Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Spherical evolution for solving continuous optimization problems

Deyu Tang *

School of Medical Information and Engineering, Guangdong Pharmaceutical University, Guangzhou 510006, PR China
School of Computer Science & Engineering, South China University of Technology, Guangzhou, 510006, PR China
Guangdong Province Precise Medicine Big Data of Traditional Chinese Medicine Engineering Technology Research Center, 510006, PR China

HIGHLIGHTS

- Search pattern and search style are proposed by a mathematical model.
- Spherical search style is proposed.
- Spherical evolution algorithm is proposed.
- Data clustering problem is achieved by the proposed algorithm.

ARTICLE INFO

Article history:
Received 8 September 2018
Received in revised form 22 March 2019
Accepted 12 May 2019
Available online 18 May 2019

Keywords:
Differential evolution
Spherical evolution
Search pattern
Spherical search style
Data clustering optimization

ABSTRACT

In these years, more and more nature-inspired meta-heuristic algorithms have been proposed; search operators have been their core problem. The common characteristics or mechanism of search operators in different algorithms have not been represented by a standard format. In this paper, we first propose the concept of a search pattern and a search style represented by a mathematical model. Second, we propose a new search style, namely a spherical search style, inspired by the traditional hypercube search style. Furthermore, a spherical evolution algorithm is proposed based on the search pattern and spherical search style. At the end, 30 benchmark functions of CEC2017 and a real-world optimization problem are tested. Experimental results and analysis demonstrate that the proposed method consistently outperforms other state-of-the-art algorithms.

1. Introduction

Optimization has been an active area of research to provide satisfactory solutions for complex real-world problems over the past few decades. These problems have been more complex with the association of differentiability, multi-modality, increasing dimensionality, rotation characteristics and lack of knowledge of their mathematical expressions. This promotes researchers to develop accurate, fast and computationally efficient optimization algorithms.

More and more nature-inspired meta-heuristic approaches have been proposed and have achieved their goals successfully for solving some problems. The Genetic Algorithm (GA) [1] was an Evolutionary Algorithm (EA) based on Darwin's theory of natural selection. The Particle Swarm Optimization (PSO) [2] was a well-known optimization algorithm mimicking the social behavior of birds flocking or fish schooling. The Differential Evolution (DE) [3] was an evolutionary algorithm, which was achieved by the mutation operator, crossover operator, and selection operator. The Ant Colony Optimization (ACO) [4] imitated the foraging behavior of an ant colony. The Artificial Bee Colony Algorithm (ABC) [5] was inspired by the foraging behavior of a bee colony, including onlookers, employed bees, and scouts. The Biogeography-Based Optimization Algorithm (BBO) [6] was achieved by a mathematical model of biogeography, which described how species migrate from one island to another, how new species arise, and how species become extinct. The Teaching-Learning-Based Optimization (TLBO) [7] was inspired by the study process between teachers and students. The Gravitational Search Algorithm (GSA) [8] was based on the law of gravity and mass interactions. The searcher agents were a collection of masses that interact with each other based on the Newtonian gravity theory and laws of motion. The Artificial Algae Algorithm (AAA) [9] was inspired by the living behaviors of a microalgae, photosynthetic species. The algorithm was based on its evolutionary process, adaptation process, and the movement of microalgae. The Thermal Exchange Optimization (TEO) [10] was based on Newton's law of cooling. Each agent was considered as a cooling object, and by associating another agent as environment, a heat transference and thermal exchange happens between them. The Ant Lion Optimizer (ALO) [11] mimics the hunting mechanism of ant lions in nature. The five main steps of hunting prey (the random walk

* Correspondence to: School of Medical Information and Engineering, Guangdong Pharmaceutical University, Guangzhou 510006, PR China.
E-mail address: scutdy@126.com.

https://doi.org/10.1016/j.asoc.2019.105499
1568-4946/© 2019 Elsevier B.V. All rights reserved.

of ants, building traps, entrapment of ants in traps, catching preys, and re-building traps) were implemented. The Whale Optimization Algorithm (WOA) [12] mimics the social behavior of humpback whales. The algorithm was inspired by the bubble-net hunting strategy. The Grasshopper Optimization Algorithm (GOA) [13] mathematically modeled and mimicked the behavior of grasshopper swarms in nature. The Gray Wolf Optimizer [14] mimicked the leadership hierarchy and hunting mechanism of gray wolves in nature. Four types of gray wolves, namely, alpha, beta, delta, and omega, were employed for simulating the leadership hierarchy. The Sine Cosine Algorithm (SCA) [15] created multiple initial random candidate solutions and required them to fluctuate outwards or towards the best solution using a mathematical model based on sine and cosine functions. The Firefly Algorithm [16] was based on the behavior of flashing lights of fireflies. A less bright firefly gets attracted to the nearest bright firefly within its visual range, and the brightness of a firefly was determined by the objective function. The Vortex Search Algorithm (VS) [17] was inspired from the vortex pattern created by the vortical flow of the stirred fluids. To provide a good balance between the explorative and exploitative behavior of a search, the proposed method modeled its search behavior as a vortex pattern by using an adaptive step size adjustment scheme. The Harmony Search (HS) [18] emulated the process of playing different musical instruments based on an analogy to physical phenomena. The Charge System Search (CSS) [19] was based on electrostatic and Newtonian mechanics laws. The Mine Blast Algorithm (MBA) [20] was inspired by the mine bomb explosion. The Water Cycle Algorithm [21] was inspired from nature and based on the observation of the water cycle process. Streams flow to rivers or directly flow to the sea, which was modeled to solve optimization problems. The Water Wave Optimization (WWO) [22] was inspired by the shallow water wave theory. The phenomena of water waves, such as propagation, refraction, and breaking, were used to derive effective mechanisms for searching in a high-dimensional solution space. The Symbiotic Organisms Search (SOS) [23] was modeled based on three fundamental relationship structures, namely mutualism, commensalism and parasitism. These natural phenomena can be associated with the specific problem objective function to be optimized or solved. The Kidney-inspired Algorithm (KA) [24] was inspired by the kidney process in the human body. In this algorithm the solutions were filtered at a rate that was calculated based on the mean of objective functions of all solutions in the current population of each iteration. The Krill Herd Algorithm (KH) [25] was based on the simulation of the herding behavior of krill individuals. The minimum distances of each individual krill from food and from highest density of the herd were considered as the objective function for the krill movement. The Bird Mating Optimizer (BMO) [26] was inspired by mating strategies of bird species during mating season. BMO imitated the behavior of bird species metaphorically to breed broods with superior genes for designing optimum searching techniques. The Salp Swarm Algorithm (SSA) [27] was inspired by the swarming behavior of salps when navigating and foraging in oceans. The Invasive Tumor Growth Optimization (ITGO) [28] was inspired by the mechanism of tumor growth. The Across Neighborhood Search (ANS) [29] was motivated by a neighborhood search strategy. An individual directly searches across the neighborhoods of multiple superior solutions with the guidance of a Gaussian distribution. The Black Hole (BH) [30] was inspired by the black hole phenomenon. At each iteration of the Black Hole Algorithm, the best candidate was selected to be the black hole, which then started pulling other candidates around it, called stars. The Lightning Attachment Procedure Optimization (LAPO) [31] mimics the lightning attachment procedure including the downward leader movement, the upward leader propagation, the unpredictable trajectory of lightning downward leader, and the branch fading feature of lightning. The Forest Optimization Algorithm (FOA) [32] was inspired by the few trees in the forests that can survive for several decades, while other trees live only for a limited period. In FOA, the seeding procedure of the trees was simulated so that some seeds fall just under the trees, while others were distributed in wide areas by natural forces and the animals that feed on the seeds or fruits. Spotted hyena optimizer [33] was inspired by the behavior of spotted hyenas. The main concept behind this algorithm was the social relationship between spotted hyenas and their collaborative behavior. The three basic steps of SHO was searching for prey, encircling, and attacking prey; all three were mathematically modeled and implemented. Evolution strategies [34] belong to the general field of genetic algorithms. Evolutionary programming [35] was inspired by the inherent relationship and behavior between parents and offspring, which simulated the evolution at species level without a crossover operator. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [36] was an improved version of Evolution Strategies, in which a covariance matrix adaptation was used.

By researching the mechanisms of nature-inspired metaheuristic approaches (NMH), we can discover that the core problems of NMH algorithm's design are focused on two aspects. The first issue is the connection between the search operator and the search pattern and search style. The search operators of an NMH algorithm can determine that if a solution of an individual in population can find a better solution. In fact, there are different operators for the different NMH algorithms. Are there some common characteristics or principles of NMH algorithms such as search pattern or search style? To date, we have not found the research to verify this. The second issue is the individual selection method for the search operators. Many individual selection methods, such as roulette wheel selection [37], tournament selection [38], and stochastic universal sampling method [39], have been proposed. More individual selection methods are inspired by the characteristics of individuals in nature. In this paper, we focus on research of these issues and attempt to propose the search pattern and search style. Moreover, we design a spherical evolution algorithm based on the search pattern and search style.

The rest of the paper is structured as follows. Section 2 introduces reviews of the related literatures. Section 3 proposes search pattern and search style. Section 4 presents the spherical evolution algorithm based on search pattern and search style. Section 5 introduces the experimental results and analysis. Section 6 concludes this study and gives directions for future research.

2. Related reviews

A search operator is a core task, which can guarantee each individual in a population of finding a better solution for an optimization problem. Many operators of NMH algorithms are achieved by an initial position (solution) and one or more updating units (mutation items). In Particle Swarm Optimization (PSO) [2], a particle *i* updates its position by a velocity item and an initial position of *i* as the initial point and two updating units. The first updating unit is the perturbation of difference between the global best particle *gbest* and particle *i*. The second updating unit is the perturbation of difference between the current best particle *pbest* and particle *i*. In the Differential Evolution (DE) Algorithm [3], an individual *i* updates its position by an initial position of individual *i* or a randomly chosen individual *j* in the population as the initial point with one or two updating units. Generally speaking, an updating unit is often the perturbation of difference between two individuals chosen randomly from a population. In contrast to PSO, the dimension selection method is used in DE. In the Artificial Bee Colony (ABC) [5], a bee i updates its position using its initial position as the initial point and the perturbation of difference between a bee randomly chosen and a bee i as an updating unit. In addition, this search operator is achieved in one dimensional space. In Teaching-Learning-Based Optimization (TLBO) [7], student i updates its score (solution) by two stages. In teaching stage, student i is updated by the initial score and the perturbation of difference between teacher and mean score. In the learning stage, student i is updated by the initial score and the perturbation of difference between student i and a student randomly chosen from class. In the Cuckoo Search Algorithm (CS) [40], a cuckoo i updates its nest by two search operators. First, cuckoo i is updated by its initial nest (solution) and a levy flight's perturbation of difference between cuckoo i and the global best cuckoo. Second, cuckoo i is updated by its initial nest and the perturbation of difference between two randomly selected cuckoos. In the Gravitational Search Algorithm (GSA) [8], a particle i is updated by its initial position and perturbation of difference between the best particle j and particle i. The number of the best particle is larger than 1, and the perturbation function is achieved according to the principle of gravitation. In the Gray Wolf Optimizer (GWO) [14], gray wolf i updates its position by its initial position and the perturbation of difference between three best wolves and a wolf i. In the Whale Optimization Algorithm (WOA) [12], the search operator can be seen as the improved version of GWO by some different perturbation function, such as cosine function. In Spotted Hyena Optimization [33], the search operator can also be considered as the improved version of GWO. In the Artificial Algae Algorithm (AAA) [9], algae i updates its position by its initial position and the perturbation of difference between a better algae selected by some principle and an algae i. In contrast to other algorithms, the perturbation function is achieved by helical style. In Black Hole (BH) algorithm [30], particle i updates its position by its initial position and the perturbation of difference between the best particle j and the particle i. In the Across Neighborhood Search (ANS) Algorithm [29], particle i is updated by two operators with dimension selection like DE. First, particle i updates its position by the best particle j as the initial position and the perturbation of difference between the best particle j and the particle i. Second, particle i updates its position by randomly selected particle j as initial position and the perturbation of difference between the particle j and the particle i. In the Sine Cosine Algorithm (SCA) [15], particle i is updated by its initial position and the perturbation of difference between the best particle j and the particle i. In contrast to BH, the perturbation function is achieved by sine function, cosine function, and absolute function. In Salp Swarm Algorithm (SSA) [27], a salp i is updated by two operators. First, the leading salp i updates its position by a perturbation function. Second, the follower salp i updates its position by its initial position i and the historical position j. In fact, a search pattern achieved by an initial point and one or more updating units with perturbed differences between two individuals has been used in many NMH algorithms, such as the Thermal Exchange Optimization (TEO) [10], the Grasshopper Optimization Algorithm (GOA) [13], the Firefly Algorithm (FA) [16], and the Invasive Tumor Growth Optimization (ITGO) [28].

The second core problem is the individual selection method for search operators. Generally speaking, many NMH algorithms use some individual selection method, such as Roulette Wheel Selection Method (RWM) [37], Tournament Selection Method (TSM) [38], Stochastic Universal Sampling Method (SUSM) [39], Randomly Selection Method (RSM) [3], or Designated Individual Selection Method (DISM) [2] and Individual Selection by Bee's Behavior (ISBB) [5]. RWM uses the cumulative probability calculated with selection probabilities of all individuals. The selection of roulette wheel gives the fitness values of all the individuals, and then each individual is put on a wheel based on the percentage value of the total fitness sum. In TSM, two random individuals are chosen; as the most appropriate individual among them, even in the worst-case scenario, the second lowest matching individual participates in the mating pool. SUSM is a single-phase sampling algorithm with minimum spread and zero bias. Instead of a single selection pointer employed in roulette wheel methods, SUSM uses N equally spaced pointers, where N is the number of selections required. RSM uses the random number of uniform distributions. DISM is achieved by choosing the designated individual with some characteristics. ISBB chooses the individual by a probability of gathering honey in artificial the Bee Colony Algorithm. RSM and DISM or the combination of both are the most popular methods for many NMH algorithms, such as PSO, DE, TLBO, CS, GSA, BH, GWO, and SCA, etc.

The features and effectiveness of NMH are summarized in Table 1. Some common search patterns and search styles can be observed. However, the search pattern and search style of NMH have not been represented by mathematical formation, which make it difficult to understand for researchers. In addition, all the search styles are represented by the first order difference, and it is a hypercube search style. Therefore, a new search style of NMH, such as spherical search style, has not been proposed for NMH algorithms.

3. Search pattern and search style

Search operators of NMH algorithms have some common characteristics or patterns, which have not been represented in standard form. Standard form of a search operator can help researchers better understand the existing NMH algorithms so that new NMH algorithms can be proposed. Therefore, we attempt to propose the concept of search pattern and search style by mathematical form.

3.1. Search pattern

As reviewed above, search pattern can be represented as Eq. (1).

$$A_{i,j}^{new} = B_{i,j}^{0} + \sum_{k=1}^{n} SS\,(C_{i,j}^{k}, D_{i,j}^{k}),\ i = 1, 2, \ldots popsize;\quad j = 1, 2, \ldots, Dim \tag{1}$$

where A, B, C and D denote four solution sets by four matrixes, in which each matrix is represented with $popsize$ rows and Dim columns. The number of individuals in a population corresponding to the number of solutions is denoted as $popsize$, while Dim denotes the dimensional size of a solution corresponding to an individual. B^0 denotes the initial solution set for a search operator. $SS(C^k, D^k)$ represents the updating units in the search operator, which decides the search style. In addition, n denotes the number of updating units. In some cases (such as TLBO, ABC etc.), B^0 can be represented as itself (A^0).

3.2. Search style

Indeed, updating units of search operators in many NMH algorithms uses the first order difference method to update initial solution. It is a hypercube search style in high dimensional space, which can be represented as Eq. (2).

$$\begin{aligned} SS(C_{i,j}^{k} D_{i,j}^{k}) = {} & ScaleFun01_{i,j}() \cdot (ScaleFun02_{i,j}() \cdot C_{i,j}^{k} \\ & - ScaleFun03_{i,j}() \cdot D_{i,j}^{k}), \\ & i = 1, 2, \ldots, popsize;\ j = 1, 2, \ldots, dim;\ k = 1, 2, \ldots, n. \end{aligned} \tag{2}$$

Table 1
Characteristics of NMH algorithms.

NMH algorithms	Number of search operators	Initial position	Number of updating unit	Feature of updating unit	Perturbation function	Dimension selection method	Individual selection methods
PSO	1	Itself	2	**First order difference**	Uniform distribution	No	DISM
DE	1	Random	1∼2	**First order difference**	Constant	Yes	RSM
ABC	2	Itself	1	**First order difference**	Uniform distribution	Yes	ISBB
TLBO	2	Itself	1	**First order difference**	Uniform distribution	No	DISM
CS	2	Itself	1	**First order difference**	Levy flight and uniform distribution	No	DISM and RSM
GSA	1	Itself	K>=2	**First order difference**	Gravitational function	No	DISM
GWO	1	Itself	1	**First order difference**	Linear function	No	DISM
WOA	2	Itself	1	**First order difference**	Linear function and hybrid function	No	DISM
AAA	2	Itself	1	**First order difference**	Hybrid function and uniform distribution	Yes	TSM
BH	1	Itself	1	**First order difference**	Uniform distribution	No	DISM
SCA	2	Itself	1	**First order difference**	Hybrid function	No	DISM
SSA	3	Itself	1	**First order difference**	Hybrid function	No	DISM
TEO	2	Itself	1	**First order difference**	Hybrid function	No	DISM
GOA	1	Itself	N>2	**First order difference**	Hybrid function	No	RSM
FA	1	Itself	1	**First order difference**	Hybrid function	No	RSM
ITGO	4	Itself	1 or 2	**First order difference**	Hybrid function and uniform distribution	Yes	DISM

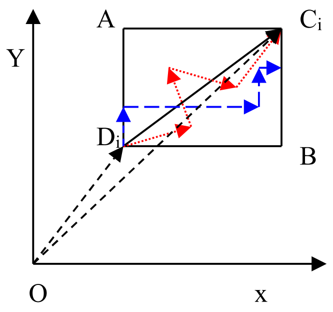


Fig. 1. Hypercube search style in two-dimension space.

where the*ScaleFun*01 () function is used to adjust the scale of the difference between *C* and *D*; it is often a constant function, a simple linear function, a probability function such as uniform distribution, normal distribution or levy distribution, or any other complex nonlinear function (such as GSA). Similarly, *ScaleFun*02 () and *ScaleFun*03 () functions are often used to adjust the scale of *C* and *D* (such as TLBO, GWO etc.). Generally speaking, the *ScaleFun*02 () or *ScaleFun*03 () functions are often as constant functions. An example is provided to simplify the explanation of the hypercube search style according to Eq. (2). Suppose *ScaleFun*01 () returns a random number of uniform distribution as *rand*∼(0,1). *ScaleFun*02 () or *ScaleFun*03 () is equal to 1. Fig. 1 shows the different search trajectories in two-dimensional space. Dotted lines OD_i and OC_i denote two solution vectors. The black line with the arrow (D_iC_i) represents a search trajectory of an updating unit in DE when the cross rate (CR) is equal to 1 in two-dimensional space. An individual can search only within the black line area because the scale factor (F) is equal to the same scale value in x-dimension and y-dimension. The blue line with an arrow represents a search trajectory of an updating unit in DE in dimensional space when we suppose only one dimension can be selected. Note that it is also a search trajectory of an updating unit in ABC. That is to say, the search style of ABC is a special case of DE. The red line with an arrow represents a search trajectory of an updating unit in PSO (or TLBO etc.) because the scale values are different in x-dimension and y-dimension for each search process.

