) Evaluating the fitness of the population needs $O(N)$.

(3) The time complexity of dividing the population into k clusters is $O(kN^2)$.

(4) Selecting one cluster center to be replaced by a randomly generated center needs $O(N^2)$.

(5) The individual selection procedure costs $O(N^2)$ and the generation of steplength also needs $O(N^2)$.

(6) New individual generation needs $O(N^2)$.

(7) If the criterion of implementing chaotic local search is satisfied, the calculation cost is $O(N^2)$. Otherwise the time complexity of generating new individual by adding random value needs $O(N^2)$.

The overall time complexity of CBSO is

$$\begin{aligned} &O(N) + O(1) + O(N) + O(kN^2) + O(N^2) + 2O(N^2) + 2O(N^2) \\ &\qquad = 2O(N) + 5O(N^2) + O(kN^2) + O(1) \end{aligned} \tag{3.18}$$

Therefore the time complexity of CBSO is $O(N^2)$. From this result, we can see that CBSO equals the BSO in time complexity which indicates the CBSO can perform better than BSO without needing more computational time.

Chapter 4

ASBSO: An Improved Brain Storm Optimization with Flexible Search Length and Memory-based Selection

4.1 Introduction

Nowadays, many swarm intelligence algorithms have been proposed to solve complex real-world problems [11, 48].Brain storm optimization algorithm (BSO) which is one of the swarm intelligence algorithm, is promising in solving complex problems [16]. It is inspired by the human brain storming behaviors. Each idea generated by the human brain represents an individual in search space. In a brain storming process, humans firstly generate some rough ideas, then exchange and discuss these ideas with each other. The inferior ideas are sifted out while the superior ones are left. This operation circles over and over, which makes ideas become more and more mature. In the meanwhile, new ideas are kept being generated and joined in the circle. With the process ends, a feasible and effective idea spurts out.

Since the announcement of BSO in 2011, it gets lots of attention from the researchers in swarm intelligence community due to its novelty and efficiency. It has been successfully applied in different scenarios, such as function optimization, engineering problems and financial prediction [49, 50, 51, 52, 53]. Moreover, some modifications for BSO have been made to enhance its performance from several per-

spectives. For example, a new multi-objective BSO (MBSO) is proposed in [54] for solving multi-objective optimization problems. The clustering strategy is applied in the objective search space to handle multi-objective optimization problems, while it is originally performed in the solution search space for solving single objective problems. With different characteristics of diverging operation, MBSO becomes a promising algorithm with an outstanding ability to solve multi-objective optimization problems. In [55], BSO in objective space (BSOOS) is proposed to cut down the computation time of the convergent operation. A clustering operation is replaced by taking p percentage individuals as elitists. An updating operation is modified to suit for an elitists mechanism in one-dimensional objective space instead of solution space. By doing so, BSOOS achieves a better convergent speed and solution quality in comparison with the traditional BSO.

Improving the population diversity is an alternative modification besides the usage of objective space. As the balance between convergence and divergence is very important to swarm intelligence optimization algorithms, a premature convergence leads to a low population diversity and bad solution quality, while the opposite brings very slow search speed. The issue of how to find the balance between convergence and divergence of solutions is still very challenging and it reflects the algorithm's exploration and exploitation ability. In [56, 57], chaotic sequences are used as variables to initialize population and generate new individuals. As a universal phenomenon of nonlinear dynamic systems, chaos has an unpredictable random behavior [21]. Thus, its randomicity and ergodicity can help BSO improve its population diversity and solution quality effectively. In [58], Cheng et al. propose a new BSO which uses different kinds of partial reinitialization strategies to increase its population diversity. Duan et al. [59] propose a novel predator-prey model to improve the population diversity of BSO for a DC brush-less motor. This model can enable the algorithm structure to explore the search space more evenly. By using the predator-prey strategy, the population can share better global information with each other to improve search efficiency in exploitation phase. In [60], quantum-behaved BSO (QBSO) which aims to improve population diversity and generate new individuals by using global infor-

mation is proposed. Moreover, QBSO for the first time combines BSO with quantum theories. It analyzes the quantum behavior and quantum state of each individual by depicting a wave function to solve the drawback of BSO that easily sticks into local optima on multimodal functions. In addition, Wang et al. [61] discover a power law distribution in BSO which opens a new way of thinking to boost the population interaction and improve population diversity via adjusting the population structure.

Although above mentioned modifications have improved the performance of BSO, they are limited and the performance of BSO is still fatigued and week [23]. Most efforts attempt to modify BSO for solving specific problems while these modifications are not suitable for other applications. It is still a great demand to enhance its search ability and robustness.

To achieve this goal, we propose an adaptive step length mechanism based on memory selection to combine with BSO (namely, ASBSO) which exhibits a notable performance. This method can modify BSO by providing strategies with various step lengths which are adaptively applied to generate new individuals. As it can supply a specific step length according to corresponding problems and convergent periods, it is more possible that ASBSO can avoid or jump out of the local optima. In other words, the search efficiency and robustness of BSO can be greatly improved.

Besides the adaptive step length mechanism, a modified selection method is also proposed based on memory. Different from the conventional storage mode in [22] which applies a success memory and a failure memory with 0 and 1 as the information stored in these memories, the modified method only employs the success memory and considers the difference between two compared fitness values instead of simple numbers (i.e., 0 and 1). This is a modification which directly demonstrates the improvement of each selected strategy and extrudes a strategy with a better performance. A detailed description is presented in Section 5.2.

The contributions of this paper can be summarized as: (1) An adaptive step length mechanism based on memory selection method is proposed to enhance the robustness of BSO evidently, therefore makes it more suitable for various applications. (2) It's for the first time that we use the difference between two compared fitness values instead

of simple numbers such as 0 and 1 to be stored in memory. This modification can increase the efficiency of the selection method, and thereby improve solution quality observably. An experimental comparison between the new storage mode and the old one brings an intuitional conclusion that the proposed method is significantly better. (3) Sufficient experimental data and statistical analyses of performance comparisons between traditional BSO and our proposed ASBSO at different dimensions show that ASBSO outperforms BSO entirely. The contrast between ASBSO and other well-known algorithms also indicates the superiority of ASBSO. (4) ASBSO is verified to be a competent and robust algorithm for different optimization problems.

The organization of this paper can be presented as follows. A brief introduction of BSO is given in Section 3.2. Section 5.2 introduces the proposed ASBSO in details. The experimental results are shown in Section 5.3. Some discussions are assigned in Section 5.4. We conclude this paper in Section 3.5.

4.2 ASBSO

4.2.1 Motivation

In the new individual generating operation of BSO introduced in Section 3.2, the search step length only varies with the current iteration number and lacks of flexibility, thus it makes a poor search efficiency and robustness. BSO only applies an invariable scale parameter $K = 20$ to render the search range to shrink during iterations, therefore the shrink is limited and inflexible. In ASBSO, an adaptive step length mechanism is motivated to alleviate this issue. Various optional scale parameters make BSO have adjustable search ranges instead of the traditional step length which only varies according to the current iteration number. As ASBSO applies multiple step lengths in the search process, the probability of getting into the gorge or jumping out of the valley in the search landscape can be increased a lot.

As we described in Section 3.2 that BSO lacks of a powerful search ability and robustness, it is a motivation for us to alleviate these drawbacks. An example is shown

below to make us further understand the utilization of search ability and robustness.

A popular approach to comprehensively observe the search ability and robustness of optimization algorithms in evolutionary community is to optimize benchmark functions. Some famous benchmark function suits such as 23 standard benchmark functions [62], CEC'05 [39], CEC'13 [63] and CEC'17 benchmark functions [64] have been widely used. These functions become more and more complicated and difficult in order to emulate the real world problems whose complexities increase in a geometric ratio. Therefore, the performances of optimization algorithms on benchmark functions have become an important standard to judge whether they can be implemented into practical applications or not. For instance, Fig. 4.1 illustrates the 3D and contour graphs of F8 and F11 in CEC'13 function suit. F8 is a rotated Ackley's function which has the same properties of multi-modal, non-separable and asymmetrical as F11 does. In addition, the local optima's number of F11 is very huge. The global optimum of F8 seems to be in a gorge surrounded by many steep precipices. The entrance of this gorge is so narrow and secluded that it could easily be missed by a search step length which is beyond the distance between $X(t)$ and $X'(t)$. Once the entrance has been missed, individual could only find a mass of similar local optimum. It will take a lot of computational time to obtain another chance for exploiting the gorge where global optima hides. On the contrary, in F11, a step length smaller than the distance between $X(t)$ and $X'(t)$ means that the individual couldn't jump out of the valley of local optima and is hard to know the global optima lays just beside it. These are two representative cases which could happen not only in benchmark functions but also in real world. Therefore, it has become an urgent task to alleviate and solve them via proposing more suitable optimization algorithms .

To address the above issues, two main modifications including multiple step lengths and new memory mechanism are proposed in ASBSO. They are interpreted in the following subsections in detail.

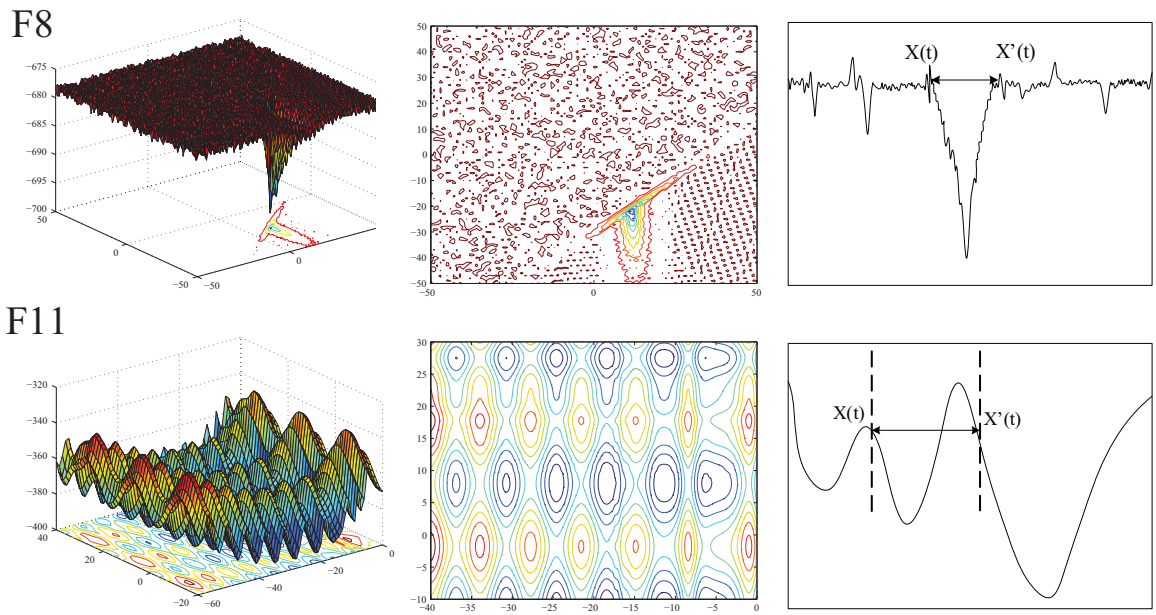

Figure 4.1: Diagrams of F8 and F11 in CEC'13.

4.2.2 Multiple Step Lengths

The parameter K in Eq. (2.2) is used to change the scale of $logsig()$. In the strategy of multiple step lengths, different K values listed in Table 4.1 is applied to provide different scales to adjust the search step length. The strategies which have relatively small K values indicate that they can provide a diffusion to search radius. It makes BSO be effective to explore the objective space and accelerate convergence. In the early search phase, optimization algorithm is required to have efficient exploration competence when facing the unknown search space. If we pay much attention to exploit local information before the whole space has been explored, the search cost will become very expensive and influence the solution quality [65]. Thus, it's necessary to provide large search step length to effectively detect the region with promising solutions. While in exploitation phase, a local search which applies short step length is needed urgently to excavate solutions with a high accuracy. Therefore, strategies with relatively large K values can improve solution quality in exploitation phase as large K values generally lead to a localized search.

As we discussed that changeless K value makes BSO only can shrink its search

Table 4.1: Illustration of the flexible multiple search length strategy.

	Strategy 1	Strategy 2	Strategy 3	...	Strategy M
K	k	$k+H$	$k+2H$	...	$k+(M-1)H$

range according to the current iteration number while couldn't flexibly adjust step length to fit various search periods and problems, assigning multiple values to K naturally equips BSO with flexible search ability to reply different situations.

4.2.3 New Memory Mechanism

To adaptively carry out multiple step lengths, we introduce an improved memory storing mechanism (IMS) which is originated from the success-failure-based memory structure (SFMS) [22, 66], In SFMS, a success memory shown in Table 4.2 and a failure memory shown in Table 4.3 is applied to store the number of succeeding or failing to generate better solutions, respectively. In the beginning, M strategies are randomly selected by roulette wheel selection method to generate new individuals. As Eqs. (4.1) and (4.2) shown, if the new individual X'_{t-1} outperforms and replaces the old individual X_{t-1}, it is indicated as a success and let $\alpha_{j,t}$ equal to 1, where j ($j = 1, 2, ..., M$) refers to the used strategy and t is the current iteration. If the opposite, it becomes a failure trial and $\beta_{j,t}$ equals to 1. If the iteration count is over the preset iteration length L (L =50 is empirically set according to [22]), the first row of Tables 4.2 and 4.3 will be removed to make space for the newest one. The selection of strategies is described as follows.

$$\alpha_{j,t} = \begin{cases} 1, & f(X'_{t-1}) < f(X_{t-1}) \\ 0, & otherwise \end{cases} \tag{4.1}$$

$$\beta_{j,t} = \begin{cases} 0, & f(X'_{t-1}) < f(X_{t-1}) \\ 1, & otherwise \end{cases} \tag{4.2}$$

The chosen probability of each strategy is calculated as shown in Eqs. (4.3) and (4.4) after the memories record the results:

Table 4.2: Traditional Success Memory

Index	Strategy 1	Strategy 2	Strategy 3	...	Strategy M
1	$\alpha_{1,t-L}$	$\alpha_{2,t-L}$	$\alpha_{3,t-L}$	...	$\alpha_{M,t-L}$
2	$\alpha_{1,t-L+1}$	$\alpha_{2,t-L+1}$	$\alpha_{3,t-L+1}$	...	$\alpha_{M,t-L+1}$
...	...	...	...	...	...
L	$\alpha_{1,t-1}$	$\alpha_{2,t-1}$	$\alpha_{3,t-1}$	...	$\alpha_{M,t-1}$

Table 4.3: Traditional Failure Memory

Index	Strategy 1	Strategy 2	Strategy 3	...	Strategy M
1	$\beta_{1,t-L}$	$\beta_{2,t-L}$	$\beta_{3,t-L}$	...	$\beta_{M,t-L}$
2	$\beta_{1,t-L+1}$	$\beta_{2,t-L+1}$	$\beta_{3,t-L+1}$	...	$\beta_{M,t-L+1}$
...	...	...	...	...	...
L	$\beta_{1,t-1}$	$\beta_{2,t-1}$	$\beta_{3,t-1}$	...	$\beta_{M,t-1}$

$$p_{j,t} = \frac{S_{j,t}}{\sum_{j=1}^{4} S_{j,t}} \tag{4.3}$$

$$S_{j,t} = \frac{\sum_{t-L}^{t-1} \alpha_{j,t}}{\sum_{t-L}^{t-1} \alpha_{j,t} + \sum_{t-L}^{t-1} \beta_{j,t}} + \delta \tag{4.4}$$

where $p_{j,t}$ denotes the probability to use the j-th strategy in current iteration t when $t > L$. $\sum_{t-L}^{t-1} \alpha_{j,t}$ calculates the total number of the j-th strategy successfully generating a new individual to replace X_{t-1}. $\sum_{t-L}^{t-1} \beta_{j,t}$ is the total number for the failure circumstances. Eq. (4.4) calculates the success rate and $\delta = 0.01$ is used for avoiding a null value. It is obvious that the strategy with higher success rate has a higher chance to be selected to generate new individuals.

However, the SFMS mechanism has one drawback that no matter how better a new individual obtained by a strategy, it only records 1 in the success memory. One case is given to interpret this drawback in detail. Let's define that D^1 represents the improvement in fitness (if $f(X'_{t-1}) < f(X_{t-1})$, $D = |f(X'_{t-1}) - f(X_{t-1})|$) obtained by Strategy 1, D^2 is that obtained by Strategy 2, and so on. Supposing $D^1 = 2D^2$, which means Strategy 1 is suitable for the current search period and can find a much better solution than Strategy 2 does in one generation. However, they score the same points (both 1) in success memory which leads to same possibilities to be selected.

Table 4.4: New Success Memory (IMS)

Index	Strategy 1	Strategy 2	Strategy 3	...	Strategy M
1	D^1_{t-L}	D^2_{t-L}	D^3_{t-L}	...	D^M_{t-L}
2	D^1_{t-L+1}	D^2_{t-L+1}	D^3_{t-L+1}	...	D^M_{t-L+1}
...	...	...	...	...	...
L	D^1_{t-1}	D^2_{t-1}	D^3_{t-1}	...	D^M_{t-1}

This mechanism evidently has relatively low efficiency which causes a slowness in convergence speed, and further decrease the solution quality. To alleviate this issue, in IMS, the improvement value in fitness D^j (j indicates the executed strategy) is recorded into a success memory to replace the numbers of 0 and 1. In the meanwhile, failure memory is not implemented in the new mechanism, since we focus on the quality not quantity that each strategy obtains. If failure memory is applied, a poor search attempt may decrease the quality of solutions and hinder the evolutionary direction of algorithm. Table 4.4 shows the structure of IMS. Each improvement value D^j_t in fitness obtained by strategy j is stored in it. The selection possibility of strategy j at iteration t can be calculated by Eq. (4.5).

$$p^{new}_{j,t} = \frac{D^j_t}{\sum_{j=1}^{M} D^j_t} \tag{4.5}$$

Algorithm 3 illustrates the main procedures of ASBSO. In each generation of new individuals, a strategy j is selected according to its selection possibility $p^{new}_{j,t}$ to produce a search step length. The new individual is generated by adding the step length to the selected X by using Eq. (2.1) and its fitness is calculated. If the new individual is better than the old one, then it will replace the old one. In the meanwhile, the selected strategy is marked as a success trial. The improvement in fitness D^j_t is stored in memory and the selection possibility for each strategy is updated.

Algorithm 3: Pseudo code of ASBSO.

```
Randomly generate a population with N individuals;
Calculate the fitness of each individual;
while termination not satisfied do
    Divide N individuals into C clusters by using k − means clustering
      method;
    Choose the best individual in each cluster as the center;
    if random(0,1) < p_c = 0.2 then
        replace one cluster center by a randomly generated individual
    end
    if random(0,1) < p_g = 0.8 then
        select one cluster;
        if random(0,1) < p_c1 = 0.4 then
            choose the cluster center as X
        else
            choose a randomly selected individual in the cluster as X
        end
    else
        randomly select two clusters;
        if random(0,1) < p_c2 = 0.5 then
            choose the combination of two centers as X
        else
            choose the combination of two randomly selected individuals in
              two clusters as X
        end
    end
    Choose a strategy to generate a search step length according to Eq. (4.5);
    Generate new individual by adding the step length to the selected X by
      using Eqs. (2.1) and (2.2);
    if new individual is better than old one then
        replace the old individual and update the memory
    end
end
```

Table 4.5: Friedman test result for $H = 10$, 20 and 30.

Algorithm	Ranking	unadjusted p	p_{Bonf}	p_{Holm}	$p_{Hochberg}$
$H = 20$ vs.	1.4298				
$H = 30$	2.2895	0.000004	0.000009	0.000009	0.000006
$H = 10$	2.2807	0.000006	0.000011	0.000009	0.000006

Table 4.6: Experimental results of CEC'13 benchmark functions (F1-F28) using BSO and ASBSO at $D = 10$ and $D = 30$.

	D=10			D=30	
	BSO	ASBSO		BSO	ASBSO
	Mean (Std Dev)	Mean (Std Dev)		Mean (Std Dev)	Mean (Std Dev)
F1	-1.40E+03 (0.00E+00)	-1.40E+03 (0.00E+00)	F1	-1.40E+03 (4.22E-14)	-1.40E+03 (1.98E-13)
F2	6.79E+04 (5.55E+04)	**3.85E+04 (3.29E+04)**	F2	1.54E+06 (4.79E+05)	1.54E+06 (4.26E+05)
F3	4.01E+07 (7.33E+07)	**2.92E+07 (4.94E+07)**	F3	1.11E+08 (1.74E+08)	**8.47E+07 (8.64E+07)**
F4	7.58E+03 (4.24E+03)	**6.00E+03 (3.58E+03)**	F4	2.17E+04 (5.65E+03)	**5.08E+03 (2.25E+03)**
F5	-1.00E+03 (1.35E-04)	-1.00E+03 (1.88E-04)	F5	-1.00E+03 (1.52E-03)	-1.00E+03 (2.98E-03)
F6	**-8.97E+02 (2.65E+00)**	-8.93E+02 (4.23E+00)	F6	**-8.66E+02 (2.47E+01)**	-8.64E+02 (2.76E+01)
F7	-7.05E+02 (3.36E+01)	**-7.24E+02 (3.17E+01)**	F7	-6.71E+02 (7.57E+01)	**-7.08E+02 (3.90E+01)**
F8	-6.80E+02(9.14E-02)	-6.80E+02 (9.48E-02)	F8	-6.79E+02 (7.60E-02)	-6.79E+02 (6.70E-02)
F9	-5.93E+02 (1.41E+00)	**-5.94E+02 (1.48E+00)**	F9	-5.68E+02 (2.90E+00)	**-5.71E+02 (2.58E+00)**
F10	-5.00E+02 (3.15E-02)	-5.00E+02 (4.81E-02)	F10	-5.00E+02 (1.93E-01)	-5.00E+02 (5.35E-02)
F11	-3.42E+02 (1.90E+01)	**-3.53E+02 (2.38E+01)**	F11	6.40E+01 (7.12E+01)	**-1.82E+02 (5.40E+01)**
F12	-2.46E+02(1.86E+01)	-2.46E+02 (2.17E+01)	F12	2.06E+02 (8.43E+01)	**-7.64E+01 (4.85E+01)**
F13	-1.30E+02 (2.14E+01)	**-1.31E+02 (2.09E+01)**	F13	3.55E+02 (8.77E+01)	**1.30E+02 (6.54E+01)**
F14	1.04E+03 (2.33E+02)	**8.73E+02 (2.96E+02)**	F14	3.88E+03 (5.17E+02)	**3.68E+03 (4.56E+02**
F15	1.17E+03 (2.79E+02)	**1.05E+03 (2.78E+02)**	F15	4.25E+03 (5.57E+02)	**3.88E+03 (5.74E+02)**
F16	2.00E+02 (2.02E-02)	2.00E+02 (7.00E-02)	F16	2.00E+02 (4.14E-02)	2.00E+02 (1.13E-01)
F17	3.55E+02 (1.86E+01)	**3.39E+02 (1.13E+01)**	F17	7.31E+02 (8.13E+01)	**5.28E+02 (5.13E+01)**
F18	4.49E+02 (2.27E+01)	**4.39E+02 (1.28E+01)**	F18	7.41E+02 (5.30E+01)	**5.98E+02 (2.85E+01)**
F19	5.02E+02 (5.95E-01)	**5.01E+02 (4.27E-01)**	F19	5.09E+02 (2.04E+00)	**5.04E+02 (7.60E-01)**
F20	6.04E+02 (6.46E-01)	**6.03E+02 (5.66E-01)**	F20	6.14E+02 (1.77E-01)	6.14E+02 (2.94E-01)
F21	1.10E+03 (4.63E-13)	1.10E+03 (2.16E-11)	F21	1.03E+03 (8.91E+01)	**1.02E+03 (8.42E+01)**
F22	2.24E+03 (3.35E+02)	**2.05E+03 (2.72E+02)**	F22	6.04E+03 (7.04E+02)	**5.36E+03 (4.70E+02)**
F23	**2.19E+03 (3.06E+02)**	2.28E+03 (3.23E+02)	F23	**5.98E+03 (7.32E+02)**	6.00E+03 (7.84E+02)
F24	1.22E+03 (1.16E+01)	1.22E+03 (1.37E+01)	F24	1.33E+03 (2.25E+01)	**1.31E+03 (2.60E+01)**
F25	1.32E+03 (4.24E+00)	1.32E+03 (1.99E+01)	F25	1.46E+03 (2.50E+01)	**1.41E+03 (1.03E+01)**
F26	1.39E+03 (3.26E+01)	1.39E+03 (2.64E+01)	F26	1.50E+03 (8.60E+01)	**1.46E+03 (7.91E+01)**
F27	1.81E+03 (1.12E+02)	**1.77E+03 (1.18E+02)**	F27	2.49E+03 (9.32E+01)	**2.42E+03 (1.07E+02)**
F28	2.26E+03 (7.51E+01)	**2.17E+03 (1.78E+02)**	F28	5.73E+03 (5.38E+02)	**2.03E+03 (8.02E+02)**

Table 4.7: Experimental results of CEC'13 benchmark functions (F1-F28) using BSO and ASBSO at $D = 50$ and $D = 100$.

