Multi-strategy serial cuckoo search algorithm for global optimization

Hu Peng[a,*], Zhaogan Zeng[a], Changshou Deng[a], Zhijian Wu[b]

[a]*School of Computer and Big Data Science, Jiujiang University, Jiujiang 332005, PR China*
[b]*School of Computer Science, Wuhan University, Wuhan 430072, PR China*

Abstract

Cuckoo search algorithm(CS) is a simple and effective nature-inspired optimization algorithm, which has been widely applied to solve the complex engineering optimization problem in the real world. However, all cuckoos have similar search behavior because of the unitary search strategy is used in CS, which easily makes the algorithm get trapped in the local optimum and enter the state of premature convergence. Inspired by the three serial behaviors of the cuckoo's growth process, a multi-strategy serial CS(MSSCS) is proposed to overcome the above problems. In MSSCS, three new learning strategies are proposed to enhance the performance of the algorithm, which observed from the cuckoo's behavior of seeking host nest, eviction and begging. Based on these three serial behaviors, we propose a multi-strategy serial framework to reduce the complexity of the multi-strategy algorithm and maximize the performance of each learning strategy. In addition, the adaptive parameter is designed to dynamically regulate the serial framework. To verify the effectiveness of MSSCS, extensive experiments are carried out on the CEC2013 test suite. The experimental results illustrate that the proposed algorithm has better performance when considering the quality of the obtained solutions.

Keywords: Cuckoo search algorithm; Meta-heuristic algorithm; Multi-strategy; Serial framework

*Corresponding author
Email address: hu_peng@whu.edu.cn (Hu Peng)

1. Introduction

Over the past few decades, meta-heuristic algorithms have been extensively applied for solving complex and highly non-linear optimization problems. These approaches are usually inspired from natural processes. There are some nature-inspired algorithms, such as, genetic algorithm (GA) [1] inspired by natural evolution, particle swarm optimization algorithm (PSO) [2] simulating the social behavior of fish or birds, artificial bee colony (ABC) [3] simulating foraging behavior of bees and cuckoo search algorithm(CS) [4] simulating some cuckoo species' obligate brood parasitic behavior. For these algorithms, operators such as mutation and crossover are usually used to generate new individuals. Then, the solution can be ameliorated by several iterations. Compared with the gradient based algorithm, meta-heuristic algorithm is proved to be more effective [5]. However, the common problem of meta-heuristic algorithms is that the convergence speed is slow, which makes the meta-heuristic algorithm less efficient when the evaluation of objective function has a large amount of calculation and time-consuming. Hence, it is still a promising domain to develop more effective meta-heuristic algorithms.

CS is a new nature-inspired meta-heuristic algorithm proposed by Yang and Deb. This global search algorithm combines the idea of some cuckoo species' obligate brood parasitism [6] with the L*é*vy flight of insects [7]. Moreover, it employs greedy selection, biased random walk and L*é*vy flight to search for the global optimum. Compared with the uniform distribution and Gaussian distribution algorithm, the long jumps provided by L*é*vy flight can better search the solution domain [8]. The combination of the advantages of L*é*vy flight and local search ability makes CS turn into one of the most effective optimization algorithms. Furthermore, the performance of CS is superior to PSO and GA, which has been proved by Yang and Deb in a previous study [4], and CS can acquire more robust results in comparison with PSO and ABC [9] in terms of conceptual comparison. In addition, CS has advantages of simple, few control parameters and so on. Considering the above advantages, CS has been applied in computational intelligence and practical engineering optimization problems with promising results.

It should be noted, for the ”No Free Lunch” theorem (NFL) [10], it has proved that for any algorithm, any performance improvement on one type of problem will be offset by the performance on another type of problem. Similarly, when a strategy has strong local search capability, its global search capability is often weak. Therefore, the optimization ability of the algorithm with single search strategy is limited. In CS, the single search strategy will result in lacking diversity in search space, which makes CS easily into the local minimum when resolving complex optimization problems. It indicates that the multi-strategy algorithm is promising.

In nature, the continuation of cuckoo offspring not only depends on the environment of habitat, but also on individual behavior during the growth process of cuckoos. As for cuckoos, the survival rate of young cuckoo is improved by three serial individual behaviors: the behavior of female cuckoo seeking the host bird's nest, the eviction behavior of nestling cuckoo after it is hatched, and the begging behavior of young cuckoo for being fed [6]. Inspired by the above cuckoo behaviors, three new learning strategies are proposed in this paper to enhance the performance of the algorithm and named saltation learning (SL), Gaussian walk learning (GWL) and single dimension learning (SDL) respectively. Then, as the number of strategies increases, the complexity of the algorithm is a problem worth considering. However, multi-strategy algorithms tend to be complicated in how to select strategies to use. Therefore, based on the three serial behaviors during the growth of the cuckoo, a multi-strategy serial framework is proposed to reduce the complexity of the multi-strategy algorithm and maximize the performance of each learning strategy.

In the multi-strategy serial framework, a serial strategy pool (*SSP*) is composed of the three learning strategies. If the conditions for using *SSP* are met, the strategy in *SSP* will be selected to update the solution, otherwise, the solution is update by using Lévy flight with strong global search performance. For the use of *SSP*, when the algorithm is in the early search phase, SL will be selected from *SSP* to obtain the promising solution and prevent premature convergence, so as to provide a solid foundation for the optimization in the later phase of the algorithm. To balance the exploration and exploitation of the algorithm, GWL is drawn from *SSP* to update the solution in the mid-term search of the algorithm. In the late search phase of the algorithm, SDL is chosen from *SSP* to improve the convergence speed of the algorithm, so that the obtained solution is closer to the global optimal solution. Besides, the adaptive parameter is designed to dynamically adjust the multi-strategy serial framework.

The main contributions of this work are outlined as follows:

- Inspired by the cuckoo's behavior of seeking host nest, eviction and begging during the growth of cuckoos, three new learning strategies are proposed to enhance the performance of the algorithm.
- Based on these three serial behaviors, we propose a multi-strategy serial framework to reduce the complexity of the multi-strategy algorithm and maximize the performance of each learning strategy. In addition, the adaptive parameter is designed to dynamically regulate the serial framework.
- The complexity, scalability and convergence rate of the MSSCS is investigated to verify its performance. Besides, the effectiveness of the multi-strategy serial framework and the three learning strategies are analyzed.

The structure of the remaining paper is as follows. The second section 2 reviews the original CS and its technical details. In the section 3, the literature on CS and its application in optimization problems are introduced. The section 4 illustrates the motivation of our proposed algorithm and the section 5 expounds the presented algorithm in detail. In the section 6, the complexity of MSSCS is analyzed. The section 7 presents the comparative analysis of digital experiments between MSSCS and CS, 9 CS variants, 4 new CS versions and several other state-of-art algorithms. Finally, in the section 8, we summarize the proposed algorithm.

2. Cuckoo search algorithm

CS is a meta-heuristic algorithm inspired by the natural behavior of some cuckoo species laying their eggs to other bird's nests. This parasitic behavior has become the reproductive strategy for cuckoos and them lay their eggs in nests of other species in most cases. As a result, the host bird may discover that the eggs are not its own, at which point it either throws away the foreign eggs or abandons the nest and builds a new one. This has led to the evolution of some cuckoos, making them very specialized at imitating the color and pattern of the host bird's egg [11]. In order to adapt the behavior of cuckoo in nature to the computer algorithm, three rules are idealized in the original CS:

- Each cuckoo lays only one egg each time and randomly places this egg in a host nest.
- The best nest with good quality egg will be passed on to the next generation.
- In the process of searching, the number of available hosts is a constant, and the host bird has the probability of $P_a (P_a \in [0,1])$ to find that the cuckoo lays its eggs in their nest. When this happens, the laid egg will be thrown away, or the host bird simply abandons the nest to build a new one.

Therefore, based on this simplified model, a solution corresponds to the position of a cuckoo, a nest, or an egg in the search space, which makes CS easier to implement. In addition, CS introduces L*é*vy flight, which simulates the foraging process of animals in nature. In essence, L*é*vy flight is a random walk and it is characterized by selecting a series of instantaneous jumps from the probability density function which has a heavy tail. [12].

Mathematically, the CS is mainly composed of local random walk and global exploratory random walk, and the parameter $P_a (P_a \in [0,1])$ controls the balance between them. The global random walk could be described in the following formula:

$$x_i^{t+1} = x_i^t + \alpha \oplus L\acute{e}vy(s,\lambda), \tag{1}$$

where

$$L\acute{e}vy(s,\lambda) \approx \frac{\lambda \Gamma(\lambda) \cdot sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \qquad (s \gg s_0 > 0), \tag{2}$$

$$\alpha = \alpha_0 (x_i^t - x_{best}^t), \tag{3}$$

where x_i^t indicates the ith solution of generation t, x_{best}^t denotes the optimal solution in t generation, $\oplus$ denotes entry-wise multiplications. Here, $L\acute{e}vy(s,\lambda)$ indicates the characteristic scale, λ represents the power coefficient $(1 < \lambda < 3)$, and Γ is the gamma function. α_0 denotes a scaling factor, which is usually taken as 0.01.

In formula 1, s is the step size of l$\acute{e}$vy flight and it was designed in Mantegna's algorithm [13] as follows:

$$s = \frac{\mu}{|\nu|^{1/\lambda}}, \tag{4}$$

where μ and ν are random numbers drawn from the normal distribution:

$$\mu \sim N(0, {\sigma_\mu}^2), \qquad \nu \sim N(0, {\sigma_\nu}^2), \tag{5}$$

$$\sigma_\mu = \left\{ \frac{\Gamma(1+\lambda) \cdot sin(\pi\lambda/2)}{\Gamma[(1+\lambda)/2] \cdot \lambda \cdot 2^{(\lambda-1)/2}} \right\}^{1/\lambda}, \qquad \sigma_\nu = 1. \tag{6}$$

s_0 denotes the initial step size of l$\acute{e}$vy flight in the formula 2.

The local random walk can be expressed as:

$$x_i^{t+1} = \begin{cases} x_i^t + r \cdot (x_{r1}^t - x_{r2}^t), & rand > P_a \\ x_i^t, & \text{otherwise} \end{cases} \tag{7}$$

where x_{r1} and x_{r2} are two different solutions chosen randomly. r and $rand(r, rand \in (0,1))$ are two stochastic numbers that obey the uniform distribution.

Based on the above description, the details of the original CS are showed in algorithm 1.

3. Related work

Up to now, new CS variants have emerged in large numbers, which ameliorate the performance of CS at different levels. Them can be roughly divided into three categories: (1) parameter control; (2) improvement and introduction of strategy for generating new solutions; and (3) hybridization.

Much significant work has been completed on control parameters of CS [12, 14–19]. In Ref. [12], authors enhanced the local search ability by nonlinearly reducing the

Algorithm 1 The original CS

Input: Population scale N, Maximum number of iterations $MaxIt$, problem dimension D, the switch parameter P_a

```
t = 1;
Randomly initialize population of N solutions x_i^t (i = 1,2,...,N);
Evaluate all solutions f_i^t = f(x_i^t);
while t < MaxIt do
    for i = 1 : N do
        Generate new solutions x_new using Lévy flight as showed in equation 1;
        Calculate its fitness f_new = f(x_new);
        if f_new^i < f_i^t then
            x_i^{t+1} = x_new^i;
            f_i^{t+1} = f_new^i;
        end if
    end for
    Desert a fraction(P_a) of worst solutions and generate new solutions by using equation 7;
    Update the global optimal solution;
    t = t + 1;
end while
```

Output: Optimal solution

step size of Lévy flight with the increase of iteration number. To improve the accuracy and convergence rate of CS, the parameters α and P_a were adjusted adaptively in [14]. In another study [15], the step size of Lévy flight was adaptively regulated according to the fitness of solution in order to accelerate the convergence speed of the CS. Li and Yin [16] introduced two new mutation rules and combined them by a linear descent rule to balance exploration and exploitation. Then, self-adaptive parameter setting was introduced to increase the diversity of the population. In Ref. [17], Guerrero et al. utilized a fuzzy system to dynamically adapt the parameters of CS. Simulation results show that their presented algorithm performs better than CS. Mareli et al. [18] proposed an adaptive CS by dynamically increasing the switching parameter to enhance the performance of algorithm. Mlakar et al. [19] presented a novel hybrid self-adaptive CS (HSA-CS). They added three features to the original CS: parameters adaptive control, linear population reduction and the mechanism for balancing the random exploration strategies during the CS search process. Experimental results demonstrate that the HSA-CS outperforms CS and several other state-of-the-art algorithms.

Some works have focused on the improvement and introduction of strategy for generating new solutions [20–26]. Peng et al. [20] inspired by the PSO in which individuals will converge to the weighted average of global optimal solution and personal optimal solution, and proposed a novel Gaussian bare-bones CS. Then, the experimental results illustrate that the presented algorithm is promising. Huang et al. [21] initialized the population by utilizing five chaotic sequences. The reset of solutions beyond the boundary and the adjustment of the L*é*vy flight step size were also realized by using these sequences. He et al. [22] proposed a new CS variant called GDCS which combines dynamic parameter selection and Gaussian bare-bones strategy. Besides, They combined GDCS with the Spark framework (SparkGDCS) to overcome the premature convergence of CS. Then, the experiment prove that SparkGDCS has excellent performance. Bilal H et al. [23] proposed seven novel selection schemes to replace the uniformly random-based selection method of original CS and good results were obtained. Based on quantum mechanism, cheuag et al. [24] proposed a nonhomogeneous search strategy to prevent the premature convergence of original CS. Liu et al. [25] added the inertia weight into the L*é*vy flight to improve global search capability of algorithm and employed chaos theory to increase the diversity of the initial population. Then, in order to further improve the search speed and convergence accuracy, the local search mechanism of frog leaping algorithm [27] was introduced. In Ref. [26], orthogonal learning strategy [28, 29] was introduced into CS in order to enhance the performance of CS.

One active research trend has been to hybridize CS with other methods and evolutionary computation techniques [30–40]. To overcome the slow convergence of CS, Shehab et al. [30] developed a hybrid CS by combining it with the bat algorithm [41]. Ding et al. [31] hybridized CS and PSO to deal with nonlinear optimization problems with multiple constraints. In another study, Abed-alguni et al. [32] proposed a hybrid CS and β-hill climbing algorithm [42] for maintaining the diversity of the solutions and balancing between the effectiveness and computational time of algorithm. Zhang et al. [33] presented a hybrid optimization algorithm by integrating CS with differential evolution algorithm [43] for solving constrained engineering problems. In their method, the shortcoming of these two algorithms was complemented by utilizing each other's advantages. In Ref. [34], CS and grey wolf optimizer [44] were combined to extract the parameters of solar photovoltaic models, and experimental results show that the proposed algorithm is a promising candidate technique. In Ref. [35], Shah et al. hybridized CS with extreme learning machine [45] to classify digitally modulated signals. In Ref. [36], the k-means operator [46] and CS were combined to solve the counterfort retaining wall optimization problem. In another study [37], CS was integrated with squirrel search algorithm [47] for brain magnetic resonance image analysis. Cui et al. [38] presented a hybrid many-objective CS for many objective optimization problems. Based on Support Vector Machine and hybrid multi-objective CS, Cai et al. [39] proposed a new method for software defect prediction. In Ref. [40], the performance of hybrid many-objective CS was improved by the combination of L*é*vy and exponential distributions.