4. Spherical evolution algorithm

To date, search operators of many NMH algorithms are based on search pattern and search style as in Eqs. (1) and (2). Almost all search styles are based on the hypercube search style by the first order difference, which restricts the development of NMH algorithms. Therefore, we attempt to propose a spherical style instead of the hypercube search style for the search pattern (1). Later in this report, we propose a spherical evolution algorithm.

4.1. Spherical search style

As shown in Fig. 1, the hypercube (a rectangle in 2-dimensional space) search style is achieved by adjusting the length of $|D_iB|$ or $|D_iA|$ in a rectangular region. In contrast to it, the spherical search style is achieved by adjusting the angle and radius. To explain the search mechanism of spherical search style, we give an instance in 2-dimensional space as shown in Fig. 2 (a sphere can be represented by a circle in 2-dimensional space). The dotted line with an arrow OD_i and OC_i denotes two solution vectors. The area of the circle (D_i is the center of a circle, and the distance between D_i and C_i is radius) is the search space. Adjusting the angle from 0 to α and the length of the radius $|D_iC_i|C$, we can obtain a new point C'_i. It is obvious that we can search the whole region of the circle when the angle is adjusted from 0 to 2π and the length of the radius is adjusted from 0 to $|D_iC_i|C$. In addition, it can be observed that the sphere search style has a larger search region than that of the hypercube search style as shown in Figs. 1 and 2, which indicates that the spherical search style has a stronger ability of exploration than the hypercube search style. The spherical search style can be represented in high dimensional space as Eqs. (3) (4) (5) according to literature [41]. In Eqs. (3) (4) (5), $\|C_{i,*} - D_{i,*}\|_2$ is the Euclidean norm that denotes the distance of vector $C_{i,*}$ and vector $D_{i,*}$. It is the radius of a sphere in high dimension. $ScaleFun_{i,j}()\cdot$ denotes a scale function that can appropriately adjust the radius length. The dimension size is denoted as *dim* (*dim*>=3). θ is the angle between vector $C_{i,*}$ and $D_{i,*}$.

$$SS_3(C_{i,j}, D_{i,j}) = ScaleFun_{i,j}() \cdot \|C_{i,*} - D_{i,*}\|_2 \cdot \prod_{k=j}^{dim-1} sin(\theta_j), j = 1 \quad (3)$$

$$SS_3(C_{i,j}, D_{i,j}) = ScaleFun_{i,j}() \cdot \|C_{i,*} - D_{i,*}\|_2 \cdot \cos(\theta_{j-1})$$

$$\cdot \prod_{k=j}^{dim-1} sin(\theta_j),\ 1 < j \leq dim-1 \tag{4}$$

$$SS_3(C_{i,j}, D_{i,j}) = ScaleFun_{i,j}() \cdot \|C_{i,*} - D_{i,*}\|_2.cos(\theta_{j-1}), j = dim \tag{5}$$

In two-dimensional space, the spherical search can be represented as follows:

$$SS_2(C_{i,j}, D_{i,j}) = ScaleFun_{i,j}() \cdot \|C_{i,*} - D_{i,*}\|_2.sin(\theta) \tag{6}$$

$$SS_2(C_{i,j}, D_{i,j}) = ScaleFun_{i,j}() \cdot \|C_{i,*} - D_{i,*}\|_2.cos(\theta) \tag{7}$$

where θ is a random number of uniform distribution between $[0, 2\pi]$.

In a special case, the search style in one-dimensional space can be represented as follows:

$$SS_1(C_{i,j}, D_{i,j}) = ScaleFun_{i,j}() \cdot \mathrm{abs}(C_{i,j} - D_{i,j}) \cdot \cos(\theta) \tag{8}$$

where θ is a random number of uniform distribution between $[0, 2\pi]$.

4.2. Spherical evolution operator

Using the search pattern as Eq. (1) and spherical search style as Eq. (3) $\sim$ Eq. (8), we can propose different search operators. The second problem is how to select the appropriate individual in the population for the search task. In this paper, we adopt the hybrid methods of RSM and DISM, similar to the differential evolution algorithm. Seven spherical evolution operators can be represented by Eq. (9) $\sim$ Eq. (15), respectively. Where $m \in \{1, 2, 3\}$ denotes the three different expressions of spherical search style corresponding to the 1-dimension, 2-dimension and D-dimension (D>=3), respectively. Seven search operators (SE01$\sim$SE07) can be represented by 'SE/current-to-best/1,' 'SE/best/1,' 'SE/best/2,' 'SE/rand/1,' 'SE/rand/2,' 'SE/current/1,' and 'SE/current/2.' The 'current,' 'best,' and 'rand' denotes the 'current individual,' 'global best individual,' and 'randomly selected individuals' independently. /1 or /2 means the number of spherical search units by the randomly selected individuals.

SE/current-to-best/1

$$X_{i,j}^{new} = X_{i,j} + SS_m(X_{i,j}, X_{g,j}) + SS_m(X_{r1,j}, X_{r2,j}) \tag{9}$$

SE/best/1

$$X_{i,j}^{new} = X_{g,j} + SS_m(X_{r1,j}, X_{r2,j}) \tag{10}$$

SE/best/2

$$X_{i,j}^{new} = X_{g,j} + SS_m(X_{r1,j}, X_{r2,j}) + SS_m(X_{r3,j}, X_{r4,j}) \tag{11}$$

SE/rand/1

$$X_{i,j}^{new} = X_{r1,j} + SS_m(X_{r2,j}, X_{r3,j}) \tag{12}$$

SE/rand/2

$$X_{i,j}^{new} = X_{r1,j} + SS_m(X_{r2,j}, X_{r3,j}) + SS(X_{r4,j}, X_{r5,j}) \tag{13}$$

SE/current/1

$$X_{i,j}^{new} = X_{i,j} + SS_m(X_{r2,j}, X_{r3,j}) \tag{14}$$

SE/current/2

$$X_{i,j}^{new} = X_{i,j} + SS_m(X_{r2,j}, X_{r3,j}) + SS_m(X_{r4,j}, X_{r5,j}) \tag{15}$$

where $r1$, $r2$, and $r3$ are three randomly selected integers from 1 population size. The index of an individual is i, and j is the index of dimension.

Another problem is how to select appropriate dimensions for the spherical search. As we know, DE algorithm uses the binomial recombination method or exponential recombination method to select appropriate dimensions corresponding to the hypercube search style. In contrast to the hypercube style, the spherical search style has a larger search region. Therefore, it requires a different dimension selection method. In this paper, we propose a randomly selected dimension method like the randomly selected individual method. First, we set a parameter $DSF \in \{1, 2, \ldots, Dim\}$; Dim is the maximum size of the dimension of an individual. Second, we randomly select p dimensions ($p \in \{1, 2, \ldots, DSF\}$) for the search task.

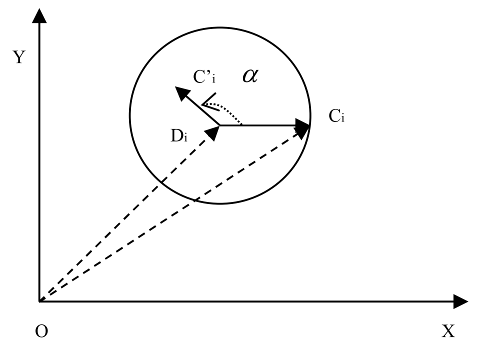


Fig. 2. Spherical search style in two-dimension space, $\theta_j (j = 1, 2, \ldots, \mathrm{dim}-2)$ is a random number of uniform distribution between $[0, \pi]$, and $\theta_j (j = \mathrm{dim}-1)$ is a random number of uniform distribution between $[0, 2\pi]$.

4.3. Analysis of spherical evolution algorithm

A spherical evolution algorithm is a simple and efficient heuristic approach. The search mechanism of SE/rand/1(SE04) is explained in Fig. 3. The three solution vectors randomly selected from the population are X_{r1}, X_{r2} and X_{r3}. The initial solution of the search is X_{r1}. The updating unit is achieved through the spherical search style (that is, a circular search in a two-dimensional space). If we search the region of a circle by its radius $|X_{r2}X_{r3}|X$, we can obtain a new vector (blue line with arrow). Then, a new solution vector X_{new} is produced, which replaces the old solution X_{old} in the population. Adjusting the radius and angle of the circle, we can search the whole region of it. The source codes of two applied examples can be downloaded from the website: https://ww2.mathworks.cn/matlabcentral/fileexchange/70993-spherical-evolution-se.

Pseudo code of SE/rand/1(SE04) is as follows (see Fig. 4):

4.4. Theoretical analysis of spherical search style

To analyze the spherical search style, the following questions must be answered: Can a search operator using the spherical search style obtain the same position (solution) produced by the same search operator using the hypercube search style or reduced hypercube search style? Does the spherical search style have a larger search space than the hypercube search styles? Take these questions into consideration, the theoretical analysis is as follows:

Definition 1. Search space can be represented as:

$$S^d = S \times S \times \cdots \times S = \{(x_1, x_2, \ldots, x_d) | x_i \in [u, v],\ u \in R, v \in R, i = 1, 2, \ldots, d\} \tag{16}$$

where d denotes the dimension $d \geq 2$, $u < v$.

Definition 2. Suppose $\vec{a}, \vec{b} \in S^d$, individuals $\vec{a}, \vec{b}$ search by a basic hypercube search style in t_k time can be represented as

$$\vec{a}(t_k) = \vec{a}(t_{k-1}) + \vec{r} \cdot (\vec{b}(t_{k-1}) - \vec{a}(t_{k-1})) \tag{17}$$

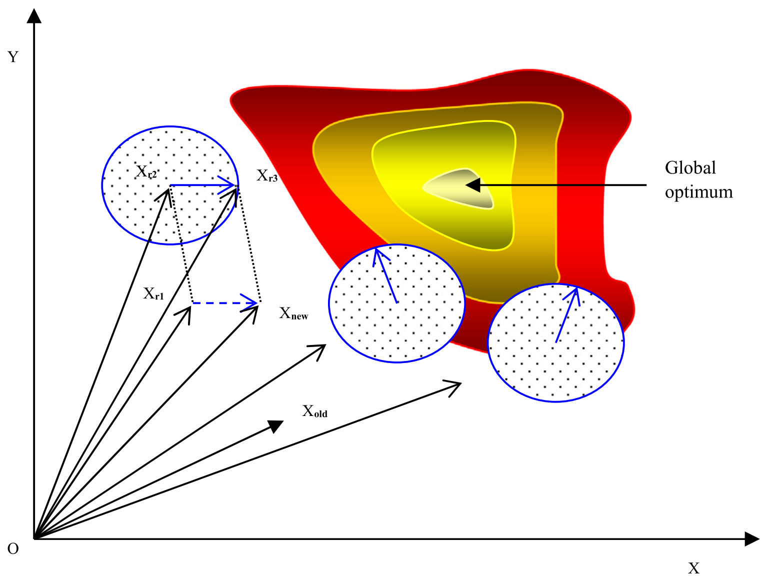


Fig. 3. Spherical search process.

where $\vec{r} = \{r_j | r_j \sim rand[0,1], j \in M\}$, $rand[0,1]$ represents a uniformly distributed random variable within the range [0,1].

$$\vec{a} = \{a_j | j \in M\} \tag{18}$$

$$\vec{b} = \{b_j | j \in M\} \tag{19}$$

$$M = \{c_i | i = 1, 2, \ldots., m, m \le d\} \tag{20}$$

c_i is an index (integer) randomly selected from 1 to d, and d denotes the dimension. In other words, c_i represents a discrete, uniformly distributed random variable within the set $T = \{1, 2, \ldots, d\}$ and $M \subseteq T$. If m is equal to d, formula (17) represents the basic hypercube search style. Otherwise, if m is less than d, formula (17) represents the reduced hypercube search style.

Definition 3. Suppose $\vec{a}, \vec{b} \in S^d$, individuals $\vec{a}, \vec{b}$ search by a basic spherical search style in t_k time can be represented as:

$$a_1(t_k) = r_j \cdot \|\vec{a}_*(t_{k-1}) - \vec{b}_*(t_{k-1})\|_2 \cdot \prod_{k=j}^{m-1} sin(\theta_j), j = 1 \tag{21}$$

$$a_j(t_k) = r_j \cdot \|\vec{a}_*(t_{k-1}) - \vec{b}_*(t_{k-1})\|_2 \cdot \cos(\theta_{j-1}) \cdot \prod_{k=j}^{m-1} sin(\theta_j),\ 1 < j \le m-1 \tag{22}$$

$$a_m(t_k) = r_j \cdot \|\vec{a}_*(t_{k-1}) - \vec{b}_*(t_{k-1})\|_2 . cos(\theta_{j-1}), j = m \tag{23}$$

As mentioned above, r_j represents a uniformly distributed random variable within the range [0, 1]. In addition, $\theta_j (j = 1, 2, \ldots, m-2)$ is a random number of uniform distribution between $[0, \pi]$, and $\theta_j (j = m-1)$ is a random number of uniform distribution between $[0, 2\pi]$. The index sequence in set M is denoted by *, and m denotes the selection dimension number and $j = c_i (c_i \in M)$ as formula (20). If m is equal to d, formulas (21) (22) (23) represent the basic hyperspherical search style (d>=3). Otherwise, if m is less than d *(m>=3)*, formulas (21) (22) (23) represent the reduced hyperspherical search style. If m is equal to 2, the spherical search style can be represented as formulas (24) (25).

$$a_j(t_k) = r_j \cdot \|\vec{a}_*(t_{k-1}) - \vec{b}_*(t_{k-1})\|_2 \cdot sin(\theta), j = 1 \tag{24}$$

$$a_j(t_k) = r_j \cdot \|\vec{a}_*(t_{k-1}) - \vec{b}_*(t_{k-1})\|_2 \cdot \cos(\theta), j = 2 \tag{25}$$

θ is a random number of uniform distribution between $[0, 2\pi]$.

In a special case, if m is equal to 1, the spherical search style can be represented as formula (26).

$$a_j(t_k) = r_j \cdot \text{abs}(\vec{a}_*(t_{k-1}) - \vec{b}_*(t_{k-1})) \cdot \cos(\theta), j = 1 \tag{26}$$

Theorem 1. *Individual $\vec{a}(t_k)$ can obtain the same position (solution) whether it uses the hypercube search style or the reduced hypercube search style.*

Proof. Suppose an individual $\vec{a}$ is updated by a hypercube search style and obtains a position (solution) in t_k time as follows:

$$\vec{a}' = \vec{a}'_1 + \vec{a}'_2 + \cdots + \vec{a}'_i + \cdots + \vec{a}'_d \tag{27}$$

where $\vec{a}'_i = (0, 0, \ldots, 0, a'_i, 0, \ldots, 0)$, a'_i denotes a component of vector $\vec{a}'$.

$$a'_i(t_k) = a'_i(t_{k-1}) + r'_i(t_{k-1}) \cdot (b'_i(t_{k-1}) - a'_i(t_{k-1})) \tag{28}$$

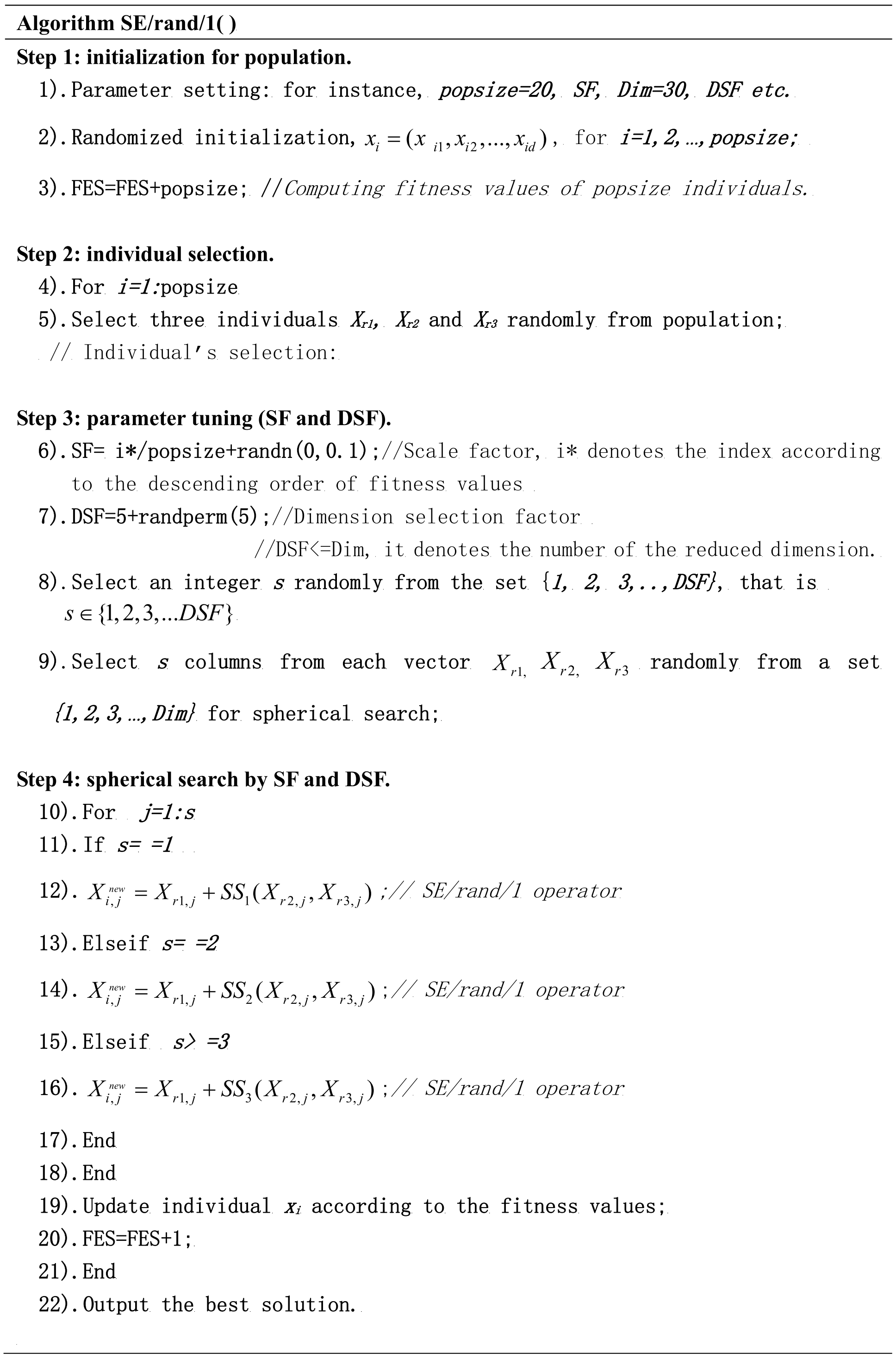
With a reduced hypercube search style, *m (m <d)* components randomly selected from individual $\vec{a}'$ can be updated in each time. This is a multi-step search method due to m components of $\vec{a}'$ being chosen by a discrete uniform distribution in {1,2,…,d}, and r_i representing a uniformly distributed random variable within the range [0,1]. It is obvious that all components of $\vec{a}'$ can be chosen, and the same a'_i value can be obtained in countless samplings. That is to say, the reduced hypercube search can produce the same position $\vec{a}'$ produced by the hypercube search style a countless number of times.