	D=50			D=100	
	BSO	ASBSO		BSO	ASBSO
	Mean (Std Dev)	Mean (Std Dev)		Mean (Std Dev)	Mean (Std Dev)
F1	-1.40E+03 (1.25E-06)	-1.40E+03 (1.54E-02)	F1	-1.40E+03 (5.99E-02)	-1.40E+03 (8.62E-01)
F2	**2.25E+06 (7.00E+05)**	2.33E+06 (7.15E+05)	F2	**1.03E+07 (2.17E+06)**	1.25E+07 (2.24E+06)
F3	**2.09E+08 (1.52E+08)**	2.68E+08 (1.55E+08)	F3	**1.45E+09 (5.98E+08)**	2.73E+09 (1.39E+09)
F4	1.58E+04 (5.04E+03)	**7.93E+03 (2.63E+03)**	F4	6.38E+03 (1.79E+03)	**2.12E+03 (7.60E+02)**
F5	-1.00E+03 (4.09E-03)	-1.00E+03 (1.31E-02)	F5	-1.00E+03 (2.86E-02)	-1.00E+03 (5.38E-01)
F6	**-8.27E+02 (3.53E+01)**	-8.17E+02 (3.26E+01)	F6	**-7.05E+02 (4.96E+01)**	-6.73E+02 (5.28E+01)
F7	-6.37E+02 (6.43E+01)	**-6.82E+02 (3.23E+01)**	F7	-6.71E+02 (2.76E+01)	**-7.05E+02 (2.05E+01)**
F8	-6.79E+02 (5.55E-02)	-6.79E+02 (4.13E-02)	F8	-6.79E+02 (4.28E-02)	-6.79E+02 (4.98E-02)
F9	-5.43E+02 (3.70E+00)	**-5.49E+02 (4.38E+00)**	F9	-4.72E+02 (4.81E+00)	**-4.82E+02 (6.31E+00)**
F10	-4.99E+02 (1.71E-01)	-4.99E+02 (4.52E-01)	F10	**-4.95E+02 (6.45E-01)**	-4.94E+02 (2.06E+00)
F11	3.27E+02 (9.56E+01)	**2.22E+02 (6.95E+01)**	F11	1.54E+03 (1.94E+02)	**1.40E+03 (1.72E+02)**
F12	**4.63E+02 (1.17E+02)**	4.66E+02 (1.08E+02)	F12	1.89E+03 (2.56E+02)	**1.83E+03 (2.34E+02)**
F13	**6.84E+02 (1.04E+02)**	6.98E+02 (1.22E+02)	F13	2.29E+03 (2.18E+02)	**2.15E+03 (2.15E+02)**
F14	7.00E+03 (7.93E+02)	**6.70E+03 (7.94E+02)**	F14	1.52E+04 (1.12E+03)	**1.46E+04 (9.60E+02)**
F15	7.93E+03 (1.01E+03)	**7.41E+03 (5.66E+02)**	F15	1.53E+04 (1.27E+03)	**1.45E+04 (1.11E+03)**
F16	2.00E+02 (9.01E-02)	2.00E+02 (1.55E-01)	F16	2.01E+02 (1.39E-01)	**2.00E+02 (3.84E-01)**
F17	1.14E+03 (9.10E+01)	**7.69E+02 (6.23E+01)**	F17	2.36E+03 (1.89E+02)	**1.36E+03 (1.58E+02)**
F18	1.01E+03 (7.72E+01)	**7.38E+02 (5.20E+01)**	F18	1.94E+03 (1.47E+02)	**1.25E+03 (1.62E+02)**
F19	5.16E+02 (2.43E+00)	**5.11E+02 (2.81E+00)**	F19	5.45E+02 (4.78E+00)	**5.33E+02 (9.08E+00)**
F20	6.24E+02 (4.50E-01)	6.24E+02 (4.15E-01)	F20	6.50E+02 (2.11E-14)	6.50E+02 (4.71E-12)
F21	1.65E+03 (3.18E+02)	**1.44E+03 (4.27E+02)**	F21	1.14E+03 (6.13E+01)	1.14E+03 (6.00E+01)
F22	1.08E+04 (1.36E+03)	**1.04E+04 (1.20E+03)**	F22	2.34E+04 (1.87E+03)	**2.22E+04 (2.42E+03)**
F23	1.09E+04 (1.07E+03)	**1.04E+04 (1.35E+03)**	F23	2.21E+04 (1.59E+03)	**2.19E+04 (1.57E+03)**
F24	1.44E+03 (6.62E+01)	**1.38E+03 (2.00E+01)**	F24	2.44E+03 (5.16E+02)	**1.72E+03 (2.97E+02)**
F25	1.59E+03 (3.53E+01)	**1.58E+03 (3.03E+01)**	F25	1.96E+03 (1.01E+02)	1.96E+03 (1.12E+02)
F26	1.64E+03 (6.94E+01)	**1.60E+03 (9.24E+01)**	F26	1.85E+03 (1.88E+01)	**1.82E+03 (2.40E+01)**
F27	3.48E+03 (1.65E+02)	**3.26E+03 (1.50E+02)**	F27	5.57E+03 (2.44E+02)	**5.10E+03 (2.59E+02)**
F28	9.11E+03 (7.24E+02)	**8.92E+03 (5.80E+02)**	F28	1.97E+04 (1.72E+03)	**1.93E+04 (1.48E+03)**

Table 4.8: Results obtained by the Wilcoxon signed-rank test for ASBSO vs. BSO on CEC'13.

Dimension	R^+	R^-	p-value	α=0.05	α=0.01
10	330.5	75.5	2.782E-3	YES	YES
30	319.0	59.0	1.132E-3	YES	YES
50	319.0	87.0	7.072E-3	YES	YES
100	321.0	85.0	6.06E-3	YES	YES

Table 4.9: Experimental results of CEC'17 benchmark functions (F29-F57) using BSO and ASBSO at $D = 10$ and $D = 30$.

	D=10			D=30	
	BSO	ASBSO		BSO	ASBSO
	Mean (Std Dev)	Mean (Std Dev)		Mean (Std Dev)	Mean (Std Dev)
F29	8.95E+02 (1.09E+03)	**8.60E+02 (1.12E+03)**	F29	2.47E+03 (1.95E+03)	**2.21E+03 (2.00E+03)**
F30	3.00E+02 (0.00E+00)	3.00E+02 (1.67E-09)	F30	5.34E+02 (2.66E+02)	**3.95E+02 (1.10E+02)**
F31	4.04E+02 (7.05E+00)	**4.03E+02 (1.50E+00)**	F31	**4.67E+02 (2.19E+01)**	4.72E+02 (2.92E+01)
F32	5.35E+02 (1.43E+01)	**5.34E+02 (1.30E+01)**	F32	6.87E+02 (4.05E+01)	**6.86E+02 (3.45E+01)**
F33	6.24E+02 (8.60E+00)	6.24E+02 (7.25E+00)	F33	6.52E+02 (7.01E+00)	**6.51E+02 (7.77E+00)**
F34	7.57E+02 (1.79E+01)	**7.54E+02 (2.15E+01)**	F34	**1.15E+03 (9.65E+01)**	1.16E+03 (9.94E+01)
F35	8.23E+02 (9.45E+00)	**8.22E+02 (9.17E+00)**	F35	9.47E+02 (2.82E+01)	**9.41E+02 (3.19E+01)**
F36	1.09E+03 (1.58E+02)	1.09E+03 (1.03E+02)	F36	3.98E+03 (6.95E+02)	**3.93E+03 (6.39E+02)**
F37	2.10E+03 (2.57E+02)	**2.08E+03 (3.22E+02)**	F37	5.30E+03 (5.16E+02)	**5.20E+03 (5.67E+02)**
F38	**1.15E+03 (3.21E+01)**	1.16E+03 (3.52E+01)	F38	1.23E+03 (4.05E+01)	1.23E+03 (4.75E+01)
F39	1.19E+05 (1.29E+05)	**5.90E+04 (5.40E+04)**	F39	1.77E+06 (1.25E+06)	**1.41E+06 (8.00E+05)**
F40	8.96E+03 (5.30E+03)	**7.80E+03 (5.55E+03)**	F40	5.36E+04 (2.85E+04)	**5.04E+04 (2.64E+04)**
F41	1.72E+03 (1.06E+03)	**1.71E+03 (3.43E+02)**	F41	**6.40E+03 (4.66E+03)**	7.08E+03 (5.23E+03)
F42	4.12E+03 (1.91E+03)	**4.10E+03 (3.34E+03)**	F42	**2.95E+04 (1.61E+04)**	3.01E+04 (2.25E+04)
F43	1.92E+03 (1.12E+02)	**1.87E+03 (1.23E+02)**	F43	3.20E+03 (4.24E+02)	**3.01E+03 (2.25E+02)**
F44	1.77E+03 (4.19E+01)	1.77E+03 (4.48E+01)	F44	2.48E+03 (2.56E+02)	**2.40E+03 (2.44E+02)**
F45	1.03E+04 (1.26E+04)	**9.78E+03 (1.01E+04)**	F45	**1.21E+05 (1.03E+05)**	1.23E+05 (1.21E+05)
F46	3.28E+03 (2.12E+03)	**3.15E+03 (1.71E+03)**	F46	1.52E+05 (6.43E+04)	**1.25E+05 (6.31E+04)**
F47	2.13E+03 (6.33E+01)	2.13E+03 (6.22E+01)	F47	2.67E+03 (1.74E+02)	2.67E+03 (2.17E+02)
F48	2.29E+03 (6.44E+01)	**2.27E+03 (5.96E+01)**	F48	2.50E+03 (4.48E+01)	**2.49E+03 (3.14E+01)**
F49	2.30E+03 (1.03E+01)	2.30E+03 (1.14E+01)	F49	6.03E+03 (1.77E+03)	**5.79E+03 (2.04E+03)**
F50	2.70E+03 (3.40E+01)	**2.69E+03 (2.86E+01)**	F50	3.29E+03 (1.27E+02)	**3.26E+03 (1.24E+02)**
F51	2.78E+03 (1.19E+02)	**2.73E+03 (1.46E+02)**	F51	3.50E+03 (1.13E+02)	**3.49E+03 (9.56E+01)**
F52	**2.92E+03 (2.25E+01)**	2.93E+03 (2.19E+01)	F52	2.89E+03 (1.46E+01)	2.89E+03 (1.25E+01)
F53	**3.33E+03 (3.90E+02)**	3.34E+03 (3.37E+02)	F53	8.16E+03 (1.57E+03)	**7.84E+03 (1.80E+03)**
F54	**3.16E+03 (3.16E+01)**	3.17E+03 (3.40E+01)	F54	**3.82E+03 (2.91E+02)**	3.85E+03 (2.17E+02)
F55	**3.21E+03 (8.38E+01)**	3.23E+03 (1.81E+02)	F55	3.21E+03 (2.57E+01)	**3.18E+03 (3.60E+01)**
F56	3.26E+03 (8.93E+01)	3.26E+03 (6.44E+01)	F56	**4.38E+03 (2.79E+02)**	4.40E+03 (3.39E+02)
F57	**5.81E+04 (3.46E+04)**	2.95E+05 (6.09E+05)	F57	5.74E+05 (3.50E+05)	**5.16E+05 (2.90E+05)**

Table 4.10: Experimental results of CEC'17 benchmark functions (F29-F57) using BSO and ASBSO at $D = 50$ and $D = 100$.

	D=50			D=100	
	BSO	ASBSO		BSO	ASBSO
	Mean (Std Dev)	Mean (Std Dev)		Mean (Std Dev)	Mean (Std Dev)
F29	2.30E+03 (2.15E+03)	**1.20E+03 (1.21E+03)**	F29	5.23E+05 (1.62E+05)	**3.34E+05 (9.55E+05)**
F30	7.44E+03 (2.73E+03)	**6.53E+03 (2.25E+03)**	F30	8.99E+04 (1.69E+04)	**8.88E+04 (1.71E+04)**
F31	5.47E+02 (5.86E+01)	**5.42E+02 (5.06E+01)**	F31	6.81E+02 (4.83E+01)	**6.80E+02 (4.62E+01)**
F32	8.17E+02 (3.72E+01)	**8.12E+02 (4.79E+01)**	F32	1.31E+03 (7.77E+01)	**1.30E+03 (8.56E+01)**
F33	6.61E+02 (4.98E+00)	**6.60E+02 (5.32E+00)**	F33	6.65E+02 (4.33E+00)	**6.64E+02 (3.50E+00)**
F34	1.66E+03 (1.22E+02)	**1.65E+03 (1.34E+02)**	F34	3.34E+03 (2.43E+02)	**3.32E+03 (2.94E+02)**
F35	1.13E+03 (4.31E+01)	1.13E+03 (4.61E+01)	F35	1.73E+03 (8.70E+01)	**1.69E+03 (7.42E+01)**
F36	1.12E+04 (1.32E+03)	**1.09E+04 (1.56E+03)**	F36	2.69E+04 (3.07E+03)	**2.43E+04 (3.87E+03)**
F37	**8.17E+03 (1.01E+03)**	8.24E+03 (8.64E+02)	F37	1.65E+04 (1.16E+03)	**1.62E+04 (1.14E+03)**
F38	1.31E+03 (3.87E+01)	**1.30E+03 (4.78E+01)**	F38	2.45E+03 (2.72E+02)	**2.39E+03 (1.41E+02)**
F39	1.15E+07 (6.98E+06)	1.15E+07 (5.00E+06)	F39	7.74E+07 (1.56E+07)	**6.92E+07 (1.34E+07)**
F40	5.26E+04 (2.34E+04)	**5.86E+04 (2.70E+04)**	F40	3.84E+04 (1.60E+04)	**3.83E+04 (1.16E+04)**
F41	4.11E+04 (2.81E+04)	**3.17E+04 (1.70E+04)**	F41	3.54E+05 (1.24E+05)	**2.81E+05 (1.20E+05)**
F42	3.06E+04 (2.08E+04)	**2.50E+04 (1.47E+04)**	F42	**3.07E+04 (1.29E+04)**	3.11E+04 (1.08E+04)
F43	3.86E+03 (4.72E+02)	**3.82E+03 (4.50E+02)**	F43	6.76E+03 (7.95E+02)	**6.73E+03 (7.46E+02)**
F44	3.61E+03 (3.70E+02)	3.61E+03 (3.71E+02)	F44	5.58E+03 (6.39E+02)	**5.49E+03 (5.70E+02)**
F45	3.53E+05 (1.31E+05)	**2.97E+05 (9.96E+04)**	F45	5.04E+05 (1.67E+05)	**5.03E+05 (2.06E+05)**
F46	5.35E+05 (2.17E+05)	**4.09E+05 (1.92E+05)**	F46	2.41E+06 (1.20E+06)	**2.24E+06 (1.07E+06)**
F47	3.63E+03 (2.15E+02)	**3.42E+03 (3.69E+02)**	F47	**5.69E+03 (4.49E+02)**	5.76E+03 (5.01E+02)
F48	2.75E+03 (6.79E+01)	**2.73E+03 (7.92E+01)**	F48	**3.99E+03 (1.67E+02)**	4.02E+03 (1.78E+02)
F49	1.03E+04 (6.94E+02)	**9.86E+03 (6.91E+02)**	F49	**1.88E+04 (9.48E+02)**	1.91E+04 (1.16E+03)
F50	**4.00E+03 (1.52E+02)**	4.01E+03 (2.17E+02)	F50	**5.44E+03 (3.08E+02)**	5.50E+03 (2.81E+02)
F51	**4.14E+03 (1.37E+02)**	4.16E+03 (2.01E+02)	F51	6.35E+03 (5.21E+02)	**6.22E+03 (4.96E+02)**
F52	**2.95E+03 (2.86E+01)**	3.07E+03 (2.77E+01)	F52	3.27E+03 (7.41E+01)	**3.32E+03 (4.75E+01)**
F53	1.29E+04 (2.04E+03)	**1.26E+04 (2.06E+03)**	F53	**3.07E+04 (1.45E+03)**	3.08E+04 (1.90E+03)
F54	5.50E+03 (5.93E+02)	**5.43E+03 (4.31E+02)**	F54	7.79E+03 (1.49E+03)	**7.46E+03 (1.28E+03)**
F55	**3.29E+03 (2.16E+01)**	3.31E+03 (2.64E+01)	F55	**3.36E+03 (3.28E+01)**	3.38E+03 (2.36E+01)
F56	5.61E+03 (3.71E+02)	**5.59E+03 (4.17E+02)**	F56	9.11E+03 (5.95E+02)	**9.01E+03 (6.78E+02)**
F57	**1.67E+07 (1.54E+06)**	1.74E+07 (1.97E+06)	F57	1.26E+07 (4.04E+06)	**1.03E+07 (3.48E+06)**

Table 4.11: Results obtained by the Wilcoxon signed-rank test for ASBSO vs. BSO on CEC'17.

Dimension	R^+	R^-	p-value	α=0.05	α=0.01
10	295.0	111.0	3.576E-2	YES	NO
30	300.5	105.5	2.555E-2	YES	NO
50	302.0	104.0	2.322E-2	YES	NO
100	338.0	97.0	8.008E-3	YES	YES

4.3 Experimental Results

Two groups of comparisons have been carried out which include internal comparisons and external comparisons using CEC'13 and CEC'17 test functions. It should be noticed that F2 in CEC'17 has been excluded because it shows unstable behavior especially for higher dimensions, and significant performance variations for the same algorithm implemented in Matlab, or C Language [63, 64]. The internal comparison aims to demonstrate that ASBSO can achieve better performance than BSO not only at low dimension, but also at high dimension. Therefore, these comprehensive

Table 4.12: Experimental results of CEC'13 (F1-F28) using ASBSO, CGSA-M, MABC, ABC, DE, WOA and SCA.

Algorithm	F1	F2	F3	F4
ASBSO	-1.40E+03 ± 1.98E-13	**1.54E+06 ± 4.26E+05**	**8.47E+07 ± 8.64E+07**	**5.08E+03 ± 2.25E+03**
CGSA-M	-1.40E+03 ± 0.00E+00	7.31E+06 ± 1.14E+06	5.82E+09 ± 1.89E+09	6.60E+04 ± 4.02E+03
MABC	-1.40E+03 ± 0.00E+00	2.06E+08 ± 3.74E+07	6.86E+10 ± 1.90E+10	7.36E+04 ± 8.16E+03
ABC	-1.40E+03 ± 1.03E-13	2.19E+08 ± 3.21E+07	6.66E+10 ± 1.59E+10	7.03E+04 ± 9.91E+03
DE	-7.26E+02 ± 3.32E+02	1.13E+08 ± 2.05E+07	1.32E+10 ± 2.63E+09	5.72E+04 ± 9.55E+03
WOA	-1.40E+03 ± 1.60E-01	3.47E+07 ± 1.64E+07	1.35E+10 ± 7.98E+09	5.69E+04 ± 2.12E+04
SCA	9.08E+03 ± 1.72E+03	1.31E+08 ± 3.26E+07	3.12E+10 ± 8.79E+09	3.17E+04 ± 6.61E+03
Algorithm	F5	F6	F7	F8
ASBSO	-1.00E+03 ± 2.98E-03	-8.64E+02 ± 2.46E+01	-7.08E+02 ± 3.90E+01	-6.79E+02 ± 6.70E-02
CGSA-M	-1.00E+03 ± 8.64E-13	-8.35E+02 ± 1.49E+01	**-7.28E+02 ± 3.33E+01**	-6.79E+02 ± 5.31E-02
MABC	-1.50E+02 ± 1.44E+02	**-8.67E+02 ± 1.32E+01**	-5.73E+02 ± 2.37E+01	-6.79E+02 ± 7.13E-02
ABC	-4.83E+01 ± 1.43E+02	-8.57E+02 ± 1.25E+01	-5.76E+02 ± 2.82E+01	-6.79E+02 ± 4.36E-02
DE	-9.32E+02 ± 1.42E+01	-7.37E+02 ± 2.44E+01	-6.88E+02 ± 1.16E+01	-6.79E+02 ± 4.22E-02
WOA	-9.20E+02 ± 1.77E+01	-7.99E+02 ± 3.72E+01	-9.41E+01 ± 2.06E+03	-6.79E+02 ± 4.45E-02
SCA	9.15E+02 ± 3.40E+02	-2.06E+02 ± 2.22E+02	-6.27E+02 ± 4.28E+01	-6.79E+02 ± 6.99E-02
Algorithm	F9	F10	F11	F12
ASBSO	**-5.71E+02 ± 3.23E+00**	**-5.00E+02 ± 5.35E-02**	-1.82E+02 ± 5.40E+01	**-7.64E+01 ± 4.85E+01**
CGSA-M	-5.69E+02 ± 3.41E+00	-5.00E+02 ± 7.86E-02	-1.11E+02 ± 2.41E+01	3.68E+01 ± 2.31E+01
MABC	-5.61E+02 ± 1.38E+00	-2.82E+02 ± 3.27E+01	**-1.94E+02 ± 1.33E+01**	-7.01E+01 ± 1.05E+01
ABC	-5.61E+02 ± 1.44E+00	-2.37E+02 ± 3.51E+01	-1.84E+02 ± 1.78E+01	-6.84E+01 ± 1.41E+01
DE	-5.61E+02 ± 1.10E+00	-4.30E+01 ± 1.05E+02	-1.59E+02 ± 2.11E+01	-9.86E+00 ± 1.22E+01
WOA	-5.64E+02 ± 2.78E+00	-4.45E+02 ± 2.21E+01	7.52E+01 ± 9.86E+01	1.46E+02 ± 1.03E+02
SCA	-5.61E+02 ± 1.21E+00	1.01E+03 ± 3.28E+02	-4.43E+01 ± 2.90E+01	7.99E+01 ± 3.52E+01
Algorithm	F13	F14	F15	F16
ASBSO	1.30E+02 ± 6.54E+01	**3.68E+03 ± 4.56E+02**	3.88E+03 ± 5.74E+02	2.00E+02 ± 1.13E-01
CGSA-M	2.61E+02 ± 3.66E+01	3.89E+03 ± 4.67E+02	**3.78E+03 ± 4.89E+02**	2.00E+02 ± 4.63E-03
MABC	**1.80E+01 ± 1.40E+01**	7.11E+03 ± 2.23E+02	7.45E+03 ± 2.00E+02	2.02E+02 ± 2.54E-01
ABC	2.24E+01 ± 9.10E+00	7.15E+03 ± 2.18E+02	7.46E+03 ± 2.37E+02	2.02E+02 ± 2.93E-01
DE	9.85E+01 ± 8.77E+00	6.61E+03 ± 4.55E+02	7.47E+03 ± 2.43E+02	2.02E+02 ± 3.33E-01
WOA	2.99E+02 ± 8.96E+01	4.88E+03 ± 7.84E+02	5.49E+03 ± 1.02E+03	2.02E+02 ± 4.34E-01
SCA	1.67E+02 ± 3.69E+01	7.00E+03 ± 3.40E+02	7.49E+03 ± 2.06E+02	2.02E+02 ± 2.55E-01
Algorithm	F17	F18	F19	F20
ASBSO	5.28E+02 ± 5.13E+01	5.98E+02 ± 2.85E+01	**5.04E+02 ± 7.60E-01**	6.14E+02 ± 2.94E-01
CGSA-M	**3.66E+02 ± 8.04E+00**	**4.55E+02 ± 5.69E+00**	5.11E+02 ± 2.40E+00	6.15E+02 ± 2.23E-01
MABC	5.31E+02 ± 1.23E+01	6.42E+02 ± 1.05E+01	1.50E+03 ± 5.36E+02	6.15E+02 ± 1.25E-01
ABC	5.38E+02 ± 1.08E+01	6.44E+02 ± 8.95E+00	1.74E+03 ± 5.62E+02	6.15E+02 ± 1.37E-01
DE	6.31E+02 ± 3.76E+01	7.47E+02 ± 3.22E+01	5.37E+02 ± 1.24E+01	**6.13E+02 ± 1.37E-01**
WOA	8.89E+02 ± 1.11E+02	1.01E+03 ± 1.20E+02	5.58E+02 ± 1.90E+01	6.15E+02 ± 3.05E-01
SCA	7.88E+02 ± 4.68E+01	8.88E+02 ± 3.94E+01	2.99E+03 ± 1.30E+03	6.14E+02 ± 3.54E-01
Algorithm	F21	F22	F23	F24
ASBSO	1.02E+03 ± 8.42E+01	**5.36E+03 ± 4.70E+02**	**6.00E+03 ± 7.84E+02**	1.31E+03 ± 2.60E+01
CGSA-M	1.01E+03 ± 4.38E+01	7.42E+03 ± 5.68E+02	6.81E+03 ± 3.14E+02	1.35E+03 ± 6.98E+01
MABC	**9.94E+02 ± 2.19E+01**	8.69E+03 ± 2.71E+02	8.84E+03 ± 3.22E+02	**1.28E+03 ± 7.18E+00**
ABC	1.00E+03 ± 1.33E-02	8.74E+03 ± 2.01E+02	8.82E+03 ± 3.04E+02	1.29E+03 ± 5.63E+00
DE	1.57E+03 ± 1.87E+02	7.76E+03 ± 4.13E+02	8.44E+03 ± 3.02E+02	1.30E+03 ± 2.66E+00
WOA	1.03E+03 ± 6.87E+01	6.76E+03 ± 1.08E+03	7.61E+03 ± 8.55E+02	1.31E+03 ± 1.00E+01
SCA	2.58E+03 ± 1.79E+02	8.35E+03 ± 4.43E+02	8.70E+03 ± 3.72E+02	1.32E+03 ± 5.05E+00
Algorithm	F25	F26	F27	F28
ASBSO	**1.41E+03 ± 1.03E+01**	1.46E+03 ± 7.82E+01	2.42E+03 ± 1.07E+02	2.03E+03 ± 8.02E+02
CGSA-M	1.49E+03 ± 7.05E+00	1.55E+03 ± 3.14E+01	**2.23E+03 ± 8.05E+01**	5.00E+03 ± 2.48E+02
MABC	1.44E+03 ± 3.82E+00	1.42E+03 ± 5.30E+00	2.66E+03 ± 4.42E+01	**1.70E+03 ± 9.31E-05**
ABC	1.44E+03 ± 5.31E+00	1.42E+03 ± 6.02E+00	2.67E+03 ± 4.60E+01	1.70E+03 ± 1.47E+00
DE	1.42E+03 ± 3.38E+00	**1.41E+03 ± 2.06E+00**	2.63E+03 ± 2.62E+01	2.66E+03 ± 1.29E+02
WOA	1.42E+03 ± 9.66E+00	1.53E+03 ± 9.64E+01	2.61E+03 ± 6.95E+01	5.36E+03 ± 7.53E+02
SCA	1.43E+03 ± 4.21E+00	1.41E+03 ± 5.66E+00	2.66E+03 ± 4.51E+01	3.92E+03 ± 1.98E+02

Table 4.13: Experimental results of CEC'17 (F29-F57) using ASBSO, CGSA-M, MABC, ABC, DE, WOA and SCA.