Besides, CS has been applied in various domains with promising results, including engineering design [48, 49], economic dispatch problems [50, 51], clustering and data mining [52, 53], medical applications [54, 55], image processing [56, 57] and so on. In Ref. [48], an improved CS was proposed for solving electric distribution network reconfiguration problem. Compared with other existing methods and software, it can generate better distribution network configuration. In Ref. [49], Inci et al. proposed a dynamic CS for improving the energy extraction ability of fuel cell implementations. It was verified that the proposed method provides better performance than conventional methods. Zhao et al. [50] presented a modified CS to tackle economic power dispatch optimization problems. The simulation results reveal that the method is efficiency, especially for large-scale problems. In another study [51], a valid CS was developed for combined heat and power economic dispatch problem. Experimental results illustrate the proposed algorithm is feasible and credible. Based on quantum theory and chaotic map, Boushaki et al. [52] presented an improved CS for data clustering. Then, the significant superiority of the presented method was demonstrated by the results on six different real-life datasets. For mixed data, the partitional clustering algorithms are easy to fall into the local optimum. In order to solve this issue, Ji et al. [53] combined CS and k-prototypes clustering algorithm [58] to cluster mixed numeric and categorical data. In Ref. [54], CS was integrated with rough sets to build a model of heart disease diagnosis. Simulation results show the presented model has better performance than its competitors. In Ref. [55], Cristin et al. introduced CS into deep neural network to detect plant diseases. Then, experimental results prove that the presented method is superior to other existing methods. Kamoona et al. [56] presented an enhanced CS for image contrast enhancement. Compared with several image enhancement algorithms, the proposed algorithm shows superior performance in handling this work. In Ref. [57], a modified CS was applied on image de-noising. The results show that the presented method equips with better denoising performance for images with higher peak signal-to-noise ratio.

It is worth noting that the work in this paper belongs to a new variant of CS, which emphasizes on the diversity of strategies and the design of strategy usage. In this work, we propose three learning strategies and a multi-strategy serial framework to enhance the performance of CS, which inspired by the cuckoo's behavior of seeking host nest, eviction and begging.

Although the multi-strategy search mechanisms are already studied for other swarm intelligence algorithms such as SaDE [59], ABCVSS [60], ABCX [61] and iTSA [62], MSSCS is somewhat different from them. ABCX and iTSA tend to randomize when considering which strategy will be used for the update of solutions. Therefore, the strategy may not fully utilize its advantages in the search process, while MSSCS will maximize the performance of each strategy under the role of multi-strategy serial framework. For SaDE and ABCVSS, which need to record the situation that each strategy improves the solution, so as to determine the probability that each strategy will be used for searching in the next generation. However, MSSCS adopts specific and appropriate strategy to update the solution in different evolution stages, which makes the use of strategy more simplified. These differences mean that MSCS is a simple and efficient multi-strategy algorithm. Hence, different algorithmic characteristics and performance can be expected.

4. Motivation

According to the NFL, for any algorithm, the performance improvement of one type of problem will be offset by the performance of another problem. Similarly, when the algorithm focuses on local search capability, its global search capability will be ignored. The single search strategy in CS leads to the loss of the diversity of the search space. When faced with complex optimization problems, CS is prone to fall into local optimum. Although many scholars have done a great deal of work in improving the performance of the single strategy, the optimization ability of single strategy is always limited. Therefore, the multi-strategy algorithm is promising, and it has the potential to deal with complex situations.

In nature, the continuation of cuckoo offspring not only depends on the environment of habitat, but also on individual behavior during the growth process of cuckoos. As for cuckoos, there are three behaviors that increase the survival rate of young cuckoo: the behavior of female cuckoo seeking the host bird's nest, the eviction behavior of nestling cuckoo after it is hatched, and the begging behavior of young cuckoo for being fed. Inspired by these three behaviors, three new learning strategies are proposed to enhance the performance of CS.

Then, as the number of strategies increases, the complexity of the algorithm is a problem worth considering. In the multi-strategy algorithm, how to utilize the strategies is one of the keys to enhance the performance of the algorithm. However, multi-strategy algorithms tend to be complicated in how to select strategies to use. MSACS [63] uses strategies adaptively by calculating the probability of each strategy being selected, and the selection probability depends on the performance of the strategy in the past generations. Therefore, MSACS needs to use multiple memory libraries to store the performance data of each strategy separately, and then use these data to calculate the selection probability of each strategy, which undoubtedly increases the space and time complexity of the algorithm. In MsDE [64], the selection strategy needs to calculate the sum of the absolute difference of each dimension between the target and trial vectors. When dealing with high-dimensional problems, the computational cost of the algorithm is huge.

To overcome these limitations, based on the three serial behaviors during the growth of the cuckoo, a multi-strategy serial framework is proposed to reduce the complexity of the multi-strategy algorithm and maximize the performance of each learning strategy.

5. The proposed MSSCS

As mentioned earlier, the continuation of cuckoo offspring not only depends on the environment of habitat, but also on individual behavior during the growth process of cuckoos. Inspired by the cuckoo's behavior of seeking host nest, eviction and begging, three new learning strategies are proposed to enhance the performance of the algorithm and named saltation learning, Gaussian walk learning and single dimension learning respectively. Then, based on these three serial behaviors, we propose a multi-strategy serial framework to reduce the complexity of the multi-strategy algorithm and maximize the performance of each learning strategy. Besides, the adaptive parameter is designed to dynamically regulate the serial framework. The details of MSSCS can be found below.

5.1. Saltation learning

The selection of host nest is an important step of cuckoo's brood parasitism, which lays the foundation for the survival of cuckoo cubs. When female cuckoos looking for host nests, them often perched on high branches to observe the host bird's every move, determine the location of the nest and look for opportunities to lay eggs into it. Female cuckoos also search for nests in vegetation where the host bird may nest [65]. Once the location of the nest is determined, the female cuckoo will slide from the perch to the nest and lay eggs when the owner is not present. Egg-laying is rapid, 10 seconds or less [6]. Inspired by this behavior of cuckoos, a strategy suitable for the early search stage of algorithm is proposed, called saltation learning (SL). It is a strategy that the global search ability is slightly stronger than the local search ability and used to improve the overall convergence speed of the algorithm without causing the algorithm to fall into the premature convergence state, so as to obtain promising solutions and lay the solid foundation for further optimization in the later stage.

In SL, only one dimension is randomly updated for each individual in each generation, and the update of this dimension is implemented around a random dimension of the current best individual, which ensures that the algorithm has considerable convergence performance in the early stage of search. It is worth noting that the information used to update this dimension is derived from other dimensions different from this one. This saltation learning between dimensions greatly enhances the diversity of the search space and reduces the possibility of premature convergence. SL can be modeled as:

$$x_{i,j}^{t+1} = x_{best,m}^{t+1} + r \cdot (x_{r_1,n}^{t} - x_{worst,n}^{t}) \tag{8}$$

where x_{best}^{t} and x_{worst}^{t} are the best solution and the worst solution of the tth generation respectively. j, m and n are three different integers randomly selected from $[1, D](j \neq m \neq n)$. D is the dimension of the problem. r represents a random number subject to uniform distribution, and its value range is [-1,1]. r_1 is a random integer chosen randomly from $[1,N]$, and N represents the population scale.

The SL framework is given in figure 1, and it shows the learning object and learning manner of the candidate solution. We use a large square to represent an individual, and the number of small squares in the large square represents the dimension of the problem. The red squares denote the information used for learning, and the green squares denote the one-dimensional variable of the generated candidate solution. For simplicity, we assume that the problem dimension is 3. From figure 1, we can see that SL generates candidate solutions by learning the different dimensions of the best individual, worst individual and a random individual in the population. Therefore, this learning manner improves the exploration ability of algorithm.

5.2. Gaussian walk learning

After the cuckoo lay eggs successfully, once the nestling cuckoo is hatched, the nestling cuckoo will take the initiative to remove the host eggs from the nest, which increases the share of the host bird's food for the nestling cuckoo, so as to improve its survival rate [6]. Inspired by this eviction behavior of the nestling cuckoo, the Gaussian walk learning (GWL) is proposed for the intermediate search phase of the

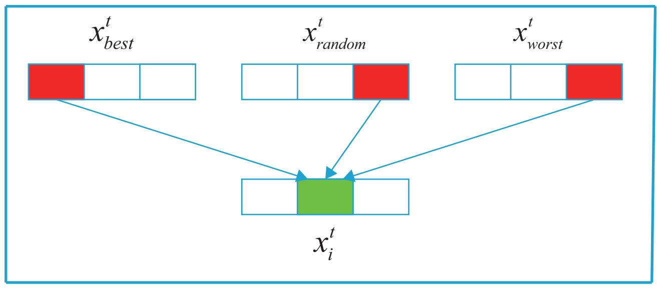


Figure 1: Illustration of SL

algorithm to balance the exploration and exploitation. Unlike SL, which only updates one dimension of solution in each generation, GWL updates the solution as a whole.

The GWL can be formulated as follows:

$$x_i^{t+1} = Gaussian(x_{best}^t, \sigma) + (r_1 \cdot x_{best}^t - r_2 \cdot x_i^t), \sigma = c \cdot exp(-t/t_{max}) \cdot (x_i^t - x_{worst}^t) \quad (9)$$

where t, t_{max} represent the current number of iterations and the maximum number of iterations respectively. r_1 and $r_2 (r_1, r_2 \in [0,1])$ are two random numbers that obey the uniform distribution. $Gaussian(x_{best}^t, \sigma)$ is a Gaussian distribution with x_{best}^t as expectation and σ as standard deviation, where $\sigma = c \cdot exp(-t/t_{max}) \cdot (x_i^t - x_{worst}^t)$. c is a constant between 0 and 1. The term $c \cdot exp(-t/t_{max})$ is designed to control the step size of GWL. The search direction of GWL is adjusted by the term $(r_1 \cdot x_{best}^t - r_2 \cdot x_i^t)$.

5.3. Single dimension learning

The young cuckoo will loudly and persistently beg its host parents for care after the stage of the nestling cuckoo's eviction behavior. Whenever the host bird approaches, the young cuckoo increases the frequency and duration of its call. This call is so loud and frequent that the cuckoo makes as much noise as the entire brood of its host species [6]. This undoubtedly increases the probability of the young cuckoo being fed, thereby increasing its survival rate. Enlightened by the begging behavior of the young cuckoo, the single dimension learning (SDL) is presented for the late search phase of the algorithm to increase the convergence speed so that the solution obtained is closer to the global optimal solution.

In SDL, each individual in each generation is also randomly updated in only one dimension, but the information used to update this dimension comes from the same dimension as it. Although the learning behavior of SDL in the same dimension weakens the correlation between dimensions, strengthens the local search ability of the algorithm, which speeds up the convergence speed of the algorithm at the end of the search. SDL is performed as follows:

$$x_{i,j}^{t+1} = x_{best,j}^{t+1} + r \cdot (x_{r_1,j}^{t} - x_{worst,j}^{t}) \tag{10}$$

where r_1 is the an integer chosen at random in the range [1, N], N denotes the population size. Here, j represents an integer selected randomly from the set $\{1,2,3,...,D\}$, where D is the problem' dimension. r denotes a random number subject to uniform distribution, and its value range is [-1,1], which could perform the bidirectional search and enhance the local search capability of algorithm.

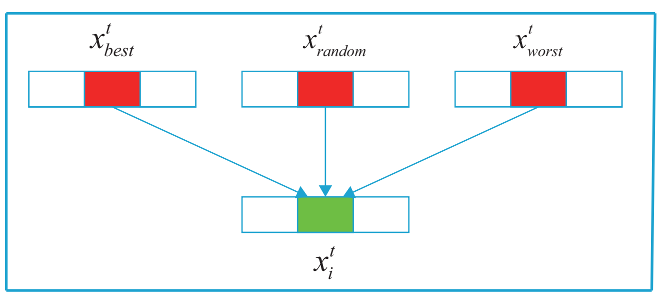


Figure 2: Illustration of SDL

From figure 2, we can observe that the learning objects of candidate solutions in SDL are the best individual, the worst individual and a random individual in the population, and SDL has learning behaviors in the same dimension, which improves the exploitation ability of algorithm.

5.4. Multi-strategy serial framework

Based on the three serial behaviors of the cuckoo's growth process, a multi-strategy serial framework is proposed to maximize the performance of each learning strategy and reduce the complexity of the multi-strategy algorithm. In this serial framework, we introduce a switching parameter *SP* to determine whether to use the strategy draw from the serial strategy pool (*SSP*). In this pool, each strategy SSP_i is defined as follows:

$$SSP_i = \begin{cases} \text{SL8}, & \text{if } t \in [1, PA * MaxIt] \\ \text{GWL9}, & \text{if } t \in [PA * MaxIt, (1 - PA) * MaxIt] \\ \text{SDL10}, & \text{otherwise} \end{cases} \tag{11}$$

Where SSP_i denotes the ith strategy in the *SSP*, and $i \in \{1, 2, 3\}$. *PA* is a probability, and t, $MaxIt$ represent the current number of iterations and the maximum number of iterations respectively.

As for *SP*, when the candidate solution produced by using the strategy of the first part of the algorithm is worse than the current solution, the value of *SP* will increase by 1. In other words, *SP* is used to record the number of unimproved solutions. If *SP* is not less than the threshold T, the algorithm will update the solution by the strategy draw from *SSP*. Otherwise, the algorithm will continuously use lévy flight with powerful global search capability to update the solution L times. The idea is that if the number of unimproved solutions is small, it is worthwhile for the algorithm to consume several iterations to search for a better solution using a strategy that is biased towards global search. On the contrary, if the number of unimproved solutions is large, the algorithm needs to be optimized using the strategy suitable for the corresponding search stage to obtain the higher quality solution.

For the use of *SSP*, when the algorithm is in the early search phase, SL will be selected from *SSP* to acquire the promising solution and prevent the algorithm from entering the state of premature convergence, so as to provide a solid foundation for the optimization in the later phase of the algorithm. To balance the exploration and exploitation of the algorithm, GWL is drawn from *SSP* to update the solution in the intermediate search phase of the algorithm. In the late search phase of the algorithm, SDL is chosen from *SSP* to increase the convergence speed of the algorithm, so that the obtained solution is closer to the global optimal solution. The details of MSSCS are presented in algorithm 2.

Algorithm 2 Multi-strategy serial CS algorithm

Input: Population scale N, Maximum number of iterations $MaxIt$, problem dimension D, the switch parameter P_a

$t = 1$;
Randomly initialize population of N solutions $x_i^t (i = 1, 2, ..., N)$;
Evaluate all solutions $f_i^t = f(x_i^t)$;
The values of the parameters T and PA are determined by equations 12 and 13 respectively;
$SP = 0$
while $t < MaxIt$ **do**
    if $SP \geq T$ **then**
        Using equation 11 to gain new solutions x_{new};
    else
        Using Lévy flight 1 L times continuously to generate new solutions;
    end if
    Calculate the fitness of all new solutions $f_{new} = f(x_{new})$;
    $SP = 0$
    for $i = 1 : N$ **do**
        if $f_{new}^i < f_i^t$ **then**
            $x_i^{t+1} = x_{new}^i$;
            $f_i^{t+1} = f_{new}^i$;
        else
            $SP = SP + 1$;
        end if
    end for
    Desert a fraction(P_a) of worst solutions and generate new solutions by using equation 7;
    Update the global optimal solution;
    $t = t + 1$
end while

Output: Optimal solution

5.5. *Parameter dynamic tuning*

The parameter T implies the minimum condition for using *SSP*, that is, the number of unimproved solutions must be greater than or equal to T. It controls the probability of *SSP* and Lévy flight being used, and affects the exploration and exploitation of algorithm. From another perspective, T is related to the population scale N, so it should be dynamically fine-tuned as the population scale changes. The fine-tuning of T is defined as follows:

$$T = \frac{N}{2.5} \tag{12}$$

For the parameter PA, it determines the share of each learning strategy in the SSP. The parameter PA is dynamically fine-tuned as follows:

$$PA = PA_{min} + (PA_{max} - PA_{min}) \cdot \frac{1}{D} \tag{13}$$

where PA_{max} and PA_{min} are the maximum and minimum of PA, respectively. D is the dimension of problem.

