Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.040967

ARTICLE

A Spider Monkey Optimization Algorithm Combining Opposition-Based Learning and Orthogonal Experimental Design

Weizhi Liao[1], Xiaoyun Xia[1,3], Xiaojun Jia[1], Shigen Shen[2,*], Helin Zhuang[4,*] and Xianchao Zhang[1]

[1]College of Information Science and Engineering, Jiaxing University, Jiaxing, 314001, China

[2]School of Information Engineering, Huzhou University, Huzhou, 313000, China

[3]Technology Research and Development Centre, Xuelong Group Co., Ltd., Ningbo, 315899, China

[4]School of Internet, Jiaxing Vocational and Technical College, Jiaxing, 314036, China

*Corresponding Authors: Shigen Shen. Email: shigens@zjhu.edu.cn; Helin Zhuang. Email: zhuanghl1998@163.com

Received: 06 April 2023 Accepted: 29 June 2023 Published: 08 October 2023

ABSTRACT

As a new bionic algorithm, Spider Monkey Optimization (SMO) has been widely used in various complex optimization problems in recent years. However, the new space exploration power of SMO is limited and the diversity of the population in SMO is not abundant. Thus, this paper focuses on how to reconstruct SMO to improve its performance, and a novel spider monkey optimization algorithm with opposition-based learning and orthogonal experimental design (SMO^3) is developed. A position updating method based on the historical optimal domain and particle swarm for Local Leader Phase (LLP) and Global Leader Phase (GLP) is presented to improve the diversity of the population of SMO. Moreover, an opposition-based learning strategy based on self-extremum is proposed to avoid suffering from premature convergence and getting stuck at locally optimal values. Also, a local worst individual elimination method based on orthogonal experimental design is used for helping the SMO algorithm eliminate the poor individuals in time. Furthermore, an extended SMO^3 named CSMO^3 is investigated to deal with constrained optimization problems. The proposed algorithm is applied to both unconstrained and constrained functions which include the CEC2006 benchmark set and three engineering problems. Experimental results show that the performance of the proposed algorithm is better than three well-known SMO algorithms and other evolutionary algorithms in unconstrained and constrained problems.

KEYWORDS

Spider monkey optimization; opposition-based learning; orthogonal experimental design; particle swarm

1 Introduction

The real-world optimization problems are to select a group of parameters and make the design target reach the optimal value under a series of given constraints. It is well known that many optimization problems are comparatively hard to solve [1–4]. Nature-inspired optimization algorithms are part of the computer intelligence disciplines, which have become increasingly popular over the past decades [5]. A lot of optimization algorithms, such as Evolutionary Algorithm (EAs) [6], Particle

Swarm Optimization (PSO) [7], Ant Colony Optimization (ACO) [8], Artificial Bee Colony (ABC) [9], Pigeon-Inspired Optimization Algorithm (PIO) [10], Slime Mould Algorithm (SMA) [11] and Crow Search Algorithm (CSA) [12] have been developed to deal with difficult optimization problems. These intelligent biological systems have similar characteristics in which the single individual behavior is simple and random, but the biological groups consisting of these individuals can cooperate to complete a series of complex tasks. Research shows that bionic algorithms can effectively handle numerous kinds of optimization problems.

Inspired by the food-searching behavior of spider monkeys, Bansal et al. developed a new bionic algorithm, called Spider Monkey Optimization (SMO) [13]. Since the SMO algorithm was proposed, this algorithm has been widely used in various complex optimization problems. It has been shown that it is superior concerning reliability, effectiveness, and accuracy to the regular ABC, Distribution Estimation Algorithm (DEA), PSO, and other intelligent algorithms. However, the SMO algorithm has some shortcomings. Typically, the new space exploration power of the original SMO is limited, i.e., it cannot eliminate the poor individuals in time and the diversity of the population is not abundant. These shortcomings seriously affect the performance of the SMO algorithm.