Algorithm	F29	F30	F31	F32
ASBSO	2.21E+03 ± 2.00E+03	**3.95E+02 ± 1.10E+02**	**4.72E+02 ± 2.92E+01**	**6.86E+02 ± 3.45E+01**
CGSA-M	**1.82E+03 ± 9.25E+02**	8.51E+04 ± 6.02E+03	5.34E+02 ± 1.24E+01	7.33E+02 ± 1.99E+01
MABC	2.01E+03 ± 1.82E+03	9.71E+04 ± 1.28E+04	5.17E+02 ± 2.31E+00	7.19E+02 ± 1.48E+01
ABC	4.72E+04 ± 7.74E+04	1.03E+05 ± 1.09E+04	5.19E+02 ± 2.79E+00	7.18E+02 ± 9.44E+00
DE	1.32E+09 ± 4.55E+08	8.04E+04 ± 1.05E+04	6.25E+02 ± 2.90E+01	7.50E+02 ± 1.31E+01
WOA	2.48E+06 ± 1.73E+06	1.60E+05 ± 8.12E+04	5.45E+02 ± 3.70E+01	7.64E+02 ± 6.16E+01
SCA	1.18E+10 ± 1.77E+09	3.50E+04 ± 6.31E+03	1.40E+03 ± 2.74E+02	7.71E+02 ± 2.17E+01
Algorithm	F33	F34	F35	F36
ASBSO	6.51E+02 ± 7.77E+00	1.16E+03 ± 9.94E+01	**9.41E+02 ± 3.19E+01**	3.93E+03 ± 6.39E+02
CGSA-M	6.50E+02 ± 3.78E+00	**7.86E+02 ± 1.11E+01**	9.52E+02 ± 1.34E+01	2.96E+03 ± 2.48E+02
MABC	**6.00E+02 ± 7.55E-04**	9.39E+02 ± 9.74E+00	1.02E+03 ± 1.08E+01	**1.41E+03 ± 3.36E+02**
ABC	6.00E+02 ± 6.69E-03	9.43E+02 ± 9.71E+00	1.02E+03 ± 1.16E+01	1.90E+03 ± 4.45E+02
DE	6.24E+02 ± 4.43E+00	1.17E+03 ± 9.96E+01	1.06E+03 ± 1.14E+01	4.14E+03 ± 7.85E+02
WOA	6.67E+02 ± 1.12E+01	1.21E+03 ± 9.33E+01	9.99E+02 ± 3.20E+01	6.54E+03 ± 2.35E+03
SCA	6.49E+02 ± 5.34E+00	1.12E+03 ± 2.88E+01	1.05E+03 ± 1.63E+01	5.52E+03 ± 1.10E+03
Algorithm	F37	F38	F39	F40
ASBSO	5.20E+03 ± 5.67E+02	**1.23E+03 ± 4.75E+01**	**1.41E+06 ± 8.00E+05**	5.04E+04 ± 2.64E+04
CGSA-M	**4.83E+03 ± 4.17E+02**	1.46E+03 ± 7.26E+01	1.49E+07 ± 2.37E+07	3.02E+04 ± 5.39E+03
MABC	8.15E+03 ± 3.09E+02	4.31E+03 ± 6.19E+02	7.79E+07 ± 2.79E+07	8.46E+07 ± 2.90E+07
ABC	8.10E+03 ± 3.19E+02	4.37E+03 ± 7.31E+02	1.17E+08 ± 2.66E+07	8.02E+07 ± 3.32E+07
DE	8.17E+03 ± 2.51E+02	1.33E+03 ± 2.16E+01	5.43E+07 ± 1.60E+07	**4.13E+03 ± 5.37E+02**
WOA	6.06E+03 ± 9.74E+02	1.45E+03 ± 1.15E+02	4.47E+07 ± 3.11E+07	1.35E+05 ± 1.44E+05
SCA	8.12E+03 ± 3.34E+02	2.19E+03 ± 3.99E+02	1.21E+09 ± 2.30E+08	4.07E+08 ± 1.98E+08
Algorithm	F41	F42	F43	F44
ASBSO	7.08E+03 ± 5.23E+03	3.01E+04 ± 2.25E+04	**3.01E+03 ± 2.25E+02**	**2.40E+03 ± 2.44E+02**
CGSA-M	4.79E+05 ± 1.35E+05	1.21E+04 ± 1.59E+03	3.16E+03 ± 2.60E+02	2.81E+03 ± 2.33E+02
MABC	3.62E+05 ± 1.64E+05	1.96E+07 ± 7.41E+06	3.68E+03 ± 1.57E+02	2.50E+03 ± 1.17E+02
ABC	3.04E+05 ± 1.25E+05	2.08E+07 ± 8.65E+06	3.76E+03 ± 1.87E+02	2.49E+03 ± 1.19E+02
DE	**1.49E+03 ± 7.57E+00**	**1.72E+03 ± 3.06E+01**	3.19E+03 ± 3.08E+02	2.41E+03 ± 2.16E+02
WOA	7.27E+05 ± 6.79E+05	6.97E+04 ± 4.48E+04	3.47E+03 ± 5.17E+02	2.53E+03 ± 2.16E+02
SCA	1.19E+05 ± 7.05E+04	1.56E+07 ± 1.29E+07	3.64E+03 ± 2.13E+02	2.42E+03 ± 1.64E+02
Algorithm	F45	F46	F47	F48
ASBSO	1.23E+05 ± 1.21E+05	1.25E+05 ± 6.31E+04	2.67E+03 ± 2.17E+02	**2.49E+03 ± 3.14E+01**
CGSA-M	2.99E+05 ± 1.33E+05	1.56E+04 ± 5.55E+03	3.01E+03 ± 2.02E+02	2.56E+03 ± 2.59E+01
MABC	6.77E+06 ± 2.63E+06	2.67E+07 ± 1.04E+07	2.75E+03 ± 1.06E+02	2.51E+03 ± 1.18E+01
ABC	6.34E+06 ± 3.20E+06	2.39E+07 ± 1.03E+07	2.74E+03 ± 8.15E+01	2.52E+03 ± 1.18E+01
DE	**6.90E+03 ± 1.84E+03**	**1.96E+03 ± 4.88E+00**	**2.31E+03 ± 2.03E+02**	2.54E+03 ± 1.26E+01
WOA	3.02E+06 ± 2.58E+06	2.45E+06 ± 2.03E+06	2.78E+03 ± 1.76E+02	2.56E+03 ± 6.24E+01
SCA	2.80E+06 ± 1.21E+06	2.48E+07 ± 1.13E+07	2.61E+03 ± 1.29E+02	2.56E+03 ± 1.92E+01
Algorithm	F49	F50	F51	F52
ASBSO	5.79E+03 ± 2.04E+03	3.26E+03 ± 1.24E+02	3.49E+03 ± 9.56E+01	2.89E+03 ± 1.25E+01
CGSA-M	6.20E+03 ± 1.84E+03	3.61E+03 ± 1.59E+02	3.27E+03 ± 5.85E+01	2.93E+03 ± 1.24E+01
MABC	2.52E+03 ± 1.76E+02	2.88E+03 ± 1.65E+01	3.04E+03 ± 1.19E+01	**2.89E+03 ± 1.29E-01**
ABC	2.64E+03 ± 2.08E+02	2.89E+03 ± 1.60E+01	3.04E+03 ± 1.17E+01	2.89E+03 ± 1.73E-01
DE	**2.52E+03 ± 4.78E+01**	**2.88E+03 ± 1.38E+01**	**3.04E+03 ± 1.07E+01**	3.01E+03 ± 3.38E+01
WOA	6.65E+03 ± 1.87E+03	3.05E+03 ± 8.54E+01	3.16E+03 ± 9.15E+01	2.94E+03 ± 2.73E+01
SCA	8.25E+03 ± 2.37E+03	2.99E+03 ± 2.34E+01	3.16E+03 ± 2.96E+01	3.20E+03 ± 4.91E+01
Algorithm	F53	F54	F55	F56
ASBSO	7.84E+03 ± 1.80E+03	3.85E+03 ± 2.17E+02	**3.18E+03 ± 3.60E+01**	4.40E+03 ± 3.39E+02
CGSA-M	6.75E+03 ± 6.48E+02	4.51E+03 ± 3.18E+02	3.31E+03 ± 5.88E+01	4.71E+03 ± 2.26E+02
MABC	5.71E+03 ± 1.13E+02	3.46E+03 ± 2.89E+01	3.23E+03 ± 1.95E+01	4.86E+03 ± 1.75E+02
ABC	5.74E+03 ± 1.28E+02	3.46E+03 ± 3.87E+01	3.26E+03 ± 2.59E+01	4.93E+03 ± 1.31E+02
DE	**3.04E+03 ± 1.07E+01**	**3.01E+03 ± 3.38E+01**	6.15E+03 ± 1.47E+02	**3.26E+03 ± 1.20E+01**
WOA	7.24E+03 ± 1.00E+03	3.36E+03 ± 8.88E+01	3.31E+03 ± 3.96E+01	5.00E+03 ± 4.58E+02
SCA	6.87E+03 ± 2.56E+02	3.39E+03 ± 4.83E+01	3.78E+03 ± 1.36E+02	4.62E+03 ± 2.62E+02
Algorithm	F57			
ASBSO	5.16E+05 ± 2.90E+05			
CGSA-M	**1.48E+05 ± 8.05E+04**			
MABC	2.38E+07 ± 7.73E+06			
ABC	2.67E+07 ± 1.04E+07			
DE	1.94E+05 ± 7.07E+04			
WOA	1.04E+07 ± 6.42E+06			
SCA	7.44E+07 ± 2.59E+07			

comparisons can show the search ability and robustness of ASBSO for solving the problems with different difficulty levels.

After proving the superiority of ASBSO, in the external comparison, some meta-heuristic algorithms have been taken into account to further evaluate the performance of ASBSO. Artificial bee colony algorithm (ABC) [67] is very popular in literature and its influence is next only to particle swarm optimization (PSO) [2] in swarm-based meta-heuristic algorithms [68, 69, 70]. Differential evolution (DE) [40, 71, 72] is the most famous optimization algorithm with very powerful search ability. MABC and CGSA-M [22, 73] which are two variations based on ABC and gravitational search algorithm (GSA) [14, 25, 74] implement memory-based selection strategies. Thus, they are very suitable to be chosen to compare with ASBSO. Furthermore, two newly proposed effective swarm intelligence based algorithms, i.e., whale optimization algorithm (WOA) [41] and sine cosine algorithm (SCA) [75], have been implemented. The population size of all compared algorithms is 100. All these contrast experiments are run for 30 times to reduce the random error, and the maximum number of function evaluation is set to $10000D$ (D is the dimension number).

4.3.1 Parameter Analysis

The aim of implementing multiple strategies and memory based selection method is to provide multiple step lengths in order to suit different search phases. Too few strategies couldn't satisfy this demand while too many strategies are redundant and will increase computational cost. Thus, we attempt $M = 4$ in this paper and preliminary experiments prove the validity of this parameter setting. A parameter analysis is executed to find an applicable value for H. Three values are applied involving 10, 20 and 30. In this comparison, k is set to 10. The contrast experiment is implement on CEC'13 and CEC'17 to find the most suitable value for four strategies.

Friedman test for multiple comparison is applied to analyze the results [44]. Table 4.5 lists statistical results obtained by Friedman test and $H = 20$ is the control algorithm. *Ranking* evaluates the performance of each algorithm, and a lower ranking

indicates a better performance. Unadjusted p-value doesn't consider the probability error in a multiple comparison. Thus, two commonly used post-hoc procedures, Holm and Hochberg procedures [45], are taken into account and their conservative adjusted p-values are convincing enough to eliminate Type I error [76]. $H = 20$ which maintains the best ranking of 1.4298 indicates that it's the best value for H. Therefore, $k = 10$ and $H = 20$ are chosen to be applied into the flexible multiple search length strategy.

4.3.2 Internal Comparison

In the first experiment, the CEC'13 and CEC'17 are used to compare the performance between traditional BSO and the proposed ASBSO. The experiments are tested at dimension $D = 10, 30, 50$ and 100 respectively.

The experimental results of CEC'13 are summarized in Tables 4.6 and 4.7, while Tables 4.9 and 4.10 show the results of CEC'17. All the better Mean and standard deviation (Std Dev) values are highlighted for convenience. From these tables, we can intuitively find out that ASBSO can obtain more number of better results than BSO. The former obtains better results on F4, F7, F9, F11, F14, F15, F17-F19, F27 and F28 at all tested dimensions, while BSO only obtains better result on F6 in CEC'13. In CEC'17, ASBSO outperforms on F29, F32, F43 and F46 while BSO can't obtain better performance at all dimensions on any function.

Wilcoxon signed-rank test is conducted to prove that ASBSO can beat BSO as it's a pairwise test which is used to analyze significant difference between the performance of two algorithms. R^+ and R^- values in Tables 4.8 and 4.11 can indicate the degree that ASBSO outperforms BSO. As we conduct ASBSO versus BSO, R^+ represents the sum of ranks for the functions on which ASBSO outperforms BSO, and R^- means the opposite.With the null hypothesis H_0 for the test assumes two compared algorithms have no difference, a better performance of our proposed algorithm can be shown via a higher R^+ value and p-value indicates the possibility that the null hypothesis happens. If p-value is lower than the level of significance $\alpha = 0.05$, we can accept the hypothesis that ASBSO is significantly better than BSO. Moreover, we set a more

rigorous level $\alpha = 0.01$ to further exhibit the improvement of ASBSO in solution quality.

All the comparisons in Table 4.8 can reach the level of $\alpha = 0.01$, while in Table 4.11, ASBSO can beat BSO on the level of $\alpha = 0.05$ at all dimensions but only has a significant difference at $D = 100$ when $\alpha = 0.01$. It's understandable because CEC'17 is a newly proposed benchmark function suit, all test functions have a promotion in difficulty and complexity compared with CEC'13.

From these results, it can be concluded that ASBSO has obvious advantage in comparison with BSO in terms of search ability and solution quality.

4.3.3 External Comparison

To investigate the performance of ASBSO when comparing with other swarm intelligence optimization algorithms, some well-known meta-heuristic algorithms, involving CGSA-M, MABC, ABC, DE, WOA and SCA, are implemented into numerical tests. Parameter settings can be investigated according to [22, 41, 67, 73, 75]. In DE, we use the efficient parameter set $F = 0.9$ and $CR = 0.9$ as suggested in [43, 77]. All tests have been executed at $D = 30$ with maximum number of function evaluation equals $10000D$ for 30 runs.

The results are listed in Tables 4.12 and 4.13. The best results are marked in boldface. It's visual that ASBSO obtains the largest number of the best results among all compared algorithms and we can draw a preliminary conclusion that ASBSO is very competitive in contrast with others. To more precisely analyze the results of multiple comparisons, Friedman test [44] which is widely used in [78, 79, 80] is employed. Table 4.14 lists statistical results obtained by Friedman test and ASBSO is the control algorithm. ASBSO maintains the best ranking of 2.5 while the second best is only 3.5526 which belongs to CGSA-M. Although adjusted p-values of Holm and Hochberg procedures are multiplied bigger than unadjusted p-values, they still reach the significant level of $\alpha = 0.05$. Furthermore, in terms of MABC, ABC, WOA and SCA, adjusted p-values satisfy the level of $\alpha = 0.01$. Wilcoxon test is also

conducted to verify the results of Friedman test and obtains similar p-values in Table 4.15. From all these results, it is obvious that ASBSO is significantly better than other contrast algorithms in benchmark function tests.

Table 4.14: Adjusted p-values (FRIEDMAN).

Algorithm	Ranking	unadjusted p	p_{Holm}	$p_{Hochberg}$	$\alpha = 0.05$	$\alpha = 0.01$
ASBSO vs.	2.5					
CGSA-M	3.5526	0.009286	0.011049	0.009286	YES	NO
MABC	3.9737	0.000271	0.000812	0.000812	YES	YES
ABC	4.3509	0.000005	0.000019	0.000019	YES	YES
DE	3.6228	0.005524	0.011049	0.009286	YES	NO
WOA	4.7982	0	0	0	YES	YES
SCA	5.2018	0	0	0	YES	YES

Table 4.15: Results obtained by the Wilcoxon signed-rank test for ASBSO vs. some other typical algorithms.

Algorithm	R^+	R^-	p-value	$\alpha = 0.05$	$\alpha = 0.01$
ASBSO vs.					
CGSA-M	1046.5	549.5	4.1615E-2	YES	NO
MABC	1208.5	387.5	7.81E-4	YES	YES
ABC	1256.0	340.0	1.73E-4	YES	YES
DE	1040.5	555.5	4.195E-2	YES	NO
WOA	1473.0	123.0	0.00	YES	YES
SCA	1510.0	143.0	0.00	YES	YES

Table 4.16: Experimental results on real-world problems.

	BSO	ASBSO	CGSA-M	MABC
RF1	1.30E+01±4.98E+00	**9.93E+00±4.68E+00**	2.19E+01±4.33E+00	1.81E+01±2.13E+00
RF2	-2.11E+01± 2.95E+00	**-2.52E+01±2.12E+00**	-3.24E+00±1.08E+00	-1.17E+01±9.23E-01
RF4	1.50E+01±9.12E-01	**1.47E+01±5.15E-01**	1.99E+01±2.09E+00	1.65E+01±2.37E+00
RF7	8.72E-01±1.01E-01	**7.91E-01±1.49E-01**	2.55E+00±2.39E-01	1.63E+00±9.01E-02
	ABC	DE	WOA	SCA
RF1	1.83E+01±1.65E+00	3.15E+01±2.01E+01	2.19E+01±5.00E+00	1.83E+01±3.98E+00
RF2	-1.13E+01±5.04E-01	-4.24E+00±4.02E-01	-1.84E+01±4.89E+00	-9.20E+00±1.16E+00
RF4	1.63E+01±2.04E+00	4.99E+01±2.46E+01	1.47E+01±1.26E+00	1.51E+01±1.19E+00
RF7	1.67E+00±8.83E-02	3.31E+00±4.06E-01	1.92E+00±2.30E-01	2.12E+00±2.02E-01

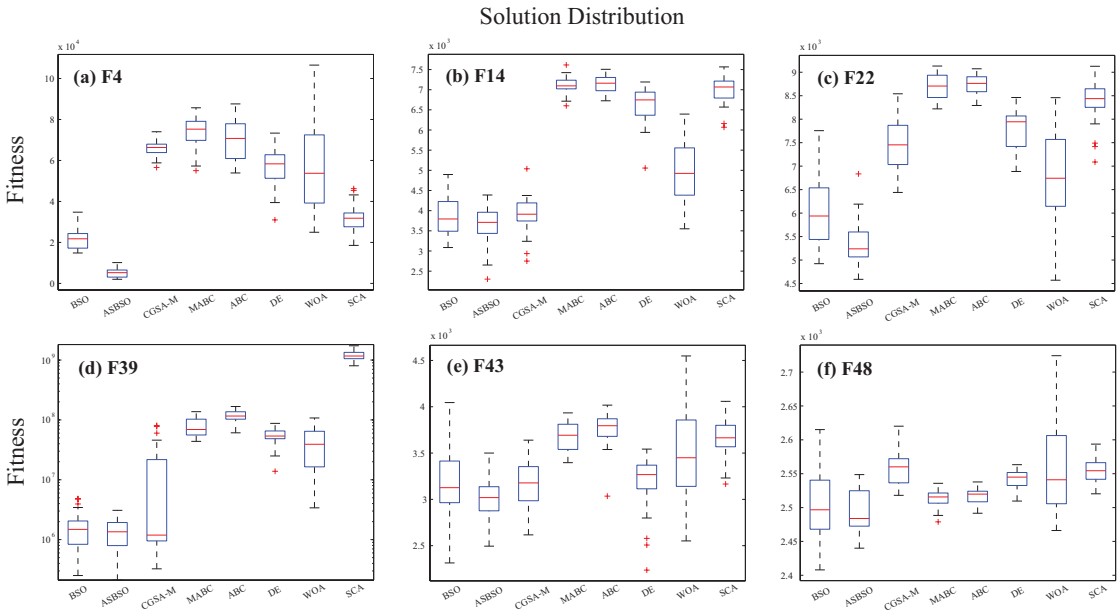

Figure 4.2: Box-and-Whisker diagrams of F4, F14, F22, F39, F43 and F48.

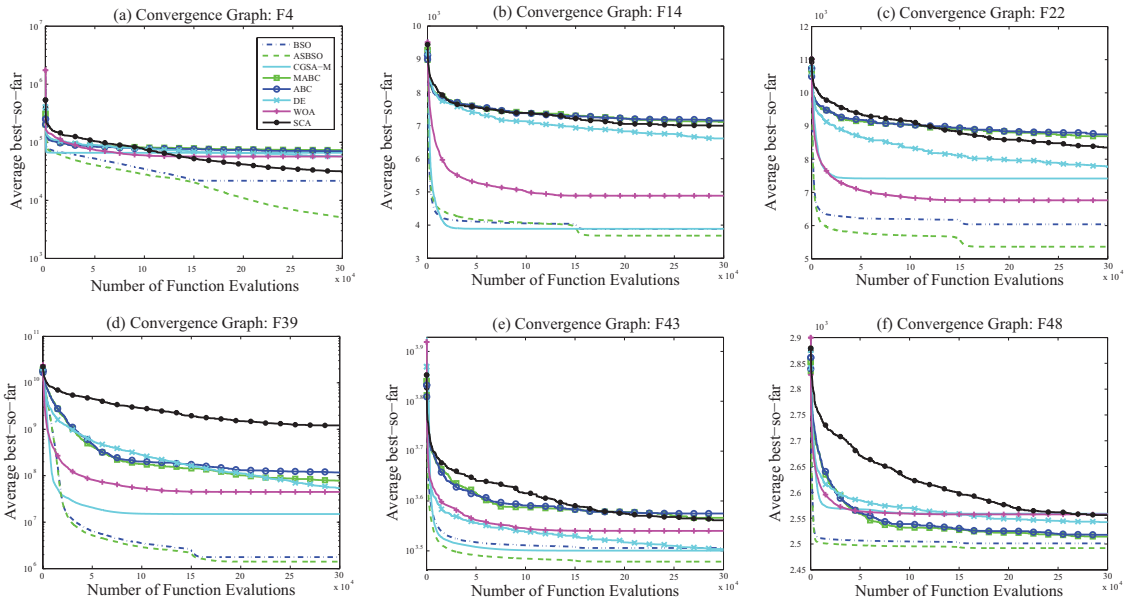

Figure 4.3: Convergence graphs of optimal solutions obtained by eight algorithms on F4, F14, F22, F39, F43 and F48.

To visually demonstrate the comparisons among ASBSO and other contrast algorithms, six functions, F4, F14, F22, F39, F43 and F48 with different properties, including unimodal, simple multimodal, hybrid and composition, are selected since they are representative to show the properties of all tested functions. The conver-

gent procedures and final solutions obtained by these algorithms in all 30 runs are exhibited.

Fig. 5.2 is the box-and-whisker diagrams and Fig. 5.3 is the convergence graphs. Five values including median, maximum, minimum, first quartile and third quartile are shown in box-and-whisker plots. The range between the first quartile and the third quartile is called interquartile range (IQR), and if the points locate either 1.5*IQR above the third quartile (i.e. 1.5*IQR) below the first quartile, they are marked as outliers. Extreme outliers refer to the points locate either 3*IQR above the third quartile or 3*IQR below the first quartile. In these six plots, the median values of ASBSO are the smallest and its IQRs are lower and shorter than most other algorithms. These indicate that the solution quality and stability obtained by ASBSO is much better than those of other contrast algorithms.

The convergence graphs can not only demonstrate the precision of solutions but also compare the convergence speeds. Fig. 5.3 shows that, ASBSO can possess the fast convergence speed. In details, all algorithms' convergence behaviors shown in Fig. 5.3 (a) are quite illuminating to further elaborate the search behavior of ASBSO. It is clear that ASBSO continues converging when other algorithms stop in the latter of the search iteration. Although ABC starts with a better initial position, it doesn't have the ability to jump out of local optima and ultimately be transcended by ASBSO. In the comparison between ASBSO and BSO, it illustrates that the former always has a better solution precision and convergence speed than the latter. When comparing with other algorithms, ASBSO also obtains fabulous performances. Thus, it can be concluded that the proposed adaptive step length based on memory selection method enhances the search ability and efficiency for ASBSO.

4.3.4 Real World Optimization Problems

It has been demonstrated that ASBSO can outperform traditional BSO and other well-known algorithms on benchmark functions. To further testify its application value, four problems introduced in CEC'11 [46] are used to execute this test: (1)

RF1: Parameter Estimation for Frequency-Modulated (FM) Sound Waves, (2) RF2: Lennard-Jones Potential Problem, (3) RF4: Optimal Control of a Non-Linear Stirred Tank Reactor, and (4) RF7: Transmission Network Expansion Planning (TNEP) problem [46]. All these problems are run for 30 independent times and the maximum function evaluation is set to $10000D$. The experimental results are presented in Table 4.16. It's obvious that ASBSO obtains dominance over all tested problems when compared with other algorithms, which well exhibiting its application value.

Table 4.17: Experimental results of using ASBSO and BSO with 1/5 Success Rule on CEC'13 and CEC'17 benchmark functions (F1-F57).