Therefore, the multi-strategy serial framework is dynamically fine-tuned according to the size of the population and the dimension of the problem.

6. Complexity analysis

To prove that the proposed algorithm does not increase the complexity of CS, we analyze the complexity of MSSCS and CS. Since the second part of MSSCS is the same as that of CS, we only analyze the time complexity of the first part of these two algorithms.

First, we analyze the time complexity of CS. Assume that the execution time of randomly initializing a variable is C_1, and the time to evaluate a D-dimensional vector is $f(D)$. The size of the population is N and the dimension of the problem is D. Hence, the time complexity of CS initialization phase is $O(N(DC_1 + f(D))) \approx O(N(D + f(D)))$. Suppose that the execution time of a variable in the new solution generated by equation 1 is C_2, the time of comparing the fitness value of the new solution with the old solution is C_3, and the time to replace a variable in the old solution is C_4, then the total time complexity of the new solution's generation, the comparison fitness value and the replacement solution is $O(N(DC_2 + f(D) + C_3 + DC_4)) \approx O(N(D + f(D)))$. If the maximum iteration number $MaxIt$ is used as the termination condition, the total time complexity of the first part of CS is $MaxIt \cdot 2 \cdot O(N(D + f(D))) \approx O(MaxIt \cdot N(D + f(D)))$.

Next, the time complexity of MSSCS is analyzed. Assume that the execution time of a variable in the new solution generated by SL, MGW, and SDL is C_5, C_6, and C_7, respectively. The time of all conditional judgment is C_8, random number generation takes C_9, and the execution time of tuning the parameter SP value is C_{10}. Since the relationships among SL, MGW, and SDL are serial, the total time complexity of their solution generation is $O(MaxIt \cdot PA \cdot N \cdot DC_5) + O(MaxIt(1-PA) \cdot N \cdot DC_6) + O(MaxIt \cdot PA \cdot N \cdot DC_7) \approx O(MaxIt \cdot N \cdot D)$. However, the relationship between L$\acute{e}$vy Flight and the three learning strategies is parallel. Assuming that the probability of using L$\acute{e}$vy flight is P, the total time complexity of MSSCS is: $P \cdot O(MaxIt \cdot N \cdot D) + (1-P) \cdot O(MaxIt \cdot N(D + f(D))) + O(MaxIt \cdot N(C_8 + C_9 + C_{10} + f(D) + C_3 + DC_4)) + O(MaxIt \cdot N(D + f(D))) \approx O(MaxIt \cdot N(D + f(D)))$.

Therefore, the time complexity of MSSCS is consistent with the original CS. In terms of space complexity, MSSCS does not use additional storage space to store data, so it does not increase the spatial complexity of the algorithm. To sum up, the complexity of MSSCS is the same as that of the original CS, which belong to the same order of magnitude. In addition, the above analysis also shows that the multi-strategy serial framework reduces the complexity of the multi-strategy algorithm.

7. Result analysis and discussion

7.1. Experimental setting

In this section, MSSCS is tested on 28 benchmark functions from CEC2013 [66] to verify its performance. Among these functions, $f_1 - f_5$ are unimodal functions, $f_6 - f_{20}$ belong to the multimodal functions, $f_{21} - f_{28}$ are classified as composition functions. All experiments are carried out on Windows 10 platform and all algorithms are implemented in MATLAB 2016.

In our experiments, the function error $|f(x) - f(x^*)|$ is recorded, where $f(x^*)$ represents the standard optimal value of the objective function and $f(x)$ denotes the actual optimal value of the objective function. Therefore, the closer the function error is to 0, the better. Besides, in order to reduce the statistical error, the average error of all these functions running independently is selected as the performance metric.

To prove the effectiveness of the improved CS algorithm, we compare MSSCS with CS and 9 state-of-art variants. These 9 CS variants include: VCS [67], ICS [68], ECS [69], GBCS [20], GCS [70], SDCS [71], NACS [72], ACS [73] and ACSA [15]. The parameter settings of all algorithm are presented in Table 1.

7.2. Comparison of MSSCS with CS and nine state-of-art variants

In this subsection, MSSCS is compared with CS and its 9 variants. The parameter settings for all algorithms participating in the comparison are presented in Table 1. For MSSCS, The parameters L is equal to 1. In order to be fair, for all algorithms, the population scale N is set to 25, the dimension of the problem D equals 30, and we have determined 20000 as the maximum number of iterations because of is sufficient for most of the problems. Besides, all the algorithms are executed independently for 30 times. To better present the experimental data, the experimental results of 11 algorithms are listed in two tables(Table 2 and Table 3). In these table, "Mean" and "Std" denote the average function error and standard deviation, respectively. The best average value and the best standard deviation are shown in **bold**.

From Table 2, we can observe that MSSCS obtains the best result on 7 functions, which are f_6, f_{11}, f_{14}, f_{17}, f_{19}, f_{21} and f_{22}, and only it can find the global optimum on f_{11}. For ICS, it yields the optimal value on 3 functions(f_2, f_{10} and f_{26}). On f_1 and f_3, all the algorithms provide the same results. Specifically, only CS obtains the best values on f_8. Similarly, GBCS acquires the best result on f_4. For VCS and ECS, them can't gain the optimal result on any function. Besides, Table 3 also reveals the fact that MSSCS is better than other algorithms on most functions.

In addition, the statistical results of Wilcoxon's rank sum test are presented at the bottom of Table 2 and Table 3. The significance level for this test is set to 0.05. $+/-/\approx$ indicates the number of functions superior, inferior and similar to MSSCS compared with other algorithms, respectively. From Table 2, we can see that MSSCS is significantly better than others. Compared with CS, VCS, ICS and GBCS, MSSCS is superior to them in no less than 18 functions and worse to them in 0, 2, 3 and 2 out of 28 functions. Similarly, MSSCS is better than ECS in 16 functions, inferior to it in 4 functions and similar in 8 functions. From Table 3, the results show that MSSCS outperforms

Table 1: Parameter setting

Algorithm	Properties	Parament settings
CS	Original CS	$\alpha = 0.01$, $\beta = 1.5$, $P_a = 0.25$
VCS	In VCS, the step-size of l$\acute{e}$vy flight is adjusted by a varied scaling factor.	$\beta = 1.5$, $P_a = 0.25$
ICS	The strategy for tuning the parameters α and P_a is proposed to improve the performance of CS.	$\alpha_{max} = 0.5$, $\alpha_{min} = 0.01$, $\beta = 1.5$, $P_{amax} = 0.5$, $P_{amin} = 0.05$
ECS	To enhance the local search ability of algorithm, an enhanced CS is proposed based on Gaussian diffusion random walk and greedy selection approach.	$P_a = 0.25$
GBCS	Inspired from PSO, a novel Gaussian barebones CS algorithm is presented to enhance the performance of CS.	$\beta = 1.5$, $P_a = 0.25$
GCS	A novel CS base on Gauss distribution is proposed to accelerate the convergence speed of CS.	$\mu = 0.0001$, $\sigma_0 = 0.5$, $P_a = 1.5$
SDCS	In SDCS, two snap and drift modes are presented to balance between exploration and exploitation.	$\alpha = 0.01$, $\beta = 1.5$, $P_a = 0.25$, $\omega = 0.005$, $J = 0.3$
NACS	Based on ring topology, the neighborhood attraction scheme is designed to improve the performance of CS.	$P_a = 0.25$, $m = 3$, $P_s = 0.8$
ACS	the step-size is regulated adaptively according to the fitness of solutions.	$P_a = 0.25$
ACSA	An adaptive step-size adjustment strategy is proposed to dynamically regulate the parameter α.	$\alpha_U = 0.8$, $\alpha_L = 0.2$, $\beta = 1.5$
MSSCS	Inspired by the three serial behaviors of the cuckoo's growth process, three new learning strategies and a multi-strategy serial framework are proposed to enhance the performance of CS.	$\alpha = 0.01$, $\beta = 1.5$, $P_a = 0.25$, $c = 0.2$, $PA_{max} = 0.35$, $PA_{min} = 0.25$

significantly GCS, SDCS, NACS, ACS and ACSA in no less than 19 functions. Obviously, MSSCS is better than its peers.

Then, in order to compare the average performance of MSSCS with its competitors, the non-parametric statistical Friedman test is performed. The experimental results are displayed at the bottom of those two tables(Table 2 and Table 3). The best average ranking value is marked in **bold**. MSSCS obtains the best one. In addition, the Wilcoxon's signed rank test is also executed by using the KEEL software [74] and the statistical analysis results are showed in Table 4. From this table, we can see that the R^+ value of MSSCS is higher than the R^- value for all cases and there are significant differences between MSSCS and other algorithms in terms of the Wilcoxon's signed rank test at $\alpha = 0.05$. The above experimental results indicate that MSSCS performs better than other algorithms in handling CEC2013 benchmark functions with 30 dimensions.

Next, we conduct a rank based statistical test that shows clearly the ranking of each algorithm on each test function to show the advantages of MSSCS more intuitively. These two stack histogram in Figure 3 and Figure 4 show the test results, which presents the ranking of the "Mean" values of all algorithms in each test function. In the stack histogram, each color block represents a rank, among which the first is yellow-green, followed by red, blue, black, green and the last is orange. Each algorithm has its own color block on 28 test functions, that is, ranking. Therefore, if an algorithm has more yellow-green color blocks, its overall ranking will be higher, which means that its performance is better. Conversely, if the more orange color blocks, the lower the overall ranking, which indicates that its performance is worse. From Figure 3 and Figure 4, we can observe that most of the yellow-green blocks was obtained by MSSCS, which indicates that MSSCS remains the best performer among all algorithms.

Finally, in order to further prove the performance of MSSCS, the convergence of algorithm is researched. We choose 5 CS variants with outstanding performance and 9 functions for comparative study. Figure 5 displays the convergence graph of the six algorithms on 9 benchmark functions. The ordinate is the mean function error and the abscissa denotes the number of iterations. From Figure 5, apparently, the convergence speed of MSSCS is obviously faster than other competitors on f_9, f_{11}, f_{14} and f_{22}. For the remaining functions, although MSSCS is not significantly faster than other competitors, it is still the fastest among all algorithms. Therefore, it can be concluded that MSSCS acquires the solution with higher quality than its competitors when dealing with 30 dimensional problems.

Besides, from Figure 5, we can clearly see that the role of each learning strategy in the multi-strategy serial framework during the entire search process of MSSCS, which is reflected in f_9, f_{14}, f_{19}, f_{22}, f_{24} and f_{27}. Especially for f_9, f_{14} and f_{22}, according to their convergence graph, we can observe that MSSCS has a considerable convergence speed in the early iterations, but this does not cause premature convergence of the algorithm. It indicates that MSSCS have the strong global search performance in the early stage of search. In the middle of the iteration, the convergence speed of MSSCS is relatively stable, which reflects the balance between exploration and exploitation of MSSCS. However, at the end of the iteration, the convergence rate of MSSCS suddenly increased, making the obtained solution closer to the global optimal solution. This phenomenon precisely demonstrates the performance of each learning strategy in MSSCS. It also shows that the multi-strategy serial framework enables each learning strategy to exert its own corresponding capability and maximize its performance.

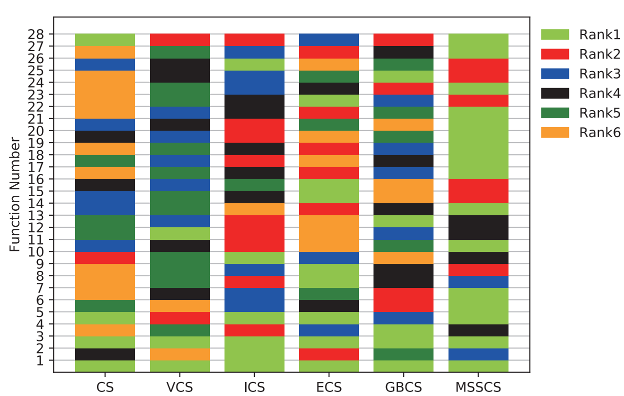


Figure 3: Stacked histogram of ranking for CS, VCS, ICS, ECS, GBCS and MSSCS on CEC2013(D = 30)

7.3. *Effectiveness analysis of multi-strategy serial framework*

To confirm the effectiveness of multi-strategy serial framework and the three learning strategies, four new CS versions is presented in this subsection.

Table 2: Comparison of MSSCS with CS, VCS, ICS, ECS and GBCS on CEC2013(D = 30)