In this paper, we focus on how to reconstruct the SMO algorithm to improve its performance. We propose a spider monkey algorithm combining opposition-based Learning (OBL) and orthogonal experimental design (OED) to cope with unconstrained and constrained optimization problems. The main contributions of our work include: (1) A position update method based on historical optimal domains and particle swarm for Local Leader Phase (LLP) and Global Leader Phase (GLP) is developed. We introduce a novel position update method that combines the particle swarm and traditional update method so that the diversity of the population can be improved. In addition, the position update is performed in the dynamic domain composed by the optimal historical individual with a certain probability to make full use of historical search experience. (2) A population regeneration method based on Opposition-Based Learning (OBL) is presented. Different from other modified SMO algorithms, our approach does not directly enter the Local Leader Learning Phase (LLLP) stage after the LLP and GLP. Instead, it first uses the OBL strategy to avoid suffering from premature convergence and getting stuck at locally optimal values. (3) A method to eliminate the worst individuals in each group of the SMO algorithm based on the orthogonal experimental design is developed. This method performs the horizontal dividing and factor determination of the worst individuals in each group and the global optimal individuals to generate new individuals by orthogonal experimental design. The individuals obtained from this hybrid method retain the historical search experience of the best and the worst spider monkey, thereby enhancing the search performance.

The rest of this paper is organized as follows. Section 2 introduces the related work on the spider monkey optimization algorithm. A spider monkey algorithm named SMO^3 that combines opposition-based learning and orthogonal experimental design for unconstrained optimization problems is presented in Section 3. Section 4 proposes a spider monkey algorithm for the constrained optimization problem based on SMO^3. The experimental results of unconstrained functions, CEC2006 benchmark sets, and a few engineering optimization problems are presented in Section 5. Section 6 concludes this paper and points out future research work.

2 Related Work

In recent years, many variants of SMO have been studied to improve the performances of the original algorithms. Kumar et al. [14] introduced the golden section search method for the position update at the local leader and global leader phases. In [15], a new position update strategy in SMO is presented. The moving distance of a spider monkey at the LLP, the GLP, and the LLDP stages is determined by the individual fitness value. Sharma et al. [16] proposed a position update method based on the age of the spider monkey, which can improve the convergence speed of the SMO algorithm. Hazrati et al. [17] evaluated the size of the position update step according to the fitness value, allowing individuals with small fitness values to quickly approach the globally optimal individual. Gupta et al. [18] developed an improved SMO named constrained SMO (CSMO) for solving constrained continuous optimization problems. Results show that CSMO can obtain better results than DE, PSO, and ABC algorithms. In [19], the position of the worst individual is updated by the fitness of the leader of LLP and GLP and thus the local searchability of SMO is enhanced. Sharam et al. [20] proposed a new method to enhance the searchability of the SMO algorithm, which can find the promising search area around the best candidate solution by iteratively reducing step size. Xia et al. [21] developed a discrete spider monkey optimization (DSMO), which gives different update position methods for the discrete coding in LLP, GLP, and LLDP. However, how to further improve the effectiveness of the SMO algorithm still requires in-depth investigation.

Since the SMO algorithm was proposed, it has been widely applied to various complex optimization problems [22]. Mittal et al. [23] proposed an SMO-based optimization algorithm to improve the network lifetime for clustering protocols. Singh et al. [24] developed an improved SMO named MSMO algorithm to synthesize the linear antenna array (LAA). Results show that the proposed algorithm is an effective way to solve complex antenna optimization problems. Bhargava et al. [25] applied the SMO algorithm to optimize the parameters of the PIDA controller to achieve the optimal control of the induction motor. Cheruku et al. [26] presented an SMO-based rule miner for diabetes classification, and the experiment results show that the classification accuracy of the presented algorithm is better than ID3, CART, and C4.5. Priya et al. [27] proposed an improved SMO algorithm called BW-SMO, which is used for optimizing the query selection of the database. It was found that the proposed method can effectively improve data security. Darapureddy et al. [28] developed a new content-based image retrieval system based on the optimal weighted hybrid pattern. A modified optimization algorithm called improved local leader-based SMO was proposed to optimize the weight that maximizes the precision and recall of the retrieved images. Sivagar et al. [29] developed an improved SMO based on elite opposition and applied it to optimize cell selection with minimal network load. Rizvi et al. [30] presented a Hybrid Spider Monkey Optimization (HSMO) algorithm to optimize the makespan and cost while satisfying the budget and deadline constraints for QoS, and the results obtained show that the effectiveness of HSMO is better than that of the ABC, Bi-Criteria PSO, and BDSD algorithms. Mageswari et al. [31] developed an enhanced SMO-based energy-aware clustering scheme to prolong the network lifetime for wireless multimedia sensor networks.