Theorem 2. *Individual $\vec{a}(t_k)$ can obtain the same position (solution) $\vec{a}'$ whether it uses the hypercube search style or the spherical search style.*

Proof. Suppose an individual $\vec{a}$ is updated by the hypercube search style and obtains a position (solution) $\vec{a}'$ in t_k time as formula (27). Then,

$$\theta_m{}^* = arccos(\frac{a_j(t_k)}{r_j \cdot \|\vec{a}_*(t_{k-1}) - \vec{b}_*(t_{k-1})\|_2}), j = m \tag{29}$$

$$\theta_{m-1}{}^* = arccos(\frac{a_j(t_k)}{r_j \cdot \|\vec{a}_*(t_{k-1}) - \vec{b}_*(t_{k-1})\|_2} \cdot \cos(\theta_m{}^*)) \tag{30}$$



```
Algorithm SE/rand/1( )
Step 1: initialization for population.
  1). Parameter setting: for instance, popsize=20, SF, Dim=30, DSF etc.
  2). Randomized initialization, $x_i = (x_{i1}, x_{i2}, ..., x_{id})$, for i=1, 2, ..., popsize;
  3). FES=FES+popsize; //Computing fitness values of popsize individuals.

Step 2: individual selection.
  4). For i=1:popsize
  5). Select three individuals $X_{r1}$, $X_{r2}$ and $X_{r3}$ randomly from population;
   // Individual's selection:

Step 3: parameter tuning (SF and DSF).
  6). SF= i*/popsize+randn(0,0.1);//Scale factor, i* denotes the index according
      to the descending order of fitness values
  7). DSF=5+randperm(5);//Dimension selection factor
                       //DSF<=Dim, it denotes the number of the reduced dimension.
  8). Select an integer s randomly from the set {1, 2, 3,.., DSF}, that is
      $s \in \{1,2,3,...DSF\}$
  9). Select s columns from each vector $X_{r1,}$ $X_{r2,}$ $X_{r3}$ randomly from a set
      {1, 2, 3, ..., Dim} for spherical search;

Step 4: spherical search by SF and DSF.
  10). For j=1:s
  11). If s= =1
  12). $X_{i,j}^{new} = X_{r1,j} + SS_1(X_{r2,j}, X_{r3,j})$ ;// SE/rand/1 operator
  13). Elseif s= =2
  14). $X_{i,j}^{new} = X_{r1,j} + SS_2(X_{r2,j}, X_{r3,j})$ ;// SE/rand/1 operator
  15). Elseif s> =3
  16). $X_{i,j}^{new} = X_{r1,j} + SS_3(X_{r2,j}, X_{r3,j})$ ;// SE/rand/1 operator
  17). End
  18). End
  19). Update individual $x_i$ according to the fitness values;
  20). FES=FES+1;
  21). End
  22). Output the best solution.
```

Fig. 4. SE/rand/1 algorithm.

$$\theta_{m-2}{}^{*} = \arcsin\Big(\frac{a_j(t_k)}{r_j \cdot \|\vec{a}_*(t_{k-1}) - \vec{b}_*(t_{k-1})\|_2} \cdot \cos(\theta_m{}^{*}) \cdot \sin(\theta_{m-1}{}^{*})\Big) \tag{31}$$

Similar to this, we can obtain another angle $\theta_{j-1}{}^{*}(m > j > 1)$, and the last angle can be computed as:

$$\theta_1^{*} = \arcsin\Big(\frac{a_j(t_k)}{r_j \cdot \|\vec{a}_*(t_{k-1}) - \vec{b}_*(t_{k-1})\|_2 \cdot \prod_{j=2}^{m-1} sin(\theta_j)}\Big) \tag{32}$$

$a_j(t_k)$ denotes a component of $\vec{a}'$, r_j is a real number produced by an uniform distribution within the range [0,1], and $\|\vec{a}_*(t_{k-1}) - \vec{b}_*(t_{k-1})\|_2$ denotes the hypersphere radius. Formulas (29) (30) (31) (32) show that we can obtain a unique angle $\theta_j{}^{*}$ for each component of $\vec{a}'$. Due to each component j of $\vec{a}'$ being chosen randomly by a discrete uniform distribution from 1 to m and $\theta_j \sim rand(0, \pi), j = 1, 2, \ldots, m-2$ and $\theta_j \sim rand(0, 2\pi), j = m-1$ are produced by a continuous uniform distribution. It is obvious that the angle $\theta_j{}^{*}$ can be obtained a countless number of times when sampling for each component of $\vec{a}'$. Therefore, the same a_i' value

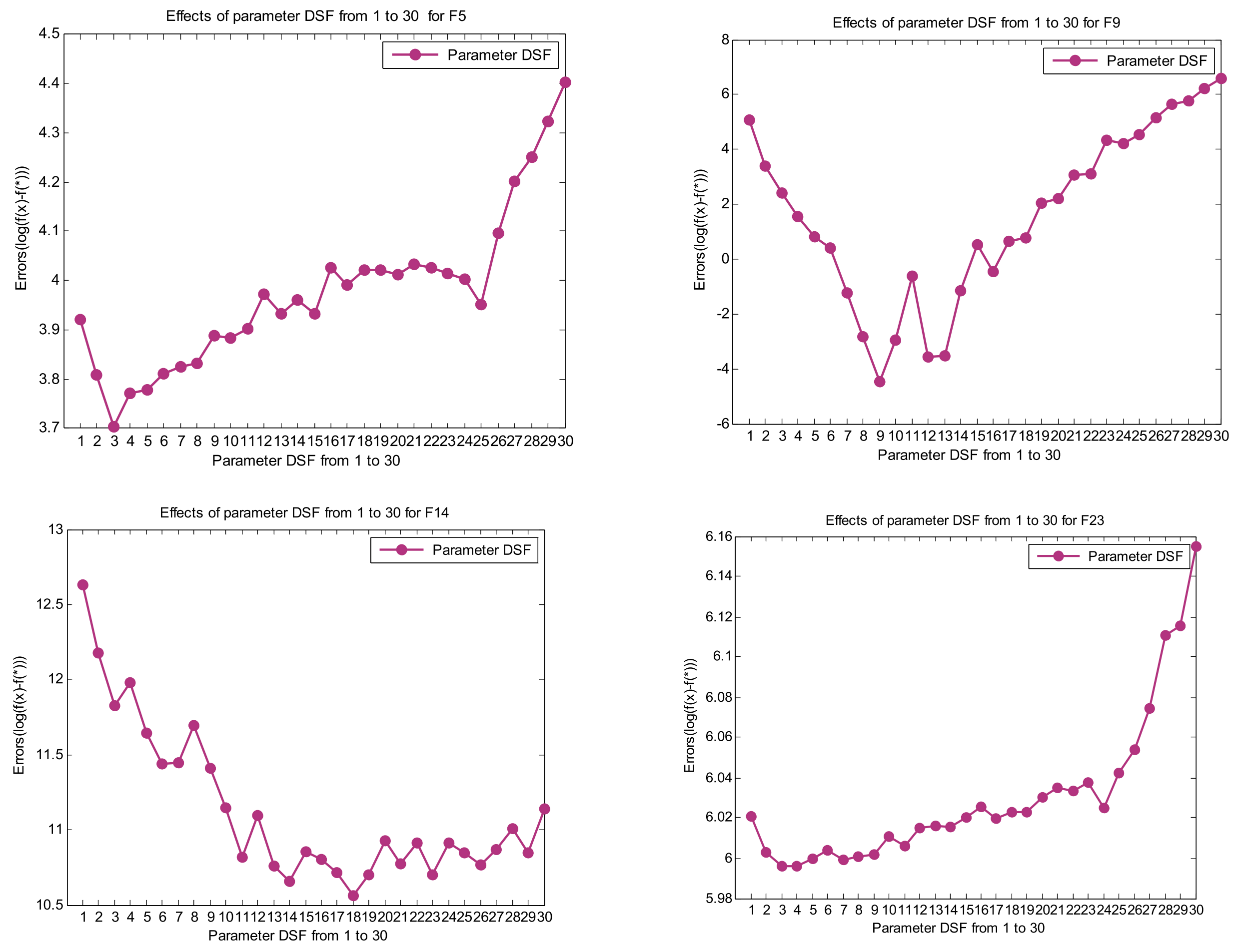


Fig. 5. Effects of parameter *DSF*.

in all the components of $\vec{a}'$ produced by the hypercube search style can also be obtained by the hyperspherical search style (d$>$ $=3$).

Similar to this, it is easy to prove that the same a_i' value in all components of $\vec{a}'$ produced by the hypercube search style can also be obtained by the spherical search style when the dimension size is equal to 1 or 2 (Eqs. (6)(7)(8)).

Theorem 3. *The scale of each component a_i' in an individual $\vec{a}(t_k)$ by spherical search style is larger than that of the hypercube search style.*

Proof. The maximum length of a component a_i' in an individual $\vec{a}(t_k)$ for a hypercube search style (hs) and spherical search style (ss) can be computed as:

$$hs = |a_i' - b_i'| \tag{33}$$

The spherical search style uses hyperspherical search style or circle search style or one-dimension search style alternately according to the size of dimension selection (m).

If $m>=3$, the hyperspherical search style is chosen to search. The maximum value of r_j, $sin(\theta_j)$ and $\cos(\theta_j)$ are equal to 1 (Eqs. (21) (22)(23)).

Therefore, the maximum length of a component a_i'

$$\begin{aligned} ss &= \|\vec{a}_*(t_{k-1}) - \vec{b}_*(t_{k-1})\|_2 \\ &= \sqrt{((a_1' - b_1')^2 + (a_2' - b_2')^2 + \cdots + (a_i' - b_i')^2 + \cdots + (a_m' - b_m')^2)} \end{aligned} \tag{34}$$

It is obvious that $ss > \sqrt{(a_i' - b_i')^2} = hs$.

Similar to this, if $m = 2$, the circle search style is chosen to search as Eqs. (6) (7). The max value of r_j, $sin(\theta_j)$ and $\cos(\theta_j)$ are equal to 1.

$$ss = \|\vec{a}_*(t_{k-1}) - \vec{b}_*(t_{k-1})\|_2 = \sqrt{(a_1' - b_1')^2 + (a_2' - b_2')^2} \tag{35}$$

$$ss > \sqrt{(a_i' - b_i')^2} = hs$$

In a special case, if $m = 1$, it is obvious that the one-dimensional search style is chosen to search as Eq. (8).

$$hs = ss = |a_i' - b_i'| \tag{36}$$

This means that the spherical search style has a larger space search space than the hypercube search style in most cases ($m>=2$). The spherical search style has the same search space as the hypercube search space only in special cases ($m=1$). This means that the spherical search style has better exploration ability than the hypercube search style.

5. Experimental results and analysis

To demonstrate the performance of the proposed algorithms, 30 benchmark functions of CEC2017 are used in the experiments. CEC 2017 benchmark suites have more complex characteristics by 3 unimodal functions (F1∼F3), 7 simple multimodal functions (F4∼F10), 10 hybrid functions (F11∼F20), and 10 composition functions (F21∼F30) [42]. More details are shown in Table 2. The performance of Spherical Evolution Algorithms (SEs) are then

Table 2
Parameters setting.

NO.	Algorithm	Parameter setting
1	ABC	SN = 10, limit = SN*2*Dimension
2	CMA-ES	σ=0.25, μ=*round*((4+*round*(2·log(*N*))/2))
3	GWO	–
4	TLBO	–
5	AAA	k = 2, Le = 0.3, ap = 0.5
6	SaDE	F ~N(0.5, 0.3), CR ~N(CRm, 0.1), mutation strategies and crossover strategies as in []
7	DE/current-to-best/1(DE01)	F = 0.5, CR = 0.9
8	DE/best/1(DE02)	F = 0.5, CR = 0.9
9	DE/best/2(DE03)	F = 0.5, CR = 0.9
10	DE/rand/1(DE04)	F = 0.5, CR = 0.9
11	SE/current-to-best/1(SE01)	–
12	SE/best/1(SE02)	–
13	SE/best/2(SE03)	–
14	SE/rand/1(SE04)	–
15	SE/rand/2(SE05)	–
16	SE/current/1(SE06)	–
17	SE/current/2(SE07)	–

compared with that of 10 state-of-the-art algorithms including four Differential Evolution Algorithms (DEs) [3], Covariance Matrix Adaptation Evolutionary Strategies (CMA-ES) [43], Artificial Bee Colony (ABC) [5], Gray Wolf Optimizer (GWO) [14], Teaching-Learning-Based Optimization (TLBO) [7], Differential Evolution Algorithm with strategy adaptation (SaDE) [44], and Artificial Algae Algorithm (AAA) [9]. Seven versions of SE are represented as SE/current-to-best/1(**SE01**), SE/best/1 (**SE02**), SE/best/2 (**SE03**), SE/rand/1 (**SE04**), SE/rand/2 (**SE05**), SE/current/1 (**SE06**), and SE/current/2 (**SE07**). Similar to SEs, four DE Algorithms are DE/current-to-best/1(**DE01**), DE/best/1 (**DE02**), DE/best/2 (**DE03**), and DE/rand/1 (**DE04**). For fair comparison, each algorithm is independently run 51 times for each function in CEC 2017. The best-so-far error (*f(x)-f(x*)*) of each run is recorded for comparison, where *f(x)* denotes the best-so-far fitness value and *f(x*)* is the global optimum. Each algorithm can run all the time if it does not exceed the limited threshold value (MAX_FES). The maximum number of the function evaluations (MAX_FES) is set to D*1E+4 according to the recommendation of CEC 2017, where D is the dimensional size. Population size is set to 20, The parameter setting of each algorithm can be found in Table 3. Traditional statistical methods such as mean and standard deviation are used in our **Tables**. To effectively analyze the experimental results obtained by the compared algorithms, two nonparametric statistical tests including T-test [45] and Friedman's test [46] with a significance level of 0.05 are utilized in the experiments. The real-world optimization problem for data clustering is solved by the proposed algorithm. The experimental comparison can be found in the last subsection.

5.1. Parameter tuning

Scale factor (SF) and dimension selection factor (DSF) are the two important parameters in SEs, which possibly affect the performance of SEs. In order to tune these parameters better, sensitivity analysis is needed. SF controls the radius size of spherical search, so it should be a real number in a range [0, 1]. DSF decides which dimension should be selected for a search, and it should be an integer from 1 to D. D is the maximum dimensional size corresponding to the optimization problem. In order to analyze the influence of parameters SF and DSF on the performance of the SE algorithms, four functions are selected to test for the SE/rand/1 (Dim = 30): F5 (simple multimodal function), F9 (simple multimodal function), F14 (hybrid functions), and F23 (composition functions). Population size and other settings are as mentioned

Table 3
CEC Suite 2017.

Func	Function name	Range	Type	Optimum
F1	Shifted and Rotated Bent Cigar Function	[−100,100]	U	100
F2	Shifted and Rotated Sum of Different Power Function	[−100,100]	U	200
F3	Shifted and Rotated Zakharov Function	[−100,100]	U	300
F4	Shifted and Rotated Rosenbrock's Function	[−100,100]	M	400
F5	Shifted and Rotated Rastrigin's Function	[−100,100]	M	500
F6	Shifted and Rotated Expanded Scaffer's F6 Function	[−100,100]	M	600
F7	Shifted and Rotated Lunacek Bi_Rastrigin Function	[−100,100]	M	700
F8	Shifted and Rotated Non-Continuous Rastrigin's Function	[−100,100]	M	800
F9	Shifted and Rotated Levy Function	[−100,100]	M	900
F10	Shifted and Rotated Schwefel's Function	[−100,100]	M	1000
F11	Hybrid Function 1 (N = 3)	[−100,100]	H	1100
F12	Hybrid Function 2 (N = 3)	[−100,100]	H	1200
F13	Hybrid Function 3 (N = 3)	[−100,100]	H	1300
F14	Hybrid Function 4 (N = 4)	[−100,100]	H	1400
F15	Hybrid Function 5 (N = 4)	[−100,100]	H	1500
F16	Hybrid Function 6 (N = 4)	[−100,100]	H	1600
F17	Hybrid Function 6 (N = 5)	[−100,100]	H	1700
F18	Hybrid Function 6 (N = 5)	[−100,100]	H	1800
F19	Hybrid Function 6 (N = 5)	[−100,100]	H	1900
F20	Hybrid Function 6 (N = 6)	[−100,100]	H	2000
F21	Composition Function 1 (N = 3)	[−100,100]	C	2100
F22	Composition Function 2 (N = 3)	[−100,100]	C	2200
F23	Composition Function 3 (N = 4)	[−100,100]	C	2300
F24	Composition Function 4 (N = 4)	[−100,100]	C	2400
F25	Composition Function 5 (N = 5)	[−100,100]	C	2500
F26	Composition Function 6 (N = 5)	[−100,100]	C	2600
F27	Composition Function 7 (N = 6)	[−100,100]	C	2700
F28	Composition Function 8 (N = 6)	[−100,100]	C	2800
F29	Composition Function 9 (N = 3)	[−100,100]	C	2900
F30	Composition Function 10(N = 3)	[−100,100]	C	3000

above. First, we set SF = 0.5, and then tune all options of DSF from 1 to 30. Mean and standard deviation are recorded in Table 4. The black font indicates the winner of SE/rand/1 by the DSF. It can be observed that SE/rand/1 obtained the better performance when DSF is equal to 3 for F5, 4 for F23, 9 for F9, and 18 for F14, respectively. Fig. 5 shows the errors of SE/rand/1 when DSF is 1 to 30. It can be observed that better DSF values are located in a 'mountain valley' for the four optimization problems. For F5 function, SE/rand/1 obtains the global best points in a 'mountain valley' when DSF = 3. For F9 function, SE/rand/1 obtains more local best points in many 'mountain valleys' when DSF = 9 or 12 or 13. For F14 function, SE/rand/1 obtains the global best point or a local best point in a 'mountain valley' when DSF = 18 or DSF = 14. For 23 function, SE/rand/1 obtains the global best point in a 'many valley' when DSF = 3 or 4. It seems that the DSF value can be chosen from a set as {3, 4, 9, 12, 13, 14, 18}. However, DSF should not be set as a fixed value such as DSF = 3. If DSF = 3, we can find that F9 and F14 cannot obtain satisfactory performance. Therefore, DSF should be set in a suitable range for different optimization problems. *DSF* function can be represented as Eq. (37), when the dimensional size of the optimization problem is greater than 10.

$$DSF = 5 + randperm\,(5) \tag{37}$$

In other cases, *DSF* function can be represented as a constant such as DSF = 3, when the dimensional size of the optimization problem is less than or equal to 10.