Function	Mean/Std	CS	VCS	ICS	ECS	GBCS	MSSCS
f_1	Mean	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_2	Mean	7.25E+05	1.55E+06	**5.43E+04**	1.41E+05	1.54E+06	6.02E+05
	Std	3.24E+05	8.80E+05	**3.76E+04**	5.02E+04	7.00E+05	2.73E+05
f_3	Mean	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_4	Mean	1.77E+03	7.25E+02	2.31E+02	2.37E+02	**1.40E+02**	3.04E+02
	Std	7.12E+02	2.68E+02	8.73E+01	2.61E+02	**5.06E+01**	1.90E+02
f_5	Mean	**0.00E+00**	7.58E-15	**0.00E+00**	**0.00E+00**	3.03E-14	**0.00E+00**
	Std	**0.00E+00**	2.84E-14	**0.00E+00**	**0.00E+00**	5.03E-14	**0.00E+00**
f_6	Mean	3.44E+00	4.35E+00	3.25E+00	3.33E+00	3.06E+00	**1.52E-01**
	Std	6.09E+00	9.21E+00	5.22E+00	8.54E+00	7.87E+00	**7.13E-01**
f_7	Mean	**1.07E+02**	6.15E+01	5.90E+01	9.47E+01	5.26E+01	3.74E+01
	Std	1.85E+01	1.98E+01	**1.13E+01**	1.94E+01	2.31E+01	1.52E+01
f_8	Mean	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**
	Std	**3.76E-02**	5.30E-02	5.06E-02	5.09E-02	4.25E-02	5.47E-02
f_9	Mean	2.87E+01	2.76E+01	2.43E+01	2.31E+01	2.70E+01	**1.96E+01**
	Std	1.98E+00	1.58E+00	1.56E+00	3.36E+00	**1.49E+00**	4.19E+00
f_{10}	Mean	1.25E-02	1.75E-02	**6.01E-03**	1.33E-02	3.04E-02	1.35E-02
	Std	9.90E-03	1.30E-02	**5.81E-03**	1.00E-02	1.94E-02	1.04E-02
f_{11}	Mean	4.21E+00	8.39E+00	3.78E+00	4.65E+01	1.77E+01	**0.00E+00**
	Std	2.26E+00	3.41E+00	1.48E+00	3.01E+01	5.96E+00	**0.00E+00**
f_{12}	Mean	1.50E+02	**7.15E+01**	7.43E+01	1.85E+02	7.59E+01	8.36E+01
	Std	3.17E+01	1.44E+01	1.41E+01	5.04E+01	**1.28E+01**	2.04E+01
f_{13}	Mean	1.84E+02	1.15E+02	1.15E+02	2.52E+02	**1.12E+02**	1.28E+02
	Std	3.73E+01	1.84E+01	**1.77E+01**	4.28E+01	2.36E+01	3.09E+01
f_{14}	Mean	7.09E+02	8.34E+02	8.41E+02	1.25E+00	7.31E+02	**1.05E-01**
	Std	2.25E+02	3.24E+02	1.94E+02	3.98E+00	2.95E+02	**3.87E-02**
f_{15}	Mean	4.08E+03	4.44E+03	4.18E+03	**3.32E+03**	4.51E+03	3.54E+03
	Std	**2.76E+02**	2.81E+02	3.53E+02	4.35E+02	4.27E+02	4.30E+02
f_{16}	Mean	1.44E+00	1.44E+00	1.57E+00	**7.36E-01**	1.57E+00	9.95E-01
	Std	**1.97E-01**	2.10E-01	2.34E-01	2.24E-01	2.34E-01	2.65E-01
f_{17}	Mean	6.38E+01	4.96E+01	4.90E+01	3.91E+01	4.70E+01	**3.05E+01**
	Std	7.89E+00	6.65E+00	6.51E+00	2.12E+01	5.86E+00	**7.21E-02**
f_{18}	Mean	2.00E+02	1.29E+02	1.23E+02	2.27E+02	1.31E+02	**1.02E+02**
	Std	3.01E+01	1.45E+01	**1.40E+01**	6.82E+01	1.60E+01	1.73E+01
f_{19}	Mean	7.46E+00	4.38E+00	3.79E+00	2.22E+00	3.50E+00	**1.00E+00**
	Std	1.99E+00	1.35E+00	7.25E-01	6.22E-01	1.06E+00	**1.78E-01**
f_{20}	Mean	1.22E+01	1.19E+01	1.18E+01	1.31E+01	1.23E+01	**1.10E+01**
	Std	3.66E-01	**3.17E-01**	3.25E-01	1.72E+00	1.05E+00	6.82E-01
f_{21}	Mean	2.42E+02	2.44E+02	2.28E+02	3.07E+02	3.11E+02	**2.13E+02**
	Std	6.73E+01	5.91E+01	4.27E+01	7.56E+01	6.82E+01	**3.40E+01**
f_{22}	Mean	1.09E+03	8.52E+02	9.36E+02	1.02E+02	9.92E+02	**7.80E+00**
	Std	3.83E+02	3.55E+02	4.43E+02	9.15E+01	3.49E+02	**3.91E+00**
f_{23}	Mean	4.98E+03	4.80E+03	4.77E+03	**3.92E+03**	4.61E+03	4.29E+03
	Std	4.44E+02	4.06E+02	**3.61E+02**	6.29E+02	4.36E+02	5.22E+02
f_{24}	Mean	2.82E+02	2.75E+02	2.72E+02	2.75E+02	2.68E+02	**2.61E+02**
	Std	1.13E+01	4.89E+00	**3.98E+00**	9.79E+00	8.02E+00	8.18E+00
f_{25}	Mean	2.98E+02	2.85E+02	2.82E+02	2.93E+02	2.77E+02	**2.75E+02**
	Std	5.00E+00	5.73E+00	4.24E+00	**1.21E+01**	7.23E+00	1.04E+01
f_{26}	Mean	**2.00E+02**	**2.00E+02**	**2.00E+02**	2.91E+02	2.06E+02	**2.00E+02**
	Std	1.41E-02	2.24E-02	**2.61E-03**	8.02E+01	3.15E+01	9.33E-03
f_{27}	Mean	1.04E+03	1.03E+03	9.56E+02	9.26E+02	9.84E+02	**8.21E+02**
	Std	1.81E+02	**3.96E+01**	1.11E+02	9.50E+01	6.83E+01	1.71E+02
f_{28}	Mean	2.93E+02	3.00E+02	3.00E+02	1.19E+03	3.00E+02	**2.87E+02**
	Std	3.59E+01	**0.00E+00**	5.87E-14	1.12E+03	5.84E-13	4.99E+01
Average ranking		4.55	4.20	2.95	3.55	3.73	**2.02**
$+/-/\approx$		19/0/9	18/2/8	18/3/7	16/4/8	21/2/5	

Table 3: Comparison of MSSCS with GCS, SDCS, NACS, ACS and ACSA on CEC2013(D = 30)

Function	Mean/Std	GCS	SDCS	NACS	ACS	ACSA	MSSCS
f_1	Mean	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_2	Mean	1.36E+06	**3.33E+04**	7.20E+04	1.11E+06	1.61E+06	6.02E+05
	Std	6.11E+05	**2.09E+04**	3.57E+04	3.41E+05	7.98E+05	2.73E+05
f_3	Mean	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_4	Mean	3.98E+03	**1.82E+01**	4.11E+03	1.10E+03	1.83E+03	3.04E+02
	Std	1.14E+03	**2.03E+01**	1.78E+03	4.99E+02	5.02E+02	1.90E+02
f_5	Mean	**0.00E+00**	2.65E-14	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Std	**0.00E+00**	4.81E-14	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_6	Mean	3.69E+00	1.29E+01	7.52E+00	4.36E+00	6.89E+00	**1.52E-01**
	Std	7.28E+00	8.92E+00	6.58E+00	7.98E+00	9.68E+00	**7.13E-01**
f_7	Mean	8.08E+01	1.11E+02	8.74E+01	1.07E+02	6.79E+01	**3.74E+01**
	Std	2.82E+01	2.59E+01	1.85E+01	2.19E+01	1.89E+01	**1.52E+01**
f_8	Mean	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**
	Std	4.05E-02	5.34E-02	5.12E-02	**3.25E-02**	3.62E-02	5.47E-02
f_9	Mean	2.90E+01	2.61E+01	2.85E+01	2.66E+01	2.80E+01	**1.96E+01**
	Std	1.85E+00	2.75E+00	2.27E+00	2.00E+00	**1.30E+00**	4.19E+00
f_{10}	Mean	2.47E-02	4.97E-02	3.05E-01	1.40E-02	1.74E-02	**1.35E-02**
	Std	2.05E-02	3.45E-02	1.96E-01	9.35E-03	1.36E-02	**1.04E-02**
f_{11}	Mean	1.95E+01	2.25E+01	7.41E+01	7.33E+00	5.94E+00	**0.00E+00**
	Std	9.10E+00	6.50E+00	2.00E+01	4.51E+00	2.70E+00	**0.00E+00**
f_{12}	Mean	1.12E+02	1.29E+02	1.53E+02	1.28E+02	**7.03E+01**	8.36E+01
	Std	2.05E+01	3.49E+01	3.63E+01	2.28E+01	**1.40E+01**	2.04E+01
f_{13}	Mean	1.56E+02	2.03E+02	2.35E+02	1.58E+02	**1.11E+02**	1.28E+02
	Std	2.79E+01	4.09E+01	3.64E+01	2.47E+01	**1.40E+01**	3.09E+01
f_{14}	Mean	1.39E+03	1.12E+03	1.69E+03	1.21E+03	1.03E+03	**1.05E-01**
	Std	4.03E+02	2.59E+02	5.25E+02	4.28E+02	3.50E+02	**3.87E-02**
f_{15}	Mean	4.83E+03	3.56E+03	3.85E+03	4.10E+03	4.78E+03	**3.54E+03**
	Std	4.05E+02	3.76E+02	5.87E+02	**3.34E+02**	3.70E+02	4.30E+02
f_{16}	Mean	1.67E+00	1.22E+00	1.15E+00	**9.90E-01**	1.63E+00	9.95E-01
	Std	2.23E-01	2.48E-01	2.87E-01	**1.54E-01**	2.09E-01	2.65E-01
f_{17}	Mean	6.92E+01	1.29E+02	9.86E+01	9.06E+01	5.77E+01	**3.05E+01**
	Std	1.06E+01	2.66E+01	1.88E+01	1.04E+01	7.75E+00	**7.21E-02**
f_{18}	Mean	1.75E+02	2.12E+02	1.56E+02	2.34E+02	1.40E+02	**1.02E+02**
	Std	2.28E+01	5.80E+01	3.63E+01	2.67E+01	**1.36E+01**	1.73E+01
f_{19}	Mean	7.81E+00	4.91E+00	1.21E+01	9.48E+00	5.30E+00	**1.00E+00**
	Std	2.47E+00	1.53E+00	3.59E+00	2.12E+00	1.28E+00	**1.78E-01**
f_{20}	Mean	1.20E+01	1.20E+01	1.18E+01	1.22E+01	1.20E+01	**1.10E+01**
	Std	5.49E-01	4.72E-01	7.71E-01	3.94E-01	**3.52E-01**	6.82E-01
f_{21}	Mean	2.67E+02	2.48E+02	3.21E+02	2.65E+02	2.68E+02	**2.13E+02**
	Std	6.26E+01	6.05E+01	9.88E+01	5.91E+01	5.75E+01	**3.40E+01**
f_{22}	Mean	1.88E+03	1.32E+03	2.19E+03	1.70E+03	1.46E+03	**7.80E+00**
	Std	5.56E+02	3.56E+02	7.35E+02	5.12E+02	5.89E+02	**3.91E+00**
f_{23}	Mean	5.30E+03	4.40E+03	4.46E+03	4.68E+03	5.05E+03	**4.29E+03**
	Std	**4.34E+02**	5.85E+02	5.58E+02	4.89E+02	4.54E+02	5.22E+02
f_{24}	Mean	2.75E+02	2.72E+02	2.78E+02	2.74E+02	2.76E+02	**2.61E+02**
	Std	1.08E+01	7.06E+00	1.18E+01	9.39E+00	**5.58E+00**	8.18E+00
f_{25}	Mean	2.95E+02	2.78E+02	2.96E+02	2.91E+02	2.81E+02	**2.75E+02**
	Std	6.18E+00	6.05E+00	**5.76E+00**	6.55E+00	7.59E+00	1.04E+01
f_{26}	Mean	**2.00E+02**	2.06E+02	2.11E+02	**2.00E+02**	**2.00E+02**	**2.00E+02**
	Std	7.41E-02	3.00E+01	4.29E+01	1.52E-02	1.20E-01	**9.33E-03**
f_{27}	Mean	1.10E+03	9.71E+02	1.05E+03	9.96E+02	1.03E+03	**8.21E+02**
	Std	5.37E+01	6.60E+01	5.93E+01	1.26E+02	**4.46E+01**	1.71E+02
f_{28}	Mean	3.00E+02	3.00E+02	3.72E+02	3.00E+02	3.00E+02	**2.87E+02**
	Std	8.30E-14	2.49E-13	3.00E+02	**0.00E+00**	**0.00E+00**	4.99E+01
Average ranking		4.25	3.32	4.66	3.71	3.46	**1.59**
$+/-/\approx$		22/0/6	20/2/6	21/1/6	20/0/8	19/2/7	

Table 4: Statistical results based on the Wilcoxon's signed rank test(D = 30)

MSSCS vs.	R^+	R^-	p-value	$\alpha = 0.05$
CS	373.5	4.5	0.000009	Yes
VCS	375.5	30.5	0.000082	Yes
ICS	287.5	90.5	0.017384	Yes
ECS	272.5	105.5	0.043581	Yes
GBCS	346.5	59.5	0.001041	Yes
GCS	376.5	1.5	0.000006	Yes
SDCS	347.5	58.5	0.00096	Yes
NACS	346.5	31.5	0.000147	Yes
ACS	371.5	6.5	0.000011	Yes
ACSA	344.5	33.5	0.000178	Yes

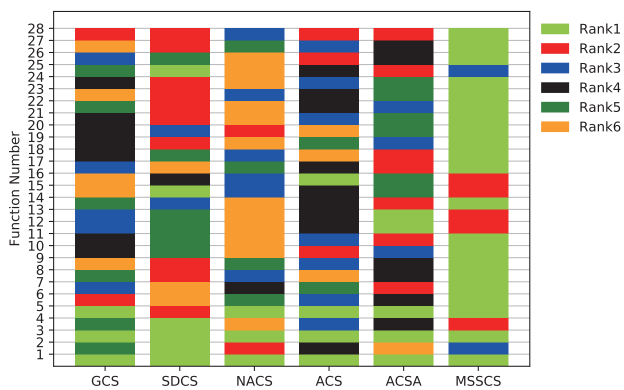
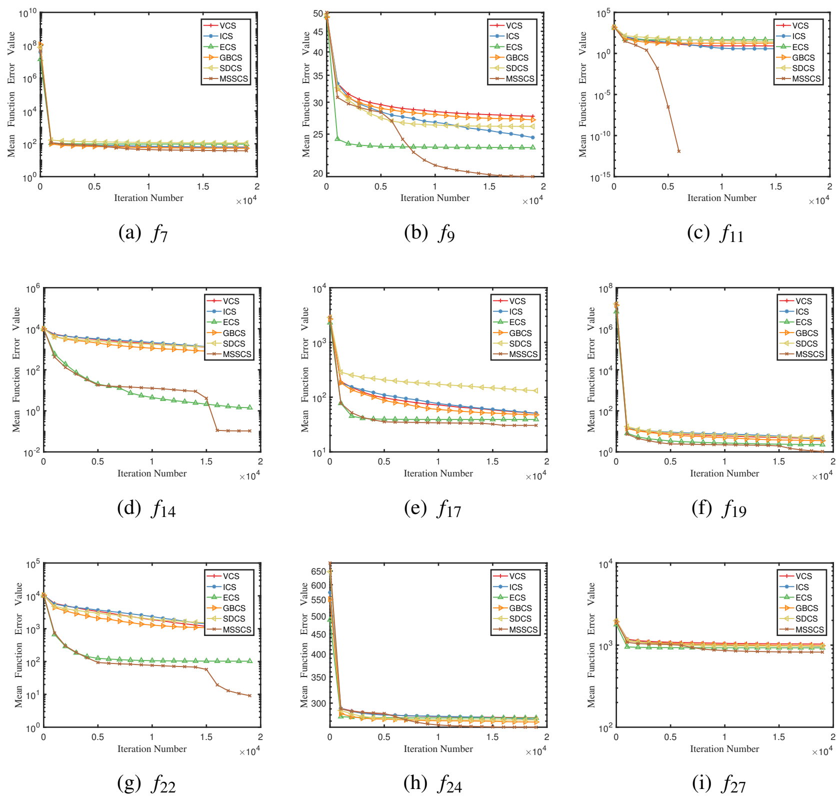


Figure 4: Stacked histogram of ranking for GCS, SDCS, NACS, ACS, ACSA and MSSCS on CEC2013(D = 30)

The new CS versions can be described as:

- SL-CS: Replace Lévy flight in CS with SL.
- GWL-CS: Lévy flight is replaced by GWL.
- SDL-CS: SDL displaces the Lévy flight.
- EPSS-CS: It is an equal probability selection strategy, which selects Lévy flight, SL, GWL and SDL with equal probability during the search process to update the solution.

In this subsection, these four new CS versions are also implemented on CEC2013.