Compared with some classical SMO algorithms, the proposed method in this paper can obtain the optimal solution more times by running multiple times on unconstrained functions, and the optimal solution has higher accuracy. Also, the algorithm in this paper successfully obtains a higher proportion of feasible and optimal solutions on constrained functions. It is shown that the proposed algorithm is easy to jump out of the local optima and higher solving accuracy.

3 Spider Monkey Algorithm for Unconstrained Optimization

Real-world optimization problems usually can be described as mathematical models of unconstrained functions or constrained functions [32–34]. In this section, we first propose a spider monkey algorithm named SMO[3] that combines opposition-based learning (OBL) and orthogonal experimental design (OED) for unconstrained optimization problems.

3.1 Local Leader Phase Based on Historical Optimal Domain and Particle Swarm

The local Leader Phase (LLP) is an important stage in the SMO algorithm. In this phase, the position of a spider monkey will be updated according to the local optimum. Different from the position update method of other spider monkey algorithms in LLP, the SMO[3] algorithm has two new position update methods in LLP: one is based on the historical optimal domain, and another is based on particle swarms. It compares the pros and cons of the positions obtained by both update methods. The better new position will be compared with the old position, and the position with the better fitness value will be adopted as the current position for a spider monkey.

Definition 1. Let $G_1 = (g_{11}, g_{12}, \ldots, g_{1M})$, $G_2 = (g_{21}, g_{22}, \ldots, g_{2M}), \ldots, G_n = (g_{n1}, g_{n2}, \ldots, g_{nM})$ be n historically optimal individuals, and the historical optimal domain is defined as follows:

$$ld_j = \min_i \left(g_{ij}\right) \tag{1}$$

and

$$ud_j = \max_i \left(g_{ij}\right) \tag{2}$$

where $[ld_j, ud_j]$ is the j-th component of the historical optimal domain, $1 \le i \le n$.

The position update method based on the historical optimal domain not only retains the update method of the traditional SMO algorithm in the LLP stage but also adds a random generation of spider monkey positions in the historical optimal domain. This method allows the individual component values to be limited in the historical optimal domain with a higher probability. Thus, the search experience of the better individual could be used to find new solutions. The mathematical model of the update method in the SMO[3] algorithm is as follows:

If $U\,(0,1) \ge pr$, then

$$SM_{ij}^{new} = SM_{ij} + U\,(0,1) \times \left(LL_{kj} - SM_{ij}\right) + U\,(-1,1) \times \left(SM_{rj} - SM_{ij}\right) \tag{3}$$

If $U\,(0,1) < pr$, then

$$SM_{ij}^{new} = \begin{cases} ld_j + U\,(0,1) \times \left(ud_j - ld_j\right) & U\,(0,1) > cr \\ SM_{ij} & else \end{cases} \tag{4}$$

where SM_{ij} is the position of the j-th component of the i-th spider monkey, $U\,(0,1)$ is a random number in [0,1], pr is the perturbation rate of the SMO algorithm, LL_{kj} is the j-th component of the local leader of the k-th group, while ud_j and ld_j are the upper and lower bounds of the historical optimal domain, $cr \in (0,1)$ and $r \neq i$.