Table 4
Comparison results of parameter *DSF*.

Func	F5 Means	Std.	F9 Means	Std.	F14 Mean	Std.	F23 Means	Std.
$DSF = 1$	5.0460e+01	8.0455e+00	1.6230e+02	1.3770e+02	3.0477e+05	2.3486e+05	4.1190e+02	1.0544e+01
$DSF = 2$	4.5116e+01	8.0321e+00	3.0235e+01	3.1419e+01	1.9451e+05	1.3827e+05	4.0460e+02	8.9156e+00
$DSF = 3$	**4.0559e+01**	**7.2624e+00**	1.1192e+01	1.0800e+01	1.3715e+05	9.9555e+04	4.0192e+02	7.8115e+00
$DSF = 4$	4.3469e+01	7.0776e+00	4.8131e+00	5.5086e+00	1.6013e+05	1.5895e+05	**4.0174e+02**	**7.4911e+00**
$DSF = 5$	4.3792e+01	7.8979e+00	2.2957e+00	4.4155e+00	1.1377e+05	1.0373e+05	4.0340e+02	8.7359e+00
$DSF = 6$	4.5187e+01	7.1952e+00	1.5036e+00	3.8448e+00	9.3113e+04	5.6845e+04	4.0511e+02	8.4384e+00
$DSF = 7$	4.5887e+01	6.8554e+00	2.9233e−01	7.8379e−01	9.3861e+04	6.9599e+04	4.0308e+02	9.0522e+00
$DSF = 8$	4.6179e+01	6.9922e+00	5.9000e−02	2.6592e−01	1.1958e+05	8.8538e+04	4.0376e+02	9.3080e+00
$DSF = 9$	4.8829e+01	7.5178e+00	**1.1295e−02**	**3.7327e−02**	9.0336e+04	9.0104e+04	4.0417e+02	9.2169e+00
$DSF = 10$	4.8606e+01	7.2904e+00	5.2551e−02	2.5527e−01	6.9385e+04	5.8863e+04	4.0780e+02	9.4835e+00
$DSF = 11$	4.9572e+01	8.0661e+00	5.4124e−01	3.7234e+00	5.0014e+04	3.6339e+04	4.0582e+02	8.7566e+00
$DSF = 12$	5.3152e+01	6.0187e+00	2.8375e−02	1.3588e−01	6.6080e+04	5.4296e+04	4.0963e+02	1.0348e+01
$DSF = 13$	5.1077e+01	7.9486e+00	2.9501e−02	1.3013e−01	4.7204e+04	4.1831e+04	4.0997e+02	1.0232e+01
$DSF = 14$	5.2502e+01	7.9572e+00	3.1993e−01	1.1333e+00	4.2477e+04	3.8139e+04	4.0975e+02	1.0652e+01
$DSF = 15$	5.1015e+01	7.9780e+00	1.6903e+00	9.8472e+00	5.1783e+04	4.2214e+04	4.1176e+02	8.7099e+00
$DSF = 16$	5.6081e+01	8.5227e+00	6.4829e−01	1.6256e+00	4.9011e+04	4.3783e+04	4.1397e+02	1.1565e+01
$DSF = 17$	5.4104e+01	7.5637e+00	1.9291e+00	4.7453e+00	4.5129e+04	4.4547e+04	4.1154e+02	9.1885e+00
$DSF = 18$	5.5801e+01	8.8963e+00	2.1980e+00	4.8383e+00	**3.8504e+04**	**3.8935e+04**	4.1270e+02	1.0481e+01
$DSF = 19$	5.5845e+01	9.7021e+00	7.8393e+00	1.7806e+01	4.4562e+04	4.7543e+04	4.1287e+02	1.2121e+01
$DSF = 20$	5.5285e+01	1.0328e+01	9.0336e+00	1.9395e+01	5.5734e+04	5.0258e+04	4.1587e+02	1.1721e+01
$DSF = 21$	5.6504e+01	1.0300e+01	2.1702e+01	3.5042e+01	4.7905e+04	4.2843e+04	4.1772e+02	1.2786e+01
$DSF = 22$	5.5993e+01	1.1386e+01	2.2410e+01	2.7053e+01	5.4963e+04	4.5304e+04	4.1704e+02	1.5642e+01
$DSF = 23$	5.5462e+01	1.0987e+01	7.5828e+01	1.1780e+02	4.4343e+04	4.5068e+04	4.1900e+02	1.8712e+01
$DSF = 24$	5.4796e+01	1.2370e+01	6.7317e+01	8.7480e+01	5.5017e+04	5.2926e+04	4.1357e+02	1.3837e+01
$DSF = 25$	5.2038e+01	1.2324e+01	9.4098e+01	1.2238e+02	5.1576e+04	4.4275e+04	4.2088e+02	2.4813e+01
$DSF = 26$	6.0142e+01	1.6821e+01	1.7275e+02	1.6073e+02	4.7413e+04	3.6921e+04	4.2582e+02	1.8912e+01
$DSF = 27$	6.6743e+01	1.6982e+01	2.7907e+02	2.0751e+02	5.2496e+04	4.9217e+04	4.3462e+02	2.2731e+01
$DSF = 28$	7.0118e+01	1.9726e+01	3.1936e+02	1.9289e+02	6.0392e+04	4.9793e+04	4.5077e+02	3.0475e+01
$DSF = 29$	7.5465e+01	1.8510e+01	5.0340e+02	2.9314e+02	5.1420e+04	4.1257e+04	4.5272e+02	3.2617e+01
$DSF = 30$	8.1715e+01	2.1709e+01	7.1832e+02	4.5475e+02	6.8798e+04	5.0534e+04	4.7105e+02	3.8919e+01

To analyze the performance of SE/rand/1 on the parameter SF, we chose a fixed DSF value (such as DSF = 14) from a set {3, 4, 9, 12, 13, 14, 18} as mentioned previously. Table 5 shows the performance of SE/rand/1 for SF from 0.1 to 1. The black fonts indicate the winner of SE/rand/1 by the SF. It can be observed that SE/rand/1 obtains better performance for F5, F9, F14, and F21 when SF is equal to 0.2, 0.3, 0.3 and 0.6, respectively. Fig. 6 shows the errors of SE/rand/1 for ten SF values from 0.1 to 1.0. It can be observed that SF should be set as a small value (such as 0.2 or 0.3) for some optimization problems such as F5, F14, and F21. In contrast to them, SF can be set as a large value (such as 0.6 or 0.7) for some optimization problems, such as F9 and F14. Different optimization problems should set different SF values. In addition, we know that SF controls the length of the search directly. Therefore, different individuals should use the different SF values. For instance, the global best individual should be set as the minimal SF value, and the bad individuals should be set at the maximal SF. Considering SF is a real number from 0 to 1, SF can be represented as Eq. (38).

$$SF_i = i * /popsize, i = 1, 2, \ldots, popsize \tag{38}$$

where $i*$ denotes the rank of each individual in the population according to the fitness values. In fact, relative to the superior individuals, the inferior individuals may be closer to the global minimum.

Based on this reasoning, we randomize the parameter SF_i using a normal distribution with the mean specified to the original value and the standard deviation specified to 0.1.

5.2. Comparison of SE and other state-of-the-art algorithm

In this subsection, we compare the SE/rand/1(SE04) to the six state-of-the-art algorithms including CMAES, SaDE, ABC, GWO, TLBO, and AAA. Comparison results can be seen in Table 6. In contrast to CMAES, the performance of SE/rand/1 outperforms that of it for 15 functions including 6 simple multimodal functions, 4 hybrid functions and 5 composition functions. The performance of CMAES outperforms that of SE/rand/1 for 3 unimodal functions, 1 simple multimodal function, 6 hybrid functions, and 5 composition functions. It can be observed that SE/rand/1 has stronger ability to solve more different multimodal optimization problems than CMAES. In contrast to SaDE, the performance of SE/rand/1 outperforms that of it for 16 functions, including 5 simple multimodal functions, 4 hybrid functions, and 7 composition functions. In contrast to ABC, the performance of SE/rand/1 outperforms that of it for 17 optimization problems, including 1 unimodal function, 5 simple multimodal functions, 6 hybrid functions, and 5 composition functions. In contrast to GWO, the performance of SE/rand/1 significantly exceeds that of it except for F3, F14, F29 and F30. In contrast to TLBO, the performance of SE/rand/1 outperforms that of it for 21 functions, including 3 unimodal functions, 6 simple multimodal functions, 7 hybrid functions, and 5 composition functions. In contrast to AAA, the performance of SE/rand/1 outperforms that of it for 22 functions, including 2 unimodal functions, 4 simple multimodal functions, 9 hybrid functions, and 7 composition functions. The last line of Table 6 shows the number of winner, loser or equation of SE/rand/1 as 15/15/0, 17/13/0, 26/4/0, 21/9/0, 22/8/0 and 16/14/0 for CMAES, ABC, GWO, TLBO, AAA and SaDE, respectively. This means that the performance of SE/rand/1 exceeds that of the other 6 algorithms. To test the scalability of SE/rand/1, we increase the dimension to 50. The comparison results can be seen in Table 7. It can be observed that the performance of SE/rand/1 still outperforms the CMAES, ABC, GWO, TLBO, AAA and SaDE for many functions like those in Table 6 with a few exceptions. The last line shows the number of the winner, loser and equation of SE/rand/1 for CMAES, ABC, GWO, TLBO and AAA algorithms corresponding to 14/16/0, 27/3/0, 23/7/0, 21/9/0 and 16/14/0.

5.3. Comparison of SEs

In this subsection, we compare the performance of seven SE algorithms including SE/current-to-best(SE01), SE/best/1(SE02), SE/best/2(SE03), SE/rand/1(SE04), SE/rand/2(SE05), SE/current/1

Table 5
Comparison results of parameter *SF*.

Func	F5 Means	Std.	F9 Means	Std.	F14 Means	Std.	F23 Means	Std.
$SF = 0.1$	5.0778e+01	1.0570e+01	3.9043e+02	1.7402e+02	1.2089e+05	2.3801e+05	4.1255e+02	1.7689e+01
$SF = 0.2$	**4.7136e+01**	**8.3373e+00**	1.0201e+02	8.2682e+01	4.0632e+04	4.4672e+04	4.1487e+02	1.7131e+01
$SF = 0.3$	4.8229e+01	7.9830e+00	2.2891e+01	2.6737e+01	**3.5821e+04**	**3.2563e+04**	**4.0569e+02**	**1.2704e+01**
$SF = 0.4$	5.1030e+01	8.2552e+00	4.3717e+00	1.2834e+01	4.6952e+04	3.9039e+04	4.0746e+02	1.1873e+01
$SF = 0.5$	5.4113e+01	7.5650e+00	1.7463e−01	4.9050e−01	5.0281e+04	4.6120e+04	4.1081e+02	9.6421e+00
$SF = 0.6$	5.6746e+01	8.8507e+00	**2.1329e−02**	**9.8692e−02**	5.4990e+04	5.5465e+04	4.1442e+02	8.6789e+00
$SF = 0.7$	5.9058e+01	7.3051e+00	4.8684e−01	1.6399e+00	4.3706e+04	3.0911e+04	4.1751e+02	8.8038e+00
$SF = 0.8$	6.3011e+01	6.5778e+00	2.8144e+00	6.2970e+00	4.6382e+04	4.2665e+04	4.1852e+02	8.9242e+00
$SF = 0.9$	6.6030e+01	7.3960e+00	8.8193e+00	8.7475e+000	4.7178e+04	3.8923e+04	4.2254e+02	8.6673e+00
$SF = 1.0$	7.0600e+01	9.3597e+00	2.2657e+01	1.9129e+01	5.8683e+04	4.9192e+04	4.2597e+02	9.3226e+00

Table 6
Comparison results of SEs and other state-of-the-art-algorithms (Dimension = 30).

Func	CMAES	SE04	ABC	SE04	GWO	SE04	TLBO	SE04	AAA	SE04	SaDE	SE04
F1Means	**1.6719E-15**	3.2930E+03	**3.0676e+002**	3.2930E+03	7.3049E+08	**3.2930E+03**	3.9572e+003	**3.2930E+03**	3.4951e+003	**3.2930E+03**	**3.0714E+03**	3.2930E+03
Std	**4.6242E-15**	4.2328E+03	**4.8995e+002**	4.2328E+03	6.4952E+08	**4.2328E+03**	4.6128e+003	**4.2328E+03**	5.6711e+003	**4.2328E+03**	**3.5072E+03**	4.2328E+03
F2 Means	**0.0000E+00**	3.0802E+13	**3.9753e+007**	3.0802E+13	1.2812E+26	**3.0802E+13**	6.3107e+025	**3.0802E+13**	**1.2150e+004**	3.0802E+13	**8.6275E-01**	3.0802E+13
Std	**0.0000E+00**	1.1694E+14	**2.2321e+008**	1.1694E+14	8.0603E+26	**1.1694E+14**	2.5869e+026	**1.1694E+14**	**5.4409e+004**	1.1694E+14	**4.9357E+00**	1.1694E+14
F3 Means	**1.2260E-14**	9.7974E+03	1.2830e+005	**9.7974E+03**	**3.4885E+03**	9.7974E+03	**2.2485e-005**	9.7974E+03	1.1926e+004	**9.7974E+03**	**3.0045E+02**	9.7974E+03
Std	**2.3612E-14**	3.4377E+03	2.8179e+004	**3.4377E+03**	**2.3985E+03**	3.4377E+03	**1.4115e-004**	3.4377E+03	8.5233e+003	**3.4377E+03**	**7.3017E+02**	3.4377E+03
F4 Means	**2.6340E+01**	8.5881E+01	**3.7728e+001**	8.5881E+01	1.5050E+02	**8.5881E+01**	**7.3327e+001**	8.5881E+01	**3.8675e+001**	8.5881E+01	**6.0423E+01**	8.5881E+01
Std	**2.8846E+01**	1.1251E+01	**3.1512e+001**	1.1251E+01	3.2951E+01	**1.1251E+01**	**2.7216e+001**	1.1251E+01	**3.3380e+001**	1.1251E+01	**2.9825E+01**	1.1251E+01
F5 Means	7.2199E+02	**4.1688E+01**	9.8155e+001	**4.1688E+01**	1.1524E+02	**4.1688E+01**	1.2841e+002	**4.1688E+01**	1.0167e+002	**4.1688E+01**	5.6192E+01	**4.1688E+01**
Std	1.7548E+02	**8.1545E+00**	1.9322e+001	**8.1545E+00**	5.8574E+01	**8.1545E+00**	2.9532e+001	**8.1545E+00**	2.9942e+001	**8.1545E+00**	1.4216E+01	**8.1545E+00**
F6 Means	9.7782E+01	**7.5481E-06**	**1.1146e-014**	7.5481E-06	5.9782E+00	**7.5481E-06**	2.4527e+001	**7.5481E-06**	**2.6841e-009**	7.5481E-06	8.9317E-02	**7.5481E-06**
Std	1.2586E+01	**3.9880E-05**	**3.4143e-014**	3.9880E-05	2.4669E+00	**3.9880E-05**	6.5207e+000	**3.9880E-05**	**1.9166e-008**	3.9880E-05	1.3955E-01	**3.9880E-05**
F7 Means	3.7423E+03	**7.2448E+01**	1.1192e+002	**7.2448E+01**	1.9317E+02	**7.2448E+01**	2.3949e+002	**7.2448E+01**	1.3036e+002	**7.2448E+01**	9.4945E+01	**7.2448E+01**
Std	1.3060E+03	**7.3495E+00**	1.3485e+001	**7.3495E+00**	5.0185E+01	**7.3495E+00**	4.7785e+001	**7.3495E+00**	2.6156e+001	**7.3495E+00**	1.9879E+01	**7.3495E+00**
F8 Means	5.8163E+02	**4.4194E+01**	1.0927e+002	**4.4194E+01**	9.0562E+01	**4.4194E+01**	1.0244e+002	**4.4194E+01**	9.4010e+001	**4.4194E+01**	5.3942E+01	**4.4194E+01**
Std	1.4821E+02	**6.5834E+00**	1.7445e+001	**6.5834E+00**	4.5530E+01	**6.5834E+00**	2.2617e+001	**6.5834E+00**	2.3230e+001	**6.5834E+00**	1.2792E+01	**6.5834E+00**
F9 Means	1.4560E+04	**3.0839E-01**	1.1725e+003	**3.0839E-01**	1.9870E+02	**3.0839E-01**	1.3967e+003	**3.0839E-01**	1.7190e+003	**3.0839E-01**	8.3556E+01	**3.0839E-01**
Std	2.9753E+03	**8.4139E-01**	6.7223e+002	**8.4139E-01**	1.2369E+02	**8.4139E-01**	6.5883e+002	**8.4139E-01**	9.7132e+002	**8.4139E-01**	6.2643E+01	**8.4139E-01**
F10Means	5.0532E+03	**2.3267E+03**	2.4491e+003	**2.3267E+03**	6.3735E+03	**2.3267E+03**	5.0120e+003	**2.3267E+03**	2.7665e+003	**2.3267E+03**	**2.3253E+03**	2.3267E+03
Std	7.1430E+02	**2.8457E+02**	2.9333e+002	**2.8457E+02**	1.3443E+03	**2.8457E+02**	1.3827e+003	**2.8457E+02**	5.5761e+002	**2.8457E+02**	**4.9247E+02**	2.8457E+02
F11Means	1.7509E+02	**4.1343E+01**	4.0776e+002	**4.1343E+01**	1.9046E+02	**4.1343E+01**	1.4733e+002	**4.1343E+01**	6.9151e+001	**4.1343E+01**	1.0032E+02	**4.1343E+01**
Std	6.5756E+01	**2.7994E+01**	4.4656e+002	**2.7994E+01**	4.2964E+01	**2.7994E+01**	5.5291e+001	**2.7994E+01**	3.3418e+001	**2.7994E+01**	4.3101E+01	**2.7994E+01**
F12Means	**1.4872E+03**	1.1143E+06	**7.7482e+005**	1.1143E+06	4.8575E+07	**1.1143E+06**	**3.7900e+004**	1.1143E+06	**3.2116e+005**	1.1143E+06	**6.8629E+04**	1.1143E+06
Std	**4.9023E+02**	8.1422E+05	**5.2106e+005**	8.1422E+05	4.9276E+07	**8.1422E+05**	**3.3616e+004**	8.1422E+05	**2.8826e+005**	8.1422E+05	**3.8252E+04**	8.1422E+05
F13Means	**1.7035E+03**	4.6063E+03	**4.4534e+003**	4.6063E+03	8.9123E+06	**4.6063E+03**	1.3581e+004	**4.6063E+03**	1.5874e+004	**4.6063E+03**	1.1211E+04	**4.6063E+03**
Std	**7.1605E+02**	4.8590E+03	**4.5213e+003**	4.8590E+03	2.4288E+07	**4.8590E+03**	1.5260e+004	**4.8590E+03**	1.8048e+004	**4.8590E+03**	1.0535E+04	**4.8590E+03**
F14Means	**2.0354E+02**	7.1204E+04	1.2560e+005	**7.1204E+04**	**6.8160E+04**	7.1204E+04	**2.4806e+003**	7.1204E+04	7.8786e+004	**7.1204E+04**	**4.3238E+03**	7.1204E+04
Std	**5.8421E+01**	5.9323E+04	1.2771e+005	**5.9323E+04**	**5.5965E+04**	5.9323E+04	**2.9738e+003**	5.9323E+04	9.7179e+004	**5.9323E+04**	**5.7159E+03**	5.9323E+04
F15Means	**2.7588E+02**	2.2013E+03	**1.4138e+003**	2.2013E+03	1.2585E+05	**2.2013E+03**	5.9838e+003	**2.2013E+03**	1.0041e+004	**2.2013E+03**	**2.1676E+03**	2.2013E+03
Std	**1.0901E+02**	1.9756E+03	**1.7692e+003**	1.9756E+03	1.3592E+05	**1.9756E+03**	8.5705e+003	**1.9756E+03**	1.1527e+004	**1.9756E+03**	**3.0178E+03**	1.9756E+03

(*continued on next page*)

(SE06), and SE/current/2(SE07). Comparison results can be seen in Table 8. Black fonts denote the winner and the last line shows the winner numbers of seven SE algorithms. In Table 8, the winner numbers of seven SE algorithms are 2, 1, 0, 15, 2, 8 and 2 for SE01, SE02, SE03, SE04, SE05, SE06 and SE07, respectively. It can be observed that SE04 obtains better performance than others for 15 multimodal functions. This means that SE04 has stronger exploring ability, especially for multimodal optimization problems. SE06 obtains better performance than the other 6 algorithms for the 2 unimodal functions and 6 multimodal functions. It means that SE06 has better local search ability and global search ability for solving some unimodal or multimodal problems. SE 02 and SE03 are global-best-guided operators, which have better local

Table 6 (*continued*).