D=30					
	ASBSO	BSO with 1/5 Rule		ASBSO	BSO with 1/5 Rule
	Mean (Std Dev)	Mean (Std Dev)		Mean (Std Dev)	Mean (Std Dev)
F1	-1.40E+03 (1.98E-13)	-1.40E+03 (7.26E-13)	F29	**2.21E+03 (2.00E+03)**	2.93E+03 (3.12E+03)
F2	**1.54E+06 (4.26E+05)**	3.96E+06 (1.25E+06)	F30	**3.95E+02 (1.10E+02)**	5.39E+02 (2.13E+02)
F3	**8.47E+07 (8.64E+07)**	4.28E+08 (4.44E+08)	F31	**4.72E+02 (2.92E+01)**	5.00E+02 (1.92E+01)
F4	5.08E+03 (2.25E+03)	**1.86E+03 (1.34E+03)**	F32	6.86E+02 (3.45E+01)	**6.79E+02 (2.61E+01)**
F5	-1.00E+03 (2.98E-03)	-1.00E+03 (3.99E-04)	F33	**6.51E+02 (7.77E+00)**	6.52E+02 (9.75E+00)
F6	**-8.64E+02 (2.46E+01)**	-8.50E+02 (2.90E+01)	F34	1.16E+03 (9.94E+01)	**1.11E+03 (1.01E+02)**
F7	**-7.08E+02 (3.90E+01)**	-6.52E+02 (4.02E+01)	F35	**9.41E+02 (3.19E+01)**	9.35E+02 (2.82E+01)
F8	-6.79E+02 (6.70E-02)	-6.79E+02 (9.75E-02)	F36	**3.93E+03 (6.39E+02)**	4.59E+03 (8.43E+02)
F9	**-5.71E+02 (3.23E+00)**	-5.65E+02 (2.53E+00)	F37	**5.20E+03 (5.67E+02)**	5.78E+03 (9.50E+02)
F10	**-5.00E+02 (5.35E-02)**	-4.99E+02 (4.49E-01)	F38	**1.23E+03 (4.75E+01)**	1.26E+03 (5.13E+01)
F11	**-1.82E+02 (5.40E+01)**	-6.33E+01 (7.12E+01)	F39	**1.41E+06 (8.00E+05)**	3.77E+06 (2.16E+06)
F12	**-7.64E+01 (4.85E+01)**	4.40E+01 (9.33E+01)	F40	**5.04E+04 (2.64E+04)**	7.04E+04 (4.01E+04)
F13	**1.30E+02 (6.54E+01)**	2.04E+02 (8.10E+01)	F41	**7.08E+03 (5.23E+03)**	1.69E+04 (1.37E+04)
F14	**3.68E+03 (4.56E+02)**	4.27E+03 (5.94E+02)	F42	**3.01E+04 (2.25E+04)**	3.59E+04 (2.55E+04)
F15	**3.88E+03 (5.74E+02)**	4.78E+03 (7.29E+02)	F43	3.01E+03 (2.25E+02)	**3.00E+03 (2.90E+02)**
F16	**2.00E+02 (1.13E-01)**	2.01E+02 (5.29E-01)	F44	2.40E+03 (2.44E+02)	**2.22E+03 (1.91E+02)**
F17	**5.28E+02 (5.13E+01)**	6.43E+02 (1.03E+02)	F45	**1.23E+05 (1.21E+05)**	2.09E+05 (2.49E+05)
F18	**5.98E+02 (2.85E+01)**	7.64E+02 (7.58E+01)	F46	**1.25E+05 (6.31E+04)**	2.66E+05 (1.56E+05)
F19	**5.04E+02 (7.60E-01)**	5.27E+02 (1.13E+01)	F47	**2.67E+03 (2.17E+02)**	2.70E+03 (2.23E+02)
F20	**6.14E+02 (2.94E-01)**	6.15E+02 (5.43E-01)	F48	2.49E+03 (3.14E+01)	**2.45E+03 (2.66E+01)**
F21	1.02E+03 (8.42E+01)	1.02E+03 (7.70E+01)	F49	5.79E+03 (2.04E+03)	**5.60E+03 (2.60E+03)**
F22	**5.36E+03 (4.70E+02)**	6.21E+03 (7.47E+02)	F50	3.26E+03 (1.24E+02)	**2.87E+03 (5.16E+01)**
F23	**6.00E+03 (7.84E+02)**	6.39E+03 (8.77E+02)	F51	3.49E+03 (9.56E+01)	**3.01E+03 (4.49E+01)**
F24	1.31E+03 (2.60E+01)	**1.30E+03 (1.30E+01)**	F52	**2.89E+03 (1.25E+01)**	2.93E+03 (2.13E+01)
F25	**1.41E+03 (1.03E+01)**	1.42E+03 (8.86E+00)	F53	7.84E+03 (1.80E+03)	**6.46E+03 (6.20E+02)**
F26	**1.46E+03 (7.82E+01)**	1.48E+03 (8.69E+01)	F54	3.85E+03 (2.17E+02)	**3.34E+03 (5.66E+01)**
F27	**2.42E+03 (1.07E+02)**	2.58E+03 (8.02E+01)	F55	**3.18E+03 (3.60E+01)**	3.23E+03 (2.33E+01)
F28	**2.03E+03 (8.02E+02)**	3.63E+03 (1.45E+03)	F56	**4.40E+03 (3.39E+02)**	4.62E+03 (3.21E+02)
			F57	**5.16E+05 (2.90E+05)**	1.54E+06 (7.63E+05)

Table 4.18: Results obtained by the Wilcoxon signed-rank test for ASBSO vs. BSO with 1/5 Rule.

vs.	R^+	R^-	p-value	α=0.05	α=0.01
BSO with 1/5 Rule	1282.0	371.0	2.22E-4	YES	YES

Table 4.19: Experimental results of using ASBSO and SFMS on CEC'13 and CEC'17 benchmark functions (F1-F57).

	D=30				
	ASBSO	SFMS		ASBSO	SFMS
	Mean (Std Dev)	Mean (Std Dev)		Mean (Std Dev)	Mean (Std Dev)
F1	-1.40E+03 (1.98E-13)	-1.40E+03 (4.72E-13)	F29	**2.21E+03 (2.00E+03)**	3.13E+03 (2.72E+03)
F2	**1.54E+06 (4.26E+05)**	1.89E+06 (4.58E+05)	F30	**3.95E+02 (1.10E+02)**	4.03E+02 (1.18E+02)
F3	**8.47E+07 (8.64E+07)**	1.16E+08 (1.19E+08)	F31	**4.72E+02 (2.92E+01)**	4.98E+02 (2.58E+01)
F4	**5.08E+03 (2.25E+03)**	1.64E+04 (4.84E+03)	F32	**6.86E+02 (3.45E+01)**	6.94E+02 (3.34E+01)
F5	-1.00E+03 (2.98E-03)	-1.00E+03 (2.41E-03)	F33	**6.51E+02 (7.77E+00)**	6.54E+02 (7.62E+00)
F6	**-8.64E+02 (2.46E+01)**	-8.61E+02 (2.83E+01)	F34	**1.16E+03 (9.94E+01)**	1.17E+03 (9.13E+01)
F7	**-7.08E+02 (3.90E+01)**	-7.07E+02 (3.46E+01)	F35	**9.41E+02 (3.19E+01)**	9.43E+02 (2.14E+01)
F8	-6.79E+02 (6.70E-02)	-6.79E+02 (9.63E-02)	F36	**3.93E+03 (6.39E+02)**	4.05E+03 (7.25E+02)
F9	-5.71E+02 (3.23E+00)	-5.71E+02 (3.42E+00)	F37	**5.20E+03 (5.67E+02)**	5.28E+03 (6.97E+02)
F10	-5.00E+02 (5.35E-02)	-5.00E+02 (8.43E-02)	F38	1.23E+03 (4.75E+01)	1.23E+03 (5.01E+01)
F11	**-1.82E+02 (5.40E+01)**	2.34E+01 (7.94E+01)	F39	1.41E+06 (8.00E+05)	**1.35E+06 (7.99E+05)**
F12	**-7.64E+01 (4.85E+01)**	1.73E+02 (8.46E+01)	F40	5.04E+04 (2.64E+04)	**4.91E+04 (2.42E+04)**
F13	**1.30E+02 (6.54E+01)**	3.44E+02 (7.70E+01)	F41	**7.08E+03 (5.23E+03)**	8.35E+03 (7.67E+03)
F14	**3.68E+03 (4.56E+02)**	3.88E+03 (5.09E+02)	F42	3.01E+04 (2.25E+04)	**2.64E+04 (1.34E+04)**
F15	**3.88E+03 (5.74E+02)**	4.26E+03 (5.32E+02)	F43	**3.01E+03 (2.25E+02)**	3.10E+03 (3.99E+02)
F16	2.00E+02 (1.13E-01)	2.00E+02 (1.99E-01)	F44	2.40E+03 (2.44E+02)	**2.39E+03 (2.77E+02)**
F17	**5.28E+02 (5.13E+01)**	5.54E+02 (3.81E+01)	F45	**1.23E+05 (1.21E+05)**	1.35E+05 (8.48E+04)
F18	**5.98E+02 (2.85E+01)**	6.02E+02 (3.11E+01)	F46	**1.25E+05 (6.31E+04)**	1.32E+05 (5.56E+04)
F19	**5.04E+02 (7.60E-01)**	5.06E+02 (1.14E+00)	F47	**2.67E+03 (2.17E+02)**	2.73E+03 (2.13E+02)
F20	6.14E+02 (2.94E-01)	6.14E+02 (1.23E-01)	F48	**2.49E+03 (3.14E+01)**	2.50E+03 (4.34E+01)
F21	1.02E+03 (8.42E+01)	1.02E+03 (7.70E+01)	F49	**5.79E+03 (2.04E+03)**	6.11E+03 (1.70E+03)
F22	**5.36E+03 (4.70E+02)**	5.56E+03 (7.10E+02)	F50	**3.26E+03 (1.24E+02)**	3.28E+03 (1.01E+02)
F23	6.00E+03 (7.84E+02)	6.00E+03 (6.64E+02)	F51	**3.49E+03 (9.56E+01)**	3.51E+03 (1.25E+02)
F24	1.31E+03 (2.60E+01)	1.31E+03 (2.27E+01)	F52	2.89E+03 (1.25E+01)	2.89E+03 (7.17E+00)
F25	**1.41E+03 (1.03E+01)**	1.45E+03 (1.68E+01)	F53	7.84E+03 (1.80E+03)	**7.48E+03 (2.17E+03)**
F26	1.46E+03 (7.82E+01)	**1.44E+03 (7.15E+01)**	F54	3.85E+03 (2.17E+02)	**3.84E+03 (1.96E+02)**
F27	**2.42E+03 (1.07E+02)**	2.44E+03 (1.30E+02)	F55	**3.18E+03 (3.60E+01)**	3.19E+03 (4.05E+01)
F28	**2.03E+03 (8.02E+02)**	5.66E+03 (5.57E+02)	F56	**4.40E+03 (3.39E+02)**	4.47E+03 (3.37E+02)
			F57	**5.16E+05 (2.90E+05)**	5.31E+05 (2.89E+05)

Table 4.20: Results obtained by the Wilcoxon signed-rank test for IMS vs. SFMS.

vs.	R^+	R^-	p-value	α=0.05	α=0.01
SFMS	1343.5	309.5	2.0E-5	YES	YES

4.3.5 ASBSO vs. previous BSO variants

To further discuss the competitiveness of ASBSO, more comparisons between it and previous BSO variants should be executed. In this part, two BSO variants: BSO in objective space (BSOOS) [55] and global-best BSO (GBSO) [81] are tested on CEC'13 and 17 benchmark functions. The results are listed in Tables 4.21 and 4.22.

From the results, ASBSO shows a great advantage comparing with BSOOS, and can be competitive with GBSO. Although the p-value for ASBSO vs. GBSO is not less than 0.05, ASBSO still obtains a greater R^+ value, which indicates that it has a better overall performance than GBSO on total 57 test functions. Moreover, GBSO adopts multiple modifications, i.e., fitness-based grouping, per-variable updates, the global-best update and the re-initialization step, but ASBSO using fewer modifica-

Table 4.21: Experimental results of using ASBSO, BSOOS and GBSO on CEC'13 benchmark functions (F1-F28).

	ASBSO	BSOOS	GBSO
	Mean (Std Dev)	Mean (Std Dev)	Mean (Std Dev)
F1	-1.40E+03 (1.98E-13)	**-1.40E+03 (1.64E-13)**	-1.40E+03 (1.06E-10)
F2	**1.54E+06 (4.26E+05)**	1.67E+06 (6.03E+05)	2.08E+06 (4.53E+05)
F3	**8.47E+07 (8.64E+07)**	1.77E+08 (2.89E+08)	5.76E+08 (7.05E+08)
F4	5.08E+03 (2.25E+03)	3.31E+04 (9.12E+03)	**-1.04E+03 (3.33E+01)**
F5	-1.00E+03 (2.98E-03)	-1.00E+03 (1.72E-03)	**-1.00E+03 (8.05E-04)**
F6	**-8.64E+02 (2.46E+01)**	-8.62E+02 (2.76E+01)	-8.50E+02 (3.16E+01)
F7	**-7.08E+02 (3.90E+01)**	-6.82E+02 (6.08E+01)	-6.99E+02 (2.76E+01)
F8	**-6.79E+02 (6.70E-02)**	-6.79E+02 (7.04E-02)	-6.79E+02 (7.42E-02)
F9	**-5.71E+02 (3.23E+00)**	-5.69E+02 (3.36E+00)	-5.71E+02 (3.40E+00)
F10	**-5.00E+02 (5.35E-02)**	-5.00E+02 (1.39E-01)	-5.00E+02 (9.27E-02)
F11	**-1.82E+02 (5.40E+01)**	3.86E+01 (7.84E+01)	-1.48E+02 (5.90E+01)
F12	**-7.64E+01 (4.85E+01)**	1.42E+02 (7.87E+01)	-6.60E+01 (6.99E+01)
F13	1.30E+02 (6.54E+01)	3.74E+02 (1.10E+02)	**7.53E+01 (5.32E+01)**
F14	**3.68E+03 (4.56E+02)**	4.19E+03 (5.57E+02)	3.98E+03 (5.46E+02)
F15	**3.88E+03 (5.74E+02)**	4.23E+03 (5.11E+02)	4.22E+03 (6.66E+02)
F16	2.00E+02 (1.13E-01)	**2.00E+02 (3.26E-02)**	2.01E+02 (2.31E-01)
F17	5.28E+02 (5.13E+01)	5.78E+02 (4.43E+01)	**4.12E+02 (2.11E+01)**
F18	5.98E+02 (2.85E+01)	5.99E+02 (3.31E+01)	**5.09E+02 (1.96E+01)**
F19	**5.04E+02 (7.60E-01)**	5.05E+02 (7.76E-01)	5.07E+02 (1.72E+00)
F20	**6.14E+02 (2.94E-01)**	6.15E+02 (3.16E-01)	6.14E+02 (6.14E+02)
F21	1.02E+03 (8.42E+01)	1.05E+03 (8.37E+01)	**1.02E+03 (7.70E+01)**
F22	**5.36E+03 (4.70E+02)**	5.84E+03 (9.16E+02)	5.89E+03 (8.82E+02)
F23	**6.00E+03 (7.84E+02)**	6.30E+03 (6.82E+02)	6.21E+03 (9.41E+02)
F24	1.31E+03 (2.60E+01)	1.35E+03 (3.49E+01)	**1.29E+03 (8.55E+00)**
F25	1.41E+03 (1.03E+01)	1.45E+03 (2.30E+01)	**1.40E+03 (1.40E+03)**
F26	1.46E+03 (7.82E+01)	1.54E+03 (7.40E+01)	**1.45E+03 (8.19E+01)**
F27	2.42E+03 (1.07E+02)	2.53E+03 (1.08E+02)	**2.39E+03 (1.07E+02)**
F28	**2.03E+03 (8.02E+02)**	5.83E+03 (5.81E+02)	2.11E+03 (1.00E+03)

Table 4.22: Experimental results of using ASBSO, BSOOS and GBSO on CEC'17 benchmark functions (F29-F57).

	ASBSO	BSOOS	GBSO
	Mean (Std Dev)	Mean (Std Dev)	Mean (Std Dev)
F29	2.21E+03 (2.00E+03)	**1.96E+03 (1.57E+03)**	3.31E+03 (4.07E+03)
F30	3.95E+02 (1.10E+02)	8.78E+03 (3.04E+03)	**3.69E+02 (5.23E+02)**
F31	4.72E+02 (2.92E+01)	**4.66E+02 (2.29E+01)**	4.76E+02 (1.19E+01)
F32	6.86E+02 (3.45E+01)	**6.82E+02 (2.88E+01)**	6.92E+02 (2.95E+01)
F33	6.51E+02 (7.77E+00)	6.50E+02 (5.86E+00)	**6.46E+02 (7.49E+00)**
F34	1.16E+03 (9.94E+01)	1.12E+03 (6.90E+01)	**8.54E+02 (4.02E+01)**
F35	9.41E+02 (3.19E+01)	**9.37E+02 (2.95E+01)**	9.47E+02 (3.06E+01)
F36	3.93E+03 (6.39E+02)	3.74E+03 (4.78E+02)	**2.63E+03 (1.02E+03)**
F37	**5.20E+03 (5.67E+02)**	5.20E+03 (8.49E+02)	5.26E+03 (5.85E+02)
F38	1.23E+03 (4.75E+01)	**1.23E+03 (4.18E+01)**	1.26E+03 (6.49E+01)
F39	**1.41E+06 (8.00E+05)**	1.88E+06 (1.30E+06)	3.56E+06 (2.70E+06)
F40	**5.04E+04 (2.64E+04)**	5.84E+04 (3.64E+04)	8.29E+04 (6.33E+04)
F41	7.08E+03 (5.23E+03)	9.92E+03 (8.88E+03)	**6.09E+03 (4.28E+03)**
F42	**3.01E+04 (2.25E+04)**	3.23E+04 (1.81E+04)	4.94E+04 (3.19E+04)
F43	3.01E+03 (2.25E+02)	3.10E+03 (2.70E+02)	**2.95E+03 (2.73E+02)**
F44	**2.40E+03 (2.44E+02)**	2.42E+03 (2.96E+02)	2.41E+03 (2.18E+02)
F45	**1.23E+05 (1.21E+05)**	1.51E+05 (1.21E+05)	1.61E+05 (1.20E+05)
F46	1.25E+05 (6.31E+04)	**1.15E+05 (4.55E+04)**	4.66E+05 (1.66E+05)
F47	**2.67E+03 (2.17E+02)**	2.72E+03 (2.10E+02)	2.70E+03 (1.20E+02)
F48	**2.49E+03 (3.14E+01)**	2.51E+03 (3.67E+01)	2.50E+03 (2.14E+01)
F49	5.79E+03 (2.04E+03)	6.42E+03 (1.54E+03)	**4.17E+03 (2.19E+03)**
F50	3.26E+03 (1.24E+02)	3.31E+03 (9.91E+01)	**3.03E+03 (9.78E+01)**
F51	3.49E+03 (9.56E+01)	3.47E+03 (2.09E+02)	**3.14E+03 (1.04E+02)**
F52	2.89E+03 (1.25E+01)	**2.88E+03 (8.40E+00)**	2.90E+03 (2.81E+01)
F53	7.84E+03 (1.80E+03)	7.65E+03 (1.89E+03)	**5.99E+03 (1.60E+03)**
F54	3.85E+03 (2.17E+02)	3.86E+03 (2.64E+02)	**3.25E+03 (8.17E+01)**
F55	**3.18E+03 (3.60E+01)**	3.21E+03 (1.35E+01)	3.22E+03 (2.58E+01)
F56	4.40E+03 (3.39E+02)	**4.37E+03 (2.71E+02)**	4.41E+03 (3.32E+02)
F57	**5.16E+05 (2.90E+05)**	7.73E+05 (4.79E+05)	1.36E+06 (7.50E+05)

tions obtains competitive results, which could be regarded as a successful variant of BSO.

Table 4.23: Results obtained by the Wilcoxon test for algorithm ASBSO vs. BSOOS and GBSO.

Algorithms	R^+	R^-	p-value	α=0.05	α=0.01
ASBSO vs.					
BSOOS	1292.5	303.5	0.000031	YES	YES
GBSO	961.0	635.0	0.163962	NO	NO

4.4 Discussion

As shown fully detailed in Section 5.3, our proposed ASBSO outperforms traditional BSO and other meta-heuristic optimization algorithms. Especially in comparison with MABC and CGSA-M which also implement memory-based selection mechanism, ASBSO obtains much better results in solution accuracy. It is interpreted in Section 5.2 that ASBSO has two main novelties: first, it adapts several step length update methods to deal with different situations; second, these methods are adaptively selected via a new memory storing mechanism. In this section, we will further discuss the effectiveness of these two modifications by comparing them with the classical 1/5 success rule used in evolutionary strategy (ES) [82] and SFMS used in [22, 66], respectively. These tests are executed at $D = 30$ with maximum number of function evaluation equals $10000D$ for 30 runs.

4.4.1 Comparison with 1/5 Success Rule

1/5 success rule is a parameter adaptive strategy proposed by Rechenberg [82] which is used to adjust deviation δ in order to make mutational step size be dynamically adapted according to the search performance.

The offspring generation equation can be exhibited as follow:

$$X_{offspring} = X + N(0, \delta(t)) \tag{4.6}$$

where X is the parent and $X_{offspring}$ is the offspring. It is generated by adding a Gaussian noise $N(0, \delta(t))$ of which mean value equals 0 and deviation $\delta(t)$ changes according to iteration t.

Its variation equation can be shown as:

$$\delta(t+1) = \begin{cases} \frac{\delta(t)}{r} & if \quad s_r > 0.2 \\ \delta(t) * r & if \quad s_r < 0.2 \\ \delta(t) & if \quad s_r = 0.2 \end{cases} \tag{4.7}$$

where r is a scale factor that is usually set in interval $[0.85, 0.99]$, and s_r is a success rate to represent the rate that mutation procedure successfully generates a better offspring in a certain period. If the success rate s_r is larger than 0.2, deviation δ will increase; in the opposite, if s_r is smaller than 0.2, δ will decrease. As an adaptive mechanism, it makes algorithm can adjust its search radius to be suitable for specific problems and different search periods. Not only in ES, but also in some other newly proposed algorithms, such as negatively correlated search proposed by Tang et al. [83], 1/5 success rule has exhibited a great performance in search ability. Thus, we combine BSO with 1/5 success rule to conduct a contrast experiment to assess the effectiveness of ASBSO.

Table 4.17 lists the experimental results between ASBSO and BSO with 1/5 success rule on 57 test functions. It is obvious that although 1/5 success rule can obtain better solutions on a few problems, ASBSO still dominates most number of the problems. Table 4.18 shows the Wilcoxon statistical analysis result between ASBSO and BSO with 1/5 success rule, where ASBSO is the control algorithm. p-value that is smaller than significant level $\alpha = 0.01$ demonstrates that the multiple step length update method proposed in ASBSO can provide more adaptive and suitable search mechanisms than the 1/5 success rule to be applied to various problems.

4.4.2 IMS vs. SFMS

The second modification of the proposed method is that a new memory storing mechanism IMS replaces the traditional memory mechanism (SFMS). Both mechanisms are introduced in Section 5.2 and it is necessary to discuss whether the former can provide a better search efficiency than the latter. Hence, a comparison between ASBSO and the BSO with adaptive step length based on SFMS is conducted and the results are listed in Table 4.19. Visually, ASBSO maintains most better results especially on CEC'13. Table 4.20 also can prove that IMS is significantly better than SFMS.

4.4.3 Computational Complexity

ASBSO has shown a superior ability for a majority of benchmark functions. In this subsection, we calculate its computational time complexity together with BSO's.

The time complexity in each procedure of BSO is described as follows:

(1) In BSO, the time complexity for initializing is $O(N)$ where N is the population size.

(2) Evaluating the fitness of population is $O(N)$.

(3) Using K-means to divide the population into c clusters needs $O(cN^2)$.

(4) The process of individual selection and step length generation both cost $O(N^2)$.

(5) The generation of new individuals and the fitness calculation need $O(N^2)$, respectively.

Thus, the overall time complexity of BSO is

$$O(N) + O(N) + O(cN^2) + O(N^2) + O(N^2) = 2O(N^2) + O(cN^2) + 2O(N) \quad (4.8)$$

To be simplified, its overall time complexity is $O(N^2)$.

ASBSO is modified based on BSO. Its procedure is shown as:

(1) The initialization needs $O(N)$.

(2) Evaluating the fitness of population is $O(N)$.

(3) Using K-means to divide the population into c clusters needs $O(cN^2)$.

(4) Generate multiple step lengths needs $O(4N^2)$.

(5) The memory selection costs $O(N)$.

(6) The generation of new individuals and the fitness calculation need $O(N^2)$, respectively.

Thus, the overall time complexity of ASBSO is

$$\begin{aligned} &O(N) + O(N) + O(cN^2) + O(4N^2) + O(N) + O(N^2) \\ &\qquad = O(cN^2) + O(4N^2) + O(N^2) + 3O(N) \end{aligned} \tag{4.9}$$

The overall time complexity of ASBSO can be seen as $O(N^2)$. The main differences between ASBSO and BSO are in Steps (4) and (5). As ASBSO applies multiple step length strategies, it costs $O(4N^2)$ which is greater than $O(N^2)$ of BSO, and the memory selection needs $O(N)$. Thus, ASBSO and BSO have the same time complexity, which indicates that both are competitive in computational efficiency.

Chapter 5

A Multiple Diversity-driven Brain Storm Optimization Algorithm with Adaptive Parameters

5.1 Introduction

In recent years, various swarm intelligence (SI) algorithms have been proposed for solving diverse optimization problems. The main property of this kind of algorithms is that they mimic the social behaviors of nature creatures. As far as we know, it is full of wisdom and intelligence when animals are hunting, foraging and navigating in nature. Survival instincts drive them to improve search ability for creating more suitable living environment. Their behaviors gradually arouse great interests among researchers in the field of artificial intelligence [1]. Particle swarm optimization (PSO) which is one of the most popular SI algorithms is modeled based on the social behaviors of flocks of birds and schools of fish [2]. It supposes that a swarm of particles fly randomly in a multidimensional search space. Each of them represents a candidate solution for the optimization task. Their trajectories change according to the best position of the individual and the global best position of the whole population. Particles can effectively search for better solutions by taking advantage of this mechanism.

In addition to PSO, more and more SI optimization algorithms progressively spring into our view. Ant colony optimization (ACO) [12], fireworks algorithm (FA) [13], gravitational search algorithm (GSA) [14], artificial bee colony algorithm (ABC) [15]

and brain storm optimization (BSO) [16] are some powerful optimization algorithms. These SI algorithms can be roughly divided into three categories according to the types of behaviors they take inspiration from.

The first category is called bio-inspired. Classical algorithms in this category such as ACO and ABC emulate the foraging behaviors of ant colony and bee colony, respectively. In ACO, individuals utilize a special chemical substance called pheromone to mark their search trajectory. The trajectory with more pheromone is considered as a preferred path to the global optimum, and further attracts other individuals [11]. ABC simulates the organizational structure of bees to categorize individuals into three groups: employed artificial bees, onlookers and scouts. The employed artificial bees represent candidate solutions and the onlookers are responsible for sharing the information of employed bees. After these steps, scouts are sent to diverse search area for discovering new solutions. This sophisticated idea of giving different functions to individuals makes the search procedure of ABC efficient and effective [15].

The second category can be named as physics-inspired. The algorithms belong to this category such as FA and GSA straightly take inspiration from physical phenomena or laws. For examples, the explosion processes of fireworks are utilized to design the search mechanism of FA, in which the distribution of individuals is analogized by the sparks in firework explosion. In GSA, the law of gravity is used to depict the relationship among individuals in search space. They are attracted by each other and the gravitational force is directly proportional to their fitness and inversely proportional to the square of the distance between them. The performance of GSA in different kinds of problems implies its powerful search ability [20, 21, 22].