(a) f_7 (b) f_9 (c) f_{11}

(d) f_{14} (e) f_{17} (f) f_{19}

(g) f_{22} (h) f_{24} (i) f_{27}

Figure 5: Convergence curves of VCS, ICS, ECS, GBCS, SDCS and MSSCS(D = 30)

The parameter settings are as same as last subsection. Each function runs 30 times individually and the results are summarized in Table 5. The lowest "Mean" is marked in **bold**. Meanwhile, the Friedman test and Wilcoxon's rank sum test are executed, and the results are showed at the bottom of Table 5. According to the results of Friedman test, the ranking values of SL-CS, GWL-CS and SDL-CS are all lower than CS. This shows that the performance of our proposed new CS versions outperforms CS, that is, the learning strategies we proposed is effective. Meanwhile, the ranking value of EPSS-CS is lower than CS and three other CS versions, which shows that the use of multi-strategy is an effective way to enhance the performance of the algorithm. However, the ranking value of MSSCS is lower than EPSS-CS, which shows each learning strategy is reasonably used under the action of multi-strategy serial framework, that is, the framework is effective in improving the performance of the algorithm. On the basis of the results of Wilcoxon's rank sum test, MSSCS is significantly better than SL-CS, GWL-CS, SDL-CS and EPSS-CS on 14, 16, 16 and 11 out of 28 functions, and inferior to them on 2, 3, 1 and 1 out of 28 ones, respectively. This further proves the effectiveness of multi-strategy serial framework. In addition, the Wilcoxon's signed rank test is also conducted. Experimental results are showed in Table 6, which indicates that MSSCS is significantly different from all the other algorithms. In conclusion, the proposed multi-strategy serial framework and the three learning strategies is effective.

7.4. The worst is not the worst

There is a problem which is why each learning strategy uses the information of the worst solution found so far to generate new solutions. The reason is that the worst solution may not have a negative impact on the evolution of the algorithm, and sometimes it actually has a positive effect on the algorithm under certain circumstances. Next, in order to prove this viewpoint, we replace the worst solution in each learning strategy with the optimal solution and the random solution to form new strategies and compare their performance. At the same time, the original strategy of MSSCS is replaced by the corresponding new strategy to form a comparison between different MSSCS versions. In this subsection, we still experiment with these new algorithm versions on CEC2013 to verify our viewpoint. The parameter settings are consistent with the section 7.2.

Table 5: Comparison between MSSCS and four new versions of CS on CEC2013(D = 30)

Function	CS Mean	SL-CS Mean	GWL-CS Mean	SDL-CS Mean	EPSS-CS Mean	MSSCS Mean
f_1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_2	8.39E+05	1.35E+06	**3.73E+05**	1.32E+06	6.28E+05	5.84E+05
f_3	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**
f_4	1.88E+03	4.21E+03	**7.73E+01**	4.20E+03	2.82E+02	3.10E+02
f_5	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.02E-13	7.58E-15	**0.00E+00**
f_6	2.89E+00	1.99E+00	4.41E+00	2.29E+00	7.76E+00	**2.00E-02**
f_7	1.06E+02	7.03E+01	6.26E+01	7.46E+01	5.94E+01	**4.13E+01**
f_8	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**
f_9	2.96E+01	2.69E+01	**1.81E+01**	2.71E+01	2.07E+01	1.96E+01
f_{10}	1.22E-02	2.10E-02	2.66E-02	**9.92E-03**	1.37E-02	1.23E-02
f_{11}	4.58E+00	**0.00E+00**	8.16E+01	4.97E-01	1.33E-01	**0.00E+00**
f_{12}	1.50E+02	8.45E+01	1.25E+02	8.50E+01	1.16E+02	**7.66E+01**
f_{13}	1.82E+02	1.31E+02	1.97E+02	1.32E+02	1.51E+02	**1.27E+02**
f_{14}	7.63E+02	**9.55E-02**	8.76E+02	2.75E+01	1.54E+00	1.89E-01
f_{15}	4.25E+03	3.79E+03	**3.41E+03**	3.83E+03	3.54E+03	3.53E+03
f_{16}	1.39E+00	1.07E+00	1.30E+00	1.02E+00	**9.63E-01**	1.07E+00
f_{17}	6.49E+01	**3.05E+01**	1.01E+02	3.07E+01	3.06E+01	**3.05E+01**
f_{18}	2.00E+02	1.19E+02	1.77E+02	**1.02E+02**	1.38E+02	**1.02E+02**
f_{19}	7.38E+00	1.18E+00	7.21E+00	**5.36E-01**	9.82E-01	1.01E+00
f_{20}	1.22E+01	1.19E+01	1.25E+01	1.19E+01	**1.12E+01**	**1.12E+01**
f_{21}	2.37E+02	**2.11E+02**	3.38E+02	2.38E+02	2.93E+02	2.13E+02
f_{22}	1.08E+03	1.14E+01	8.57E+02	5.24E+01	2.63E+01	**8.94E+00**
f_{23}	5.17E+03	4.35E+03	**3.55E+03**	4.28E+03	4.09E+03	4.29E+03
f_{24}	2.84E+02	2.77E+02	2.62E+02	2.76E+02	**2.59E+02**	**2.59E+02**
f_{25}	2.99E+02	2.93E+02	2.81E+02	2.93E+02	2.76E+02	**2.70E+02**
f_{26}	**2.00E+02**	**2.00E+02**	2.74E+02	**2.00E+02**	2.70E+02	**2.00E+02**
f_{27}	1.06E+03	9.61E+02	8.34E+02	1.03E+03	8.42E+02	**7.96E+02**
f_{28}	3.00E+02	**2.88E+02**	7.05E+02	3.00E+02	3.74E+02	2.93E+02
Average ranking	4.73	3.39	4.09	3.48	3.25	**2.05**
$+/-/\approx$	20/0/8	14/2/12	16/3/9	16/1/11	11/1/16	

In addition, we conduct experiments on two different dimensions(D = 30 and D = 50). The comparative results of the Friedman test are presented in Table 7. In this Table, SL-CS/best and SL-CS/rand respectively represent the algorithm formed by replacing the worst individual of SL from SL-CS with the best individual and a random individual. In addition, MSSCS/SL/best and MSSCS/SL/rand respectively represent the algorithm that the worst individual of SL from MSSCS is replaced by the best individual and a random individual. The meaning of other names can be deduced by analogy.

From table 7, on the 30 dimensions, we can see that among the new versions of SL-CS and MSSCS, the version that uses the worst individual information has better average performance. This is more significant in the GWL-CS versions and the corresponding MSSCS versions. For the SDL-CS versions, although the SDL-CS with the worst individual is worse than the one with the rand individual, its performance is not the worst among all SDL-CS versions. It is worth noting that when this operation is

Table 6: Statistical results based on the Wilcoxon's signed rank test(Comparison of MSSCS with CS, SL-CS, GWL-CS, SDL-CS and EPSS-CS)

MSSCS vs.	R^+	R^-	p-value	a = 0.05
CS	368.5	9.5	0.000015	Yes
SL-CS	345	61	0.001175	Yes
GWL-CS	276.5	101.5	0.034498	Yes
SDL-CS	360.5	45.5	0.000321	Yes
EPSS-CS	333.5	72.5	0.002854	Yes

applied to the corresponding MSSCS, the average performance of the MSSCS version with the worst individual is the best. On the 50 dimensions, it has similar phenomenon observe from the results of Table 7.

Based on the above findings, we can know that using the information of the worst individual does not make the performance of the algorithm worse, on the contrary, it promotes the performance of the algorithm in some cases.

Table 7: The worst is not the worst

D=30

Algorithms	Average ranking	Algorithms	Average ranking
SL-CS	**1.82**	MSSCS	**1.84**
SL-CS/best	2.25	MSSCS/SL/best	2.27
SL-CS/rand	1.93	MSSCS/SL/rand	1.89
GWL-CS	**1.50**	MSSCS	**1.57**
GWL-CS/best	2.50	MSSCS/GWL/best	2.20
GWL-CS/rand	2.00	MSSCS/GWL/rand	2.23
SDL-CS	2.04	MSSCS	**1.86**
SDL-CS/best	2.21	MSSCS/SDL/best	2.11
SDL-CS/rand	**1.75**	MSSCS/SDL/rand	2.04

D=50

Algorithms	Average ranking	Algorithms	Average ranking
SL-CS	1.98	MSSCS	**1.84**
SL-CS/best	**1.95**	MSSCS/SL/best	2.11
SL-CS/rand	2.07	MSSCS/SL/rand	2.05
GWL-CS	**1.39**	MSSCS	**1.63**
GWL-CS/best	2.61	MSSCS/GWL/best	2.41
GWL-CS/rand	2.00	MSSCS/GWL/rand	1.96
SDL-CS	**1.93**	MSSCS	**1.82**
SDL-CS/best	2.14	MSSCS/SDL/best	2.36
SDL-CS/rand	**1.93**	MSSCS/SDL/rand	**1.82**

7.5. Comparison of MSSCS with other state-of-art algorithms

Artificial bee colony algorithm, brain storm optimization algorithm and differential evolution algorithm are extensively used algorithms in optimization researches. In this subsection, MSSCS is compared with them and their multi-strategy versions in order to further verify the performance of MSSCS. The concerned algorithms are enumerated as follows:

- Artificial bee colony algorithm (ABC) [3].
- Brain storm optimization algorithm (BSO) [75].
- Differential evolution algorithm (DE) [76].
- An improved ABC for balancing local and global search behaviors in continuous optimization (ABCX) [61].
- Multi-strategy ensemble ABC (MEABC) [77].
- DE with ensemble of parameters and mutation strategies (EPSDE) [78].

For all algorithms, the population size N is set to 25 and the problem dimension D is equal to 30. For ABC and ABCX, the parameter *limit* is set to 50 and $(N/2)\cdot D$ respectively. For BSO, the number of clusters is set to 5. For the parameter settings of DE and ESPDE, we follow the literature [79]. For MEABC, the parameter settings are described in [77]. The parameter settings of MSSCS are same as subsection 7.3. For the sake of fairness, 5.0E + 5 is decided as the maximum number of function evaluations of each algorithm. Besides, all the algorithms are executed independently for 30 times and the experimental results is presented in Table 8. The best result is shown in **bold**.

According to the results from Table 8, the performance of MSSCS for 7 functions is better than other algorithms. It is worth mentioning that only MSSCS can acquire global optimum on f_5 and f_{11}. For ABC and BSO, them achieve the best result on f_{27} and f_{16}, respectively. On f_2, f_7 and f_{26}, DE obtains higher quality solution than others. Similarly, ABCX is superior to other algorithms on f_{14} and f_{17}. In terms of obtained solution accuracy, MEABC performs best on f_8 and f_{19}. However, EPSDE can't gain the best result once.

Based on the statistical results of Wilcoxon's rank sum test are showed at the bottom of Table 8, we can find that MSSCS outperforms other algorithms in no less than 18 functions. Meanwhile, MSSCS also wins the best ranking according to the experimental results of Friedman test that are also presented at the bottom of Table 8. Besides, the Wilcoxon's signed rank test is also conducted and results are listed in Table 9. Results illustrate that MSSCS is significantly different from other algorithms except DE. The reason for the above situation may be that DE has the crossover operation but CS does not. However, based on the results of all the experiments above, we can still observe that the performance of MSSCS is better than that of DE. To sum up, compared with other algorithms, MSSCS provides better performance in dealing with CEC2013 benchmark functions with 30 dimensions.

7.6. Sensitivity test of parameters

In MSSCS, parameter L is a key parameter, which affect the exploration and exploitation of MSSCS. To analyze the influence of this parameter on MSSCS, MSSCS with different parameter values of L is performed on CEC2013 test suite. In this experiment, population size N, problem dimension D, switch parameter P_a and maximum number of iterations *MaxIt* are set to 25, 30, 0.25 and 20000, respectively. In addition, in order to reduce the statistical error, 28 benchmark functions of CEC2013 have been executed 30 times independently.

We test four different L parameter values($L \geq 1$), and the experimental results are summarized in Table 10. For clarity, the best experimental results are shown in **bold**. From this table, we can observe that all parameter values converge to the global optimal value in f_1, f_5 and f_{11}. Compared with other parameter values, $L = 1$ get better results on 5 functions, which are f_6, f_{10}, f_{12}, f_{18} and f_{25}. For $L = 2$, which performs best on f_2, f_{14}, f_{23}, f_{24} and f_{27} in terms of solution accuracy. On f_4 and f_{13}, $L = 3$ acquires higher quality solutions. Similarly, $L = 4$ gains the optimal result on f_7, f_{16}, f_{19}, f_{21}, f_{22} and f_{28}. Especially, all the values have the same effect on f_3, f_8 and f_{17}. Then, the Friedman test is used for determining the best value of L. The experimental results at the bottom of the Table 10 show that $L = 1$ acquires the best ranking. Consequently, L is set to 1 in other subsections.

Table 8: Comparison of MSSCS with ABC, BSO, DE, ABCX, MEABC and EPSDE on CEC2013(D = 30)