Func	CMAES	SE04	ABC	SE04	GWO	SE04	TLBO	SE04	AAA	SE04	SaDE	SE04
F16Means	5.1900E+02	**4.9392E+02**	7.7828e+002	**4.9392E+02**	8.0842E+02	**4.9392E+02**	8.0684e+002	**4.9392E+02**	8.7681e+002	**4.9392E+02**	5.6072E+02	**4.9392E+02**
Std	3.1378E+02	**1.7309E+02**	2.0959e+002	**1.7309E+02**	4.4051E+02	**1.7309E+02**	2.7046e+002	**1.7309E+02**	2.3576e+002	**1.7309E+02**	2.0850E+02	**1.7309E+02**
F17Means	2.7372E+02	**1.4116E+02**	2.8491e+002	**1.4116E+02**	2.9552E+02	**1.4116E+02**	3.3876e+002	**1.4116E+02**	3.3118e+002	**1.4116E+02**	**8.7684E+01**	1.4116E+02
Std	1.7005E+02	**8.5026E+01**	1.1468e+002	**8.5026E+01**	2.0325E+02	**8.5026E+01**	1.5507e+002	**8.5026E+01**	1.4803e+002	**8.5026E+01**	**9.1289E+01**	8.5026E+01
F18Means	**1.9828E+02**	2.1361E+05	2.3146e+005	**2.1361E+05**	7.8490E+05	**2.1361E+05**	**1.5400e+005**	2.1361E+05	3.2853e+005	**2.1361E+05**	**1.0034E+05**	2.1361E+05
Std	**7.5970E+01**	1.3261E+05	1.6093e+005	**1.3261E+05**	7.5453E+05	**1.3261E+05**	**1.0854e+005**	1.3261E+05	2.7726e+005	**1.3261E+05**	**1.1019E+05**	1.3261E+05
F19Means	**2.1420E+02**	2.0723E+03	**1.1550e+003**	2.0723E+03	5.7327E+05	**2.0723E+03**	7.7199e+003	**2.0723E+03**	8.7457e+003	**2.0723E+03**	5.9612E+03	**2.0723E+03**
Std	**1.1798E+02**	2.1685E+03	**1.3123e+003**	2.1685E+03	8.2020E+05	**2.1685E+03**	9.1804e+003	**2.1685E+03**	1.0674e+004	**2.1685E+03**	7.1112E+03	**2.1685E+03**
F20Means	1.3932E+03	**1.7303E+02**	3.1742e+002	**1.7303E+02**	3.5915E+02	**1.7303E+02**	2.8789e+002	**1.7303E+02**	3.8878e+002	**1.7303E+02**	**1.2989E+02**	1.7303E+02
Std	3.1892E+02	**7.2015E+01**	1.1947e+002	**7.2015E+01**	1.7016E+02	**7.2015E+01**	8.8240e+001	**7.2015E+01**	1.4706e+002	**7.2015E+01**	**7.0970E+01**	7.2015E+01
F21Means	5.7411E+02	**2.5047E+02**	2.9453e+002	**2.5047E+02**	2.9224E+02	**2.5047E+02**	3.1177e+002	**2.5047E+02**	2.9898e+002	**2.5047E+02**	**2.4896E+02**	2.5047E+02
Std	2.8419E+02	**8.4442E+00**	5.2990e+001	**8.4442E+00**	4.4992E+01	**8.4442E+00**	3.1881e+001	**8.4442E+00**	2.6154e+001	**8.4442E+00**	**1.3195E+01**	8.4442E+00
F22Means	5.8972E+03	**1.0211E+03**	1.9040e+003	**1.0211E+03**	5.4027E+03	**1.0211E+03**	**5.0774e+002**	1.0211E+03	1.8408e+003	**1.0211E+03**	1.0228E+02	**1.0211E+03**
Std	8.6788E+02	**1.2872E+03**	1.4741e+003	**1.2872E+03**	2.4205E+03	**1.2872E+03**	**1.4758e+003**	1.2872E+03	1.5246e+003	**1.2872E+03**	3.2279E+00	**1.2872E+03**
F23Means	1.9082E+03	**4.0247E+02**	4.3396e+002	**4.0247E+02**	4.8213E+02	**4.0247E+02**	5.2860e+002	**4.0247E+02**	4.4544e+002	**4.0247E+02**	4.1472E+02	**4.0247E+02**
Std	8.9273E+02	**8.1687E+00**	2.7182e+001	**8.1687E+00**	6.0917E+01	**8.1687E+00**	5.1158e+001	**8.1687E+00**	2.0444e+001	**8.1687E+00**	1.8742E+01	**8.1687E+00**
F24Means	**4.6210E+02**	4.9840E+02	5.7108e+002	**4.9840E+02**	5.5032E+02	**4.9840E+02**	5.7628e+002	**4.9840E+02**	5.7662e+002	**4.9840E+02**	**4.8169E+02**	4.9840E+02
Std	**1.1443E+01**	1.3899E+01	1.5897e+002	**1.3899E+01**	6.2889E+01	**1.3899E+01**	4.1010e+001	**1.3899E+01**	3.0236e+001	**1.3899E+01**	**2.0610E+01**	1.3899E+01
F25Means	**3.8679E+02**	3.8779E+02	**3.8416e+002**	3.8779E+02	4.4206E+02	**3.8779E+02**	**4.1266e+002**	4.1543E+002	3.8807e+002	**3.8779E+02**	4.0124E+02	**3.8779E+02**
Std	**6.3854E-01**	1.1319E+00	**1.0093e+000**	1.1319E+00	1.8872E+01	**1.1319E+00**	**2.5702e+001**	1.9864E+001	5.3517e+000	**1.1319E+00**	1.9489E+01	**1.1319E+00**
F26Means	**1.3911E+03**	1.5337E+03	**1.3722e+003**	1.5337E+03	1.9916E+03	**1.5337E+03**	2.6744e+003	**1.6473E+003**	1.8302e+003	**1.5337E+03**	1.7344E+03	**1.5337E+03**
Std	**1.0417E+03**	1.9051E+02	**1.0790e+003**	1.9051E+02	4.0323E+02	**1.9051E+02**	1.5918e+003	**1.4604E+002**	8.1504e+002	**1.9051E+02**	7.1347E+02	**1.9051E+02**
F27Means	7.6785E+02	**5.0744E+02**	5.1636e+002	**5.0744E+02**	5.3094E+02	**5.0744E+02**	5.7347e+002	**5.1046E+002**	5.1324e+002	**5.0744E+02**	5.4289E+02	**5.0744E+02**
Std	1.2966E+03	**3.6242E+00**	6.7963e+000	**3.6242E+00**	1.3878E+01	**3.6242E+00**	3.8376e+001	**7.8472E+000**	8.4123e+000	**3.6242E+00**	1.7086E+01	**3.6242E+00**
F28Means	**3.4728E+02**	4.1364E+02	**3.5681e+002**	4.1364E+02	5.3592E+02	**4.1364E+02**	**3.8744e+002**	5.4069E+002	**3.5163e+002**	4.1364E+02	**3.3257E+02**	4.1364E+02
Std	**5.4933E+01**	2.5577E+01	**4.1842e+001**	2.5577E+01	5.0685E+01	**2.5577E+01**	**5.0966e+001**	5.3437E+001	**5.8679e+001**	2.5577E+01	**5.2165E+01**	2.5577E+01
F29Means	8.1090E+02	**5.4778E+02**	2.5163e+002	**5.4778E+02**	**4.2261E+02**	5.4778E+02	**2.7784e+002**	5.4640E+002	**2.5095e+002**	5.4778E+02	5.5826E+02	**5.4778E+02**
Std	1.8646E+02	**8.1960E+01**	4.3559e+001	**8.1960E+01**	**4.9438E+01**	8.1960E+01	**5.5200e+001**	6.8540E+001	**6.0766e+001**	8.1960E+01	1.0040E+02	**8.1960E+01**
F30Means	**2.3063E+03**	4.9671E+03	**1.6442e+002**	4.9671E+03	**3.3984E+02**	4.9671E+03	**1.8979e+002**	2.5584E+004	**1.5606e+002**	4.9671E+03	5.0147E+03	**4.9671E+03**
Std	**5.2376E+02**	2.0934E+03	**4.2533e+001**	2.0934E+03	**5.6481E+01**	2.0934E+03	**5.6428e+001**	6.9017E+004	**6.5112e+001**	2.0934E+03	1.9712E+03	**2.0934E+03**
w/l/e	-	**15/15/0**	-	**17/13/0**	-	**26/4/0**	-	**21/9/0**	-	**22/8/0**		**16/14/0**

search ability but have weak global search ability. In Table 8, SE02 obtains the best performance of F3 function, which verifies its local search ability. The SE01 operator is achieved by global-best-guided item and rand-guided item corresponding to the local search and global search. Therefore, SE01 has achieved the balance between exploitation and exploration.

5.4. Summary of experimental analysis

In this subsection, convergence curves of 10 algorithms can be seen in Fig. 7. It can be observed that SE04 converges faster than the other 10 algorithms, including DE01, DE02, DE03, DE04, ABC, CMAES, GWO, TLBO, AAA and SaDE, for the six multimodal functions (F5, F7, F9, F20, F23 and F27). For the F5 function (Rotated Rastrigin's Function), DE04 converges more quickly than SE04 in the early stage when FES< 50000. However, SE04 obtains better performance than DE04 in the last stage when FES>100000. We know that the operator of SE04 is more similar to DE04 with a one rand-guided item. SE04 uses the spherical search style, whereas DE04 uses the hypercube search style. The spherical search style has a larger search space than the hypercube search style, which causes the SE04 to converge more slowly in the early stage and more quickly in the last stage, especially for multi-modal functions. For F7 (Shifted and Rotated Lunacek Bi_Rastrigin Function), ABC converges faster than other algorithms except for SE04 and CMAES, obtaining bad performance. For F9 (Shifted and Rotated Levy Function), GWO converges better in the last stage and obtains better performance than other algorithms except for SE04. For F20 (Hybrid Function 6 (N = 6)), TLBO converges better than other algorithms except for SE04 and CMAES obtaining poor performance. For F23 (Composition Function 3 (N = 4)) and F27 (Composition Function 7 (N = 6)), AAA converges more rapidly than other algorithms except for SE04. Table 9 shows the numbers of the winner of SE04 for 10 algorithms by t-test. It can be observed that SE04 has better performance than other algorithms to solve different optimization problems, especially for high dimension optimization problems.

At the end of this experiment, the performances of 17 algorithms are compared by the Friedman's test as Figs. 8 and 9. In Fig. 8, it can be observed that the ranks of 17 algorithms are SaDE, SE04, SE05, SE01, SE07, SE06, SE03, ABC, SE02, CMAES, AAA, TLBO, DE04, GWO, DE03, DE01, and DE02. The performance of almost

Table 7
Comparison results of SEs and other state-of-the-art-algorithms (Dimension = 50).

Func	CMAES	SE04	ABC	SE04	GWO	SE04	TLBO	SE04	AAA	SE04	SaDE	SE04
F1 Means	**1.4211e-014**	2.7828E+03	**1.3927e+003**	2.7828E+03	3.7239E+09	**2.7828E+03**	3.5109e+003	**2.7828E+03**	6.2396e+003	**2.7828E+03**	**2.5647E+03**	2.7828E+03
Std	**00000E+0000**	4.0415E+03	**1.9432e+003**	4.0415E+03	2.1706E+09	**4.0415E+03**	4.7691e+003	**4.0415E+03**	7.9256e+003	**4.0415E+03**	**3.1799E+03**	4.0415E+03
F2 Means	**00000E+0000**	7.1383E+27	**9.4410e+013**	7.1383E+27	4.4314E+46	**7.1383E+27**	2.0739e+056	**7.1383E+27**	**7.8823e+015**	7.1383E+27	**1.8669E+08**	7.1383E+27
Std	**00000E+0000**	3.0934E+28	**3.4590e+014**	3.0934E+28	1.7768E+47	**3.0934E+28**	1.4435e+057	**3.0934E+28**	**5.6291e+016**	3.0934E+28	**7.9763E+08**	3.0934E+28
F3 Means	**5.6843e-014**	5.7932E+04	2.8799e+005	**5.7932E+04**	**1.6951E+04**	5.7932E+04	**3.4243e+002**	5.7932E+04	**5.1001e+004**	5.7932E+04	**1.8496E+03**	5.7932E+04
Std	**00000E+0000**	1.1667E+04	4.4607e+004	**1.1667E+04**	**5.4058E+03**	1.1667E+04	**7.6418e+002**	1.1667E+04	1.6758e+004	**1.1667E+04**	**1.5425E+03**	1.1667E+04
F4 Means	**4.1439e+001**	1.1033E+02	**3.6439e+001**	1.1033E+02	4.5661E+02	**1.1033E+02**	**9.4260e+001**	1.1033E+02	**6.4220e+001**	1.1033E+02	**1.0533E+02**	1.1033E+02
Std	**4.4804e+001**	3.0110E+01	**2.0478e+001**	3.0110E+01	2.6437E+02	**3.0110E+01**	5.3997e+001	**3.0110E+01**	**4.8641e+001**	3.0110E+01	**4.3336E+01**	3.0110E+01
F5 Means	1.0683e+003	**1.0296E+02**	2.2190e+002	**1.0296E+02**	2.0602E+02	**1.0296E+02**	2.4340e+002	**1.0296E+02**	2.3476e+002	**1.0296E+02**	1.2886E+02	**1.0296E+02**
Std	1.9431e+002	**1.2848E+01**	3.1497e+001	**1.2848E+01**	9.4361E+01	**1.2848E+01**	3.5196e+001	**1.2848E+01**	4.9893e+001	**1.2848E+01**	2.3960E+01	**1.2848E+01**
F6 Means	9.4658e+001	**1.6338E-04**	**1.3375e-014**	1.6338E-04	1.0550E+01	**1.6338E-04**	3.9893e+001	**1.6338E-04**	1.6398e-004	**1.6338E-04**	5.3123E-01	**1.6338E-04**
Std	9.2749e+000	**8.6708E-04**	**3.6993e-014**	8.6708E-04	2.2885E+00	**8.6708E-04**	6.0137e+000	**8.6708E-04**	1.1710e-003	**8.6708E-04**	5.8275E-01	**8.6708E-04**
F7 Means	6.6939e+003	**1.5363E+02**	2.3917e+002	**1.5363E+02**	3.8620E+02	**1.5363E+02**	5.9158e+002	**1.5363E+02**	2.6165e+002	**1.5363E+02**	2.4910E+02	**1.5363E+02**
Std	1.2922e+003	**1.1661E+01**	2.7085e+001	**1.1661E+01**	1.0484E+02	**1.1661E+01**	8.7735e+001	**1.1661E+01**	4.5363e+001	**1.1661E+01**	4.8736E+01	**1.1661E+01**
F8 Means	1.1079e+003	**1.0514E+02**	2.4039e+002	**1.0514E+02**	2.0954E+02	**1.0514E+02**	2.6359e+002	**1.0514E+02**	2.2611e+002	**1.0514E+02**	1.3161E+02	**1.0514E+02**
Std	1.8478e+002	**1.2364E+01**	3.3464e+001	**1.2364E+01**	9.2107E+01	**1.2364E+01**	4.1902e+001	**1.2364E+01**	4.5002e+001	**1.2364E+01**	2.0238E+01	**1.2364E+01**
F9 Means	3.1413e+004	**5.7434E+01**	6.0397e+003	**5.7434E+01**	2.0699E+03	**5.7434E+01**	7.5876e+003	**5.7434E+01**	1.0838e+004	**5.7434E+01**	1.1715E+03	**5.7434E+01**
Std	5.1376e+003	**5.5695E+01**	2.1949e+003	**5.5695E+01**	8.9047E+02	**5.5695E+01**	4.1636e+003	**5.5695E+01**	3.9327e+003	**5.5695E+01**	6.8168E+02	**5.5695E+01**
F10Means	8.4489e+003	**4.4496E+03**	**4.4188e+003**	**4.4496E+03**	1.1852E+04	**4.4496E+03**	7.3580e+003	**4.4496E+03**	5.2940e+003	**4.4496E+03**	4.7308E+03	**4.4496E+03**
Std	1.0101e+003	**3.9287E+02**	4.4125e+002	**3.9287E+02**	3.1054E+03	**3.9287E+02**	2.2542e+003	**3.9287E+02**	6.3490e+002	**3.9287E+02**	7.3879E+02	**3.9287E+02**
F11Means	2.7474e+002	**6.0701E+01**	1.5037e+003	**6.0701E+01**	4.1869E+02	**6.0701E+01**	2.1813e+002	**6.0701E+01**	1.4823e+002	**6.0701E+01**	1.2689E+02	**6.0701E+01**
Std	5.9009e+001	**1.2012E+01**	1.6142e+003	**1.2012E+01**	1.0986E+02	**1.2012E+01**	4.9835e+001	**1.2012E+01**	4.5330e+001	**1.2012E+01**	3.5636E+01	**1.2012E+01**
F12Means	**2.5881e+003**	6.6352E+06	**2.4835e+006**	6.6352E+06	3.7611E+08	**6.6352E+06**	**3.0438e+005**	6.6352E+06	**1.8284e+006**	6.6352E+06	**7.4458E+05**	6.6352E+06
Std	**4.9548e+002**	3.7940E+06	**1.2046e+006**	3.7940E+06	4.5658E+08	**3.7940E+06**	**4.9122e+005**	3.7940E+06	**1.2084e+006**	3.7940E+06	**4.2569E+05**	3.7940E+06
F13Means	**2.4701e+003**	2.7844E+03	**2.6388e+003**	2.7844E+03	5.7058E+07	**2.7844E+03**	9.1664e+003	**2.7844E+03**	7.1070e+003	**2.7844E+03**	**2.5361E+03**	2.7844E+03
Std	**8.2966e+002**	2.7302E+03	**2.8458e+003**	2.7302E+03	7.3579E+07	**2.7302E+03**	6.9657e+003	**2.7302E+03**	9.4774e+003	**2.7302E+03**	**2.5305E+03**	2.7302E+03
F14Means	**3.3621e+002**	8.8121E+05	1.3081e+006	**8.8121E+05**	**3.6648E+05**	8.8121E+05	**4.2120e+004**	8.8121E+05	**2.5169e+005**	8.8121E+05	**4.7140E+04**	8.8121E+05
Std	**8.4557e+001**	5.3387E+05	8.8294e+005	**5.3387E+05**	**2.6035E+05**	5.3387E+05	**3.4894e+004**	5.3387E+05	**1.8733e+005**	5.3387E+05	**3.1170E+04**	5.3387E+05
F15Means	**4.9343e+002**	4.1274E+03	6.1355e+003	**4.1274E+03**	2.9602E+06	**4.1274E+03**	1.0567e+004	**4.1274E+03**	1.2401e+004	**4.1274E+03**	5.6216E+03	**4.1274E+03**
Std	**1.6229e+002**	3.9161E+03	4.2146e+003	**3.9161E+03**	5.2809E+06	**3.9161E+03**	7.1272e+003	**3.9161E+03**	7.5944e+003	**3.9161E+03**	4.7469E+03	**3.9161E+03**

(*continued on next page*)

all the SEs outperforms the other algorithms except SaDE. Fig. 9 shows the ranks of 17 algorithms for high dimensional problems: SE04, SaDE, SE05, SE01, SE07, SE03 ABC, SE06, CMAES, AAA, SE02, TLBO, DE04, GWO, DE03, DE01, and DE02. Only the performances of SE06 and SE02 have declined when dimension was increased from 30 to 50. More SE algorithms still exhibit good performance when the dimension is increased from 30 to 50.