The last category is called sociology-inspired. The major property of the algorithms in this category is that they are inspired by human social behaviors. BSO is very notable among SI algorithms and has already achieved great success in various applications [23]. Its operations of generating new individuals adopt the brainstorming process in human social behaviors. In reality, a group of people should be called together to figure out a solution when we encounter problems that can not be solved alone. This brainstorming process needs repetitive discussions and debates. BSO is

enlightened by this feature and obtains an elaborated search process. At the rudimentary stage of optimization, individuals are divided into multiple clusters, then each cluster selects the best individual as the center. BSO has four independent individual generation methods and the selections of corresponding method are depending on three preset parameters p_1, p_2 and p_3. p_1 decides the usage of one or two clusters. In the condition of using one cluster, p_2 is adopted to choose the center or one random individual in the selected cluster. Otherwise, when two clusters are selected, p_3 determines the adoption of two centers or two random individuals. Being beneficial from this sophisticated selection mechanism, BSO can avoid sticking into local optima and outperform other optimization algorithms when dealing with multimodal problems [23]. However, the inherent feature of BSO that can not maintain good diversity reduces its robustness and deteriorates the performance of solving different problems. In the meanwhile, the parameter adjustment is very important in designing algorithms but it generally costs much time to find an acceptable parameter set. Therefore, more and more researchers prefer making parameters adaptive or self-adaptive to enhance the robustness and performance of algorithms [24, 25, 26, 27, 28].

Many modifications have been facilitated to improve the optimization performance of BSO but little work tries to make parameters be adaptive and keep the diversity staying in a high stage. BSO in objective space (BSO-OS) [55] aims to accelerate its convergent speed by replacing k-means clustering method with an elitist selection mechanism. Its mutation operation focuses on one-dimension objective space instead of the whole solution space. In [84], a random grouping BSO (RGBSO) is proposed to balance exploration and exploitation via adopting a new dynamic parameter in the generation of step-size. Besides, it replaces k-means clustering by a random grouping strategy so that the time complexity is decreased. Global-best BSO (GBSO) [81] tries to improve the performance of BSO from multiple aspects, including the clustering method, individual selection and mutation. Different from the k-means and mentioned random grouping methods, GBSO ranks the population according to their fitness and makes good and bad individuals equally distribute in different clusters. In original BSO, at most two individuals participate in generating new individuals,

while in GBSO, more individuals can contribute to enhance the information exchange in this step. GBSO also adopts the global-best guidance strategy in PSO to modify its mutation mechanism. In our previous work [57], a chaotic local search method is combined with BSO (CBSO) to enhance its search ability and improve the solution quality. Besides the mentioned works, there are many other effective modifications for BSO, In [50], a self-adaptive multiobjective BSO (SMOBSO) is proposed. It adopts an adaptive mutation method to give an uneven distribution of solutions, but parameters still need to be set according to empirical data. Similarly, other works [85, 86] mainly focus on the adaptations of search step length in the mutation operator.

Table 5.1: The main parameters in BSO and MDBSO.

BSO	*Parameters*	n	p_0	p_1	p_2	p_3	k	μ	δ
	Values	5	0.2	0.8	0.4	0.5	20	1	0.5
MDBSO	*Parameters*	n	μ						
	Values	5	0.5						

Overall, most existing works that improve the performance of BSO focus on the adjustments of search and mutation strategies, but as we emphasized before, one drawback of BSO is that it has too many user-defined parameters. Presetting these parameters is a nontrivial task and generally difficult to find the best parameter set for solving different problems. Table 5.1 lists the main parameters of BSO and their corresponding values. It's widely accepted that the variation in parameter values of an algorithm could cause considerable fluctuation in performance [87]. Taking differential evolution (DE) as an example [40], the number of control parameters in DE is very few, including the scaling factor F, crossover rate CR and population size NP. The effects of these parameters on the performance of DE are well studied and it is reported that different value set for F and CR could obtain significant performance variations [88]. The most successful modifications for DE, such as JADE [24] and SHADE [89], employ parameter adaptation strategy to automatically update the control parameters. Besides, some researches indicate that diversity plays a significant role in improving search performance of SI algorithms [47, 90, 91].

In the design of optimization algorithms, the balance between exploration and ex-

ploitation is a crucial factor for the search performance. A good balance can make the algorithms fast converge and avoid local optimal solutions. Contrarily, the solution quality could be badly deteriorated when the relation is unbalanced. Therefore, the researches about keeping the balance between exploration and exploitation become crucial in recent years [92, 93, 94]. The key point of keeping balance is the preservation of population diversity in optimization process [91]. The population diversity can be explained as the extent of variation in the population based on the distribution or fitness performance obtained by individuals [95]. There are various methods that can be used to calculate the population diversity [47, 91]. The diversity is named as distance-based when it is measured according to the distance between each individual in decision space. While the fitness-based diversity is obtained by evaluating the performance of individuals in the objective space.

Both kinds of diversity have been incorporated into other techniques to improve the performance of corresponding algorithms. In [92, 93], the distance-based diversity is considered as an explicit objective. In other words, diversity and fitness are combined as a multi-objective problem to be solved. In this way, the balance between exploration and exploitation can be well maintained by searching for Pareto optimal solutions. The experimental results [92, 93] also demonstrate controlling diversity can evidently improve the performance of algorithm. With regard to the fitness-based diversity, it is mainly used to obtain good fitness spread among individual solutions. In [96], a variable relocation technique based on fitness diversity is applied to make the converged population restart convergence from another promising location. A fast adaptive memetic algorithm is proposed in [97] and the fitness diversity is investigated to control the utilization of local search strategies. Other techniques such as fitness sharing [98] and adaptive grid [99] also apply fitness diversity to improve the performance of algorithms [91]. Motivated by these prior studies, it can be expected to make the parameters adaptive via diversity control. Therefore, a multiple diversity-driven BSO (MDBSO) with well-balanced diversity and adaptive parameters is proposed.

The main contributions of this study can be summarized as: (1) We make the first attempt to use both distance-based diversity D_d and fitness-based diversity D_f

to control the mutation process to the best of our knowledge. (2) Two new mutation strategies are adopted, including a local search strategy called BLX-α [65, 100] and a Gaussian mutation strategy. Additionally, D_f is utilized to adjust the standard deviation δ in Gaussian distribution. (3) Both D_d and D_f participate in generating new individuals. (4) Extensive experiments are conducted to verify the performance of MDBSO based on CEC2017 [64] benchmark function test suit and a neuron model training task [18, 101]. The results indicate that MDBSO has much better search ability than its peers.

The organization of this paper is arranged as follows. Section 3.2 briefly introduces the BSO. The proposed MDBSO is presented in Section 5.2. In Section 5.3, the experimental results of benchmark function suit and neuron model training data set are reported to show the performance of MDBSO in comparison with other SI algorithms. Some discussions are given in Section 5.4. Finally, we conclude this paper in Section 3.5.

5.2 Multiple Diversity-driven BSO (MDBSO)

5.2.1 Diversity-driven Strategy

Although the distance-based diversity and fitness-based diversity are investigated in some researches, they are for the first time to be studied simultaneously as control parameters in this study. Before introducing the specific roles of two kinds of diversity in MDBSO, their formulas are given as follows.

$$D_d^j = \frac{1}{N_j}\sqrt{\sum_{i=1}^{N_j}(||X_i^j - X_{center}^j||)^2} \tag{5.1}$$

where j ($j = 1, 2, ..., n$) refers to the cluster number and D_d^j is the distance-based diversity of the jth cluster. N_j is the number of individuals in the jth cluster, X_i^j and X_{center}^j are the ith individual and the center in the current cluster, respectively.

The fitness-based diversity D_f is calculated as

$$D_f^j = \frac{1}{N_j}\sqrt{\sum_{i=1}^{N_j}(||F_i^j - F_{center}^j||)^2} \tag{5.2}$$

where F_i^j and F_{center}^j are the fitness of the ith individual and the center in the jth cluster, respectively. It should be noticed that we choose the centers and their fitness values instead of using the mean values as the subtrahends to calculate corresponding diversities. The intention is to increase convergence rate during optimization process as the centers are the best individuals in the population.

In MDBSO, D_d is applied as a control parameter to replace p_1, p_2 and p_3. It is adaptive via a normalization operation and the formula is shown in Eq. (5.3).

$$p_d^j(t) = \frac{D_d^j(t) - min\{D_d(t)\}}{max\{D_d(t)\} - min\{D_d(t)\}} \tag{5.3}$$

where $p_d^j(t)$ decides which mutation strategy is called to generate new individuals in the jth cluster at the tth iteration. $max(D_d(t))$ and $min(D_d(t))$ refer to the maximum and minimum values of D_d of n clusters at tth iteration, respectively. It's obvious that p_d^j values in the interval of $[0, 1]$, and it controls a switch between two mutation strategies: BLX-α and Gaussian mutation. If a random value generated in $(0, 1)$ is smaller than $p_d^j(t)$, it indicates the jth cluster may have a good distance diversity. Therefore, a local search method BLX-α is applied to speed up its convergence. Conversely, if the random value is greater than $p_d^j(t)$, the bad distance diversity in the jth cluster may eventually deteriorate solution quality and cause a premature convergence. Thus, the function of the Gaussian mutation is used to improve the distance diversity.

5.2.2 Mutation Strategies

5.2.2.1 BLX-α

BLX-α is a local search operator to adjust the population density [102]. Firstly, two individuals $X_1 = (x_1^1 ... x_1^{dim})$ and $X_2 = (x_2^1 ... x_2^{dim})$ are selected (dim is the dimension number). Then, a new individual is generated from the interval of $[min\{X_1, X_2\} - Y \times \alpha, max\{X_1, X_2\} + Y \times \alpha]$, where $Y = max\{X_1, X_2\} - min\{X_1, X_2\}$. α is a control parameter used to limit the search space. According to [100], BLX-α can increase the distribution of individuals when $\alpha > \frac{\sqrt{3}-1}{2}$, otherwise the distribution will be decreased. In particular, BLX-0 makes the variance of the distribution decrease and reduces the distance diversity. Therefore, we use BLX-0 in MDBSO because it is a local search operator that can improve solutions' quality when a cluster maintains a good distance diversity.

Particularly, the individuals X_1 and X_2 are selected via a novel mechanism, in which the centers of the top two clusters with the highest fitness diversity D_f are specified as X_1 and X_2, respectively. The reasons we use the fitness diversity instead of distance diversity here are as follows: (1) Even if the individuals have a close distance, their fitness can vary widely as they are in different peaks of a multimodal problem. Thus, fitness diversity is more suitable for selections of mutation operators. (2) High fitness diversity means that the cluster manages good fitness spread among individual solutions. Thus, it can avoid premature convergence to a great extent. (3) Meanwhile, centers are the best individuals in the population. They are of strong reliability and promising to enable BLX-α to generate individuals with good fitness. The formula of generating individuals is shown in Eq. (5.4)

$$X_{generated} = rand \times (max\{X_1, X_2\} - min\{X_1, X_2\}) \tag{5.4}$$

where $rand$ is a random value generated in $(0, 1)$.

5.2.2.2 Gaussian Mutation

BLX-α is applied for the situation that the cluster stays in a good diversity. But it is not capable of improving diversity when the solution quality is poor. Therefore, we use a mutation strategy to generate new individuals when the distance diversity is relative low. In this part, the adopted Gaussian mutation is presented in details. A common formula which uses Gaussian distribution to generate new individuals can be described as follows.

$$X_{generated} = X_i + N(\mu, \delta) \cdot (X_{selected1} - X_{selected2}) \tag{5.5}$$

where X_i is the ith individual to be updated in the population. In MDBSO, $X_{selected1}$ and $X_{selected2}$ are two randomly selected individuals in the top two clusters with the highest distance diversity D_d, respectively. It should be pointed out that, different from the utilization of D_f in BLX-α, we use D_d because we want to increase the distance diversity here. Moreover, δ is adaptive in MDBSO and it is controlled by D_f. The adaptation mechanism is given as follows.

$$\delta = \frac{1}{e^{\omega}} \tag{5.6}$$

where e is the base of natural logarithm and ω is calculated according to Eq. (5.7).

$$\omega = |\frac{D_f^j(t) - mean\{D_f(t)\}}{max\{D_f(t)\} - min\{D_f(t)\}}| \tag{5.7}$$

where $mean\{D_f(t)\} = \frac{1}{n}\sum_{j=1}^{n} D_f^j(t)$.

It is worth emphasizing that we use $mean\{D_f(t)\}$ as the subtrahend to control ω. It is clear that ω and δ are negatively correlated, which means that the clusters with higher (or lower) D_f obtain smaller (or bigger) δ. Generally, an individual in the cluster with poor population diversity may need greater δ to provide a larger search step size in the aim of increase diversity. But too high diversity could cause the algorithm fail to converge. A contrast experiment is conducted, in which we use $min\{D_f(t)\}$ instead of $mean\{D_f(t)\}$ as the subtrahend. In this way, the cluster with the lowest

fitness diversity would generate new individuals with $N(\mu, 1)$ and it's predictable that this method could obtain considerable population diversity due to the increase in δ. However, its optimization result is not as well as it of using $mean\{D_f(t)\}$. The reason is that too high population diversity undermines the performance of algorithm. Therefore, a moderate value is more suitable for not only maintaining population diversity, but also obtaining good results.

5.2.3 MDBSO

The structure of BSO is simplified by replacing its parameters with $p_d^j(t)$. We can find the number of parameters are substantially reduced due to the proposal of adaptive parameters, as shown in Table 5.1. The number of clusters stays the same as 5 in BSO and μ is set to 0.5. Regarding the values of these two parameters, some discussions are given in Section 5.4.

The primary procedures of MDBSO is presented in Algorithm 4. In the first step, MDBSO randomly generates N individuals and calculates their fitness. If the termination is not satisfied, k-means is applied to divide the population into n clusters. The best individual in each cluster is selected as the *center*. Then, MDBSO has its specific step in which the distance diversity D_d and fitness diversity D_f of each cluster are calculated. The p_d^j calculated by D_d decides the selection of mutation strategies for each cluster. The $BLX - \alpha$ strategy would have a high possibility to be applied to generate individuals in the cluster with good distance diversity. In the opposite, the Gaussian mutation strategy is utilized by the cluster with bad distance diversity. The new generated individual with better fitness will replace the old at the end of each iteration.

Fig. 5.1 illustrates the functions of D_d and D_f in the specific steps in MDBSO. Each of them has very important role in the generation of new solutions and would be used for more than once. In most literature, population diversity is usually applied as an optimization objective rather than an approach. They focus on the maintenance of population diversity but it does not participate in the search process. Innovatively,

Algorithm 4: Flowchart of MDBSO.

```
Randomly generate a population with N individuals;
Calculate the fitness of each individual;
while maximum number of function evaluations is not reached do
    Use k-means to divide N individuals into n clusters;
    Choose the best individual in each cluster as the center;
    Calculate the D_d and D_f of each cluster;
    for the individual in the jth cluster do
        if random(0,1) < p_d^j then
            use BLX-α strategy to generate new individuals in the jth cluster;
        else
            use Gaussian mutation strategy to generate new individuals in the
             jth cluster;
        end
    end
    if the new individual is better than the old one then
        replace the old individual
    end
end
```

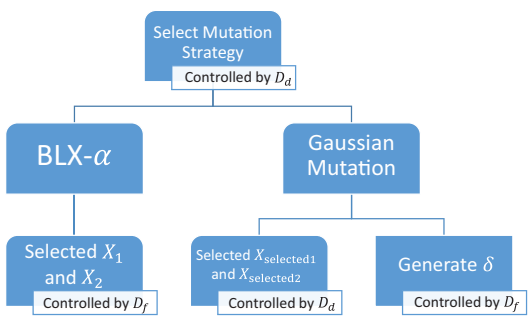

Figure 5.1: The functions of distance diversity and fitness diversity in MDBSO.

in this study, the distance and fitness diversity are simultaneously utilized and have been proven to be very effective in enhancing the performance of BSO.

Compared with the original BSO presented in Algorithm 3, MDBSO has essential modifications in two aspects. One is that the diversity in BSO is well maintained so that the search ability is enhanced. The other is the adaptations of parameters. Most steps in MDBSO are controlled by adaptive parameters, which enhances its robustness and makes it can be applied into more diverse application scenarios.

5.3 Experimental Results

5.3.1 Benchmark Function Test Suit

In this section, CEC2017 benchmark function suit is implemented to test the performance of MDBSO. It should be noticed that F2 in CEC2017 has been excluded because it shows unstable behavior especially for higher dimensions, and significant performance variations for the same algorithm implemented in Matlab and C [64]. This benchmark function suit includes 2 unimodal, 7 simple unimodal, 10 hybrid and 10 composition functions. Hence, it is very suitable for testing the search ability and robustness of optimization algorithms. The population size N is 100, and the dimension for the problems dim is 30. Each problem is run for 30 times to reduce random errors. The maximum number of function evaluations (MFE) is 10000*dim. All experiments are implemented on a PC with 3.10GHz Intel(R) Core(TM) i5-4440 CPU and 8GB of RAM using MATLAB R2013b. All parameters for the contrast SI algorithms are set up according to the values provided in the corresponding literature.

5.3.1.1 MDBSO *vs.* BSO Variants

In this part, MDBSO is compared with BSO [16] and its variants, including CBSO [57], BSO-OS [55], RGBSO [84], GBSO [81] and ASBSO [85]. The results including mean and standard deviation (Std Dev) are listed in Tables 5.2 and 5.3. The values in boldface represent the best results among compared algorithms.

We can intuitively find that MDBSO obtains much more number of the best results in comparison with other competitors from these tables. It should be emphasized that MDBSO outperforms BSO on hybrid and composition functions (F11-F30) except for F14, suggesting that the drawback of BSO's poor robustness is mitigated and the search ability of BSO is greatly improved via diversity controlled parameters. A non-parametric statistical analysis called Friedman test is employed to give the ranking that each algorithm obtained in the current comparison [45]. The lower ranking indicates the better performance. As observed, MDBSO is the algorithm

with the best performance among the compared BSO variations. To more precisely analyze its performance, a non-parametric statistical test called Wilcoxon rank-sum test is implemented [76]. Each $+/\approx/-$ indicates the performance of MDBSO is significantly better (+), not significantly better and worse ($\approx$) or worse ($-$) than its peers. According to the statistical results, the number of times MDBSO wins to others is 19 (BSO), 21 (CBSO), 20 (BSO-OS), 19 (RGBSO), 17 (GBSO) and 19 (ASBSO) out of 29 tested problems, respectively. Moreover, there are at most six problems where MDBSO underperforms another algorithm (i.e. GBSO). Considering the tested algorithms are state-of-the-art BSO variations, MDBSO has verified its superior and it executes an effective search process by the adaptive parameter system.

In addition, box-and-whisker diagrams and convergence graphs are given in Fig. 5.2 and Fig. 5.3 to directly exhibit the difference in performance between MDBSO and its peers, respectively. The box-and-whisker diagrams can illustrate the quality of solutions on 30 runs. There are five values are conventionally used: the extremes, the upper and lower hinges (quartiles), and the median. The interval between the upper and lower hinges of the box is called interquartile range (IQR) and it indicates the degree of dispersion and skewness in the results. Symbol + indicates the outliers. As observed in Fig. 5.2, MDBSO obtains the best performance on F5, F12, F13, F23, F24, and F26. Fig. 5.3 depicts the convergent performance during the whole search procedure. The horizontal axis represents the number of function evaluations, and the vertical axis denotes the average values of optimization results on 30 runs. It's obvious that the convergence speed of MDBSO is much faster than its peers. In addition, it generally obtains better results. Although RGBSO shows an ability of avoiding premature on F13, F23, and F26, its slow convergent speed deteriorates the solutions' quality. The good performance of MDBSO profits from the diversity controlled search mechanism which keeps the balance between exploration and exploitation.

Moreover, to show the population diversity obtained by each contrast algorithm graphically, four plots on F12, F13, F23, and F24 are illustrated in Fig. 5.4, respectively. The calculation of population diversity is shown in Eq. (5.8)

Table 5.2: Experimental results of MDBSO versus BSO variants on CEC'17 benchmark functions (1).

Fun.	MDBSO Mean ±Std Dev	BSO Mean ±Std Dev	CBSO Mean ±Std Dev	BSO-OS Mean ±Std Dev
F1	3.17E+03 ± 4.68E+03	2.47E+03 ± 1.95E+03 ≈	3.99E+03 ± 3.07E+03 +	**1.96E+03 ± 1.57E+03** ≈
F3	6.13E+02 ± 6.28E+02	5.34E+02 ± 2.66E+02 ≈	**3.03E+02 ± 3.56E+00** −	8.78E+03 ± 3.04E+03 +
F4	4.62E+02 ± 3.28E+01	4.67E+02 ± 2.19E+01 ≈	4.94E+02 ± 2.04E+01 +	4.66E+02 ± 2.29E+01 ≈
F5	**6.02E+02 ± 3.10E+01**	6.87E+02 ± 4.05E+01 +	6.98E+02 ± 3.91E+01 +	6.82E+02 ± 2.88E+01 +
F6	6.13E+02 ± 7.18E+00	6.52E+02 ± 7.01E+00 +	6.47E+02 ± 7.67E+00 +	6.50E+02 ± 5.86E+00 +
F7	8.88E+02 ± 6.43E+01	1.15E+03 ± 9.65E+01 +	1.13E+03 ± 8.76E+01 +	1.12E+03 ± 6.90E+01 +
F8	9.13E+02 ± 3.75E+01	9.47E+02 ± 2.82E+01 +	9.41E+02 ± 2.57E+01 +	9.37E+02 ± 2.95E+01 +
F9	1.31E+03 ± 5.38E+02	3.98E+03 ± 6.95E+02 +	3.87E+03 ± 6.65E+02 +	3.74E+03 ± 4.78E+02 +
F10	7.37E+03 ± 1.29E+03	5.30E+03 ± 5.16E+02 −	5.37E+03 ± 5.82E+02 −	5.20E+03 ± 8.49E+02 −
F11	**1.21E+03 ± 5.06E+01**	1.23E+03 ± 4.05E+01 ≈	1.23E+03 ± 4.31E+01 ≈	1.23E+03 ± 4.18E+01 ≈
F12	**4.39E+04 ± 2.26E+04**	1.77E+06 ± 1.25E+06 +	1.91E+06 ± 1.36E+06 +	1.88E+06 ± 1.30E+06 +
F13	**1.46E+04 ± 1.61E+04**	5.36E+04 ± 2.85E+04 +	5.82E+04 ± 4.12E+04 +	5.84E+04 ± 3.64E+04 +
F14	7.71E+03 ± 5.29E+03	6.40E+03 ± 4.66E+03 ≈	3.51E+03 ± 2.20E+03 −	9.92E+03 ± 8.88E+03 ≈
F15	**7.51E+03 ± 8.01E+03**	2.95E+04 ± 1.61E+04 +	2.97E+04 ± 1.67E+04 +	3.23E+04 ± 1.81E+04 +
F16	**2.44E+03 ± 4.43E+02**	3.20E+03 ± 4.24E+02 +	2.83E+03 ± 2.73E+02 +	3.10E+03 ± 2.70E+02 +
F17	**1.98E+03 ± 1.94E+02**	2.48E+03 ± 2.56E+02 +	2.20E+03 ± 2.10E+02 +	2.42E+03 ± 2.96E+02 +
F18	9.28E+04 ± 5.38E+04	1.21E+05 ± 1.03E+05 ≈	8.35E+04 ± 4.27E+04 ≈	1.51E+05 ± 1.21E+05 ≈
F19	9.70E+03 ± 9.19E+03	1.52E+05 ± 6.43E+04 +	9.59E+04 ± 5.88E+04 +	1.15E+05 ± 4.55E+04 +
F20	**2.20E+03 ± 1.19E+02**	2.67E+03 ± 1.74E+02 +	2.50E+03 ± 1.33E+02 +	2.72E+03 ± 2.10E+02 +
F21	**2.40E+03 ± 4.16E+01**	2.50E+03 ± 4.48E+01 +	2.48E+03 ± 4.08E+01 +	2.51E+03 ± 3.67E+01 +
F22	4.78E+03 ± 3.01E+03	6.03E+03 ± 1.77E+03 ≈	5.36E+03 ± 2.21E+03 ≈	6.42E+03 ± 1.54E+03 ≈
F23	**2.73E+03 ± 2.30E+01**	3.29E+03 ± 1.27E+02 +	3.02E+03 ± 1.27E+02 +	3.31E+03 ± 9.91E+01 +
F24	**2.92E+03 ± 5.27E+01**	3.50E+03 ± 1.13E+02 +	3.13E+03 ± 1.42E+02 +	3.47E+03 ± 2.09E+02 +
F25	2.89E+03 ± 1.20E+01	2.89E+03 ± 1.46E+01 −	2.89E+03 ± 6.78E+00 −	**2.88E+03 ± 8.40E+00** −
F26	**4.78E+03 ± 6.31E+02**	8.16E+03 ± 1.57E+03 +	6.26E+03 ± 2.12E+03 +	7.65E+03 ± 1.89E+03 +
F27	**3.23E+03 ± 2.01E+01**	3.82E+03 ± 2.91E+02 +	3.39E+03 ± 1.94E+02 +	3.86E+03 ± 2.64E+02 +
F28	3.19E+03 ± 5.78E+01	3.21E+03 ± 2.57E+01 ≈	3.19E+03 ± 4.02E+01 ≈	3.21E+03 ± 1.35E+01 ≈
F29	**3.70E+03 ± 1.92E+02**	4.38E+03 ± 2.79E+02 +	4.24E+03 ± 2.85E+02 +	4.37E+03 ± 2.71E+02 +
F30	**8.49E+03 ± 3.24E+03**	5.74E+05 ± 3.50E+05 +	3.91E+05 ± 2.07E+05 +	7.73E+05 ± 4.79E+05 +
Rank	1	6	4	7
+/ ≈ /−	- /- /-	19/ 8 /2	21/ 4/4	20/ 7 /2

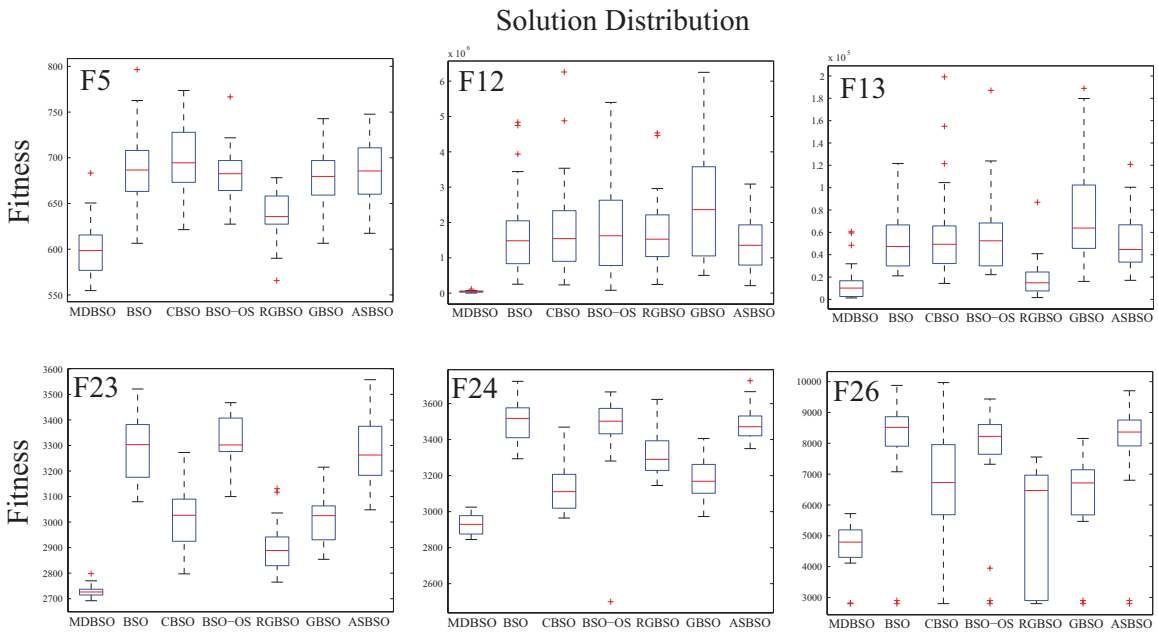

Figure 5.2: The box-and-whisker diagrams of optimal solutions obtained by seven kinds of BSOs on F5, F12, F13, F23, F24, F26.