Algorithm 1: LLP based on historical optimal domain and particle swarm

```
Step 1 Implement the update method based on the historical optimal domain for each component $j$ of the $i$-th individual $SM_i$ to obtain the position $new\_pos1$, and the update method is as follows:
    Step 1.1 If $U\,(0,1) \ge pr$, calculate $new\_pos1_j$ according to Eq. (3), otherwise go to Step 1.2.
```

(Continued)

Algorithm 1 (continued)

```
    Step 1.2 If $U\,(0,1) \geq cr$, then
            $new\_pos1_j = ld_j + U(0,1) \times (ud_j - ld_j)$
        else
            $new\_pos1_j = SM_{ij}$
Step 2 Implement the update method based on particle swarm for each component $j$ of the $i$-th individual $SM_i$ to calculate the position $new\_pos2$. The update method is as follows:
    Step 2.1 Update the speed $v_{ij}$ according to Eq. (5).
    Step 2.2 If $v_{ij} > ms_j$, then $v_{ij} = ms_j$. If $v_{ij} < -ms_j$, then $v_{ij} = -ms_j$.
    Step 2.3 $new\_pos2_j = SM_{ij} + v_{ij}$.
    Step 2.4 If $new\_pos2_j$ is out of its domain, $new\_pos2_j$ is randomly generated in $[L_j, U_j]$ with a certain probability, otherwise, $new\_pos2_j$ is randomly generated in the historical optimal domain.
    Step 3 Compare the pros and cons of $new\_pos1$, $new\_pos2$, and $SM_i$, and update $SM_i$ with the best value.
```

In this paper, we introduce a particle swarm-based update method for LLP. This update method enables the spider monkey algorithm to search the solution space in various ways, thus ensuring the diversity of individuals and avoiding the algorithm from falling into a local optimum too early. Let v_{ij} be the walking speed of the i-th spider monkey in the direction j, and the mathematical model of its updated is given by Eq. (5).

$$v_{ij} = gw \times v_{ij} + c_1 \times r_1 \times \left(SM_{rj} - SM_{ij}\right) + c_2 \times r_2 \times \left(LL_{kj} - SM_{ij}\right) \tag{5}$$

where gw is the inertia weight, $c_{1,}$ and c_2 are the learning factors, r_1 and r_2 are random numbers in [0,1], and $r \neq i$. The speed value range of a spider monkey in the direction j is $[-ms_j, ms_j]$, where $ms_j = (U_j - L_j) * 0.2$, L_j and U_j are the lower and upper bounds of the j-th decision variables respectively. The walking speed of a spider monkey is calculated by the experience of the local leader and local group member's experience, and a spider monkey can be led to a better position. According to the walking speed of spider monkey v_{ij}, the position SM_{ij} of the i-th spider monkey in the direction j is updated by Eq. (6).

$$SM_{ij}^{new} = SM_{ij} + v_{ij} \tag{6}$$

The LLP algorithm based on the historical optimal domain and particle swarm is given in Algorithm 1. Let N be the population size and D be the dimension size, Algorithm 1 requires updating all components of each individual, and running the SMO algorithm once requires updating N individuals. Thus, the time complexity of Algorithm 1 is $\mathbf{O}(n^2)$.

3.2 Global Leader Phase Based on Historical Optimal Domain and Particle Swarm

In the global leader phase, each spider monkey updates its position using the position of global leader as well as the local group individual's experience. The traditional position update equation for this phase is given by Eq. (7).

$$SM_{ij}^{new} = SM_{ij} + U\,(0,1) \times \left(GL_j - SM_{ij}\right) + U\,(-1,1) \times \left(SM_{rj} - SM_{ij}\right) \tag{7}$$

In addition to the traditional position update method, we present a position update method based on particle swarm in GLP. The position obtained by the particle swarm method is compared with the position obtained by Eq. (7), and the better one is adopted as a candidate position.