In addition, the experimental result and analysis show that the performance of SE algorithms outperforms that of basic DEs. Scale factor (F) and crossover rate (CR) are the two important parameters influencing the performance of DEs. However, each dimensional variable is updated by the same scale factor (F), which means that individual $\mathbf{D_i}$ can only search in $\overrightarrow{D_iC_i}$ as shown in Fig. 1, which reduces the diversity of population. In addition, parameter CR determines the dimension selection. The number of the selected dimensional variable is set in a range [1, Dim]. Larger CR values cause more dimensional variables to be selected. Notably, the whole dimensional variables are selected if CR=1. In contrast to DEs, SEs exhibit better performance using the spherical search style replaced the hypercube search style in DEs. In the spherical search style, SEs can search in the whole spherical search space (circle search space in 2-dimension space) due to the parameter scale factor (SF) and angle corresponding to each dimension. SF in SEs denotes the spherical radius. Because all the angles correspond to each dimension produced by the uniform distribution, each position of the spherical search space can be searched. This will enhance the exploration ability of SEs, but reduce the exploitation ability of SEs. Therefore, SEs use the dimension selection factor (DSF) to replace the crossover rate (CR) in DEs. A greater number of less dimensional variables are selected by DSF in contrast to CR, which can enhance the exploitation ability of SEs. Thus, the balance between exploration and exploitation can be achieved by SF, angles, and DSF in SEs.

5.5. Comparison of computational times

In this subsection, we compare SE04 to the 7 algorithms, including ABC, GWO, TLBO, AAA, CMASE, SaDE, and DE04, for computational times. All the algorithms are run by Matlab R2016a using the Windows 7 operating system. Hardware configuration

Table 7 (*continued*).

Func	**CMAES**	**SE04**	**ABC**	**SE04**	**GWO**	**SE04**	**TLBO**	**SE04**	**AAA**	**SE04**	**SaDE**	**SE04**
F16Means	**9.0658e+002**	1.1818E+03	1.5445e+003	**1.1818E+03**	1.6938E+03	**1.1818E+03**	1.5504e+003	**1.1818E+03**	1.5943e+003	**1.1818E+03**	**1.0043E+03**	1.1818E+03
Std	**3.3437e+002**	2.2417E+02	3.0284e+002	**2.2417E+02**	8.0021E+02	**2.2417E+02**	3.6729e+002	**2.2417E+02**	3.3343e+002	**2.2417E+02**	**3.3703E+02**	2.2417E+02
F17Means	9.4418e+002	**7.5992E+02**	1.0546e+003	**7.5992E+02**	1.0922E+03	**7.5992E+02**	1.2894e+003	**7.5992E+02**	1.1253e+003	**7.5992E+02**	**7.5575E+02**	7.5992E+02
Std	2.7180e+002	**1.5556E+02**	1.9830e+002	**1.5556E+02**	4.6524E+02	**1.5556E+02**	3.3382e+002	**1.5556E+02**	1.9941e+002	**1.5556E+02**	**1.8544E+02**	1.5556E+02
F18Means	**3.4115e+002**	1.7071E+06	**1.3748e+006**	1.7071E+06	2.6993E+06	**1.7071E+06**	**3.4764e+005**	1.7071E+06	**7.5553e+005**	1.7071E+06	**5.5399E+05**	1.7071E+06
Std	**1.3358e+002**	1.0909E+06	**9.9234e+005**	1.0909E+06	1.7422E+06	**1.0909E+06**	**2.5614e+005**	1.0909E+06	**5.4972e+005**	1.0909E+06	**4.5699E+05**	1.0909E+06
F19Means	**3.2287e+002**	6.5977E+03	**5.7591e+003**	6.5977E+03	1.9894E+06	**6.5977E+03**	1.5360e+004	**6.5977E+03**	1.3545e+004	**6.5977E+03**	1.3390E+04	**6.5977E+03**
Std	**1.8967e+002**	6.4876E+03	**3.8600e+003**	6.4876E+03	2.1627E+06	**6.4876E+03**	1.0668e+004	**6.4876E+03**	1.3217e+004	**6.4876E+03**	8.3663E+03	**6.4876E+03**
F20Means	2.5069e+003	**5.4999E+02**	8.8894e+002	**5.4999E+02**	1.1068E+03	**5.4999E+02**	6.9330e+002	**5.4999E+02**	9.4594e+002	**5.4999E+02**	5.6922E+02	**5.4999E+02**
Std	4.4078e+002	**1.8562E+02**	1.9976e+002	**1.8562E+02**	5.2154E+02	**1.8562E+02**	2.3736e+002	**1.8562E+02**	2.6810e+002	**1.8562E+02**	1.4463E+02	**1.8562E+02**
F21Means	7.9154e+002	**3.1863E+02**	4.4299e+002	**3.1863E+02**	3.8937E+02	**3.1863E+02**	4.5798e+002	**3.1863E+02**	4.4212e+002	**3.1863E+02**	**3.1678E+02**	3.1863E+02
Std	5.1803e+002	**1.1613E+01**	3.2346e+001	**1.1613E+01**	8.8289E+01	**1.1613E+01**	3.5786e+001	**1.1613E+01**	4.8199e+001	**1.1613E+01**	**2.3411E+01**	1.1613E+01
F22Means	9.3583e+003	**4.8133E+03**	5.2549e+003	**4.8133E+03**	1.2047E+04	**4.8133E+03**	8.1892e+003	**4.8133E+03**	5.9003e+003	**4.8133E+03**	**4.7690E+03**	4.8133E+03
Std	1.4043e+003	**8.0418E+02**	9.1137e+002	**8.0418E+02**	2.9430E+03	**8.0418E+02**	3.9004e+003	**8.0418E+02**	**6.8565e+002**	8.0418E+02	**1.8229E+03**	8.0418E+02
F23Means	3.0770e+003	**5.5680E+02**	6.9312e+002	**5.5680E+02**	6.8231E+02	**5.5680E+02**	8.6543e+002	**5.5680E+02**	6.6842e+002	**5.5680E+02**	5.8431E+02	**5.5680E+02**
Std	8.9430e+002	**1.3842E+01**	4.5681e+001	**1.3842E+01**	1.1279E+02	**1.3842E+01**	7.0029e+001	**1.3842E+01**	4.6004e+001	**1.3842E+01**	3.8846E+01	**1.3842E+01**
F24Means	**6.5860e+002**	7.0270E+02	1.1046e+003	**7.0270E+02**	7.5372E+02	**7.0270E+02**	8.8538e+002	**7.0270E+02**	8.8716e+002	**7.0270E+02**	**6.7632E+02**	7.0270E+02
Std	**1.6512e+002**	2.6906E+01	8.1038e+001	**2.6906E+01**	1.2677E+02	**2.6906E+01**	9.2386e+001	**2.6906E+01**	6.6579e+001	**2.6906E+01**	**4.4248E+01**	2.6906E+01
F25Means	**4.9942e+002**	5.1996E+02	**5.0470e+002**	5.1996E+02	7.9592E+02	**5.1996E+02**	5.6259e+002	**5.1996E+02**	5.3301e+002	**5.1996E+02**	5.5431E+02	**5.1996E+02**
Std	3.1355e+001	**1.5016E+01**	**2.9524e+001**	**1.5016E+01**	1.7507E+02	**1.5016E+01**	3.5980e+001	**1.5016E+01**	3.9423e+001	**1.5016E+01**	4.3335E+01	**1.5016E+01**
F26Means	**1.7481e+003**	2.4329E+03	3.4369e+003	**2.4329E+03**	3.6644E+03	**2.4329E+03**	6.7073e+003	**2.4329E+03**	3.5331e+003	**2.4329E+03**	3.9842E+03	**2.4329E+03**
Std	**7.0312e+002**	**2.1358E+02**	1.1902e+003	**2.1358E+02**	1.0288E+03	**2.1358E+02**	2.9287e+003	**2.1358E+02**	6.2392e+002	**2.1358E+02**	9.2623E+02	**2.1358E+02**
F27Means	1.0071e+003	**5.9130E+02**	7.0580e+002	**5.9130E+02**	7.3895E+02	**5.9130E+02**	1.0955e+003	**5.9130E+02**	6.6656e+002	**5.9130E+02**	8.8692E+02	**5.9130E+02**
Std	1.6648e+003	**2.4353E+01**	5.2989e+001	**2.4353E+01**	6.2891E+01	**2.4353E+01**	1.8405e+002	**2.4353E+01**	5.8134e+001	**2.4353E+01**	9.7798E+01	**2.4353E+01**
F28Means	**4.6774e+002**	5.0575E+02	**4.7754e+002**	5.0575E+02	9.4231E+02	**5.0575E+02**	5.0897e+002	**5.0575E+02**	**4.8829e+002**	5.0575E+02	**4.9924E+02**	5.0575E+02
Std	1.8023e+001	**1.1313E+01**	**1.3392e+001**	1.1313E+01	3.0222E+02	**1.1313E+01**	3.0800e+001	**1.1313E+01**	2.1135e+001	**1.1313E+01**	**2.3252E+01**	**1.1313E+01**
F29Means	8.1090E+02	**7.4488E+02**	**3.7899e+002**	7.4488E+02	8.6223E+02	**7.4488E+02**	**4.1841e+002**	7.4488E+02	**3.9193e+002**	7.4488E+02	9.6535E+02	**7.4488E+02**
Std	1.8646E+02	**1.2998E+02**	**1.4188e+001**	1.2998E+02	2.7528E+02	**1.2998E+02**	**4.1480e+001**	1.2998E+02	**2.0304e+001**	1.2998E+02	2.6606E+02	**1.2998E+02**
F30Means	**2.3063E+03**	6.9078E+05	**2.7783e+002**	6.9078E+05	**7.0036E+02**	6.9078E+05	**3.1491e+002**	6.9078E+05	**2.8430e+002**	6.9078E+05	9.2269E+05	**6.9078E+05**
Std	**5.2376E+02**	7.3390E+04	**1.4339e+001**	7.3390E+04	**2.2450E+02**	7.3390E+04	**3.3154e+001**	7.3390E+04	**1.9704e+001**	7.3390E+04	1.7675E+05	**7.3390E+04**
w/l/e	**-**	**14/16/0**	**-**	**17/13/0,**	**-**	**27/3/0**	**-**	**23/7/0**	**-**	**21/9/0**		**16/14/0**

of the computer is as follows: four CPU, Intel(R) Core (TM) i5-4200 U, CPU 1.60 GHz, and RAM 4.00GB. Six optimization problems (F5, F11, F17, F21, F26, and F29) in CEC 2017 are adopted. Each algorithm is run 51 times for each optimization problem independently, and all running times (minutes) are saved. Other parameters are as mentioned above. Means and standard deviation are adopted for comparisons. Table 10 shows the comparison results. ABC (rank = 1) and GWO (rank = 2) run with more less running times for the 6 optimization problems. The rank of AAA is equal to 3 for the 6 optimization problems except for F26. In fact, the running time of AAA is the largest one than other 7 algorithms for F26. It seems a little unexpected; however, it can be explained if we observe the standard deviation. AAA obtains the larger standard deviation value (7058.163) for F26, which means that AAA obtains unstable solutions for each time. In other words, it is not a robust algorithm. In contrast, the SE04 has fewer running times than the other four algorithms, including DE04, CMAES, SaDE, and TLBO, for the six optimization problems with a greater number of smaller standard deviation values (<1).

Theoretically speaking, the time complexity of the proposed method depends on the number of fitness evaluations, the number of iterations, and population size, so the overall time complexity can be computed as follows:

$$O(SE) = O(ps * T) \tag{39}$$

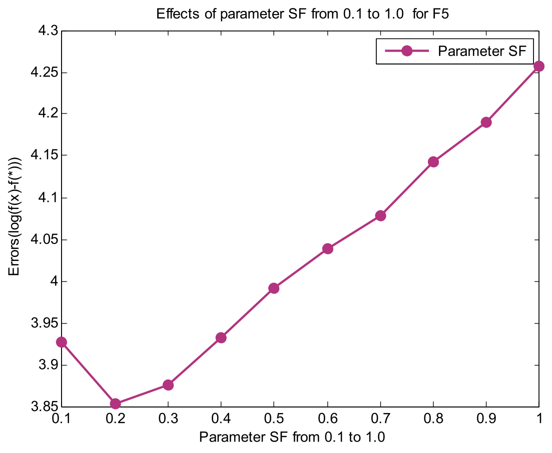
where ps denotes the population size, T denotes the number of iterations.

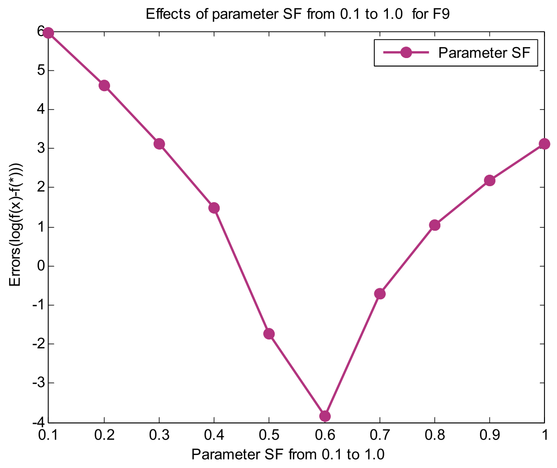
5.6. Real-world optimization problem

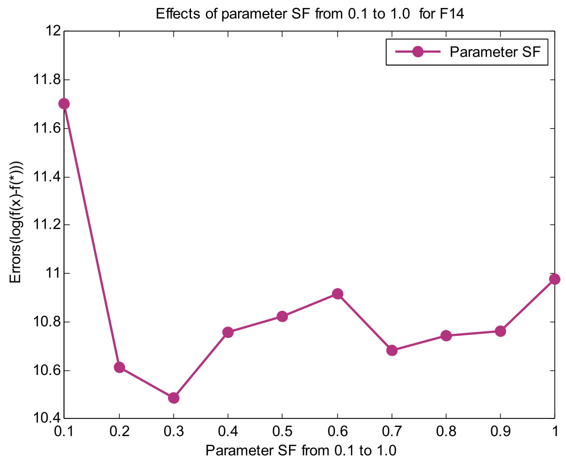
Data clustering is a most important and popular data analysis techniques, and denotes the process of grouping a set of data objects into different clusters, in which some data samples in a cluster must have greater similarity, and the data samples of different clusters must have higher dissimilarity [47]. Clustering techniques have been applied to many areas, such as security and crime detection [48], document clustering [49], image processing [50], prediction [51], and so on.

Table 8
Comparison results of SEs (Dimension = 50).