Table 5.3: Experimental results of MDBSO versus BSO variants on CEC'17 benchmark functions (2).

Fun.	RGBSO Mean ±Std Dev	GBSO Mean ±Std Dev	ASBSO Mean ±Std Dev
F1	2.50E+03 ± 2.90E+03 ≈	3.43E+03 ± 3.60E+03 ≈	2.21E+03 ± 2.00E+03 ≈
F3	2.39E+04 ± 7.78E+03 +	**3.00E+02± 2.11E-04** −	3.95E+02 ± 1.10E+02 ≈
F4	4.75E+02 ± 1.10E+01 +	**4.58E+02± 3.13E+01** ≈	4.72E+02 ± 2.92E+01 ≈
F5	6.37E+02 ± 2.49E+01 +	6.79E+02± 3.44E+01 +	6.86E+02 ± 3.45E+01 +
F6	**6.01E+02 ± 8.14E-01** −	6.43E+02± 8.74E+00 +	6.51E+02 ± 7.77E+00 +
F7	9.10E+02 ± 3.11E+01 +	**8.36E+02± 2.89E+01** −	1.16E+03 ± 9.94E+01 +
F8	**9.08E+02 ± 1.89E+01** ≈	9.31E+02 ±2.62E+01 +	9.41E+02 ± 3.19E+01 +
F9	2.35E+03 ± 5.33E+02 +	**1.08E+03± 2.56E+02** −	3.93E+03 ± 6.39E+02 +
F10	**4.22E+03 ± 5.39E+02** −	5.00E+03± 6.55E+02 −	5.20E+03 ± 5.67E+02 −
F11	1.22E+03 ± 4.44E+01 ≈	1.25E+03 ±4.45E+01 +	1.23E+03 ± 4.75E+01 ≈
F12	1.73E+06 ± 1.06E+06 +	2.52E+06± 1.66E+06 +	1.41E+06 ± 8.00E+05 +
F13	1.76E+04 ± 1.66E+04 ≈	7.94E+04 ±4.60E+04 +	5.04E+04 ± 2.64E+04 +
F14	1.55E+04 ± 1.65E+04 +	**1.92E+03 ±4.01E+02** −	7.08E+03 ± 5.23E+03 ≈
F15	4.57E+03 ± 5.42E+03 −	5.61E+04 ±4.76E+04 +	3.01E+04 ± 2.25E+04 +
F16	2.93E+03 ± 3.46E+02 +	2.87E+03 ±1.95E+02 +	3.01E+03 ± 2.25E+02 +
F17	2.40E+03 ± 2.67E+02 +	2.22E+03± 1.99E+02 +	2.40E+03 ± 2.44E+02 +
F18	1.43E+05 ± 8.64E+04 +	**6.82E+04 ±3.51E+04** −	1.23E+05 ± 1.21E+05 ≈
F19	**6.17E+03 ± 4.20E+03** ≈	1.01E+05± 6.16E+04 +	1.25E+05 ± 6.31E+04 +
F20	2.58E+03 ± 2.44E+02 +	2.68E+03± 2.11E+02 +	2.67E+03 ± 2.17E+02 +
F21	2.45E+03 ± 3.67E+01 +	2.48E+03 ±4.44E+01 +	2.49E+03 ± 3.14E+01 +
F22	**3.54E+03 ± 1.82E+03** ≈	3.72E+03± 2.26E+03 ≈	5.79E+03 ± 2.04E+03 ≈
F23	2.90E+03 ± 8.97E+01 +	3.01E+03± 9.52E+01 +	3.26E+03 ± 1.24E+02 +
F24	3.32E+03 ± 1.27E+02 +	3.17E+03 ±1.14E+02 +	3.49E+03 ± 9.56E+01 +
F25	2.89E+03 ± 1.76E+01 −	2.90E+03± 2.25E+01 ≈	2.89E+03 ± 1.25E+01 −
F26	5.47E+03 ± 1.89E+03 +	6.10E+03 ±1.61E+03 +	7.84E+03 ± 1.80E+03 +
F27	3.29E+03 ± 3.33E+01 +	3.24E+03 ±6.39E+01 ≈	3.85E+03 ± 2.17E+02 +
F28	3.23E+03 ± 2.24E+01 +	3.21E+03± 3.68E+01 ≈	**3.18E+03 ± 3.60E+01** ≈
F29	3.90E+03 ± 2.39E+02 +	4.27E+03 ±2.85E+02 +	4.40E+03 ± 3.39E+02 +
F30	5.06E+04 ± 4.62E+04 +	7.66E+05 ±4.05E+05 +	5.16E+05 ± 2.90E+05 +
Rank	2	3	5
+/ ≈ /−	19 /6 /4	17/ 6 /6	19/ 8/2

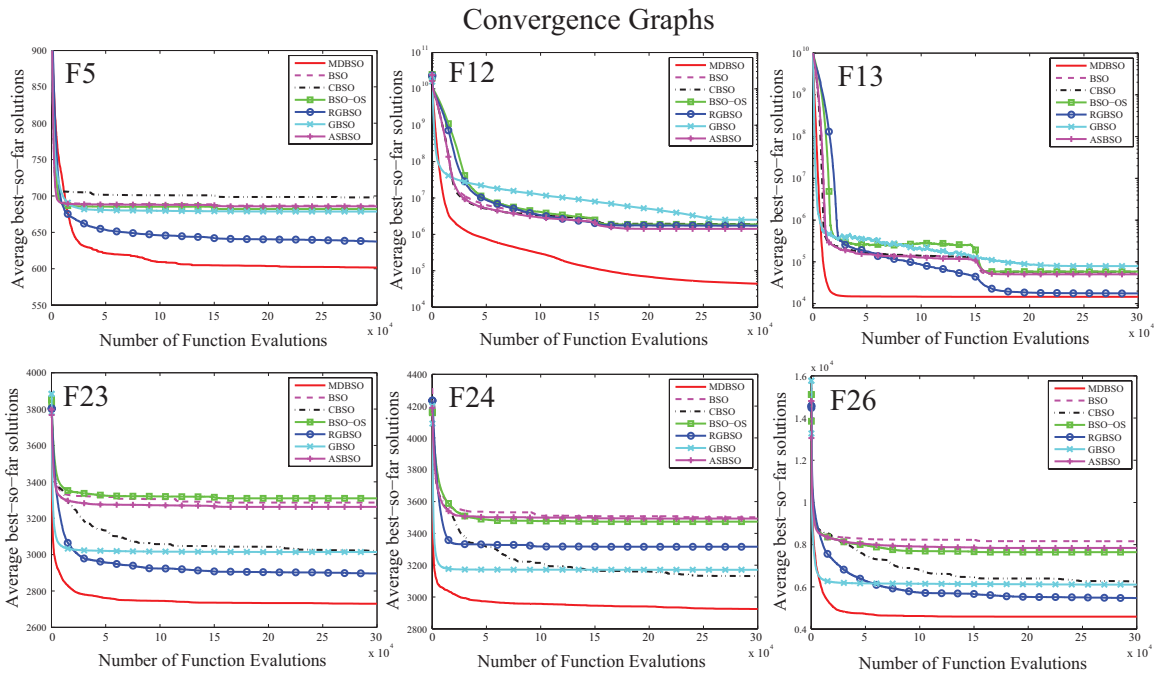

Figure 5.3: The convergence graphs of average best-so-far solutions obtained by seven kinds of BSOs on F5, F12, F13, F23, F24, F26.

$$Div = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(||X_i - X_{mean}||)^2} \tag{5.8}$$

where Div is the population diversity. N is population size and X_i is the ith individual. X_{mean} which is calculated as $X_{mean} = \frac{1}{N}\sum_{i=1}^{N} X_i$ is the average of the population.

As observed, MDBSO and CBSO are the best two algorithms that maintain population diversity at a good level in the whole process. This is owing to that MDBSO implements the diversity-driven strategy and CBSO uses a chaotic local search mechanism that can disturb the search trajectory of the individual. The diversity of GBSO keeps stable at the early stage but rapidly deteriorates, which makes it lose the capability of further improving solutions' quality. It should be emphasized that the diversity of RGBSO keeps fluctuating, which means that the dynamic step-size parameter control strategy has the efficacy of improving population diversity, but RGBSO lacks a mechanism to maintain it. Based on these analyses, one conclusion can be drawn that MDBSO can significantly preserve the population diversity in a good level during the search process.

5.3.1.2 MDBSO *vs.* GSA Variants

Besides the BSO variants, more SI algorithm are applied to further testify the effectiveness of MDBSO. GSA has been proposed for nearly a decade, and its developments are more matured than BSO's. Thus, the comparison between MDBSO and GSA variants can reflect the position of MDBSO in the whole SI algorithms. GSA uses gravity to mimic the search mechanism of individuals and it has obtained many successes in various research aspects. In this part, GSA [14] and its variants (IGSA [25], GGSA [103], MGSA [104], PSOGSA [105], HGSA [106], DNLGSA [107]) are implemented and their experimental results are presented in Tables 5.4 and 5.5. The results including mean and standard deviation (Std Dev) are exhibited and the values in boldface represent the best results among compared algorithms. The *Rank* refers to the rank of each algorithm obtained in the Friedman test. Each $+/\approx/-$ indicates

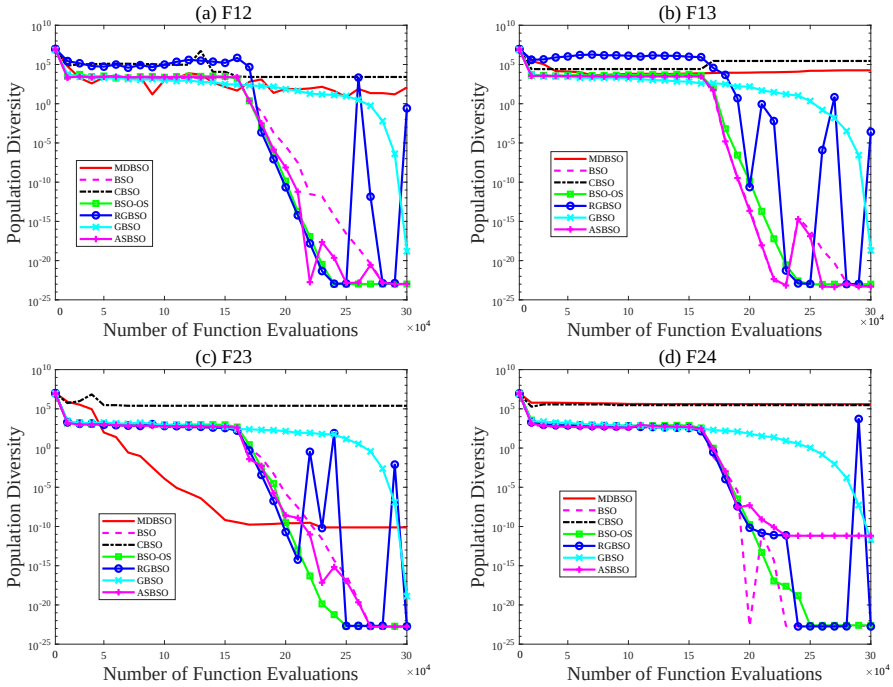

Figure 5.4: Population diversity on F12, F13, F23, and F24.

Table 5.4: Experimental results of MDBSO versus GSA variants on CEC'17 benchmark functions (1).

Fun.	MDBSO Mean ±Std Dev	GSA Mean ±Std Dev	IGSA Mean ±Std Dev	GGSA Mean ±Std Dev
F1	3.17E+03 ± 4.68E+03	2.00E+03 ± 1.03E+03 ≈	**1.88E+03 ± 1.37E+03** ≈	2.18E+03 ± 1.12E+03 ≈
F3	**6.13E+02 ± 6.28E+02**	8.30E+04 ± 4.33E+03 +	6.04E+04 ± 7.02E+03 +	6.02E+04 ± 6.73E+03 +
F4	**4.62E+02 ± 3.28E+01**	5.42E+02 ± 1.59E+01 +	5.22E+02 ± 2.10E+01 +	5.33E+02 ± 2.30E+01 +
F5	6.02E+02 ± 3.10E+01	7.26E+02 ± 2.01E+01 +	**5.42E+02 ± 8.18E+00** −	6.11E+02 ± 1.22E+01 +
F6	6.13E+02 ± 7.18E+00	6.50E+02 ± 2.75E+00 +	**6.00E+02 ± 1.29E-02** −	6.09E+02 ± 5.29E+00 −
F7	8.88E+02 ± 6.43E+01	7.87E+02 ± 1.19E+01 −	**7.43E+02 ± 5.60E+00** −	7.37E+02 ± 1.49E+00 −
F8	9.13E+02 ± 3.75E+01	9.51E+02 ± 1.31E+01 +	**8.33E+02 ± 7.72E+00** −	8.88E+02 ± 9.79E+00 −
F9	1.31E+03 ± 5.38E+02	2.93E+03 ± 3.92E+02 +	9.00E+02 ± 2.11E-14 −	**9.00E+02 ± 0.00E+00** −
F10	7.37E+03 ± 1.29E+03	4.87E+03 ± 4.34E+02 −	**3.58E+03 ± 4.60E+02** −	4.38E+03 ± 3.89E+02 −
F11	1.21E+03 ± 5.06E+01	1.45E+03 ± 8.92E+01 +	1.28E+03 ± 7.44E+01 +	1.25E+03 ± 3.23E+01 +
F12	**4.39E+04 ± 2.26E+04**	1.03E+07 ± 1.93E+07 +	1.40E+06 ± 7.32E+05 +	4.83E+05 ± 2.11E+05 +
F13	1.46E+04 ± 1.61E+04	3.10E+04 ± 6.45E+03 +	3.06E+04 ± 7.97E+03 +	1.87E+04 ± 4.70E+03 +
F14	7.71E+03 ± 5.29E+03	4.74E+05 ± 1.31E+05 +	1.96E+05 ± 1.37E+05 +	1.96E+05 ± 7.59E+04 +
F15	7.51E+03 ± 8.01E+03	1.17E+04 ± 1.93E+03 +	1.31E+04 ± 3.65E+03 +	4.12E+03 ± 1.57E+03 ≈
F16	**2.44E+03 ± 4.43E+02**	3.18E+03 ± 2.84E+02 +	2.71E+03 ± 2.16E+02 +	2.88E+03 ± 3.22E+02 +
F17	**1.98E+03 ± 1.94E+02**	2.90E+03 ± 1.70E+02 +	2.22E+03 ± 2.14E+02 +	2.67E+03 ± 2.06E+02 +
F18	9.28E+04 ± 5.38E+04	3.20E+05 ± 1.76E+05 +	3.81E+05 ± 3.85E+05 +	1.68E+05 ± 7.28E+04 +
F19	9.70E+03 ± 9.19E+03	1.42E+04 ± 5.13E+03 +	1.57E+04 ± 8.10E+03 +	5.93E+03 ± 1.46E+03 ≈
F20	**2.20E+03 ± 1.19E+02**	3.03E+03 ± 2.36E+02 +	2.41E+03 ± 1.70E+02 +	2.82E+03 ± 1.64E+02 +
F21	2.40E+03 ± 4.16E+01	2.56E+03 ± 1.95E+01 +	**2.35E+03 ± 6.54E+00** −	2.41E+03 ± 2.11E+01 +
F22	4.78E+03 ± 3.01E+03	6.39E+03 ± 1.69E+03 +	**2.30E+03 ± 0.00E+00** −	2.30E+03 ± 2.05E-10 -
F23	**2.73E+03 ± 2.30E+01**	3.56E+03 ± 1.23E+02 +	2.74E+03 ± 2.26E+01 ≈	2.86E+03 ± 3.94E+01 +
F24	2.92E+03 ± 5.27E+01	3.29E+03 ± 5.57E+01 +	**2.82E+03 ± 2.24E+01** −	2.91E+03 ± 3.70E+01 ≈
F25	2.89E+03 ± 1.20E+01	2.93E+03 ± 1.22E+01 +	2.92E+03 ± 9.79E+00 +	2.93E+03 ± 1.03E+01 +
F26	4.78E+03 ± 6.31E+02	6.86E+03 ± 8.95E+02 +	**2.83E+03 ± 4.66E+01** −	2.94E+03 ± 5.28E+02 −
F27	**3.23E+03 ± 2.01E+01**	4.67E+03 ± 3.21E+02 +	3.37E+03 ± 6.87E+01 +	3.39E+03 ± 3.57E+01 +
F28	3.19E+03 ± 5.78E+01	3.31E+03 ± 4.94E+01 +	3.26E+03 ± 3.48E+01 +	3.23E+03 ± 3.28E+01 +
F29	**3.70E+03 ± 1.92E+02**	4.71E+03 ± 2.10E+02 +	4.03E+03 ± 2.22E+02 +	4.25E+03 ± 2.30E+02 +
F30	**8.49E+03 ± 3.24E+03**	1.70E+05 ± 1.24E+05 +	3.34E+05 ± 3.68E+05 +	4.39E+04 ± 1.91E+04 +
Rank	1	6	3	4
+/ ≈ /−	- /- /-	26/ 1 /2	17/ 2/10	18/ 4 /7

Table 5.5: Experimental results of MDBSO versus GSA variants on CEC'17 benchmark functions (2).

Fun.	MGSA Mean ±Std Dev	PSOGSA Mean ±Std Dev	HGSA Mean ±Std Dev	DNLGSA Mean ±Std Dev
F1	4.63E+03 ± 4.30E+03 +	4.12E+03 ± 3.26E+03 +	2.68E+03 ± 2.50E+03 ≈	1.22E+05 ± 1.81E+05 +
F3	4.23E+04 ± 1.28E+04 +	3.56E+03 ± 7.87E+03 +	4.36E+04 ± 5.49E+03 +	1.49E+04 ± 1.24E+04 +
F4	5.33E+02 ± 5.86E+01 +	1.04E+03 ± 5.05E+02 +	5.19E+02 ± 2.63E+00 +	7.11E+02 ± 1.46E+02 +
F5	6.35E+02 ± 3.01E+01 +	6.46E+02 ± 3.40E+01 +	6.53E+02 ± 1.28E+01 +	6.50E+02 ± 3.67E+01 +
F6	6.27E+02 ± 8.09E+00 +	6.24E+02 ± 8.94E+00 +	6.08E+02 ± 4.54E+00 −	6.41E+02 ± 7.86E+00 +
F7	8.38E+02 ± 2.65E+01 −	9.72E+02 ± 6.32E+01 +	7.41E+02 ± 3.01E+00 −	9.86E+02 ± 6.78E+01 +
F8	9.08E+02 ± 2.29E+01 ≈	9.36E+02 ± 3.25E+01 +	9.00E+02 ± 9.03E+00 ≈	9.16E+02 ± 2.85E+01 ≈
F9	3.41E+03 ± 8.50E+02 +	4.54E+03 ± 1.67E+03 +	9.00E+02 ± 9.67E-14 −	3.93E+03 ± 1.10E+03 +
F10	4.92E+03 ± 8.11E+02 −	4.70E+03 ± 6.23E+02 −	4.21E+03 ± 2.93E+02 −	4.96E+03 ± 8.84E+02 −
F11	1.23E+03 ± 4.53E+01 ≈	1.49E+03 ± 3.14E+02 +	**1.20E+03 ± 2.98E+01** ≈	1.51E+03 ± 2.44E+02 +
F12	5.27E+05 ± 5.78E+05 +	6.00E+07 ± 1.48E+08 +	1.29E+05 ± 8.15E+04 +	1.58E+08 ± 2.63E+08 +
F13	2.81E+05 ± 1.42E+06 +	2.39E+07 ± 7.46E+07 +	**1.46E+04 ± 5.32E+03 +**	1.62E+06 ± 8.72E+06 +
F14	1.87E+04 ± 3.80E+04 +	9.87E+04 ± 2.69E+05 ≈	**6.72E+03 ± 3.05E+03** ≈	6.07E+04 ± 1.02E+05 +
F15	6.08E+03 ± 4.72E+03 ≈	5.31E+05 ± 2.82E+06 +	**2.20E+03 ± 7.21E+02** −	1.29E+04 ± 1.02E+04 +
F16	2.83E+03 ± 2.89E+02 +	3.05E+03 ± 4.59E+02 +	2.83E+03 ± 2.32E+02 +	2.74E+03 ± 3.13E+02 +
F17	2.37E+03 ± 2.07E+02 +	2.27E+03 ± 2.29E+02 +	2.77E+03 ± 1.99E+02 +	2.30E+03 ± 2.31E+02 +
F18	1.44E+05 ± 1.28E+05 ≈	3.07E+05 ± 1.01E+06 ≈	**6.16E+04 ± 1.47E+04** −	1.88E+05 ± 1.86E+05 +
F19	9.28E+03 ± 6.23E+03 ≈	1.43E+04 ± 1.33E+04 +	**5.42E+03 ± 1.25E+03** ≈	1.72E+04 ± 5.34E+04 ≈
F20	2.67E+03 ± 1.86E+02 +	2.57E+03 ± 2.35E+02 +	2.86E+03 ± 2.24E+02 +	2.72E+03 ± 2.15E+02 +
F21	2.44E+03 ± 3.14E+01 +	2.43E+03 ± 3.53E+01 +	2.41E+03 ± 5.90E+01 +	2.43E+03 ± 3.73E+01 +
F22	4.19E+03 ± 2.22E+03 ≈	4.68E+03 ± 1.91E+03 ≈	2.30E+03 ± 3.91E-09 −	4.50E+03 ± 2.32E+03 ≈
F23	3.00E+03 ± 8.12E+01 +	2.93E+03 ± 8.75E+01 +	2.76E+03 ± 1.33E+02 +	3.00E+03 ± 8.74E+01 +
F24	3.27E+03 ± 1.12E+02 +	3.21E+03 ± 1.43E+02 +	2.92E+03 ± 3.58E+01 ≈	3.18E+03 ± 7.29E+01 +
F25	2.92E+03 ± 1.66E+01 +	3.02E+03 ± 7.53E+01 +	**2.89E+03 ± 7.59E+00** −	3.00E+03 ± 4.61E+01 +
F26	5.56E+03 ± 1.63E+03 +	5.70E+03 ± 1.30E+03 +	2.85E+03 ± 5.07E+01 −	5.98E+03 ± 1.26E+03 +
F27	3.52E+03 ± 1.19E+02 +	3.52E+03 ± 1.36E+02 +	3.25E+03 ± 2.08E+01 +	3.43E+03 ± 1.50E+02 +
F28	3.21E+03 ± 7.43E+01 ≈	3.52E+03 ± 2.00E+02 +	**3.11E+03 ± 2.82E+01** −	3.44E+03 ± 9.79E+01 +
F29	4.12E+03 ± 3.03E+02 +	4.24E+03 ± 3.80E+02 +	4.05E+03 ± 1.88E+02 +	4.47E+03 ± 3.17E+02 +
F30	7.95E+04 ± 1.81E+05 +	3.39E+06 ± 1.42E+07 +	1.10E+04 ± 2.60E+03 +	3.60E+06 ± 6.27E+06 +
Rank	5	8	2	7
+/ ≈ /−	20 /7 /2	25/ 3 /1	13/ 6 /10	25/ 3/1

Table 5.6: Experimental results of MDBSO versus ABC variants on CEC'17 benchmark functions (1).