Algorithm 2: GLP Based on Historical Optimal Domain and Particle Swarm

```
Step 1 If $U(0,1) < prob$, then go to Step 2 to update $SM_i$.
Step 2 Randomly select $j \in \{1, 2, \ldots, D\}$.
Step 3 Implement the traditional update method on the component $j$ of $SM_i$ according to Eq. (7) to calculate the position $new\_pos1$.
Step 4 Randomly select $j \in \{1, 2, \ldots, D\}$.
Step 5 Implement the particle swarm method on the component $j$ of $SM_i$ to obtain the position $new\_pos2$ as follows:
    Step 5.1 Calculate the spider monkey walking speed $v_{ij}$ according to Eq. (8).
    Step 5.2 If $v_{ij}$ is out of its range, then correct $v_{ij}$.
    Step 5.3 $new\_pos2_j = SM_{ij} + v_{ij}$
    Step 5.4 If $new\_pos2_j$ is out of its domain, $new\_pos2$ is randomly generated in $[L_j, U_j]$ with a certain probability, otherwise, $new\_pos2$ is randomly generated in the historical optimal domain.
Step 6 Find the best position among $new\_pos1$, $new\_pos2$, and $SM_i$, then update $SM_i$ with the best position.
```

Let v_{ij} be the walking speed of the i-th spider monkey in the direction j, and the updated method of v_{ij} is given by Eq. (8).

$$v_{ij} = gw \times v_{ij} + c_1 \times r_1 \times \left(SM_{rj} - SM_{ij}\right) + c_2 \times r_2 \times \left(GL_j - SM_{ij}\right) \tag{8}$$

The walking speed of a spider monkey is determined by the experience of the global leader as well as the local group individua's experience by Eq. (8), and a spider monkey has a chance to move to a better position.

According to the spider monkey's walking speed, the position of the i-th spider monkey in the direction j is updated in the same way by Eq. (6). The GLP algorithm based on the historical optimal domain and particle swarm is shown in Algorithm 2. Let N be the population size and D be the dimension size, Algorithm 2 requires updating one component of each individual, and running the SMO algorithm once requires updating N individuals. Thus, the time complexity of Algorithm 2 is $\mathbf{O}(n)$.

3.3 OBL Strategy Based on Extreme Value

The main idea of Opposition-Based Learning (OBL) is to evaluate the feasible solution and its reverse solution, and the better solution is adopted by the individuals of the next generation. Since opposition-based learning was developed, OBL has been applied to various optimization algorithms, which is capable of improving the performance of these optimization algorithms to search for the problem solution [35]. To make better use of the search experience of each spider monkey, we propose an OBL strategy based on its extreme value and apply it to the SMO^3 algorithm.

Definition 2. Let the number of spider monkeys in population G be NP, and the position of the i-th spider monkey is denoted as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, $1 \le i \le NP$, and $best_i^t = (b_{i1}, b_{i2}, \ldots, b_{iD})$ is the optimal position of the i-th spider monkey when the algorithm is iterated to the t-th generation, we define the optimal domain based on the individual's extreme value as follows:

$$lz_j = \min_j \left(b_{ij}\right) \tag{9}$$

and

$$uz_j = \max_j \left(b_{ij}\right) \tag{10}$$

where $1 \le i \le NP,\ 1 \le j \le D$.

Definition 3. Let $best_i^t = (b_{i1}, b_{i2}, \ldots, b_{iD})$ be the best position of the i-th spider monkey when the algorithm is iterated to the t-th generation, then its j-th component is updated by the OBL strategy with a certain probability, and its updating method is given by Eq. (11).

$$best_{ij}^{new} = U(0,1) \times \left(uz_j + lz_j\right) - best_{ij} \tag{11}$$

If $best_{ij}^{new}$ is out of its dynamic domain, we recalculate it according to Eq. (12).

$$best_{ij}^{new} = \begin{cases} lz_j + U(0,1) \times \left(uz_j - lz_j\right) & U(0,1) < 0.5 \\ L_j + U(0,1) \times \left(U_j - L_j\right) & else \end{cases} \tag{12}$$

where U (0,1) is a random number in [0,1].