Func	SE01	SE02	SE03	SE04	SE05	SE06	SE07
F1 Means	4.0257E+03	7.2497E+03	4.0257E+03	2.7828E+03	1.8582E+04	**4.1615E+02**	6.6878E+03
Std	5.3167E+03	7.9654E+03	7.0735E+03	4.0415E+03	1.5579E+04	**5.6436E+02**	5.0246E+03
F2 Means	2.2670E+14	2.4596E+21	7.3545E+15	7.1383E+27	3.2992E+29	**9.3701E+13**	3.5999E+16
Std	1.2598E+15	1.7518E+22	4.6516E+16	3.0934E+28	1.5563E+30	**3.2591E+14**	2.4716E+17
F3 Means	9.1624E+04	**3.1469E+04**	3.7037E+04	5.7932E+04	6.9319E+04	9.3597E+04	8.6378E+04
Std	1.2774E+04	**7.9085E+03**	9.3963E+03	1.1667E+04	1.3876E+04	1.3955E+04	1.1907E+04
F4 Means	4.8518E+01	1.0798E+02	1.0825E+02	1.1033E+02	1.0493E+02	**3.7950E+01**	4.7256E+01
Std	2.7136E+01	5.4059E+01	5.4382E+01	3.0110E+01	3.5137E+01	**1.8570E+01**	2.6023E+01
F5 Means	2.0332E+02	1.5315E+02	1.3605E+02	**1.0296E+02**	1.1169E+02	2.5366E+02	2.0407E+02
Std	1.8218E+01	2.6005E+01	2.2732E+01	**1.2848E+01**	1.5129E+01	2.2147E+01	1.8643E+01
F6 Means	6.9651E−03	6.2044E−01	1.3721E−02	1.6338E−04	**2.8756E−13**	7.9700E−01	1.8995E−01
Std	3.2538E−03	5.0027E−01	2.3115E−02	8.6708E−04	**9.4717E−14**	2.0313E−01	6.7084E−02
F7 Means	2.3786E+02	2.1844E+02	1.9802E+02	**1.5363E+02**	1.5572E+02	2.9162E+02	2.5140E+02
Std	1.5740E+01	3.0520E+01	3.0961E+01	**1.1661E+01**	1.1864E+01	1.8570E+01	2.2627E+01
F8 Means	2.0100E+02	1.4851E+02	1.3978E+02	**1.0514E+02**	1.1107E+02	2.4486E+02	2.1078E+02
Std	2.2341E+01	2.3552E+01	2.5693E+01	**1.2364E+01**	1.4008E+01	2.1270E+01	2.0267E+01
F9 Means	8.3713E+03	2.5632E+02	1.7025E+02	**5.7434E+01**	1.7850E+02	1.3870E+04	9.8644E+03
Std	1.3112E+03	2.7108E+02	1.7213E+02	**5.5695E+01**	1.3411E+02	1.4292E+03	1.6382E+03
F10Means	4.4184E+03	5.1271E+03	5.1050E+03	**4.4496E+03**	4.6061E+03	4.5103E+03	4.5019E+03
Std	3.6136E+02	7.1645E+02	6.7731E+02	3.9287E+02	4.1224E+02	3.7389E+02	**3.4295E+02**
F11Means	1.6571E+02	2.1081E+02	1.5487E+02	**6.0701E+01**	8.3654E+01	1.8259E+02	1.7356E+02
Std	5.1476E+01	4.6550E+01	3.7085E+01	**1.2012E+01**	2.1909E+01	5.2745E+01	3.8470E+01
F12Means	4.3428E+06	5.4626E+06	5.8874E+06	6.6352E+06	9.6235E+06	**3.6296E+06**	4.6836E+06
Std	2.0995E+06	3.0074E+06	3.0644E+06	3.7940E+06	4.0388E+06	**1.7933E+06**	2.1904E+06
F13Means	1.1593E+03	1.1660E+04	1.0243E+04	2.7844E+03	7.2098E+03	**5.7768E+02**	1.2527E+03
Std	1.0982E+03	8.9786E+03	1.0297E+04	2.7302E+03	4.6508E+03	**2.4637E+02**	7.6637E+02
F14Means	3.6281E+05	4.1903E+05	5.5602E+05	8.8121E+05	8.2518E+05	**2.9054E+05**	3.2148E+05
Std	2.3643E+05	3.4421E+05	3.8224E+05	5.3387E+05	4.4381E+05	**1.5944E+05**	1.9063E+05
F15Means	**7.0406E+02**	9.0667E+03	8.9111E+03	4.1274E+03	7.5762E+03	9.6959E+02	7.2496E+02
Std	9.0884E+02	6.6067E+03	6.2644E+03	3.9161E+03	5.0798E+03	1.4486E+03	**8.7008E+02**
F16Means	1.3886E+03	1.4913E+03	1.3459E+03	**1.1818E+03**	1.2273E+03	1.3935E+03	1.3805E+03
Std	**1.8435E+02**	3.6027E+02	3.1477E+02	2.2417E+02	2.2807E+02	2.4886E+02	2.1258E+02
F17Means	1.0264E+03	1.1020E+03	9.9573E+02	**7.5992E+02**	8.0778E+02	1.0676E+03	9.4700E+02
Std	1.8719E+02	1.9996E+02	2.4404E+02	**1.5556E+02**	1.7206E+02	1.9715E+02	1.9260E+02
F18Means	8.0051E+05	1.6668E+06	1.5206E+06	1.7071E+06	1.6255E+06	8.1209E+05	**6.8753E+05**
Std	4.2468E+05	1.6187E+06	1.1514E+06	1.0909E+06	8.2599E+05	4.4138E+05	**4.1674E+05**
F19Means	9.4859E+02	1.3485E+04	1.1880E+04	6.5977E+03	9.5983E+03	**6.0460E+02**	1.0255E+03
Std	1.2318E+03	1.2691E+04	8.9595E+03	6.4876E+03	5.9408E+03	**7.1408E+02**	9.1581E+02
F20Means	8.0608E+02	8.3005E+02	7.3989E+02	**5.4999E+02**	6.3750E+02	8.6405E+02	7.5262E+02
Std	1.7293E+02	2.4442E+02	2.2210E+02	1.8562E+02	**1.5822E+02**	1.7401E+02	1.6492E+02
F21Means	4.1436E+02	3.5479E+02	3.5097E+02	**3.1863E+02**	3.2220E+02	4.6589E+02	4.2824E+02
Std	1.9786E+01	3.0115E+01	2.5149E+01	**1.1613E+01**	1.5530E+01	5.4151E+01	2.0108E+01
F22Means	5.0086E+03	5.6693E+03	5.6145E+03	**4.8133E+03**	5.1437E+03	4.9425E+03	4.8517E+03
Std	8.0976E+02	1.0903E+03	7.7762E+02	8.0418E+02	**4.6177E+02**	1.2929E+03	1.0306E+03
F23Means	6.6233E+02	5.9700E+02	5.8615E+02	**5.5680E+02**	5.5902E+02	7.1844E+02	6.7069E+02
Std	2.2500E+01	2.8774E+01	2.5656E+01	**1.3842E+01**	1.5655E+01	3.5557E+01	2.4635E+01
F24Means	9.4969E+02	7.2298E+02	7.2147E+02	7.0270E+02	**7.0196E+02**	1.0341E+03	9.6273E+02
Std	5.9221E+01	4.3081E+01	4.7745E+01	**2.6906E+01**	2.9281E+01	1.3182E+02	5.7600E+01
F25Means	**4.8461E+02**	5.4398E+02	5.1430E+02	5.1996E+02	5.2326E+02	4.8662E+02	4.9071E+02
Std	2.3712E+01	4.0544E+01	3.2406E+01	**1.5016E+01**	9.5100E+00	2.5497E+01	2.1732E+01
F26Means	2.9724E+03	3.0338E+03	2.8446E+03	**2.4329E+03**	2.5126E+03	3.5191E+03	2.6311E+03
Std	1.2360E+03	2.8296E+02	2.8399E+02	2.1358E+02	**1.3097E+02**	1.3273E+03	1.5283E+03
F27Means	6.6015E+02	5.9949E+02	5.9666E+02	**5.9130E+02**	5.9394E+02	6.9116E+02	6.6160E+02
Std	2.9257E+01	3.6375E+01	2.9176E+01	**2.4353E+01**	2.6561E+01	3.0100E+01	2.7840E+01
F28Means	4.7849E+02	5.0667E+02	5.0280E+02	5.0575E+02	4.9557E+02	**4.7013E+02**	4.7803E+02
Std	1.6776E+01	**7.5844E+00**	1.4148E+01	1.1313E+01	1.5012E+01	1.3374E+01	1.6830E+01
F29Means	9.5214E+02	1.0443E+03	9.5698E+02	**7.4488E+02**	7.6944E+02	9.9522E+02	9.2243E+02
Std	1.4151E+02	2.1739E+02	1.6488E+02	**1.2998E+02**	1.5731E+02	1.4992E+02	1.3368E+02
F30Means	6.8775E+05	1.5256E+06	9.5385E+05	6.9078E+05	6.9313E+05	6.8742E+05	**6.8666E+05**
Std	5.6494E+04	5.4240E+05	2.6602E+05	7.3390E+04	8.8037E+04	**4.2307E+04**	4.5013E+04
Winner number	**2**	**1**	**0**	**15**	**2**	**8**	**2**







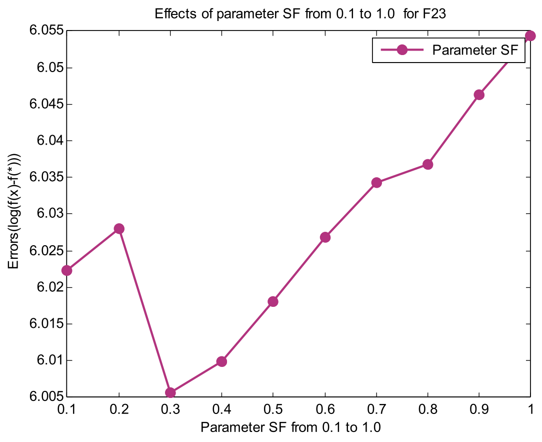


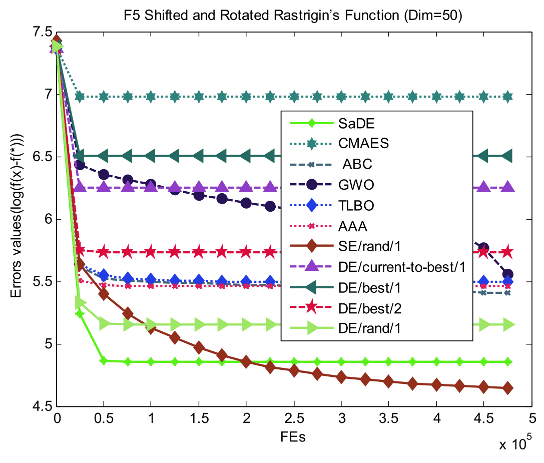
Fig. 6. Effects of parameter *F*.

Table 9
Numbers of winner of SE04 for other 10 algorithms by T-test.

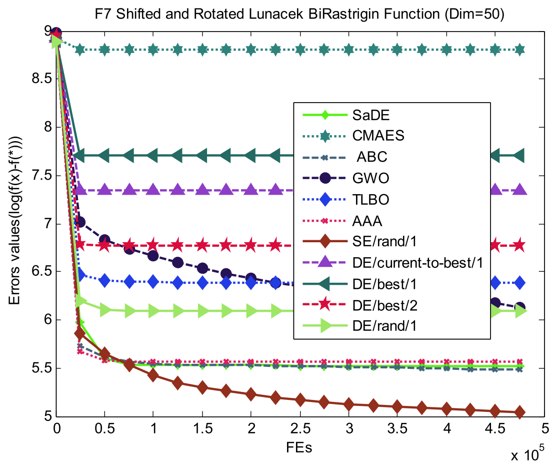
CEC2017	DE01	DE02	DE03	DE04	CMAES	ABC	GWO	TLBO	AAA	SaDE
Dimension = 30	**27**	**29**	**25**	**23**	**13**	**15**	**26**	**20**	**18**	**13**
Dimension = 50	**29**	**29**	**26**	**24**	**13**	**17**	**27**	**20**	**21**	**14**

Table 10
Comparison of computational times.

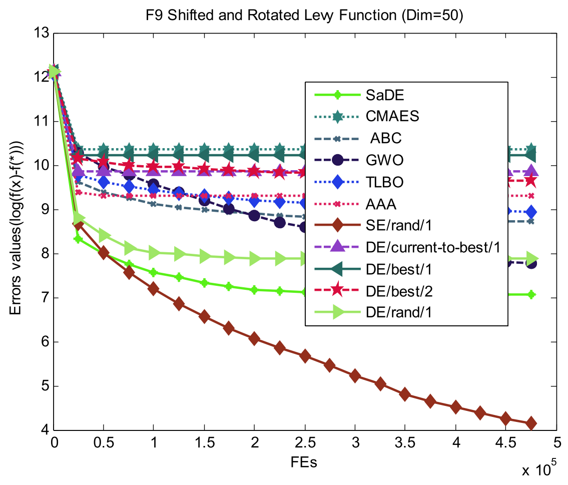
Func	ABC	GWO	TLBO	AAA	CMAES	SaDE	DE04	SE04
F5 Means	**12.12**	20.33	91.24	31.14	54.00	66.98	56.44	42.48
Std.	0.143209	0.10506	0.285409	3.151645	0.742942	0.713577	0.223672	0.373414
Rank	**1**	**2**	**8**	**3**	**5**	**7**	**6**	**4**
F11 Means	**13.57**	19.66	91.63	28.09	53.03	65.30	55.69	41.91
Std.	0.036863	0.079564	0.317669	0.171464	0.87206	0.239062	0.157713	0.212376
Rank	**1**	**2**	**8**	**3**	**5**	**7**	**6**	**4**
F17 Means	**16.16**	24.21	95.30	32.81	65.91	70.25	60.47	47.30
Std.	0.045546	0.072827	0.591348	0.253134	3.902175	0.402956	0.205413	0.305513
Rank	**1**	**2**	**8**	**3**	**6**	**7**	**5**	**4**
F21 Means	**22.07**	30.25	101.20	49.16	82.27	76.10	66.43	53.42
Std.	0.049058	0.092517	0.25178	74.99586	5.244624	0.180187	0.18758	0.144618
Rank	**1**	**2**	**8**	**3**	**7**	**6**	**5**	**4**
F26 Means	**32.48**	40.53	111.71	1039.06	93.15	86.85	77.56	64.13
Std.	0.070184	0.097278	0.432678	7058.163	6.995634	15.583	0.163423	0.150098
Rank	**1**	**2**	**7**	**8**	**6**	**5**	**4**	**3**
F29 Means	**33.36**	41.59	111.79	50.99	85.39	78.82	70.74	57.26
Std.	0.074753	0.113344	0.428947	0.236837	5.371195	0.233766	0.159169	0.193873
Rank	**1**	**2**	**8**	**3**	**7**	**6**	**5**	**4**



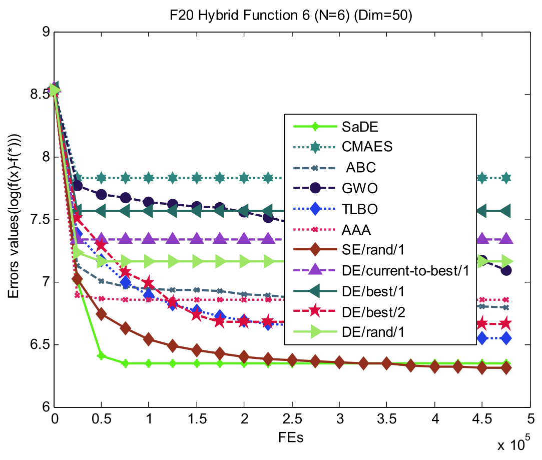
(a)



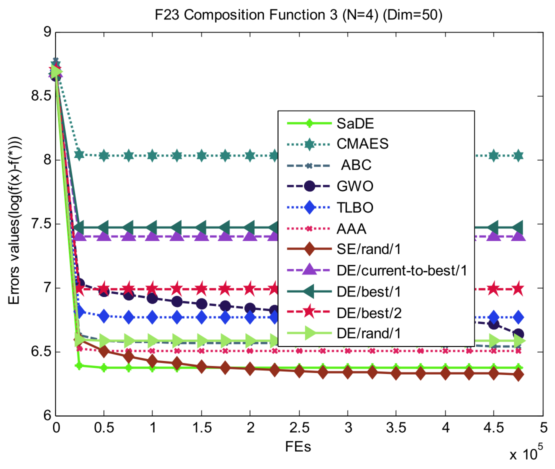
(b)



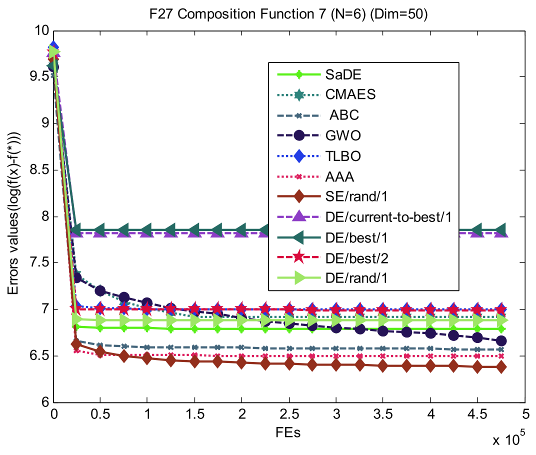
(c)



(d)



(e)



(f)

Fig. 7. Convergence curve.

Basically, a mathematical model of a data clustering problem can be represented as Eq. (40).

$$Min\ J = \sum_{i=1}^{n}\sum_{j=1}^{k} d_{ij}(X_i - C_j)^2 \tag{40}$$

where $d_{ij}(X_i - C_j)$ denotes the distance between the data sample and the centroid of a cluster, which can be computed by the Euclidean distance. The number of clusters is k, and n is the number of data samples. The object is to minimize the J by finding the k best centroids. The classic and well-known clustering algorithm is K-means algorithm. It is simple and efficient to

Table 11
The details of six dataset.

No.	Data set	Number of cluster	Dimension	Distribution (sample numbers of each class)
1	Iris	3	4	150 (50, 50, 50)
2	Wine	3	13	178 (59, 71, 48)
3	Glass	6	9	214 (70, 76, 17, 13, 9, 29)
4	Cancer	2	9	683 (444, 239)
5	Vowel	6	3	871 (72, 89, 172, 151, 207, 180)
6	CMC	3	9	1473 (629, 334, 510)

achieve data clustering. However, it is easy to fall into the local optimum due to its selection of the initial centroids. For the sake of shortcomings, many heuristic algorithms have been used to solve this problem [52–58].

In this experiment, six benchmark data sets, including Iris, Wine, Glass, Wisconsin Breast Cancer, Vowel and Contraceptive Method Choice (CMC), are selected from the repository of the machine learning databases [59]. The details of the six data sets are shown in Table 11. The SE04 and SE06 are used to compare with the 10 methods, including K-Means, PSO, MBCO, TLBO, GSA, MKCLUST, ACO, BBBC, KH, and BH. For SE04 and SE06, each algorithm is run 20 times. Population size is 20, and the MAX_FES = D*1E4 is as mentioned previously. Mean and standard deviation, the best and the worst, are recorded in Table 12. The results of K-means, ACO, KH, and PSO are cited from literature [60]; the results of TLBO and GSA are cited from literature [59]; the results of BH and BB-BC are cited from literature [30]; and the results of MBCO and MKCLUST are cited from literature [61].

Table 12 illustrates that the performances of SE06 and SE04 have outperformed those of the other ten algorithms except for the Iris data set. Fig. 10 shows the average ranks by Friedman's test as follows: SE06, SE04, BH, KH, BB-BC, ACO, MKCLUST, GSA, TLBO, MBCO, K-Means, and PSO. It is obvious that SE06 obtains better performance than others, which means that SE06 is better than SE04 for solving data clustering problems.

6. Conclusions

In this paper, we research the mechanism of many nature-inspired heuristic algorithms, including the classical algorithms and the currently proposed algorithms. Then, we propose the concept of a search pattern and search style by mathematical expression. We find that almost all algorithms are achieved by the hypercube search style. Inspired by it, we first propose the spherical search style and propose the spherical evolution algorithms, including SE/current-to-best/1, SE/best/1, SE/best/2, SE/rand/1, SE/rand/2, SE/current/1, and SE/current/2 like DE.

To verify the performance of the proposed algorithms, 9 state-of-the-art algorithms, including DE/current-to-best/1, DE/best/1, DE/best/2 and DE/rand/1, CMAES, ABC, GWO, ABC, and AAA are compared with the SEs. In addition, a real-world optimization problem, namely, the data clustering optimization problem, is achieved by SE/rand/1 and SE/current/1. Their performances are compared to those of 10 algorithms, including K-Means, PSO, MBCO, TLBO, GSA, MKCLUST, ACO, BBBC, KH, and BH. Experimental results and analysis confirm the superior performance of SEs.

Table 12
Comparison results of SEs and other state-of-art algorithms for data clustering problems.