Fun.	MDBSO Mean ±Std Dev	ABC Mean ±Std Dev	GABC Mean ±Std Dev	MABC Mean ±Std Dev
F1	3.17E+03 ± 4.68E+03	4.72E+04 ± 7.74E+04 +	5.27E+03 ± 5.71E+03 +	2.01E+03 ± 1.82E+03 ≈
F3	**6.13E+02 ± 6.28E+02**	1.03E+05 ± 1.09E+04 +	9.20E+04 ± 1.02E+04 +	9.71E+04 ± 1.28E+04 +
F4	**4.62E+02 ± 3.28E+01**	5.19E+02 ± 2.79E+00 +	4.82E+02 ± 3.32E+01 +	5.17E+02 ± 2.31E+00 +
F5	6.02E+02 ± 3.10E+01	7.18E+02 ± 9.44E+00 +	**5.96E+02 ± 2.01E+01** ≈	7.19E+02 ± 1.48E+01 +
F6	6.13E+02 ± 7.18E+00	6.00E+02 ± 6.69E-03 −	6.00E+02 ± 1.15E-01 −	6.00E+02 ± 7.55E-04 −
F7	**8.88E+02 ± 6.43E+01**	9.43E+02 ± 9.71E+00 +	8.38E+02 ± 3.40E+01 −	9.39E+02 ± 9.74E+00 +
F8	9.13E+02 ± 3.75E+01	1.02E+03 ± 1.16E+01 +	**8.91E+02 ± 2.13E+01** −	1.02E+03 ± 1.08E+01 +
F9	1.31E+03 ± 5.38E+02	1.90E+03 ± 4.45E+02 +	2.08E+03 ± 1.08E+03 +	1.41E+03 ± 3.36E+02 +
F10	**7.37E+03 ± 1.29E+03**	8.10E+03 ± 3.19E+02 +	8.20E+03 ± 2.16E+02 +	8.15E+03 ± 3.09E+02 +
F11	**1.21E+03 ± 5.06E+01**	4.37E+03 ± 7.31E+02 +	1.71E+03 ± 7.64E+02 +	4.31E+03 ± 6.19E+02 +
F12	**4.39E+04 ± 2.26E+04**	1.17E+08 ± 2.66E+07 +	1.30E+06 ± 1.06E+06 +	7.79E+07 ± 2.79E+07 +
F13	1.46E+04 ± 1.61E+04	8.02E+07 ± 3.32E+07 +	**8.32E+03 ± 7.15E+03** ≈	8.46E+07 ± 2.90E+07 +
F14	**7.71E+03 ± 5.29E+03**	3.04E+05 ± 1.25E+05 +	1.88E+05 ± 9.66E+04 +	3.62E+05 ± 1.64E+05 +
F15	7.51E+03 ± 8.01E+03	2.08E+07 ± 8.65E+06 +	**7.32E+03 ± 7.67E+03** ≈	1.96E+07 ± 7.41E+06 +
F16	**2.44E+03 ± 4.43E+02**	3.76E+03 ± 1.87E+02 +	2.48E+03 ± 2.19E+02 ≈	3.68E+03 ± 1.57E+02 +
F17	**1.98E+03 ± 1.94E+02**	2.49E+03 ± 1.19E+02 +	2.05E+03 ± 1.34E+02 +	2.50E+03 ± 1.17E+02 +
F18	**9.28E+04 ± 5.38E+04**	6.34E+06 ± 3.20E+06 +	5.10E+06 ± 2.12E+06 +	6.77E+06 ± 2.63E+06 +
F19	9.70E+03 ± 9.19E+03	2.39E+07 ± 1.03E+07 +	**6.00E+03 ± 5.19E+03** ≈	2.67E+07 ± 1.04E+07 +
F20	**2.20E+03 ± 1.19E+02**	2.74E+03 ± 8.15E+01 +	2.72E+03 ± 8.80E+01 +	2.75E+03 ± 1.06E+02 +
F21	**2.40E+03 ± 4.16E+01**	2.52E+03 ± 1.18E+01 +	2.40E+03 ± 2.35E+01 ≈	2.51E+03 ± 1.18E+01 +
F22	4.78E+03 ± 3.01E+03	2.64E+03 ± 2.08E+02 ≈	2.30E+03 ± 1.56E+00 −	2.52E+03 ± 1.76E+02 ≈
F23	**2.73E+03 ± 2.30E+01**	2.89E+03 ± 1.60E+01 +	2.78E+03 ± 3.12E+01 +	2.88E+03 ± 1.65E+01 +
F24	**2.92E+03 ± 5.27E+01**	3.04E+03 ± 1.17E+01 +	2.95E+03 ± 3.66E+01 ≈	3.04E+03 ± 1.19E+01 +
F25	2.89E+03 ± 1.20E+01	2.89E+03 ± 1.73E-01 −	2.90E+03 ± 1.45E+01 +	**2.89E+03 ± 1.29E-01** −
F26	**4.78E+03 ± 6.31E+02**	5.74E+03 ± 1.28E+02 +	5.08E+03 ± 7.13E+02 +	5.71E+03 ± 1.13E+02 +
F27	3.23E+03 ± 2.01E+01	3.46E+03 ± 3.87E+01 +	3.25E+03 ± 1.52E+01 +	3.46E+03 ± 2.89E+01 +
F28	**3.19E+03 ± 5.78E+01**	3.26E+03 ± 2.59E+01 +	3.22E+03 ± 2.59E+01 +	3.23E+03 ± 1.95E+01 +
F29	**3.70E+03 ± 1.92E+02**	4.93E+03 ± 1.31E+02 +	3.73E+03 ± 1.69E+02 ≈	4.86E+03 ± 1.75E+02 +
F30	**8.49E+03 ± 3.24E+03**	2.67E+07 ± 1.04E+07 +	1.08E+04 ± 2.81E+03 +	2.38E+07 ± 7.73E+06 +
Rank	1	7	3	6
+/ ≈ /−	- /- /-	26/ 1 /2	17/ 8/4	25/ 2 /2

the performance of MDBSO is significantly better (+), not significantly better and worse (≈) or worse (−) than its peers.

In the statistical results obtained by MDBSO and GSA variants, the number of times MDBSO wins to others is 26 (GSA), 17 (IGSA), 18 (GGSA), 20 (MGSA), 25 (PSOGSA), 13 (HGSA) and 25 (DNLGSA) out of 29 problems, respectively. MDBSO can significantly outperform most variations of GSA and is very competitive with HGSA. This indicates that MDBSO obtains a strong competitiveness in comparison with GSA and its peers.

5.3.1.3 MDBSO *vs.* ABC Variants

ABC [15] is another powerful SI algorithm and it has been successfully modified in these years. It has a very special search mechanism against classical SI algorithms like PSO and GSA. The population in ABC is divided into three categories: employed bees, onlooker bees and scout bees. Each of them has different responsibility during

Table 5.7: Experimental results of MDBSO versus ABC variants on CEC'17 benchmark functions (2).

Fun.	SeABC Mean ±Std Dev	RABC Mean ±Std Dev	SFABC Mean ±Std Dev
F1	2.87E+09 ± 6.67E+09 +	4.36E+03 ± 5.73E+03 ≈	**1.40E+03 ± 1.26E+03** ≈
F3	9.90E+04 ± 1.15E+04 +	7.73E+04 ± 1.01E+04 +	9.82E+04 ± 1.38E+04 +
F4	5.06E+02 ± 3.15E+01 +	4.93E+02 ± 1.96E+01 +	5.05E+02 ± 2.44E+01 +
F5	6.58E+02 ± 5.81E+01 +	6.55E+02 ± 1.89E+01 +	6.98E+02 ± 1.51E+01 +
F6	6.29E+02 ± 2.66E+01 +	**6.00E+02 ± 7.35E-06** −	6.00E+02 ± 7.36E-06 −
F7	9.46E+02 ± 1.86E+02 +	8.91E+02 ± 1.80E+01 +	9.27E+02 ± 1.12E+01 +
F8	9.50E+02 ± 7.14E+01 +	9.59E+02 ± 2.36E+01 +	9.96E+02 ± 1.48E+01 +
F9	4.19E+03 ± 3.90E+03 +	9.58E+02 ± 1.22E+02 −	**9.00E+02 ± 1.18E-07** −
F10	8.07E+03 ± 2.47E+02 +	7.89E+03 ± 3.37E+02 +	8.11E+03 ± 3.22E+02 +
F11	2.90E+03 ± 1.70E+03 +	1.30E+03 ± 4.74E+01 +	1.82E+03 ± 4.48E+02 +
F12	2.59E+08 ± 6.64E+08 +	2.37E+06 ± 2.08E+06 +	1.23E+06 ± 7.26E+05 +
F13	2.34E+08 ± 6.04E+08 ≈	2.04E+04 ± 2.86E+04 +	1.06E+04 ± 6.52E+03 ≈
F14	2.56E+05 ± 1.95E+05 +	1.44E+05 ± 8.34E+04 +	1.93E+05 ± 1.00E+05 +
F15	3.63E+06 ± 1.58E+07 ≈	1.11E+04 ± 2.13E+04 ≈	1.41E+04 ± 3.07E+04 ≈
F16	2.86E+03 ± 6.08E+02 +	3.10E+03 ± 2.13E+02 +	3.45E+03 ± 1.55E+02 +
F17	2.37E+03 ± 2.86E+02 +	2.13E+03 ± 1.53E+02 +	2.33E+03 ± 1.28E+02 +
F18	4.25E+06 ± 3.33E+06 +	3.94E+06 ± 1.64E+06 +	5.93E+06 ± 3.06E+06 +
F19	2.21E+07 ± 7.50E+07 ≈	7.38E+05 ± 1.83E+06 +	1.37E+04 ± 2.86E+04 ≈
F20	2.74E+03 ± 8.62E+01 +	2.51E+03 ± 8.88E+01 +	2.76E+03 ± 7.80E+01 +
F21	2.48E+03 ± 7.97E+01 +	2.45E+03 ± 2.60E+01 +	2.49E+03 ± 1.15E+01 +
F22	3.02E+03 ± 1.68E+03 ≈	2.31E+03 ± 4.19E+00 ≈	**2.30E+03 ± 1.53E-08** −
F23	2.84E+03 ± 1.17E+02 +	2.77E+03 ± 2.89E+01 +	2.84E+03 ± 2.05E+01 +
F24	3.12E+03 ± 2.01E+02 +	2.99E+03 ± 2.24E+01 +	3.01E+03 ± 1.15E+01 +
F25	3.04E+03 ± 3.94E+02 ≈	2.89E+03 ± 7.25E+00 −	2.89E+03 ± 6.79E-01 −
F26	6.07E+03 ± 1.80E+03 +	4.90E+03 ± 3.44E+02 ≈	5.29E+03 ± 2.74E+02 +
F27	3.33E+03 ± 1.59E+02 +	**3.22E+03 ± 8.70E+00** −	3.25E+03 ± 1.80E+01 +
F28	3.42E+03 ± 4.18E+02 +	3.22E+03 ± 1.98E+01 +	3.21E+03 ± 1.08E+01 ≈
F29	4.05E+03 ± 4.12E+02 +	3.92E+03 ± 2.17E+02 +	4.32E+03 ± 2.23E+02 +
F30	1.83E+07 ± 5.10E+07 +	3.63E+05 ± 2.42E+05 +	2.33E+05 ± 3.07E+05 +
Rank	5	2	4
+/ ≈ /−	24 /5 /0	21/ 4 /4	20/ 5/4

Table 5.8: Details of the classification data sets.

Classification data sets	# of attributes	# of training samples	# of test samples	# of classes
3-bits XOR	3	8	8	2
Ballon	4	16	16	2
Iris	4	150	150	2
Heart	10	297	297	2

Table 5.9: Details of the function approximation data sets.

Function approximation datasets	# of Training Samples	# of Test Samples
Cosine: $y = cos(x\pi/2)^7$	31: $x \in [1.25 : 0.05 : 2.75]$	38: $x \in [1.25 : 0.04 : 2.75]$
Sine: $y = sin(2x)$	126: $x \in [-2\pi : 0.1 : 2\pi]$	252: $x \in [-2\pi : 0.05 : 2\pi]$
Griewank: $z = \frac{1}{4000}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2} cos(\frac{x_i}{\sqrt{i}}) + 1$, $x = x_1$ and $y = x_2$	21× 21: $x, y \in [-2 : 0.1 : 2]$	41× 41: $x, y \in [-4 : 0.05 : 2]$

Table 5.10: Details of the prediction data sets.

Classification data sets	# of training samples	# of test samples
Mackey Glass	450	550
Box Jenkins	140	156
EEG	1000	1500

Table 5.11: Reasonable combination of three parameters for nine tested problems, respectively.

Problems	M	k	θ_s
XOR	6	3	0.5
Balloon	7	3	0.5
Iris	7	3	0.5
Heart	14	3	0.5
Cosine	23	3	0.5
Sine	22	3	0.5
Griewank	15	3	0.5
Mackey Glass	12	3	0.5
Box Jenkins	15	3	0.5
EEG	12	3	0.5

Table 5.12: Experimental results of DNM training by MDBSO and BSO, respectively.

	MDBSO	BSO
	Mean (Std Dev)	Mean (Std Dev)
XOR	**2.89E-02** (**2.08E-02)**	1.42E-01 + (3.86E-02)
Balloon	**2.00E-02** (**8.84E-06)**	2.41E-02 + (5.86E-03)
Iris	**1.11E-02** (**7.83E-09)**	1.11E-02 + (3.97E-05)
Heart	**5.97E-02** (**1.60E-02)**	8.89E-02 + (2.30E-02)
Cosine	**5.41E-03** (**5.84E-04)**	2.92E-02 ≈ (6.40E-02)
Sine	**2.38E-01** (**7.43E-03)**	2.38E-01 ≈ (1.42E-02)
Griewank	**8.18E-02** (**1.62E-02)**	9.97E-02 + (2.10E-02)
Mackey Glass	**3.54E-04** (**1.66E-04)**	3.99E-04 ≈ (2.01E-04)
Box Jenkins	**4.22E-03** (**1.23E-04)**	4.39E-03 + (9.38E-05)
EEG	**5.59E-03** (**1.80E-04)**	5.64E-03 + (1.46E-04)

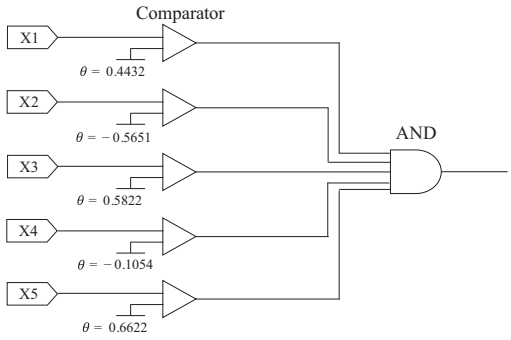

Figure 5.5: LC of Heart dataset trained by MDBSO.

the search process. Therefore, ABC is quite successful in optimizing multivariable and multimodal problems. As we mentioned that the proposed MDBSO is superior than BSO in solving such kinds of problems [108]. Using ABC variants as contrasts, including GABC [109], SeABC [110], MABC [73], RABC [111] and SFABC [112], is a very suitable conduct to prove the promising search ability of MDBSO.

Tables 5.6 and 5.7 exhibit the results obtained by MDBSO and ABC variants. The results including mean and standard deviation (Std Dev) are exhibited and the values in boldface represent the best results among compared algorithms. The *Rank* refers to the rank of each algorithm obtained in the Friedman test. Each $+/\approx/-$ indicates the performance of MDBSO is significantly better (+), not significantly better and worse ($\approx$) or worse ($-$) than its peers. To be precise, MDBSO significantly outperforms ABC and its variants on 26 (ABC), 17 (GABC), 25 (MABC), 24 (SeABC), 21 (RABC) and 20 (SFABC) problems, respectively. The result reveals the fact that MDBSO has better performance in solving diverse optimization problems than most state-of-the-art variants of ABC.

5.3.2 Artificial Neuron Network (ANN) Training Data Set

Benchmark functions are widely used to test the preliminary performance of the proposed algorithms because of its simple practicality. They can directly exhibit the pros and cons of the tested algorithms. However, it is far from enough to only use

benchmark functions as they are surrogate models and can not closely reflect real-world challenges to cross the big gap between academia and industries. Thus, in this section, we make an attempt to apply MDBSO for training a dendritic neuron model (DNM) [101].

DNM is proposed by considering the nonlinearity of synapses and has achieved great success in classification and prediction problems [113, 114, 115, 116]. It is composed of four layers, including a synaptic layer, a dendrite layer, a membrane layer and a soma layer. Gao *et al.* [18] conclude that DNM with learning algorithms can outperform the traditional multilayer perceptron (MLP) model with the same algorithms. Besides, training a neural network is a complex and tough optimization problem as it requires high computational cost. The goal is to minimize the sum of errors (between the practical and desired values) by optimizing the weight ω and threshold θ [117]. Thus, the application of using MDBSO can closely reflect its practical value in real-world challenges. We use BSO and MDBSO to train DNM for classification, approximation and prediction problems to systemically investigate the effectiveness of MDBSO in ANN training.

Four classification, three function approximation, and three prediction problems are used to testify the effectiveness of MDBSO for training DNM. The classification problems, i.e., XOR, ballon, iris and heart are acquired from the University of California at Irvine Machine Learning Repository [118]. The number of attributes, training samples, test samples and classes for these problems are summarized in Table 5.8. The function approximation problems include 1-D cosine with one peak, 1-D sine with four peaks, and 2-D Griewank problems. Their formulas, number of training samples and test samples are listed in Table 5.9. With regard to the prediction problems, their details are presented in Table 5.10, including Mackey Glass, Box Jenkins and EEG times series data. Besides, the reasonable combination of three DNM parameters for these tested problems are given in Table 5.13, respectively.

The experimental results obtained by BSO and MDBSO for training DNM are shown in Table 5.13 where better results are highlighted. A Wilcoxon rank-sum test is used to analyze the significant difference between the performance of BSO and

Table 5.13: Wilcoxon rank-sum test results of different numbers of clusters in MDBSO.

n=5 *vs.*	n=3	n=7	n=9
$+/\approx/-$	21 /8 /0	10/ 19 /0	12/ 17/0

Table 5.14: Wilcoxon rank-sum test results of different numbers of μ in MDBSO.

$\mu = 0.5$ *vs.*	$\mu = -0.5$	$\mu = 0$	$\mu = 1$
$+/\approx/-$	5 / 24/0	10 / 19 /0	14/ 14/1

MDBSO. It can be observed that MDBSO obtains better results than BSO on all test problems, and seven out of ten are significantly better. Thus, the conclusion can be drawn that MDBSO is more superior than BSO in training DNM, which exhibits that it is promising to be applied to more fields.

In addition, Fig. 5.5 presents a logic circuit (LC) of DNM trained by MDBSO for the heart dataset, after implementing the neuronal pruning function [116, 119]. In LCs, comparator which is an analog-to-digital converter and logical "NOT", "AND" and "OR" gates are the major components. Comparator outputs 1 when the input is greater than the threshold θ, otherwise it outputs 0. More details about the pruning method can be referred in [116, 119]. In this LC, the number of attributes is decreased from 10 to 5, which means the structure is greatly simplified and it can be easily implemented in hardware. By doing so, this model achieves a high computational speed and exhibits its practical value.

5.4 Discussion

5.4.1 Analysis of Preset Parameters

The number of parameters of MDBSO is reduced to two, including the number of cluster n and the mean value of Gaussian distribution μ. This indicates the effectiveness of the proposed concrete structure and adaptive parameters. Moreover, to be more precise, n and μ need to be analyzed to find the best parameter set for MDBSO.

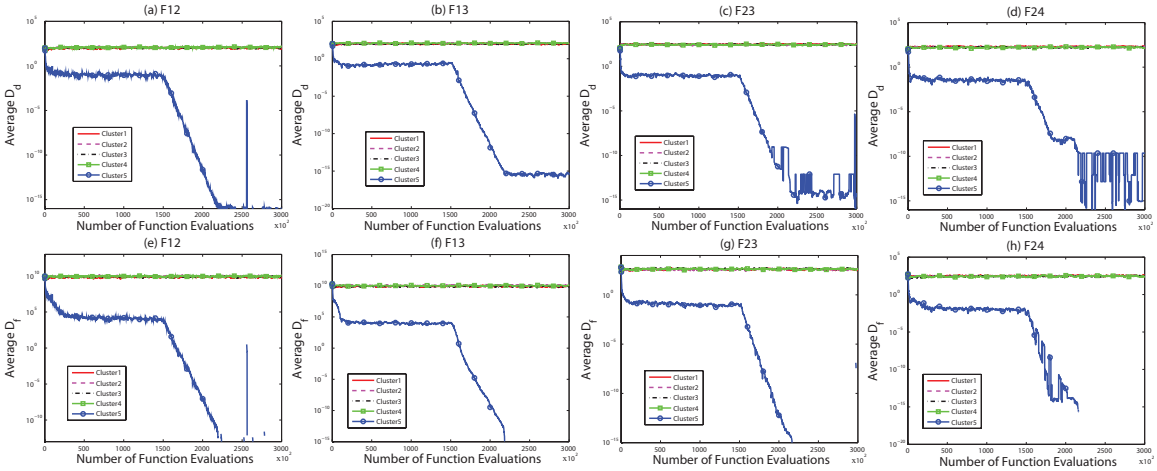

Figure 5.6: The curves of distance diversity and fitness diversity of BSO on F12, F13, F23 and F24.

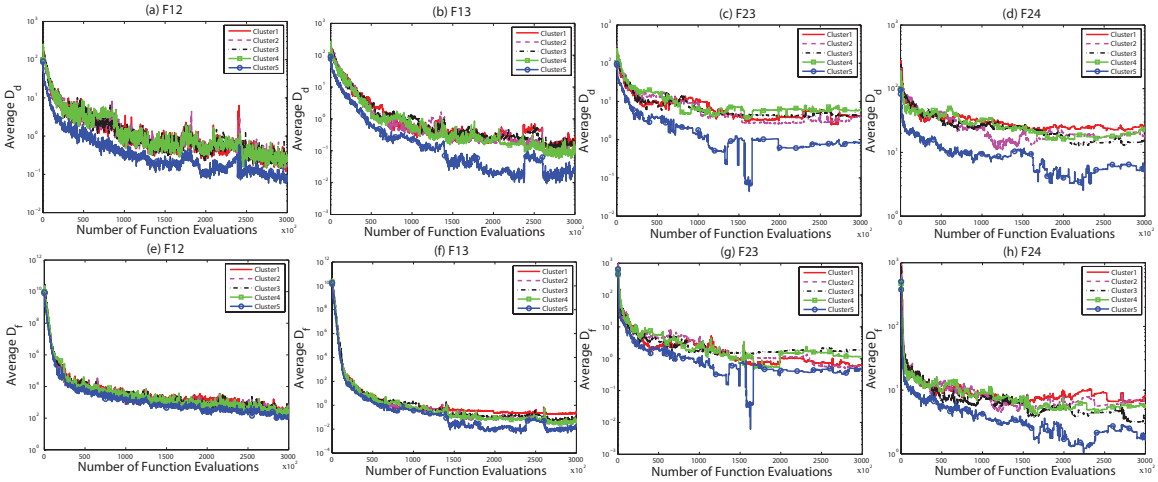

Figure 5.7: The curves of distance diversity and fitness diversity of MDBSO on F12, F13, F23 and F24.

5.4.1.1 Analysis of the Number of Clusters

In this part, n is first investigated. Four values, 3, 7, 9 and the original number 5 in BSO, are tested. The Wilcoxon rank-sum result is given in Table 5.13 and we can find that 5 is the value with the best performance.

5.4.1.2 Analysis of μ

Besides $\mu = 0.5$, we also investigate the values of -0.5, 0 and 1 to decide the best parameter for Gaussian distribution. In Table 5.15, $\mu = 0.5$ is significantly better

than 0 and 1. Although $\mu = -0.5$ has similar performance with 0.5, $\mu = 0.5$ is still the best choice for MDBSO.

5.4.2 Analysis of Population Diversity

The experimental results have proven that the MDBSO has superior performance than BSO and other SI algorithms. It is owing to the multiple diversity-driven strategy which keeps a well balance between exploration and exploitation. In this part, the graphs of population diversity are illustrated to deeply analyze its effectiveness in optimization process. Figs. 5.6 and 5.7 are the curves of distance and fitness diversities of BSO and MDBSO on F12, F13, F23 and F24, respectively. In each subgraph, the population diversity of each cluster is depicted based on the average value of 30 runs. The horizontal axis is the number of function evaluations and the vertical axis is the average population diversity (including distance diversity D_d and fitness diversity D_f).

It should be noticed that in Fig. 5.6 the population diversity of the last cluster in BSO is always decreasing fast and stays in a very low order of magnitude until the end of search process. It indicates that only a few individuals remain in this cluster. As we introduced in Section 3.2, the best individual in each cluster is selected as the center after k-means clustering. The situation of individuals keeping emigrating reports that the center in the last cluster obtains the worst quality in comparison with others. Thus, it is not competitive in attracting other individuals, which reflects the deficiency of the original individual generation strategy of BSO. If a cluster in BSO can not generate individuals with better fitness at the early stage of optimization, its size will continue to decease and never has a chance to rebound. However, in Fig. 5.7, the curves of MDBSO have more coordination than these of BSO. The population diversity of the last cluster keeps the same level with other cluster from the beginning to the end. Even if there is a deterioration in the middle, it will rebound quickly after several generations. This is due to the proposed mutation strategies in MDBSO which can efficiently generate new individuals with good fitness and endow the cluster

Table 5.15: Wilcoxon rank-sum test result of MDBSO *vs.* MDBSO-FG.

MDBSO *vs.*	MDBSO-FG
$+/\approx/-$	28 / 1 /0

with considerable attraction.

5.4.3 MDBSO with Fitness-based Grouping

It is introduced above that many attempts are made to improve the efficiency of BSO in clustering as the k-means is a time-cost clustering method. The modifications in BSO-OS, RGBSO and GBSO include improvements to clustering methods. For example, BSO-OS and RGBSO apply random grouping to minimize the clustering overhead. Although this method divides the population into elitists and normal individuals, it can not provide any specific measures for clustering. Thus, it is not suitable for MDBSO since the population diversity of each cluster is not available. In GBSO, a fitness-based grouping strategy is presented in which the individuals are ranked according to their fitness. The individuals with good and bad fitness are equally distributed into different groups. Fitness-based grouping is proven to be a less time-cost and effective method in [81]. Therefore, in this part, we will discuss whether the fitness-based grouping method could further improve the performance of MDBSO by replacing k-means clustering. The combination is named as MDBSO-FG.

Table 5.15 provides the Wilcoxon rank-sum test result and one conclusion can be drawn that the fitness-based grouping method is not suitable for MDBSO. In current situation, although k-means clustering is a computational cost method, it still has good performance in optimization results. Finding a method which can outperform k-means in both efficiency and effectiveness becomes a challenge in our future research.

5.4.4 Comparison with MIIBSO

BSO with multi-information interactions (MIIBSO) [120] is a newly proposed state-of-the-art BSO variation. It proposes a multi-information interaction (MII) strategy which contains three patterns to enhance the information interaction capability

between individuals. Moreover, it uses a random grouping strategy to replace the k-means clustering method. The comparison result between MDBSO and MIIBSO is shown in Table 5.16. The results including mean and standard deviation (Std Dev) are exhibited and the values in boldface represent the better results. Each $+/\approx/-$ indicates the performance of MDBSO is significantly better (+), not significantly better and worse ($\approx$) or worse ($-$) than MIIBSO.

As observed, MDBSO can significantly outperform MIIBSO on 19 out of the total of 29 functions. Specifically, the search ability of MDBSO is much better than that of MIIBSO on F1-F10 as they are unimodal and simple multimodal functions. The good performance of MIIBSO on F14-F20 reveals its effect for solving hybrid functions. This encourages us to further enhance the search ability of MDBSO on complex problems.

5.4.5 Computational Complexity

In this part, we compare the computational complexity of MDBSO with BSO to show its efficiency.

The time complexity of BSO is calculated as follows:

(1) The initialization of population and parameters in BSO needs the time complexity $O(N)$ where N is the population size.

(2) The population evaluation process needs $O(N)$.

(3) Using k-means clustering method to divide the population into 5 clusters needs $O(5N^2)$.

(4) The process of individual selection and generation of step length both cost $O(N^2)$.

(5) The generation of new individuals and the fitness calculation needs the time complexity $O(N^2)$, respectively.

Therefore, the overall time complexity of BSO is

Table 5.16: Experimental results of MDBSO versus MIIBSO on CEC'17 benchmark functions.