Function	Mean/Std	ABC	BSO	DE	ABCX	MEABC	EPSDE	MSSCS
f_1	Mean	9.25E-13	1.34E-02	1.89E-13	2.50E-13	5.08E-13	2.35E-13	**0.00E+00**
	Std	1.43E-13	1.24E-02	8.47E-14	6.82E-14	9.62E-14	9.25E-14	**0.00E+00**
f_2	Mean	1.48E+07	8.20E+05	**1.32E+05**	1.77E+07	8.39E+06	8.98E+05	5.84E+05
	Std	3.32E+06	2.36E+05	**6.32E+04**	3.72E+06	2.25E+06	3.25E+06	3.09E+05
f_3	Mean	1.06E+09	1.29E+08	**6.45E+05**	9.11E+08	4.10E+08	1.32E+07	1.00E+10
	Std	5.30E+08	2.27E+08	1.47E+06	2.66E+08	3.50E+08	2.84E+07	**0.00E+00**
f_4	Mean	6.19E+04	1.19E+03	3.30E+02	6.05E+04	9.95E+04	1.86E+03	**3.10E+02**
	Std	5.27E+03	1.08E+03	2.41E+02	6.44E+03	1.26E+04	4.55E+03	**1.93E+02**
f_5	Mean	2.94E-11	5.39E-02	2.05E-13	2.69E-13	6.14E-13	3.79E-13	**0.00E+00**
	Std	2.05E-11	1.38E-02	1.19E-13	5.48E-14	9.09E-14	1.48E-13	**0.00E+00**
f_6	Mean	1.53E+01	3.50E+01	7.18E+00	1.50E+01	1.34E+01	5.61E+00	**2.00E-02**
	Std	2.28E+00	2.56E+01	6.85E+00	4.63E+00	4.10E+00	5.85E+00	**1.39E-02**
f_7	Mean	1.06E+02	1.25E+02	**2.60E+00**	8.42E+01	1.09E+02	4.07E+01	4.13E+01
	Std	1.38E+01	5.04E+01	**3.02E+00**	8.83E+00	1.67E+01	1.88E+01	1.53E+01
f_8	Mean	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**
	Std	4.32E-02	1.06E-01	4.45E-02	5.17E-02	**3.47E-02**	5.05E-02	4.55E-02
f_9	Mean	2.98E+01	3.47E+01	3.89E+01	2.71E+01	3.07E+01	2.79E+01	**1.96E+01**
	Std	**1.20E+00**	2.84E+00	2.49E+00	1.95E+00	2.36E+00	2.20E+00	4.32E+00
f_{10}	Mean	2.46E+00	1.02E+00	2.31E-02	1.43E+00	1.76E-01	1.76E-01	**1.23E-02**
	Std	5.83E-01	1.73E-01	1.51E-02	2.20E-01	6.91E-02	1.10E-01	**1.10E-02**
f_{11}	Mean	4.12E-12	5.90E+02	2.91E+01	6.25E-14	1.66E-01	2.98E-01	**0.00E+00**
	Std	6.09E-12	1.07E+02	1.20E+01	1.71E-14	3.71E-01	8.18E-01	**0.00E+00**
f_{12}	Mean	2.42E+02	6.13E+02	**4.22E+01**	2.25E+02	1.96E+02	4.52E+01	7.66E+01
	Std	2.12E+01	9.70E+01	2.79E+01	1.55E+01	4.02E+01	**1.34E+01**	1.82E+01
f_{13}	Mean	3.05E+02	6.09E+02	1.31E+02	1.94E+02	2.50E+02	**8.77E+01**	1.27E+02
	Std	2.40E+01	8.28E+01	5.59E+01	**1.02E+01**	3.04E+01	2.70E+01	2.46E+01
f_{14}	Mean	3.56E+01	4.44E+03	1.34E+03	**4.17E-02**	4.05E+00	8.66E-01	1.89E-01
	Std	1.34E+01	6.66E+02	4.93E+02	**3.09E-02**	4.35E+00	1.53E+00	2.90E-01
f_{15}	Mean	4.65E+03	4.60E+03	7.22E+03	4.34E+03	3.85E+03	5.07E+03	**3.53E+03**
	Std	3.78E+02	7.13E+02	3.21E+02	**3.00E+02**	3.83E+02	1.29E+03	3.97E+02
f_{16}	Mean	1.79E+00	**3.40E-01**	2.42E+00	1.61E+00	9.09E-01	2.17E+00	1.07E+00
	Std	2.01E-01	**1.09E-01**	2.75E-01	2.60E-01	1.83E-01	3.21E-01	1.77E-01
f_{17}	Mean	3.17E+01	5.26E+02	7.98E+01	**3.04E+01**	3.05E+01	3.05E+01	3.05E+01
	Std	4.88E-01	1.03E+02	1.36E+01	**5.91E-03**	9.12E-02	1.89E-01	9.34E-02
f_{18}	Mean	3.77E+02	4.89E+02	2.16E+02	1.58E+02	2.12E+02	1.04E+02	**1.02E+02**
	Std	2.27E+01	7.65E+01	2.01E+01	**1.53E+01**	2.75E+01	1.89E+01	1.57E+01
f_{19}	Mean	2.08E+00	1.12E+01	3.28E+00	3.35E-01	**2.52E-01**	1.24E+00	1.01E+00
	Std	5.14E-01	2.46E+00	9.32E-01	1.98E-01	**1.04E-01**	2.74E-01	1.87E-01
f_{20}	Mean	1.44E+01	1.45E+01	1.21E+01	1.32E+01	1.47E+01	1.17E+01	**1.12E+01**
	Std	**2.31E-01**	8.82E-02	6.72E-01	4.01E-01	2.44E-01	7.56E-01	5.67E-01
f_{21}	Mean	2.57E+02	3.70E+02	3.27E+02	2.01E+02	**1.96E+02**	2.92E+02	2.13E+02
	Std	3.69E+01	1.01E+02	6.90E+01	4.39E+01	4.16E+01	7.32E+01	**3.40E+01**
f_{22}	Mean	1.58E+02	5.47E+03	1.34E+03	5.70E+01	2.25E+01	8.22E+01	**8.94E+00**
	Std	1.89E+01	1.01E+03	3.48E+02	4.81E+01	3.24E+01	4.42E+01	**2.79E+00**
f_{23}	Mean	5.37E+03	5.79E+03	7.72E+03	5.02E+03	4.86E+03	5.22E+03	**4.29E+03**
	Std	**3.22E+02**	7.18E+02	3.28E+02	3.50E+02	5.57E+02	1.07E+03	5.12E+02
f_{24}	Mean	2.83E+02	3.37E+02	**2.13E+02**	2.63E+02	2.90E+02	2.32E+02	2.59E+02
	Std	**5.65E+00**	2.51E+01	6.09E+00	7.40E+00	6.42E+00	1.75E+01	1.06E+01
f_{25}	Mean	3.12E+02	3.68E+02	**2.56E+02**	2.93E+02	3.10E+02	2.89E+02	2.70E+02
	Std	**3.70E+00**	2.16E+01	6.98E+00	4.33E+00	7.37E+00	4.89E+00	9.92E+00
f_{26}	Mean	2.01E+02	2.21E+02	**2.00E+02**	2.01E+02	**2.00E+02**	2.15E+02	**2.00E+02**
	Std	2.08E-01	5.18E+01	**4.00E-03**	1.37E-01	1.35E-01	4.29E+01	9.47E-03
f_{27}	Mean	**4.01E+02**	1.34E+03	5.78E+02	**4.01E+02**	8.76E+02	8.67E+02	7.96E+02
	Std	**6.31E-01**	8.40E+01	2.50E+02	1.29E+00	3.65E+02	1.46E+02	2.07E+02
f_{28}	Mean	**2.45E+02**	4.81E+03	3.00E+02	2.81E+02	2.49E+02	3.00E+02	2.93E+02
	Std	5.05E+01	6.42E+02	**2.64E-13**	5.68E+01	8.45E+01	3.58E-13	3.59E+01
Average ranking		4.82	5.93	3.91	3.46	4.00	3.55	**2.32**
+/−/≈		24/3/1	25/2/1	18/3/7	19/7/2	19/5/4	18/4/6	

Table 9: Statistical results based on the Wilcoxon's signed rank test(Comparison of MSSCS with ABC, BSO, DE, ABCX, MEABC and EPSDE)

MSSCS vs.	R^{+}	R^{-}	p-value	$\alpha = 0.05$
ABC	338	68	0.002032	Yes
BSO	374	32	0.000094	Yes
DE	262	144	0.17545	No
ABCX	306	100	0.018431	Yes
MEABC	330	76	0.003692	Yes
EPSDE	309	97	0.015302	Yes

Table 10: Sensitivity test of parameter L

function	$L = 1$ Mean	$L = 2$ Mean	$L = 3$ Mean	$L = 4$ Mean
f_1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_2	5.84E+05	**5.51E+05**	5.65E+05	6.74E+05
f_3	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**
f_4	3.10E+02	3.88E+02	**2.65E+02**	2.85E+02
f_5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_6	**2.00E-02**	9.00E-01	8.97E-01	2.14E-02
f_7	4.13E+01	4.34E+01	3.94E+01	**3.12E+01**
f_8	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**
$f9$	1.96E+01	**1.95E+01**	2.08E+01	**1.95E+01**
f_{10}	**1.23E-02**	1.32E-02	1.70E-02	1.80E-02
f_{11}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{12}	**7.66E+01**	8.10E+01	8.11E+01	8.42E+01
f_{13}	**1.27E+02**	1.28E+02	**1.27E+02**	1.29E+02
f_{14}	1.89E-01	**1.68E-01**	1.90E-01	2.02E-01
f_{15}	**3.53E+03**	**3.53E+03**	3.66E+03	3.60E+03
f_{16}	1.07E+00	1.04E+00	1.03E+00	**9.85E-01**
f_{17}	**3.05E+01**	**3.05E+01**	**3.05E+01**	**3.05E+01**
f_{18}	**1.02E+02**	1.09E+02	1.03E+02	1.09E+02
f_{19}	1.01E+00	1.01E+00	1.02E+00	**9.81E-01**
f_{20}	1.12E+01	**1.11E+01**	1.12E+01	**1.11E+01**
f_{21}	2.13E+02	2.13E+02	2.07E+02	**2.00E+02**
f_{22}	8.94E+00	8.80E+00	9.33E+00	**7.94E+00**
f_{23}	4.29E+03	**4.13E+03**	4.24E+03	4.33E+03
f_{24}	2.59E+02	**2.55E+02**	2.58E+02	2.60E+02
f_{25}	**2.70E+02**	2.74E+02	2.74E+02	2.73E+02
f_{26}	**2.00E+02**	2.05E+02	**2.00E+02**	**2.00E+02**
f_{27}	7.96E+02	**7.95E+02**	8.39E+02	8.40E+02
f_{28}	2.93E+02	3.00E+02	3.00E+02	**2.80E+02**
Average ranking	**2.34**	2.57	2.59	2.50

7.7. *Effect of dimension growth*

Based on the comparative analysis between the algorithms in the above subsection, it can be seen that MSSCS outperforms others in tackling the benchmark function of 30 dimensional CEC2013. But for an excellent algorithm, it should also be able to generate high-quality solutions on high-dimensional problems. To investigate the impact of dimension growth on the performance of MSSCS, the scalability research of algorithm is implemented on 28 test functions from CEC2013, that is, the problem dimension size was expanded from 30 to 50 and 100. In this subsection, VCS, ICS, ECS, GBCS and SDCS are still selected for comparative experiments, and the experimental results are display in Table 11 and Table 12.

From Table 11, we can see that MSSCS is the winner on 6 functions, which are f_{11}, f_{14}, f_{17}, f_{18}, f_{19} and f_{22}. Similarly, ICS is the champion on f_{12}, f_{13} and f_{26}. On f_1, all CS variants can find the global optimal solution except GBCS. According to the quality of the solutions, all algorithms gain the same result on f_3. Besides, ECS yields higher quality solutions on f_2 and f_{16}. Unfortunately, VCS, GBCS and SDCS can't acquire the champion once. According to Table 12, MSSCS still gains the best results in the most of functions when the dimension rises from 50 to 100.

From the Wilcoxon's rank sum test results at the bottom of Table 11 and Table 12, it can be seen that the performance of MSSCS is better than other competitors. Besides, in terms of the results of the Friedman test, MSSCS also acquires the minimum value of average ranking. According to the results of Wilcoxon's signed rank test from Table 13, we can find that MSSCS is significantly superior to VCS, ICS, ECS, GBCS and SDCS. For Table 14, MSSCS is significantly better than VCS, ECS and SDCS. From Figure 6 and Figure 7, it can be observed that MSSCS still holds the most yellow-green color blocks. Based on all the above experimental analysis, we can know that although the advantages of MSSCS decrease slightly when the dimensionality of the problem is increased to 50 and 100 dimensions, the MSSCS is still the best algorithm for dealing with these benchmark functions by combining the effects of MSSCS on all experiments.

From Figure 8, we can see that the convergence rate of MSSCS is obviously faster than other competitors on 6 functions, which are f_9, f_{11}, f_{14}, f_{17}, f_{22} and f_{27}. On f_7, f_{19} and f_{24}, although MSSCS is not significantly faster than other competitors, it is still the fastest among all algorithms. In terms of Figure 9, MSSCS still maintains a fast convergence rate on most functions when the dimension is expanded to 100. All in all, the proposed MSSCS has better performance compared with other competitors.

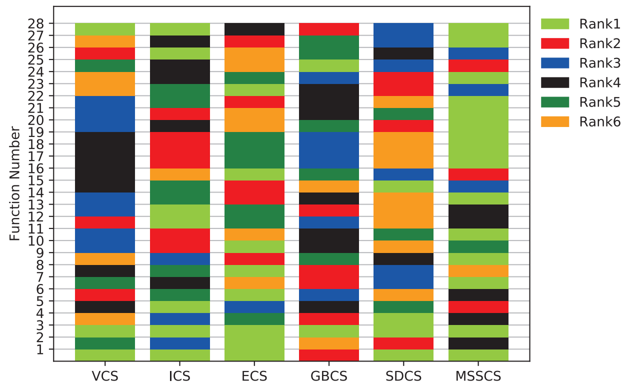


Figure 6: Stacked histogram of ranking for VCS, ICS, ECS, GBCS, SDCS and MSSCS on CEC2013(D = 50)

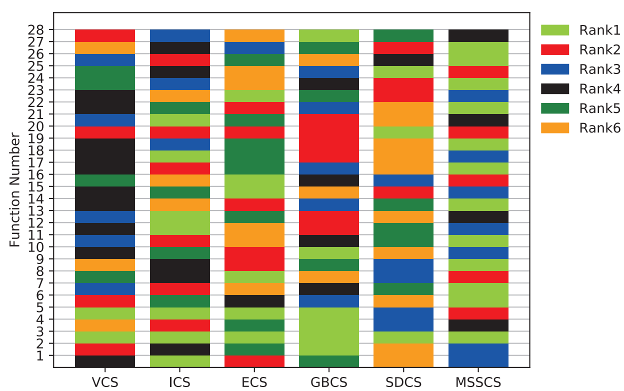
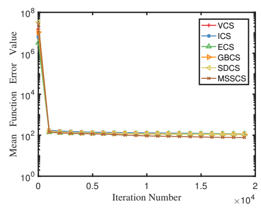
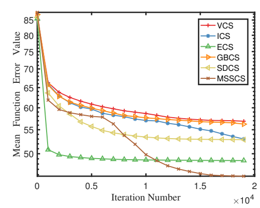


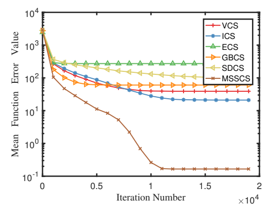
Figure 7: Stacked histogram of ranking for VCS, ICS, ECS, GBCS, SDCS and MSSCS on CEC2013(D = 100)



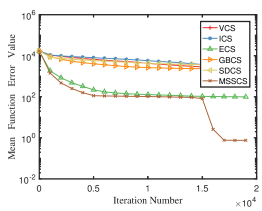
(a) f_7



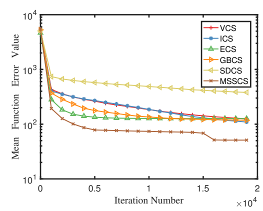
(b) f_9



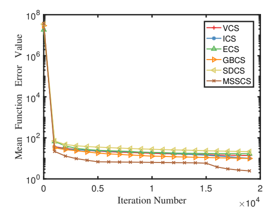
(c) f_{11}



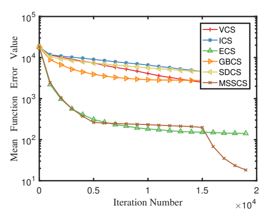
(d) f_{14}



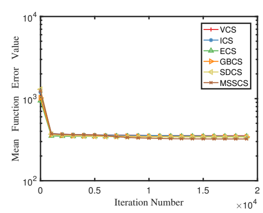
(e) f_{17}



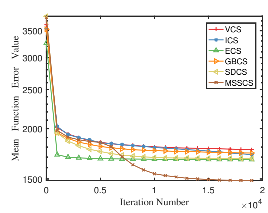
(f) f_{19}



(g) f_{22}

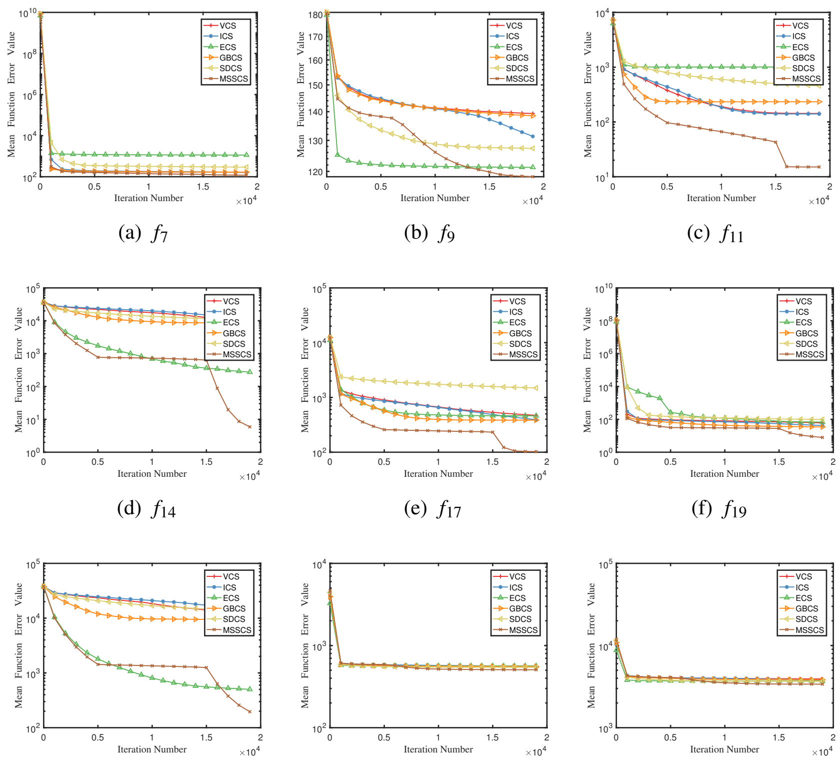


(h) f_{24}



(i) f_{27}

Figure 8: Convergence of VCS, ICS, ECS, GBCS, SDCS and MSSCS(D = 50)