Dataset	MKCLUST[4]	K-means[1]	TLBO[2]	GSA[2]	ACO[1]	KH[1]	PSO[1]	BH[3]	BB–BC[3]	MBCO[4]	SE04	SE06
(**Iris**)												
Best	95.01	97.3259	96.6554	**96.6554**	97.10077	96.6555	96.8942	96.65589	96.67648	**94.14**	96.6554	96.6554
Worst	201.00	123.9695	125.0844	127.6676	97.808466	96.6555	97.8973	96.66306	97.42865	104.22	97.1967	**96.6554**
Means	**96.01**	106.5766	103.6115	98.2060	97.171546	96.6555	97.2328	96.65681	96.76537	96.03	96.6838	96.6554
Std	3.53	12.938	11.0483	6.9345	0.367	1.9E−06	0.347168	0.00173	0.20456	2.46	0.12079	**1.06e−006**
Rank	**1**	**12**	**11**	**10**	**8**	**4**	**9**	**5**	**7**	**2**	**6**	**3**
(**Wine**)												
Best	16400.00	16,555.68	16,295.14	17,123.24	16,530.533	16,292.19	16,345.97	16,293.41995	16,298.67356	16405.00	**16292.1846**	**16292.1846**
Worst	21939.00	18,294.85	16,461.23	22,259.28	16,530.534	18,293.6	16,562.32	16,300.22613	16,310.11354	22539.00	16293.8238	**16292.1846**
Means	21488.00	17,251.35	16,329.26	19,767.68	16,530.53	16,579.66	16,417.47	16,294.31763	16,303.41207	21989.00	16292.3925	**16292.1846**
Std	41.67	874.148	52.4949	1194.7086	0.00	424.9147	85.4974	1.65127	2.66198	41.88	0.39725211	**5.090e−006**
Rank	**11**	**9**	**5**	**10**	**7**	**8**	**6**	**3**	**4**	**12**	**2**	**1**
(**Glass**)												
Best	215.00	215.73	218.3590	214.4027	273.46	210.2421	270.57	210.51549	223.89410	215.00	**210.008407**	210.428662
Worst	333.00	227.35	303.1350	265.2572	280.08	251.2749	283.52	213.95689	243.20883	330.00	**212.464646**	214.806808
Means	220.00	218.70	253.1780	239.0503	269.72	215.7225	275.71	211.49860	231.23058	225.00	**210.475064**	210.783253
Std	13.10	2.456	18.3250	16.8698	3.584829	5.44876	4.557134	1.18230	4.65013	13.07	0.48679038	1.11412839
Rank	**6**	**5**	**10**	**9**	**11**	**4**	**12**	**3**	**8**	**7**	**1**	**2**
(**CMC**)												
Best	5678.20	5703.20	5532.2446	5532.1847	5819.1347	5693.72	5700.985	5532.88323	5534.09483	5680.12	**5532.1847**	**5532.1847**
Worst	5790.21	5704.57	5598.0817	5532.9950	5912.4300	6755.956	5923.249	5534.77738	5644.70264	5798.20	5532.1849	**5532.1847**
Means	5684.80	5705.37	5542.5356	5532.2642	5701.9230	5737.234	5820.965	5533.63122	5574.75174	5685.21	**5532.1847**	**5532.1847**
Std	1.97	1.033	18.5901	0.2123	45.63470	178.0245	46.95969	0.59940	39.43494	1.98	4.201e−005	**5.56e−008**
Rank	**6**	**9**	**4**	**2**	**8**	**10**	**11**	**3**	**5**	**7**	**1**	**1**
(**Cancer**)												
Best	2969.01	2988.43	2964.3870	**2964.3869**	3046.06	2964.387	2973.50	2964.38878	2964.38753	2965.25	**2964.3869**	**2964.3869**
Worst	3076.10	2999.19	3542.0871	3071.7828	3242.01	3580.312	3318.88	2964.45074	2964.38902	3001.01	2964.3870	**2964.3869**
Means	2985.23	2988.99	3097.1951	2980.4701	2970.49	2971.977	3050.04	2964.39539	2964.38798	2990.23	**2964.3869**	**2964.3869**
Std	2.12	2.469	195.5960	27.7893	90.50028	62.26148	110.8013	0.00921	0.00048	2.12	1.364e−005	**4.15e−012**
Rank	**7**	**8**	**11**	**6**	**4**	**5**	**10**	**3**	**2**	**9**	**1**	**1**
(**Vowel**)												
Best	148989.14	149,398.66	149,056.17	219,794.07	159,458.14	**148,967.246**	148,976.01	148,985.61	149,038.51	148989.31	148968.4464	148967.2740
Worst	151067.10	162,455.69	158,615.91	302,756.54	165,939.82	158,503.045	149,121.18	153,058.98	153,090.44	151067.23	149105.13276	**148989.17779**
Means	150887.09	151,987.98	152,698.51	261,555.31	149,395.60	150,035.986	148,999.82	149,848.18	151,010.03	150890.75	149011.52726	**148968.62762**
Std	110.11	3425.250	3372.3409	21692.0729	3485.3816	1707.84248	28.8134692	1306.95375	1859.3235	110.12	42.782267359	**4.84764805**
Rank	**7**	**10**	**11**	**12**	**4**	**6**	**2**	**5**	**9**	**8**	**3**	**1**

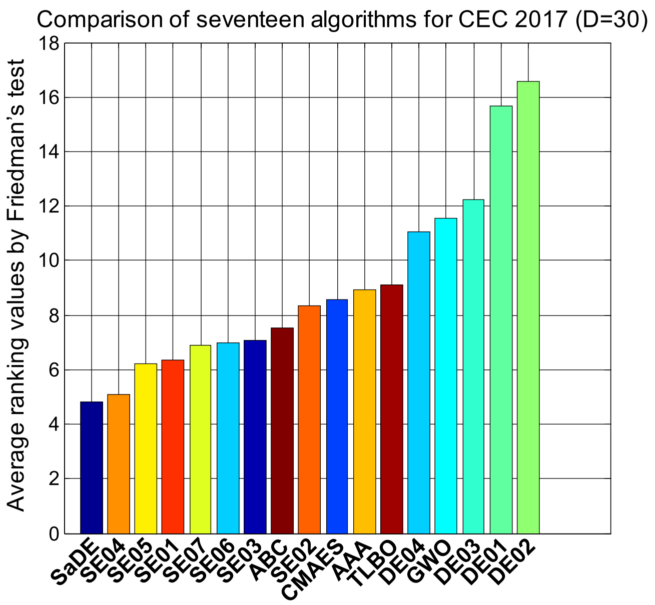


Fig. 8. Rank of 17 algorithms by Friedman's test(D = 30).

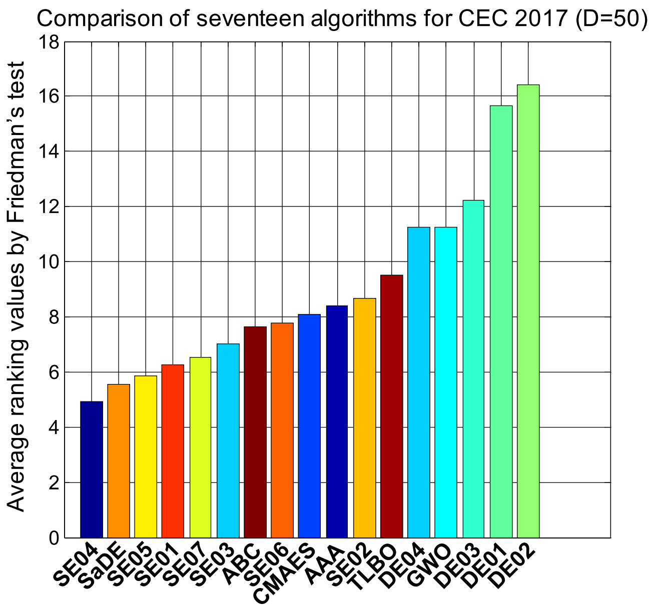


Fig. 9. Rank of 17 algorithms by Friedman's test(D = 50).

However, there are some shortcomings of SEs. Like the DE algorithm, the SE is sensitive to two parameters: scale factor (SF) and dimension selection factor (DSF). Different parameter tuning strategy decides the performance of SE algorithms. Owing to the characteristics of the spherical search style, the dimension selection factor (DSF) is different from that of DE. In addition, another important parameter of SE is ignored by the uniform distribution: the angles of the spherical search style. Therefore, we believe that it should be tuned by some method. The parameter tuning strategy will be the hot topic of SE in the future.

Acknowledgments

This work is supported by the Guang Dong Provincial Natural fund project, PR China (2016A030310300), Drug-target interaction prediction method based on collaborative intelligent optimization; Guang Dong Provincial Natural fund, PR China (2014A030313585, 2015A03031026, 2015A030310483).

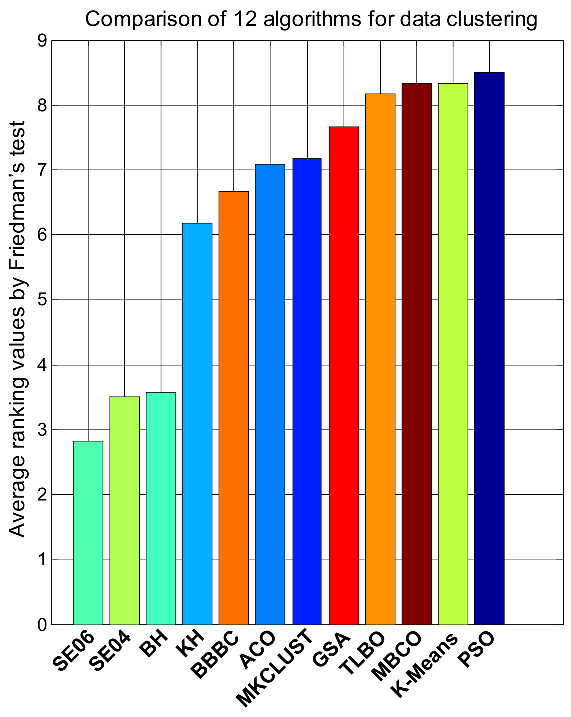


Fig. 10. Rank of 12 algorithms for data clustering by Friedman's test.

References

[1] J.J. Grefenstette, Optimization of control parameters for genetic algorithms, IEEE Trans. Syst. Man cybernet. 16 (1) (1986) 122–128.
[2] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceeding IEEE International Conference Neural Network, Perth, Western Australia, 1995, pp. 1942–1948.
[3] R.M. Storn, K.V. Price, Differential evolution -a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[4] M. Dorigo, C. Blum, Ant colony optimization theory: A survey, Theor. Comput. Sci. 344 (2) (2005) 243–278.
[5] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report TR06, Erciyes University, 2005.
[6] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (2008) 702–713.
[7] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching-learning-based optimization: An optimization method for continuous non-linear large scale problems, Inform. Sci. 183 (2012) 1–15.
[8] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Inform. Sci. 179 (2009) 2232–2248.
[9] S.A. Uymaz, G. Tezel, E. Yel, Artificial algae algorithm (AAA) for nonlinear global optimization, Appl. Soft Comput. 31 (2015) 153–171.
[10] A. Kaveh A. Dadras, A novel meta-heuristic optimization algorithm: Thermal exchange optimization, Adv. Eng. Softw. 110 (2017) 69–84.
[11] Seyedali Mirjalili, The ant lion optimizer, Adv. Eng. Softw. 83 (2015) 80–98.
[12] Seyedali Mirjalili, Andrew Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[13] Shahrzad Saremi, Seyedali Mirjalili, Andrew Lewis, Grasshopper optimisation algorithm: Theory and application, Adv. Eng. Softw. 105 (2017) 30–47.
[14] S. Mirjalili, S.M. Mirjalili, Andrew Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[15] Seyedali Mirjalili, SCA:a Sine cosine algorithm for solving optimization problems, Knowl.-Based Syst. 96 (2016) 120–133.
[16] X.-S. Yang, Firefly algorithm, in: X.-S. Yang (Ed.), Nature-Inspired Meta Heuristic Algorithms, Wiley Online, Library, 2008, pp. 79–90.
[17] Berat Dogan, Tamer Ölmez, A new metaheuristic for numerical function optimization: Vortex search algorithm, Inform. Sci. 293 (2015) 125–145.

[18] Assif Assad, Kusum Deep, A hybrid harmony search and simulated annealing algorithm for continuous optimization, Inform. Sci. 450 (2018) 246–266.
[19] A. Kaveh, K. Laknejadi, A novel hybrid charge system search and particle swarm optimization method for multi-objective optimization, Expert Syst. Appl. 38 (12) (2011) 15475–15488.
[20] Ali Sadollh, Ardeshir Bahreininejad, Hadi Eskandar, Mohd Hamdi, Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems, Appl. Soft Comput. 13 (5) (2013) 2592-1612.
[21] Hadi Eskandar, Ali Sadollah, Ardeshir Bahreininejad, Mohd Hamdi, Water cycle algorithm-A novel metaheuristic optimization method for solving constrained engineering optimization problems, Comput. Struct. 110–111 (2012) 151–166.
[22] Yu-Jun Zheng, Water wave optimization: A new nature-inspired metaheuristic, Comput. Oper. Res. 55 (2015) 1–11.
[23] Min-Yuan Cheng, Doddy Prayogo, Symbiotic organisms search: A new metaheuristic optimization algorithm, Comput. Struct. 139 (2014) 98–112.
[24] Najmeh Sadat Jaddi, Jafar Alvankarian, Salwani Abdullah, Kidney-inspired algorithm for optimization problems, Commun. Nonlinear Sci. Numer. Simul. 42 (2017) 358–369.
[25] Amir Hossein Gandomi, Amir Hossein Alavi, Krill herd: A new bio-inspired optimization algorithm, Commun. Nonlinear Sci. Numer. Simul. 17 (2012) 4831–4845.
[26] Alireza Askarzadeh, Bird mating optimizer: An optimization algorithm inspired by bird mating strategies, Commun. Nonlinear Sci. Numer. Simul. 19 (2014) 1213–1228.
[27] Seyedali Mirjalili, Amir H. Gandomi, SeyedehZahra Mirjalili, Shahrzad Saremi, Hossam Faris, SeyedMohammad Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191.
[28] Deyu Tang, Shoubin Dong, Yi Jiang, Huan Li, Yishuan Huang, ITGO: Invasive tumor growth optimization algorithm, Appl. Soft Comput. 36 (2015) 670–698.
[29] Guohua Wu, Across neighborhood search for numerical optimization, Inform. Sci. 329 (2016) 597–618.
[30] A. Hatamlou, Black hole: a new heuristic optimization approach for data clustering, Inform. Sci. 222 (2013) 175–184.
[31] Foroughi Nematollahi, A. Rahiminejad, A novel physical based metaheuristic optimization method known as lightning attachment procedure optimization, Appl. Soft Comput. 59 (2017) 596–621.
[32] Manizheh Ghaemi, Mohammad-Reza Feizi-Derakhshi, Forest optimization algorithm, Expert Syst. Appl. 41 (2014) 6676–6687.
[33] Gaurav Dhiman, Vijay Kumar, Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications, Adv. Eng. Softw. 114 (2017) 48–70.
[34] I. Rechenberg, Evolution Strategies: Optimierung Technischer Systeme Nach Prinzipien Der Biologischen Evolution, Frommann-Holzboog, Stuttgart, 1973.
[35] X. Yao, Y. Liu, Fast evolutionary programming, Evol. Program. 3 (1996) 451–460.
[36] N. Hansen, S.D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation(CMA-ES), Evol. Comput. 11 (2003) 1–18.
[37] Adam Lipowski, Dorota Lipowska, Roulette-wheel selection via stochastic acceptance, Physica A 391 (2012) 2193–2196.
[38] Brad L. Miller, David E. Goldberg, Genetic algorithms, tournament selection, and the effects of noise, Complex Syst. 9 (1995) 193–212.
[39] Tania Pencheva, Krassimir Atanassov, Anthony Shannon, Modelling of a stochastic universal sampling selection operator in genetic algorithms using generalized nets, Tenth Int. Workshop Gen. Nets Sofia 5 (2009) 1–7.
[40] X.S. Yang, S. Deb, Cuckoo search via Lévy flights, in: Proceeings of World Congress on Nature & Biologically Inspired Computing, IEEE Publications, USA, 2009.
[41] D. Mustard, Numerical integration over the n-dimensional spherical shell, Math. Comp. 18 (88) (1964) 578–589.
[42] A.M.L.J. Awad, N.H., B. Qu, P. Suganthan, Problem Definitions and Evaluation Criteria for the Cec 2017 Special Session and Competition on Single Objective Bound Constrained Real-Parameter Numerical Optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2017.
[43] Pinar Civicioglu, Artificial cooperative search algorithm for numerical optimization problems, Inform. Sci. 229 (2013) 58–76.
[44] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417.
[45] Z.H. Zhan, J. Zhang, Y. Li, et al., Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. B 39 (6) (2009) 1362–1381.
[46] G.F. Fan, L.L. Peng, W.C. Hong, Short term load forecasting based on phase space reconstruction algorithm and bi-square kernel regression model, Appl. Energy 224 (2018) 13–33.
[47] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Academic Press, 2006.
[48] T. Grubesic, On the application of fuzzy clustering for crime hot spot detection, J. Quant. Criminology 22 (2006) 77–105.
[49] M. Carullo, E. Binaghi, I. Gallo, An online document clustering technique for short web contents, Pattern Recognit. Lett. 30 (2009) 870–876.
[50] R.I. John, P.R. Innocent, M.R. Barnes, Neuro-fuzzy clustering of radiographic tibia image data using type 2 fuzzy sets, Inform. Sci. 125 (2000) 65–82.
[51] G. Cardoso, F. Gomide, Newspaper demand prediction and replacement model based on fuzzy clustering and rules, Inform. Sci. 177 (2007) 4799–4809.
[52] S. Das, A. Abraham, A. Konar, Automatic hard clustering using improved differential evolution algorithm, in: Studies in Computational Intelligence, 2009, pp. 137–174.
[53] A. Hatamlou, S. Abdullah, H. Nezamabadi-pour, Application of gravitational search algorithm on data clustering, in: Rough Sets and Knowledge Technology, Springer, Berlin/Heidelberg, 2011, pp. 337–346.
[54] A. Hatamlou, S. Abdullah, H. Nezamabadi-pour, A combined approach for clustering based on k-means and gravitational search algorithms, Swarm Evol. Comput. 6 (2012) 47–52.
[55] J. Senthilnath, S.N. Omkar, V. Mani, Clustering using firefly algorithm: performance study, Swarm Evol. Comput. 1 (2011) 164–171.
[56] A. Hatamlou, S. Abdullah, M. Hatamlou, Data clustering using big bang-big crunch algorithm, in: Communications in Computer and Information Science, 2011, pp. 383–388.
[57] M. Fathian, B. Amiri, A. Maroosi, Application of honey-bee mating optimization algorithm on clustering, Appl. Math. Comput. 190 (2007) 1502–1513.
[58] Suresh Chandra Satapathy, Anima Naik, Data Clustering Based on Teaching-Learning-Based Optimization, (2011) 148-156.
[59] Deyu Tang, Shoubin Dong, Lifang He, Yi Jiang, Intrusive tumor growth inspired optimization algorithm for data clustering, Neural Comput. Appl. 27 (2016) 349–374.
[60] R. Jensi, G. Wiselin Jiji, An improved krill herd algorithm with global exploration capabilityfor solving numerical function optimization problems and its application to data clustering r, Appl. Soft Comput. 46 (2016) 230–245.
[61] Pranesh Das, Dushmanta Kumar Das, Shouvik Dey, An improved krill herd algorithm with global exploration capability for solving numerical function optimization problems and its application to data clustering, Appl. Soft Comput. 46 (2016) 230–245.