Fun.	MDBSO Mean ±Std Dev	MIIBSO Mean ±Std Dev
F1	**3.17E+03 ± 4.68E+03**	1.95E+10 ± 5.48E+09 +
F3	**6.13E+02 ± 6.28E+02**	3.29E+04 ± 1.10E+04 +
F4	**4.62E+02 ± 3.28E+01**	1.35E+03 ± 5.67E+02 +
F5	**6.02E+02 ± 3.10E+01**	6.49E+02 ± 2.44E+01 +
F6	**6.13E+02 ± 7.18E+00**	6.23E+02 ± 4.66E+00 +
F7	**8.88E+02 ± 6.43E+01**	9.84E+02 ± 5.30E+01 +
F8	**9.13E+02 ± 3.75E+01**	9.27E+02 ± 2.03E+01 +
F9	**1.31E+03 ± 5.38E+02**	2.42E+03 ± 7.60E+02 +
F10	**7.37E+03 ± 1.29E+03**	8.04E+03 ± 5.05E+02 +
F11	**1.21E+03 ± 5.06E+01**	1.45E+03 ± 1.65E+02 +
F12	**4.39E+04 ± 2.26E+04**	7.61E+08 ± 7.02E+08 +
F13	**1.46E+04 ± 1.61E+04**	3.50E+04 ± 2.67E+04 +
F14	7.71E+03 ± 5.29E+03	**1.48E+03 ± 3.85E+01** −
F15	7.51E+03 ± 8.01E+03	**2.98E+03 ± 9.77E+02** −
F16	2.44E+03 ± 4.43E+02	**2.24E+03 ± 2.60E+02** −
F17	1.98E+03 ± 1.94E+02	**1.83E+03 ± 4.03E+01** −
F18	9.28E+04 ± 5.38E+04	**2.87E+03 ± 1.12E+03** −
F19	9.70E+03 ± 9.19E+03	**2.30E+03 ± 7.12E+02** −
F20	2.20E+03 ± 1.19E+02	**2.20E+03 ± 9.89E+01** ≈
F21	**2.40E+03 ± 4.16E+01**	2.43E+03 ± 2.31E+01 +
F22	**4.78E+03 ± 3.01E+03**	8.65E+03 ± 1.81E+03 +
F23	**2.73E+03 ± 2.30E+01**	2.86E+03 ± 6.81E+01 +
F24	**2.92E+03 ± 5.27E+01**	3.04E+03 ± 4.59E+01 +
F25	**2.89E+03 ± 1.20E+01**	3.20E+03 ± 1.70E+02 +
F26	**4.78E+03 ± 6.31E+02**	5.86E+03 ± 5.94E+02 +
F27	3.23E+03 ± 2.01E+01	**3.20E+03 ± 2.03E-04** −
F28	**3.19E+03 ± 5.78E+01**	3.30E+03 ± 2.13E-04 +
F29	3.70E+03 ± 1.92E+02	**3.41E+03 ± 1.20E+02** −
F30	8.49E+03 ± 3.24E+03	**4.67E+03 ± 9.39E+02** −
w /t/ l	- /- /-	19/1/9

$$O(N) + O(N) + O(5N^2) + O(N^2) + O(N^2) = 2O(N^2) + O(5N^2) + 2O(N) \quad (5.9)$$

To be simplified, its overall time complexity is $O(N^2)$.

The time complexity of MDBSO is

(1) The initialization process is $O(N)$.

(2) The fitness evaluation is $O(N)$.

(3) Using k-means clustering method to divide the population into 5 clusters needs $O(5N^2)$.

(4) Calculating the distance diversity (D_d) and fitness diversity (D_f) needs $O(N^2)$, respectively.

(5) Selection of mutation strategies costs $O(N^2)$.

(6) The generation of new individuals and the fitness calculation needs the time complexity $O(N^2)$, respectively.

Thus, the overall time complexity of MDBSO is

$$O(N)+O(N)+O(5N^2)+2O(N^2)+O(N^2)+O(N^2) = 4O(N^2)+O(5N^2)+2O(N) \quad (5.10)$$

The time complexity of MDBSO can be simplified as $O(N^2)$.

Although calculating D_d and D_f need more cost than BSO, the same overall time complexities of $O(N^2)$ indicate that MDBSO achieves the same computational efficiency in comparison with BSO. In other words, the multiple diversity-driven strategy can improve the performance of MDBSO and maintain efficiency. Thus, the utilization of D_d and D_f is promising to be applied to more SI algorithms.

Chapter 6

Conclusion

In this thesis, I propose multiple improvements to alleviate the drawbacks of brain storm optimization algorithms. These proposed algorithms are presented as follows.

For the drawback of lacking exploitation search ability, I propose a method which combines the brain storm optimization with chaotic local search (CLS) to enhance the search ability of BSO. CLS is implemented when the process of BSO sticks into stagnation. The dynamic mechanism of CLS makes each individual spread out and break the current position's balance. The proposed CBSO is tested using CEC'05 benchmark functions and exhibits better performance when comparing with other optimization algorithms. Wilcoxon and Friedman test results demonstrate the effectiveness of CBSO by statistical analysis. In my future work, I want to study the combination of chaos and several local search mechanisms using meta Lamarckian learning strategy to enhance the robustness of CLS for solving unimodal and multimodal benchmark functions and engineering problems.

An adaptive step length mechanism based on memory is proposed, namely AS-BSO, to enhance the flexibility and robustness of BSO. It applies multiple step length generation strategies and a new memory mechanism in aim to generate better individuals for different search periods and problems. The strategies with different step lengths are produced by using four different scale parameters and they are selected based on a memory structure in each iteration. Different from the conventional memory mechanism, the proposed memory structure method is created to record the improvement value in fitness obtained by each strategy. By implementing this, the

strategy which can increase solution quality substantially has a higher possibility to be selected compared with the original one which can similarly success while obtains only a little improvement. The performance of ASBSO has been tested by using CEC'13 and CEC'17 benchmark function suits (57 functions in total) which include different characteristics. Some well-known optimization algorithms also have been added into comparison. Experimental and statistical results show that the proposed ASBSO can succeed in improving the performance of BSO in terms of global search ability, convergence speed, robustness and solution quality. Moreover, some real-world problems in CEC'11 are introduced to present the application value of ASBSO. These results can encourage our future research into self-adaptive search mechanism. Furthermore, this will broaden our perspective of BSO for dynamic and multiobjective optimization.

As for the deterioration of population diversity during the optimization process, I propose a novel multiple diversity-driven strategy to solve this problem. Two diversity measures, including distance diversity and fitness diversity, are collaborated to control the generation of adaptive parameters in optimization procedure. The diversities of each cluster in BSO is calculated to control the utilization of mutation strategies. Moreover, new individuals are generated according to these diversities. In this way, the number of parameters in original BSO is greatly decreased. Experiments on CEC2017 benchmark function suit and ANN training task are utilized to investigate the performance of the proposed MDBSO. The results demonstrate that MDBSO obtains superior effectiveness, efficiency and robustness. In the comparison with the latest BSO variations, MDBSO exhibits overwhelming advantages, which indicates MDBSO is the current best BSO variation. Besides, diversity is a significant part in optimization search. Several researchers have realized its importance but I creatively take it into parameter adaptation and obtain desired result. It suggests that the proposed multiple diversity-driven strategy deserves more attention in SI research field.

The performance of BSO is promising, and there is still a great space to improve its search ability. In my future research, we will consistently concentrate on the improvements of BSO.

Bibliography

[1] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Robots and Biological Systems: Towards a New Bionics?* Springer, 1993, pp. 703–712.

[2] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning.* Springer, 2011, pp. 760–766.

[3] S. Gao, Y. Yu, Y. Wang, J. Wang, J. Cheng, and M. Zhou, "Chaotic local search-based differential evolution algorithms for optimization," *IEEE Transactions on Systems, Man and Cybernetics: Systems*, 2019, doi: 10.1109/TSMC.2019.2956121.

[4] J. Cheng, G. Yuan, M. Zhou, S. Gao, C. Liu, H. Duan, and Q. Zeng, "Accessibility analysis and modeling for iov in an urban scene," *IEEE Transactions on Vehicular Technology*, 2020, doi: 10.1109/TVT.2020.2970553 .

[5] J. Wang, B. Cen, S. Gao, Z. Zhang, and Y. Zhou, "Cooperative evolutionary framework with focused search for many-objective optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2018, doi: 10.1109/TETCI.2018.2849380.

[6] J. Cheng, G. Yuan, M. Zhou, S. Gao, C. Liu, and H. Duan, "A fluid mechanics-based data flow model to estimate vanet capacity," *IEEE Transactions on Intelligent Transportation Systems*, 2019, doi: 10.1109/TITS.2019.2921074.

[7] J. Cheng, M. Chen, M. Zhou, S. Gao, C. Liu, and C. Liu, "Overlapping community change point detection in an evolving network," *IEEE Transactions on Big Data*, 2018, doi: 10.1109/TBDATA.2018.2880780.

[8] ——, "Overlapping community change point detection in an evolving network," *IEEE Transactions on Big Data*, 2018, doi: 10.1109/TBDATA.2018.2880780.

[9] J. Cheng, X. Wu, M. Zhou, S. Gao, Z. Huang, and C. Liu, "A novel method for detecting new overlapping community in complex evolving networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 4, pp. 1832–1844, 2019.

[10] S. Gao, S. Song, J. Cheng, Y. Todo, and M. Zhou, "Incorporation of solvent effect into multi-objective evolutionary algorithm for improved protein structure prediction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 4, pp. 1365–1378, 2018.

[11] S. Gao, Y. Wang, J. Cheng, Y. Inazumi, and Z. Tang, "Ant colony optimization with clustering for solving the dynamic location routing problem," *Applied Mathematics and Computation*, vol. 285, pp. 149–173, 2016.

[12] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[13] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *International Conference in Swarm Intelligence*. Springer, 2010, pp. 355–364.

[14] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

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

[16] Y. Shi, "Brain storm optimization algorithm," in *International Conference in Swarm Intelligence*. Springer, Berlin, Heidelberg, 2011, pp. 303–309.

[17] J. Sun, S. Gao, H. Dai, J. Cheng, M. Zhou, and J. Wang, "Bi-objective elite

differential evolution for multivalued logic networks," *IEEE Transactions on Cybernetics*, vol. 50, no. 1, pp. 233–246, 2020.

[18] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neural model with effective learning algorithms for classification, approximation, and prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 601–614, 2019.

[19] J. Cheng, J. Cheng, M. Zhou, F. Liu, S. Gao, and C. Liu, "Routing in internet of vehicles: A review." *IEEE Trans. Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2339–2352, 2015.

[20] S. Gao, Y. Todo, T. Gong, G. Yang, and Z. Tang, "Graph planarization problem optimization based on triple-valued gravitational search algorithm," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 9, no. 1, pp. 39–48, 2014.

[21] S. Gao, C. Vairappan, Y. Wang, Q. Cao, and Z. Tang, "Gravitational search algorithm combined with chaos for unconstrained numerical optimization," *Applied Mathematics and Computation*, vol. 231, pp. 48–62, 2014.

[22] Z. Song, S. Gao, Y. Yu, J. Sun, and Y. Todo, "Multiple chaos embedded gravitational search algorithm," *IEICE Transactions on Information and Systems*, vol. 100, no. 4, pp. 888–900, 2017.

[23] S. Cheng, Q. Qin, J. Chen, and Y. Shi, "Brain storm optimization algorithm: a review," *Artificial Intelligence Review*, vol. 46, no. 4, pp. 445–458, 2016.

[24] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[25] J. Ji, S. Gao, S. Wang, Y. Tang, H. Yu, and Y. Todo, "Self-adaptive gravitational search algorithm with a modified chaotic local search," *IEEE Access*, vol. 5, pp. 17 881–17 895, 2017.

[26] C. Vairappan, H. Tamura, S. Gao, and Z. Tang, "Batch type local search-based adaptive neuro-fuzzy inference system (ANFIS) with self-feedbacks for time-series prediction," *Neurocomputing*, vol. 72, no. 7-9, pp. 1870–1877, 2009.

[27] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, 2008.

[28] Y. Ong, M. Lim, N. Zhu, and K. Wong, "Classification of adaptive memetic algorithms: a comparative study," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 1, pp. 141–152, 2006.

[29] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Computing*, vol. 8, no. 2, pp. 239–287, 2009.

[30] M. Črepinšek, S. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, p. 35, 2013.

[31] R. Caponetto, L. Fortuna, S. Fazzino, and M. G. Xibilia, "Chaotic sequences to improve the performance of evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 289–304, 2003.

[32] G. Wang, S. Deb, A. H. Gandomi, Z. Zhang, and A. H. Alavi, "Chaotic cuckoo search." *Soft Comput.*, vol. 20, no. 9, pp. 3349–3362, 2016.

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

[34] S. H. Kellert, *In the wake of chaos: Unpredictable order in dynamical systems.* University of Chicago press, 1994.

[35] A. R. Jordehi, "A chaotic artificial immune system optimisation algorithm for

solving global continuous optimisation problems," *Neural Computing and Applications*, vol. 26, no. 4, pp. 827–833, 2015.

[36] Y. Lu, J. Zhou, H. Qin, Y. Wang, and Y. Zhang, "Chaotic differential evolution methods for dynamic economic dispatch with valve-point effects," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 2, pp. 378–387, 2011.

[37] W. Jiang and B. Li, "Optimizing complex functions by chaos search," *Cybernetics & Systems*, vol. 29, no. 4, pp. 409–419, 1998.

[38] B. Liu, L. Wang, Y. H. Jin, F. Tang, and D. X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons & Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.

[39] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization," *KanGAL Report*, vol. 2005005, p. 2005, 2005.

[40] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[41] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[42] J. Wang, Y. Zhou, Y. Wang, J. Zhang, C. P. Chen, and Z. Zheng, "Multi-objective vehicle routing problems with simultaneous delivery and pickup and time windows: formulation, instances, and algorithms," *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp. 582–594, 2016.

[43] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[44] J. Luengo, S. García, and F. Herrera, "A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions and non-parametric tests," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7798–7808, 2009.

[45] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.

[46] S. Das and P. N. Suganthan, "Problem definitions and evaluation criteria for cec 2011 competition on testing evolutionary algorithms on real world optimization problems," *Jadavpur University, Nanyang Technological University, Kolkata*, 2010.

[47] W. Gao, S. Liu, and L. Huang, "Enhancing artificial bee colony algorithm using more information-based search equations," *Information Sciences*, vol. 270, pp. 112–133, 2014.

[48] G. Yang, S. Wu, Q. Jin, and J. Xu, "A hybrid approach based on stochastic competitive hopfield neural network and efficient genetic algorithm for frequency assignment problem," *Applied Soft Computing*, vol. 39, pp. 104–116, 2016.

[49] X. Guo, Y. Wu, and L. Xie, "Modified brain storm optimization algorithm for multimodal optimization," in *International Conference in Swarm Intelligence*. Springer, 2014, pp. 340–351.

[50] X. Guo, Y. Wu, L. Xie, S. Cheng, and J. Xin, "An adaptive brain storm optimization algorithm for multiobjective optimization problems," in *International conference in swarm intelligence*. Springer, 2015, pp. 365–372.

[51] L. Li and K. Tang, "History-based topological speciation for multimodal opti-

mization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 1, pp. 136–150, 2015.

[52] H. Qiu and H. Duan, "Receding horizon control for multiple uav formation flight based on modified brain storm optimization," *Nonlinear dynamics*, vol. 78, no. 3, pp. 1973–1988, 2014.

[53] Y. Sun, "A hybrid approach by integrating brain storm optimization algorithm with grey neural network for stock index forecasting," in *Abstract and Applied Analysis*, vol. 2014. Hindawi Publishing Corporation, 2014.

[54] Y. Shi, J. Xue, and Y. Wu, "Multi-objective optimization based on brain storm optimization algorithm," *International Journal of Swarm Intelligence Research (IJSIR)*, vol. 4, no. 3, pp. 1–21, 2013.

[55] Y. Shi, "Brain storm optimization algorithm in objective space," in *Evolutionary Computation (CEC), 2015 IEEE Congress on*. IEEE, 2015, pp. 1227–1234.

[56] C. Li and H. Duan, "Information granulation-based fuzzy rbfnn for image fusion based on chaotic brain storm optimization," *Optik-International Journal for Light and Electron Optics*, vol. 126, no. 15, pp. 1400–1406, 2015.

[57] Y. Yu, S. Gao, S. Cheng, Y. Wang, S. Song, and F. Yuan, "CBSO: a memetic brain storm optimization with chaotic local search," *Memetic Computing*, vol. 10, no. 4, pp. 353–367, 2018.

[58] S. Cheng, Y. Shi, Q. Qin, Q. Zhang, and R. Bai, "Population diversity maintenance in brain storm optimization algorithm," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 4, no. 2, pp. 83–97, 2014.

[59] H. Duan, S. Li, and Y. Shi, "Predator–prey brain storm optimization for dc brushless motor," *IEEE Transactions on Magnetics*, vol. 49, no. 10, pp. 5336–5340, 2013.

[60] H. Duan and C. Li, "Quantum-behaved brain storm optimization approach to solving loney's solenoid problem," *IEEE Transactions on Magnetics*, vol. 51, no. 1, pp. 1–7, 2015.

[61] Y. Wang, S. Gao, Y. Yu, and Z. Xu, "The discovery of population interaction with a power law distribution in brain storm optimization," *Memetic Computing*, pp. In Press, DOI: 10.1007/s12 293–017–0248–z, 2017.

[62] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary computation*, vol. 3, no. 2, pp. 82–102, 1999.

[63] J. Liang, B. Qu, P. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report*, vol. 201212, 2013.

[64] N. Awad, M. Ali, J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization," in *Technical Report*. NTU, Singapore, 2016.

[65] D. Molina, M. Lozano, C. García-Martínez, and F. Herrera, "Memetic algorithms for continuous optimisation based on local search chains," *Evolutionary Computation*, vol. 18, no. 1, pp. 27–63, 2010.

[66] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[67] V. Tereshko and A. Loengarov, "Collective decision making in honey-bee foraging dynamics," *Computing and Information Systems*, vol. 9, no. 3, p. 1, 2005.

[68] Y. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[69] Y. Shi, C.-M. Pun, H. Hu, and H. Gao, "An improved artificial bee colony and its application," *Knowledge-Based Systems*, vol. 107, pp. 14–31, 2016.

[70] R. Zhang, P.-C. Chang, S. Song, and C. Wu, "A multi-objective artificial bee colony algorithm for parallel batch-processing machine scheduling in fabric dyeing processes," *Knowledge-Based Systems*, vol. 116, pp. 114–129, 2017.

[71] S. Gao, Y. Wang, J. Wang, and J. Cheng, "Understanding differential evolution: A poisson law derived from population interaction network," *Journal of Computational Science*, vol. 21, pp. 140–149, 2017.

[72] R. P. Parouha and K. N. Das, "A robust memory based hybrid differential evolution for continuous optimization problem," *Knowledge-Based Systems*, vol. 103, pp. 118–131, 2016.

[73] X. Li and G. Yang, "Artificial bee colony algorithm with memory," *Applied Soft Computing*, vol. 41, pp. 362–372, 2016.

[74] G. Sun, P. Ma, J. Ren, A. Zhang, and X. Jia, "A stability constrained adaptive alpha for gravitational search algorithm," *Knowledge-Based Systems*, vol. 139, pp. 200–213, 2018.

[75] S. Mirjalili, "SCA: a sine cosine algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.

[76] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.

[77] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, 2005.

[78] Y. Cai and J. Wang, "Differential evolution with neighborhood and direction information for numerical optimization," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 2202–2215, 2013.

[79] J. Wang, J. Liao, Y. Zhou, and Y. Cai, "Differential evolution enhanced with multiobjective sorting-based mutation operators," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2792–2805, 2014.

[80] J. Wang, W. Zhang, and J. Zhang, "Cooperative differential evolution with multiple populations for multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2848–2861, 2016.

[81] M. El-Abd, "Global-best brain storm optimization algorithm," *Swarm and Evolutionary Computation*, vol. 37, pp. 27–44, 2017.

[82] I. Rechenberg, "Evolutionsstrategien," in *Simulationsmethoden in der Medizin und Biologie*. Springer, 1978, pp. 83–114.

[83] K. Tang, P. Yang, and X. Yao, "Negatively correlated search," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 542–550, 2016.

[84] Z. Cao, Y. Shi, X. Rong, B. Liu, Z. Du, and B. Yang, "Random grouping brain storm optimization algorithm with a new dynamically changing step size," in *International Conference in Swarm Intelligence*. Springer, 2015, pp. 357–364.

[85] Y. Yu, S. Gao, Y. Wang, J. Cheng, and Y. Todo, "ASBSO: An improved brain storm optimization with flexible search length and memory-based selection," *IEEE Access*, vol. 6, pp. 36 977–36 994, 2018.

[86] M. El-Abd, "Brain storm optimization algorithm with re-initialized ideas and adaptive step size," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, pp. 2682–2686.

[87] Á. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.

[88] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[89] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 71–78.

[90] Y. Shi and R. C. Eberhart, "Population diversity of particle swarms," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 2008, pp. 1063–1067.

[91] X. Chen, Y.-S. Ong, M.-H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 591–607, 2011.

[92] C. Segura, C. A. C. Coello, E. Segredo, and A. H. Aguirre, "A novel diversity-based replacement strategy for evolutionary algorithms," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 3233–3246, 2016.

[93] C. Segura, A. Hernández-Aguirre, F. Luna, and E. Alba, "Improving diversity in evolutionary algorithms: New best solutions for frequency assignment," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 539–553, 2017.

[94] S. Cheng, Y. Shi, Q. Qin, T. O. Ting, and R. Bai, "Maintaining population diversity in brain storm optimization algorithm," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 3230–3237.

[95] E. Burke, S. Gustafson, G. Kendall, and N. Krasnogor, "Advanced population diversity measures in genetic programming," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2002, pp. 341–350.

[96] Y. G. Woldesenbet and G. G. Yen, "Dynamic evolutionary algorithm with variable relocation," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 500–513, 2009.

[97] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, and M. Sumner, "A fast adaptive memetic algorithm for online and offline control design of pmsm drives," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 28–41, 2007.

[98] C. C. Coello, G. T. Pulido, and E. M. Montes, "Current and future research trends in evolutionary multiobjective optimization," in *Information Processing with Evolutionary Algorithms*. Springer, 2005, pp. 213–231.

[99] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[100] T. Nomura and K. Shimohara, "An analysis of two-parent recombinations for real-valued chromosomes in an infinite population," *Evolutionary Computation*, vol. 9, no. 3, pp. 283–308, 2001.

[101] Y. Todo, H. Tamura, K. Yamashita, and Z. Tang, "Unsupervised learnable neuron model with nonlinear interaction on dendrites," *Neural Networks*, vol. 60, pp. 96–103, 2014.

[102] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artificial Intelligence Review*, vol. 12, no. 4, pp. 265–319, 1998.

[103] S. Mirjalili and A. Lewis, "Adaptive gbest-guided gravitational search algorithm," *Neural Computing and Applications*, vol. 25, no. 7-8, pp. 1569–1584, 2014.

[104] B. Gu and F. Pan, "Modified gravitational search algorithm with particle memory ability and its application," *International Journal of Innovative Computing, Information and Control*, vol. 9, no. 11, pp. 4531–4544, 2013.

[105] S. Mirjalili and S. Z. M. Hashim, "A new hybrid PSOGSA algorithm for function optimization," in *Computer and information application (ICCIA), 2010 international conference on*. IEEE, 2010, pp. 374–377.

[106] Y. Wang, Y. Yu, S. Gao, H. Pan, and G. Yang, "A hierarchical gravitational search algorithm with an effective gravitational constant," *Swarm and Evolutionary Computation*, vol. Accept, 2019, 2019.

[107] A. Zhang, G. Sun, J. Ren, X. Li, Z. Wang, and X. Jia, "A dynamic neighborhood learning-based gravitational search algorithm," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 436–447, 2018.

[108] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[109] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Applied Mathematics and Computation*, vol. 217, no. 7, pp. 3166–3173, 2010.

[110] Y. Xue, J. Jiang, B. Zhao, and T. Ma, "A self-adaptive artificial bee colony algorithm based on global best for global optimization," *Soft Computing*, pp. 1–18, 2017.

[111] G. Li, L. Cui, X. Fu, Z. Wen, N. Lu, and J. Lu, "Artificial bee colony algorithm with gene recombination for numerical function optimization," *Applied Soft Computing*, vol. 52, pp. 146–159, 2017.

[112] J. Ji, S. Song, C. Tang, S. Gao, Z. Tang, and Y. Todo, "An artificial bee colony algorithm search guided by scale-free networks," *Information Sciences*, vol. 473, pp. 142–165, 2019.

[113] T. Zhou, S. Gao, J. Wang, C. Chu, Y. Todo, and Z. Tang, "Financial time series prediction using a dendritic neuron model," *Knowledge-Based Systems*, vol. 105, pp. 214–224, 2016.

[114] T. Jiang, S. Gao, D. Wang, J. Ji, Y. Todo, and Z. Tang, "A neuron model with synaptic nonlinearities in a dendritic tree for liver disorders," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 12, no. 1, pp. 105–115, 2017.

[115] W. Chen, J. Sun, S. Gao, J. Cheng, J. Wang, and Y. Todo, "Using a single dendritic neuron to forecast tourist arrivals to japan," *IEICE Transactions on Information and Systems*, vol. 100, no. 1, pp. 190–202, 2017.

[116] Y. Tang, J. Ji, S. Gao, H. Dai, Y. Yu, and Y. Todo, "A pruning neural network model in credit classification analysis," *Computational Intelligence and Neuroscience*, vol. 2018, Article ID 9390410, 2018.

[117] Y.-J. Gong, J. Zhang, and Y. Zhou, "Learning multimodal parameters: A barebones niching differential evolution approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 7, pp. 2944–2959, 2018.

[118] C. L. Blake, "Uci repository of machine learning databases, irvine, university of california," *http://www. ics. uci. edu/˜ mlearn/MLRepository. html*, 1998.

[119] J. Ji, S. Gao, J. Cheng, Z. Tang, and Y. Todo, "An approximate logic neuron model with a dendritic structure," *Neurocomputing*, vol. 173, pp. 1775–1783, 2016.

[120] C. Li, Z. Song, J. Fan, Q. Cheng, and P. X. Liu, "A brain storm optimization with multi-information interactions for global optimization problems," *IEEE Access*, vol. 6, pp. 19 304–19 323, 2018.