(a) f_7 (b) f_9 (c) f_{11}

(d) f_{14} (e) f_{17} (f) f_{19}

(g) f_{22} (h) f_{24} (i) f_{27}

Figure 9: Convergence of VCS, ICS, ECS, GBCS, SDCS and MSSCS(D = 100)

Table 11: Comparison of MSSCS with VCS, ICS, ECS, GBCS and SDCS on CEC2013(D = 50)

Function	Mean/Std	VCS	ICS	ECS	GBCS	SDCS	MSSCS
f_1	Mean	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.58E-15	**0.00E+00**	**0.00E+00**
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.08E-14	**0.00E+00**	**0.00E+00**
f_2	Mean	5.86E+06	1.31E+06	**5.01E+05**	6.69E+06	5.35E+05	1.56E+06
	Std	1.43E+06	4.06E+05	**1.45E+05**	2.10E+06	2.32E+05	6.56E+05
f_3	Mean	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_4	Mean	5.02E+03	3.05E+03	4.57E+03	1.80E+03	**1.07E+03**	3.16E+03
	Std	1.25E+03	7.77E+02	2.38E+03	**5.33E+02**	6.63E+02	1.70E+03
f_5	Mean	1.10E-13	**4.17E-14**	9.47E-14	1.10E-13	1.02E-07	8.72E-14
	Std	**2.04E-14**	5.48E-14	4.24E-14	**2.04E-14**	1.00E-07	4.81E-14
f_6	Mean	4.28E+01	4.36E+01	**3.19E+01**	4.36E+01	4.85E+01	4.36E+01
	Std	4.46E+00	**1.02E+00**	2.56E+01	**1.02E+00**	1.08E+01	**1.02E+00**
f_7	Mean	1.11E+02	1.11E+02	1.16E+02	1.08E+02	1.09E+02	**7.64E+01**
	Std	1.38E+01	**1.17E+01**	2.18E+01	2.05E+01	1.30E+01	1.51E+01
f_8	Mean	**2.11E+01**	**2.11E+01**	**2.11E+01**	**2.11E+01**	**2.11E+01**	**2.11E+01**
	Std	3.47E-02	**3.20E-02**	**3.20E-02**	3.38E-02	**3.20E-02**	3.24E-02
f_9	Mean	5.67E+01	5.25E+01	4.87E+01	5.61E+01	5.29E+01	**4.57E+01**
	Std	2.73E+00	**1.76E+00**	4.10E+00	2.13E+00	3.49E+00	6.52E+00
f_{10}	Mean	5.71E-02	5.43E-02	**5.17E-02**	6.03E-02	6.72E-02	6.71E-02
	Std	3.43E-02	3.08E-02	4.19E-02	**2.46E-02**	3.59E-02	4.10E-02
f_{11}	Mean	3.91E+01	2.10E+01	2.73E+02	6.04E+01	9.27E+01	**1.66E-01**
	Std	9.60E+00	4.49E+00	1.01E+02	1.46E+01	2.21E+01	**4.51E-01**
f_{12}	Mean	2.02E+02	**1.80E+02**	3.66E+02	2.16E+02	3.79E+02	2.25E+02
	Std	2.68E+01	**2.42E+01**	6.83E+01	2.54E+01	6.82E+01	3.57E+01
f_{13}	Mean	2.97E+02	**2.52E+02**	5.33E+02	2.91E+02	5.43E+02	3.23E+02
	Std	2.98E+01	**2.77E+01**	8.67E+01	3.10E+01	7.08E+01	4.56E+01
f_{14}	Mean	1.98E+03	2.43E+03	9.84E+01	2.25E+03	3.30E+03	**7.56E-01**
	Std	7.74E+02	6.78E+02	1.51E+02	5.11E+02	6.65E+02	**7.80E-01**
f_{15}	Mean	9.61E+03	9.75E+03	**6.98E+03**	1.00E+04	6.95E+03	7.59E+03
	Std	5.04E+02	**4.24E+02**	4.84E+02	5.30E+02	5.14E+02	6.25E+02
f_{16}	Mean	2.29E+00	2.47E+00	**1.35E+00**	2.33E+00	2.22E+00	1.68E+00
	Std	2.90E-01	2.89E-01	**2.26E-01**	3.42E-01	3.71E-01	2.39E-01
f_{17}	Mean	1.20E+02	1.06E+02	1.27E+02	1.17E+02	3.73E+02	**5.11E+01**
	Std	1.97E+01	1.16E+01	5.08E+01	2.31E+01	8.27E+01	**3.23E-01**
f_{18}	Mean	3.19E+02	2.92E+02	5.22E+02	3.16E+02	5.59E+02	**2.60E+02**
	Std	2.96E+01	3.12E+01	9.15E+01	2.44E+01	1.05E+02	**2.34E+01**
f_{19}	Mean	1.34E+01	9.30E+00	1.67E+01	9.60E+00	2.08E+01	**2.27E+00**
	Std	4.24E+00	2.24E+00	2.60E+01	3.89E+00	5.23E+00	**2.39E-01**
f_{20}	Mean	2.16E+01	2.16E+01	2.21E+01	2.20E+01	2.11E+01	**2.04E+01**
	Std	**3.97E-01**	6.34E-01	1.66E+00	7.29E-01	7.57E-01	8.03E-01
f_{21}	Mean	6.65E+02	3.77E+02	1.01E+03	6.84E+02	7.45E+02	**2.73E+02**
	Std	3.99E+02	3.28E+02	**1.98E+02**	4.33E+02	3.20E+02	2.24E+02
f_{22}	Mean	1.97E+03	2.94E+03	1.41E+02	2.74E+03	4.02E+03	**1.56E+01**
	Std	6.55E+02	7.85E+02	1.90E+02	6.64E+02	7.68E+02	**2.89E+00**
f_{23}	Mean	1.07E+04	1.06E+04	**8.45E+03**	1.04E+04	8.49E+03	9.07E+03
	Std	5.49E+02	6.48E+02	8.26E+02	**5.33E+02**	7.56E+02	7.83E+02
f_{24}	Mean	3.53E+02	3.46E+02	3.46E+02	3.41E+02	3.41E+02	**3.22E+02**
	Std	6.28E+00	**5.61E+00**	1.27E+01	1.24E+01	9.99E+00	1.63E+01
f_{25}	Mean	3.73E+02	3.67E+02	3.80E+02	**3.56E+02**	3.64E+02	3.60E+02
	Std	8.31E+00	**6.79E+00**	1.34E+01	1.45E+01	1.19E+01	1.39E+01
f_{26}	Mean	2.01E+02	**2.00E+02**	4.11E+02	3.38E+02	2.31E+02	2.07E+02
	Std	1.54E-01	**5.40E-02**	5.70E+01	1.20E+02	7.95E+01	3.84E+01
f_{27}	Mean	1.77E+03	1.71E+03	1.68E+03	1.74E+03	1.69E+03	**1.49E+03**
	Std	5.65E+01	**4.61E+01**	1.20E+02	6.78E+01	8.80E+01	1.48E+02
f_{28}	Mean	**4.00E+02**	**4.00E+02**	1.88E+03	7.10E+02	1.08E+03	**4.00E+02**
	Std	2.34E-13	**1.68E-13**	2.13E+03	9.30E+02	1.35E+03	1.92E-13
Average ranking		3.88	3.30	3.70	3.66	4.02	**2.45**
$+/-/\approx$		19/4/5	15/3/10	17/5/6	19/2/7	19/4/5	

Table 12: Comparison of MSSCS with VCS, ICS, ECS, GBCS and SDCS on CEC2013(D = 100)

Function	Mean/Std	VCS	ICS	ECS	GBCS	SDCS	MSSCS
f_1	Mean	2.05E-13	**0.00E+00**	1.06E-13	2.20E-13	1.54E-05	1.14E-13
	Std	6.82E-14	**0.00E+00**	1.13E-13	4.08E-14	1.34E-05	1.14E-13
f_2	Mean	3.68E+09	5.35E+09	6.33E+09	**3.34E+09**	9.00E+09	5.02E+09
	Std	4.81E+09	4.97E+09	4.82E+09	4.71E+09	**3.00E+09**	4.98E+09
f_3	Mean	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**	**1.00E+10**
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_4	Mean	2.97E+04	1.84E+04	2.85E+04	**1.63E+04**	1.91E+04	2.65E+04
	Std	5.25E+03	3.53E+03	6.07E+03	**3.29E+03**	4.41E+03	6.18E+03
f_5	Mean	**1.14E-13**	**1.14E-13**	**1.14E-13**	**1.14E-13**	3.89E-02	4.89E-13
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.77E-02	1.86E-13
f_6	Mean	1.68E+02	1.91E+02	1.78E+02	1.73E+02	2.65E+02	**1.43E+02**
	Std	4.39E+01	**3.43E+01**	4.15E+01	4.74E+01	5.65E+01	5.99E+01
f_7	Mean	1.64E+02	1.61E+02	1.13E+03	1.66E+02	2.92E+02	**1.17E+02**
	Std	2.18E+01	1.78E+01	1.08E+03	**1.77E+01**	1.63E+02	1.85E+01
f_8	Mean	2.13E+01	2.13E+01	**2.12E+01**	2.13E+01	2.13E+01	**2.12E+01**
	Std	3.29E-02	3.77E-02	5.22E-02	**3.04E-02**	3.39E-02	3.74E-02
f_9	Mean	1.39E+02	1.30E+02	1.21E+02	1.38E+02	1.27E+02	**1.18E+02**
	Std	**2.74E+00**	3.80E+00	6.83E+00	2.97E+00	4.71E+00	1.10E+01
f_{10}	Mean	1.18E-01	1.34E-01	**1.03E-01**	**1.03E-01**	9.92E+00	1.10E-01
	Std	7.65E-02	7.29E-02	6.19E-02	5.33E-02	3.90E+00	**4.44E-02**
f_{11}	Mean	1.43E+02	**1.40E+02**	1.01E+03	2.34E+02	4.50E+02	1.51E+01
	Std	2.91E+01	1.99E+01	1.85E+02	3.76E+01	5.65E+01	**5.48E+00**
f_{12}	Mean	7.51E+02	**5.56E+02**	1.17E+03	7.48E+02	1.12E+03	7.49E+02
	Std	6.07E+01	5.87E+01	1.84E+02	**5.83E+01**	1.36E+02	5.91E+01
f_{13}	Mean	9.84E+02	**7.33E+02**	1.44E+03	9.16E+02	1.68E+03	1.06E+03
	Std	7.30E+01	5.27E+01	1.45E+02	7.91E+01	1.84E+02	**1.07E+02**
f_{14}	Mean	8.93E+03	1.07E+04	2.52E+02	8.46E+03	1.05E+04	**5.31E+00**
	Std	3.26E+03	2.72E+03	2.67E+02	1.65E+03	1.00E+03	**2.48E+00**
f_{15}	Mean	2.38E+04	2.43E+04	**1.47E+04**	2.49E+04	1.49E+04	1.72E+04
	Std	6.78E+02	**5.85E+02**	1.09E+03	6.93E+02	1.16E+03	1.24E+03
f_{16}	Mean	3.28E+00	3.29E+00	**1.98E+00**	3.25E+00	3.12E+00	2.54E+00
	Std	**2.28E-01**	2.68E-01	2.53E-01	2.80E-01	5.29E-01	2.75E-01
f_{17}	Mean	4.59E+02	3.76E+02	4.61E+02	3.85E+02	1.47E+03	**1.03E+02**
	Std	8.37E+01	5.47E+01	1.34E+02	5.13E+01	1.82E+02	**5.99E-01**
f_{18}	Mean	1.06E+03	**9.29E+02**	1.59E+03	9.62E+02	2.08E+03	1.00E+03
	Std	8.60E+01	5.98E+01	1.44E+02	**4.98E+01**	2.44E+02	9.19E+01
f_{19}	Mean	5.95E+01	3.92E+01	6.74E+01	3.41E+01	9.57E+01	**7.16E+00**
	Std	1.59E+01	8.02E+00	7.66E+01	1.43E+01	2.02E+01	**6.59E-01**
f_{20}	Mean	**5.00E+01**	**5.00E+01**	**5.00E+01**	**5.00E+01**	5.00E+01	**5.00E+01**
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.98E-07	**0.00E+00**
f_{21}	Mean	3.63E+02	**3.23E+02**	3.97E+02	3.53E+02	4.27E+02	3.97E+02
	Std	4.82E+01	4.23E+01	1.80E+01	4.99E+01	**8.79E-03**	1.80E+01
f_{22}	Mean	9.99E+03	1.26E+04	4.92E+02	9.47E+03	1.32E+04	**1.60E+02**
	Std	3.60E+03	2.29E+03	4.99E+02	1.34E+03	1.35E+03	**4.30E+01**
f_{23}	Mean	2.65E+04	2.66E+04	**1.87E+04**	2.66E+04	1.95E+04	2.10E+04
	Std	**6.93E+02**	1.20E+03	1.81E+03	9.20E+02	1.86E+03	1.80E+03
f_{24}	Mean	5.63E+02	5.53E+02	5.63E+02	5.53E+02	5.44E+02	**5.07E+02**
	Std	1.39E+01	**1.03E+01**	2.67E+01	1.53E+01	1.61E+01	3.89E+01
f_{25}	Mean	6.15E+02	5.99E+02	6.30E+02	5.96E+02	**5.82E+02**	5.83E+02
	Std	**8.91E+00**	1.13E+01	2.80E+01	1.66E+01	1.49E+01	3.18E+01
f_{26}	Mean	4.99E+02	4.58E+02	6.02E+02	6.48E+02	5.93E+02	**2.80E+02**
	Std	1.98E+02	2.09E+02	7.62E+01	**9.88E+00**	1.06E+02	1.59E+02
f_{27}	Mean	3.92E+03	3.74E+03	3.70E+03	3.82E+03	3.66E+03	**3.39E+03**
	Std	9.37E+01	9.70E+01	1.73E+02	**9.32E+01**	1.34E+02	2.45E+02
f_{28}	Mean	3.80E+03	3.92E+03	1.21E+04	**3.62E+03**	1.01E+04	4.18E+03
	Std	1.47E+03	1.12E+03	1.31E+03	**1.11E+03**	2.96E+03	1.74E+03
Average ranking		3.86	3.36	3.84	3.25	4.39	**2.30**
$+/-/\approx$		17/4/7	15/8/5	15/4/9	16/5/7	21/3/4	

Table 13: Statistical results based on the Wilcoxon's signed rank test(D = 50)

MSSCS vs.	R^+	R^-	p-value	$\alpha = 0.05$
VCS	324.5	53.5	0.001085	Yes
ICS	267.5	110.5	0.057699	No
ECS	305.5	100.5	0.019004	Yes
GBCS	312	66	0.003006	Yes
SDCS	301.5	104.5	0.024173	Yes

Table 14: Statistical results based on the Wilcoxon's signed rank test(D = 100)

MSSCS vs.	R^+	R^-	p-value	$\alpha = 0.05$
VCS	322.5	83.5	0.006284	Yes
ICS	285.5	120.5	0.058754	No
ECS	325.5	80.5	0.005096	Yes
GBCS	266.5	139.5	0.145014	No
SDCS	304	74	0.005522	Yes

8. Conclusion

This paper proposed MSSCS, a multi-strategy serial CS composed of three learning strategies and a multi-strategy serial framework. In MSSCS, inspired by the cuckoo's behavior of seeking host nest, eviction and begging during the growth of cuckoos, three new learning strategies are proposed to enhance the performance of CS. Then, based on these three serial behaviors, a multi-strategy serial framework is presented to reduce the complexity of the multi-strategy algorithm and maximize the performance of each learning strategy. Besides, the serial framework is dynamically regulated by designing the adaptive parameter.

To verify the performance of MSSCS, the experimental studies in this paper are conducted on 28 well-known test functions of CEC2013. MSSCS is compared with CS, 9 CS variants and several other state-of-art algorithms, and the experimental results show that MSSCS has superior performance. Besides, MSSCS is compared with the four new versions of CS, and the results illustration the effectiveness of the three learning strategies and the multi-strategy serial framework.

For regard to future work, the real-world applications of MSSCS will be focused.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (61763019), the Science and Technology Foundation of Jiangxi Province (20202BABL202019), and the Science and Technology Plan Projects of Jiangxi Provincial Education Department (GJJ180891).

References

[1] J. H. Holland, Genetic algorithms and the optimal allocation of trials, SIAM Journal on Computing 2 (2) (1973) 88–105.

[2] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, Vol. 4, IEEE, 1995, pp. 1942–1948.

[3] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm, Journal of Global Optimization 39 (3) (2007) 459–471.

[4] X.-S. Yang, S. Deb, Cuckoo search via lévy flights, in: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), IEEE, 2009, pp. 210–214.

[5] D. A. Wood, Hybrid cuckoo search optimization algorithms applied to complex wellbore trajectories aided by dynamic, chaos-enhanced, fat-tailed distribution sampling and metaheuristic profiling, Journal of Natural Gas Science and Engineering 34 (2016) 236–252.

[6] R. B. Payne, M. D. Sorensen, The cuckoos, Vol. 15, Oxford University Press, 2005, pp. 137–153.

[7] C. T. Brown, L. S. Liebovitch, R. Glendon, Lévy flights in dobe ju/'hoansi foraging patterns, Human Ecology 35 (1) (2007) 129–138.

[8] X.-S. Yang, S. Deb, Cuckoo search: recent advances and applications, Neural Computing and Applications 24 (1) (2014) 169–174.

[9] P. Civicioglu, E. Besdok, A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms, Artificial Intelligence Review 39 (4) (2013) 315–346.

[10] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.

[11] A. H. Gandomi, X.-S. Yang, A. H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Engineering with computers 29 (1) (2013) 17–35.

[12] S. Walton, O. Hassan, K. Morgan, M. Brown, Modified cuckoo search: a new gradient free optimisation algorithm, Chaos, Solitons & Fractals 44 (9) (2011) 710–718.

[13] R. N. Mantegna, Fast, accurate algorithm for numerical simulation of levy stable stochastic processes, Physical Review E 49 (5) (1994) 4677.

[14] E. Valian, S. Mohanna, S. Tavakoli, Improved cuckoo search algorithm for global optimization, International Journal of Communications and Information Technology 1 (1) (2011) 31–44.

[15] P. Ong, Adaptive cuckoo search algorithm for unconstrained optimization, The Scientific World Journal 2014 (2014) 1–8.

[16] X. Li, M. Yin, Modified cuckoo search algorithm with self adaptive parameter method, Information Sciences 298 (2015) 80–97.

[17] M. Guerrero, O. Castillo, M. García, Fuzzy dynamic parameters adaptation in the cuckoo search algorithm using fuzzy logic, in: 2015 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2015, pp. 441–448.

[18] M. Mareli, B. Twala, An adaptive cuckoo search algorithm for optimisation, Applied computing and informatics 14 (2) (2018) 107–115.

[19] U. Mlakar, I. Fister Jr, I. Fister, Hybrid self-adaptive cuckoo search for global optimization, Swarm and Evolutionary Computation 29 (2016) 47–72.

[20] H. Peng, C. Deng, H. Wang, W. Wang, X. Zhou, Z. Wu, Gaussian bare-bones cuckoo search algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2018, pp. 93–94.

[21] L. Huang, S. Ding, S. Yu, J. Wang, K. Lu, Chaos-enhanced cuckoo search optimization algorithms for global optimization, Applied Mathematical Modelling 40 (5-6) (2016) 3860–3875.

[22] Z. He, H. Peng, C. Deng, Y. Tan, Z. Wu, S. Wu, A spark-based gaussian bare-bones cuckoo search with dynamic parameter selection, in: 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 1220–1227.

[23] B. H. Abed-Alguni, F. Alkhateeb, Novel selection schemes for cuckoo search, Arabian Journal for Science and Engineering 42 (8) (2017) 3635–3654.

[24] N. J. Cheung, X.-M. Ding, H.-B. Shen, A nonhomogeneous cuckoo search algorithm based on quantum mechanism for real parameter optimization, IEEE Transactions on Cybernetics 47 (2) (2016) 391–402.

[25] X. Liu, M. Fu, Cuckoo search algorithm based on frog leaping local search and chaos theory, Applied Mathematics and Computation 266 (2015) 1083–1092.

[26] X. Li, J. Wang, M. Yin, Enhancing the performance of cuckoo search algorithm using orthogonal learning method, Neural Computing and Applications 24 (6) (2014) 1233–1247.

[27] M. Eusuff, K. Lansey, F. Pasha, Shuffled frog-leaping algorithm: a memetic metaheuristic for discrete optimization, Engineering optimization 38 (2) (2006) 129–154.

[28] W.-f. Gao, S.-y. Liu, L.-l. Huang, A novel artificial bee colony algorithm based on modified search equation and orthogonal learning, IEEE Transactions on Cybernetics 43 (3) (2013) 1011–1024.

[29] Z.-H. Zhan, J. Zhang, Y. Li, Y.-H. Shi, Orthogonal learning particle swarm optimization, IEEE transactions on evolutionary computation 15 (6) (2010) 832–847.

[30] M. Shehab, A. T. Khader, M. Laouchedi, O. A. Alomari, Hybridizing cuckoo search algorithm with bat algorithm for global numerical optimization, The Journal of Supercomputing 75 (5) (2019) 2395–2422.

[31] J. Ding, Q. Wang, Q. Zhang, Q. Ye, Y. Ma, A hybrid particle swarm optimization-cuckoo search algorithm and its engineering applications, Mathematical Problems in Engineering 2019 (2019) 1–12.

[32] B. H. Abed-alguni, F. Alkhateeb, Intelligent hybrid cuckoo search and β-hill climbing algorithm, Journal of King Saud University-Computer and Information Sciences 32 (2) (2020) 159–173.

[33] Z. Zhang, S. Ding, W. Jia, A hybrid optimization algorithm based on cuckoo search and differential evolution for solving constrained engineering problems, Engineering Applications of Artificial Intelligence 85 (2019) 254–268.

[34] W. Long, S. Cai, J. Jiao, M. Xu, T. Wu, A new hybrid algorithm based on grey wolf optimizer and cuckoo search for parameter extraction of solar photovoltaic models, Energy Conversion and Management 203 (2020) 112243.

[35] S. I. H. Shah, S. Alam, S. A. Ghauri, A. Hussain, F. A. Ansari, A novel hybrid cuckoo search-extreme learning machine approach for modulation classification, IEEE Access 7 (2019) 90525–90537.

[36] J. García, V. Yepes, J. V. Martí, A hybrid k-means cuckoo search algorithm applied to the counterfort retaining walls problem, Mathematics 8 (4) (2020) 555.

[37] S. Agrawal, L. Samantaray, R. Panda, L. Dora, A new hybrid adaptive cuckoo search-squirrel search algorithm for brain mr image analysis, in: Hybrid Machine Intelligence for Medical Image Analysis, Springer, 2020, pp. 85–117.

[38] Z. Cui, M. Zhang, H. Wang, X. Cai, W. Zhang, A hybrid many-objective cuckoo search algorithm, soft computing 23 (21) (2019) 10681–10697.

[39] X. Cai, Y. Niu, S. Geng, J. Zhang, Z. Cui, J. Li, J. Chen, An under-sampled software defect prediction method based on hybrid multi-objective cuckoo search, Concurrency and Computation: Practice and Experience 32 (5) (2020) e5478.

[40] Z. Cui, M. Zhang, H. Wang, X. Cai, W. Zhang, J. Chen, Hybrid many-objective cuckoo search algorithm with lévy and exponential distributions, Memetic Computing 12 (3) (2020) 251–265.

[41] X.-S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature inspired cooperative strategies for optimization (NICSO 2010), Springer, 2010, pp. 65–74.

[42] M. A. Al-Betar, β-hill climbing: an exploratory local search, Neural Computing and Applications 28 (1) (2017) 153–168.

[43] Z. Meng, J.-S. Pan, K.-K. Tseng, Pade: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization, Knowledge-Based Systems 168 (2019) 80–99.

[44] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, Advances in engineering software 69 (2014) 46–61.

[45] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1-3) (2006) 489–501.

[46] K. Krishna, M. N. Murty, Genetic k-means algorithm, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 29 (3) (1999) 433–439.

[47] M. Jain, V. Singh, A. Rani, A novel nature-inspired algorithm for optimization: Squirrel search algorithm, Swarm and evolutionary computation 44 (2019) 148–175.

[48] T. T. Nguyen, T. T. Nguyen, An improved cuckoo search algorithm for the problem of electric distribution network reconfiguration, Applied Soft Computing 84 (2019) 105720.

[49] M. Inci, A. Caliskan, Performance enhancement of energy extraction capability for fuel cell implementations with improved cuckoo search algorithm, International Journal of Hydrogen Energy 45 (19) (2020) 11309–11320.

[50] J. Zhao, S. Liu, M. Zhou, X. Guo, L. Qi, Modified cuckoo search algorithm to solve economic power dispatch optimization problems, IEEE/CAA Journal of Automatica Sinica 5 (4) (2018) 794–806.

[51] T. T. Nguyen, T. T. Nguyen, D. N. Vo, An effective cuckoo search algorithm for large-scale combined heat and power economic dispatch problem, Neural Computing and Applications 30 (11) (2018) 3545–3564.

[52] S. I. Boushaki, N. Kamel, O. Bendjeghaba, A new quantum chaotic cuckoo search algorithm for data clustering, Expert Systems with Applications 96 (2018) 358–372.

[53] J. Ji, W. Pang, Z. Li, F. He, G. Feng, X. Zhao, Clustering mixed numeric and categorical data with cuckoo search, IEEE Access 8 (2020) 30988–31003.

[54] D. Acharjya, et al., A hybrid scheme for heart disease diagnosis using rough set and cuckoo search technique, Journal of Medical Systems 44 (1) (2020) 27.

[55] R. Cristin, B. S. Kumar, C. Priya, K. Karthick, Deep neural network based rider-cuckoo search algorithm for plant disease detection, Artificial Intelligence Review (2020) 1–26.

[56] A. M. Kamoona, J. C. Patra, A novel enhanced cuckoo search algorithm for contrast enhancement of gray scale images, Applied Soft Computing 85 (2019) 105749.

[57] N. Vasudevan, V. Nagarajan, et al., Efficient image de-noising technique based on modified cuckoo search algorithm, Journal of medical systems 43 (10) (2019) 307.

[58] J. Ji, W. Pang, C. Zhou, X. Han, Z. Wang, A fuzzy k-prototype clustering algorithm for mixed numeric and categorical data, Knowledge-Based Systems 30 (2012) 129–135.

[59] A. K. Qin, P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: 2005 IEEE congress on evolutionary computation, Vol. 2, IEEE, 2005, pp. 1785–1791.

[60] M. S. Kiran, H. Hakli, M. Gunduz, H. Uguz, Artificial bee colony algorithm with variable search strategy for continuous optimization, Information Sciences 300 (2015) 140–157.

[61] H. Hakli, M. S. Kiran, An improved artificial bee colony algorithm for balancing local and global search behaviors in continuous optimization, International Journal of Machine Learning and Cybernetics (2020) 1–26.

[62] I. Gungor, B. G. Emiroglu, A. C. Cinar, M. S. Kiran, Integration search strategies in tree seed algorithm for high dimensional function optimization, International Journal of Machine Learning and Cybernetics 11 (2) (2020) 249–267.

[63] S. Gao, Y. Gao, Y. Zhang, L. Xu, Multi-strategy adaptive cuckoo search algorithm, IEEE Access 7 (2019) 137642–137655.

[64] A. Yaman, G. Iacca, M. Coler, G. Fletcher, M. Pechenizkiy, Multi-strategy differential evolution, in: International Conference on the Applications of Evolutionary Computation, Springer, 2018, pp. 617–633.

[65] J. Seppä, The cuckoo's ability to find a nest where it can lay an egg, Ornis Fennica 46 (1969) 78–79.

[66] J. Liang, B. Qu, P. Suganthan, A. G. Hernández-Díaz, Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report 201212 (34) (2013) 281–295.

[67] L. Wang, Y. Yin, Y. Zhong, Cuckoo search with varied scaling factor, Frontiers of Computer Science 9 (4) (2015) 623–635.

[68] E. Valian, S. Tavakoli, S. Mohanna, A. Haghi, Improved cuckoo search for reliability optimization problems, Computers & Industrial Engineering 64 (1) (2013) 459–468.

[69] A. M. Kamoona, J. C. Patra, A. Stojcevski, An enhanced cuckoo search algorithm for solving optimization problems, in: 2018 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2018, pp. 1–6.

[70] H. Zheng, Y. Zhou, A novel cuckoo search optimization algorithm based on gauss distribution, Journal of Computational Information Systems 8 (10) (2012) 4193–4200.

[71] H. Rakhshani, A. Rahati, Snap-drift cuckoo search: A novel cuckoo search optimization algorithm, Applied Soft Computing 52 (2017) 771–794.

[72] J. Cheng, L. Wang, Cuckoo search algorithm with neighborhood attraction for numerical optimization, IEEE Access 7 (2019) 122261–122274.

[73] M. Naik, M. R. Nath, A. Wunnava, S. Sahany, R. Panda, A new adaptive cuckoo search algorithm, in: 2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS), IEEE, 2015, pp. 1–5.

[74] J. Alcalá-Fdez, L. Sanchez, S. Garcia, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, et al., Keel: a software tool to assess evolutionary algorithms for data mining problems, Soft Computing 13 (3) (2009) 307–318.

[75] Y. Shi, Brain storm optimization algorithm, in: International conference in swarm intelligence, Springer, 2011, pp. 303–309.

[76] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization 11 (4) (1997) 341–359.

[77] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, J.-s. Pan, Multi-strategy ensemble artificial bee colony algorithm, Information Sciences 279 (2014) 587–603.

[78] R. Mallipeddi, P. N. Suganthan, Q.-K. Pan, M. F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Applied soft computing 11 (2) (2011) 1679–1696.

[79] H. Peng, Z. Wu, C. Deng, Enhancing differential evolution with commensal learning and uniform local search, Chinese Journal of Electronics 26 (4) (2017) 725–733.