In this paper, we first construct the lower and upper bounds of decision variables based on their extreme values. Furthermore, opposition-based learning is applied to calculate the best position of a spider monkey with the upper and lower bounds. If the position obtained by OBL is better than the current position, it is used to replace the current position. The OBL based on its extreme value is shown in Algorithm 3, where NP is the number of spider monkeys, and $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ is the position of spider monkey i, $1 \le i \le NP$. It is not difficult to see that Algorithm 3 is composed of two nested loops, thus the time complexity of Algorithm 1 is $\mathbf{O}(n^2)$.

Algorithm 3: OBL Based on Extreme Value

```
Step 1 Construct its extreme value optimal domain according to Eqs. (9) and (10).
Step 2 Implement the OBL as follows:
  Step 2.1 $i = 1$
  Step 2.2 If $i <= NP$ then go to Step 2.3, else return results
  Step 2.3 $temp = best_i, j = 1$
  Step 2.4 If $j <= D$ then go to Step 2.5, else go to Step 2.8
  Step 2.5 If $U(0,1) < 0.2$ then $temp[j] = U(0,1) \times (uz_j + lz_j) - best_{i,j}$
  Step 2.6 If $temp[j]$ not in $[L_j, U_j]$, then recalculate according to Eq. (12)
  Step 2.7 $j = j + 1$, go to Step 2.4
  Step 2.8 If $temp$ is better than $X_i$ then $X_i = temp$
  Step 2.9 $i = i + 1$, go to Step 2.2
```

3.4 Worst Individual Elimination Mechanism Based on Orthogonal Experimental Design

Orthogonal experimental design (OED) is an important branch of statistical mathematics, based on probability theory, mathematical statistics, and the standardized orthogonal table to arrange the test plan [36]. It is another design method to study multiple factors and multiple levels. It selects some representative points from the comprehensive test according to the orthogonality. These representative points have the characteristics of uniform dispersion and comparability. OED is an efficient, fast, and economical method of experiment design. Using an orthogonal experiment design to incorporate heuristic algorithms is an effective way to improve the efficiency of heuristic algorithms [37,38].

Let the worst individual of the i-th group be $worst = (w_1, w_2, \ldots, w_D)$, and the global leader is $gbest = (g_1, g_2, \ldots, g_D)$. We first use the method presented in [37] to calculate the level of each component of the worst individual and the global leader. Let $L_{i,k}$ be the value of the k-th level of the i-th component, and S is the number of levels, and the calculation method of $L_{i,k}$ is given in Eq. (13).

$$L_{ik} = \min(w_i, g_i) + (\max(w_i, g_i) - \min(w_i, g_i)) \times \frac{k-1}{D-1} \quad (13)$$

where $i = 1, 2, \ldots, D$, $k = 1, 2, \ldots, S$.

Let the number of factors in the orthogonal experiment be F. In the process of constructing S horizontal orthogonal tables, if the number of components is small, each component can be directly used as a factor. In this case, the number of factors $F = D$. In case the value of D is large, if D is directly used as D factors, the number of orthogonal experiments will be too large, which can increase the complexity of the algorithm and the algorithm may run too slowly. For this reason, D components are divided into F groups to satisfy Eq. (14):

$$D = F \times h + n \quad (14)$$

where $h = \text{int}(D/\text{F})$, and $n = D\%F$. The components contained in each group are as follows:

$$D_i = \left(w_{(i-1)\times h+1}, \ldots, w_{i\times h}\right) 1 \le i \le F \quad (15)$$

$$D_{1,j} = \left(L_1, \ldots, L_{h,j}\right) 1 \le j \le S \quad (16)$$

After various level values of each factor are determined, the best level combination method is chosen according to the orthogonal experimental design method to obtain a new individual. If the new individual is better than the worst individual in the group, the worst individual will be replaced by the new individual. Otherwise, the new individual will not be adopted into the population.

Finally, an example is presented to show how to use OED to generate new individuals based on the worst individual and the best individual. Let *worst* = (1,3,0,8,7,4,2,6,3) be the worst individual and *gbest* = (5,4,6,1,6,0,9,3,2) be the best individual in 9 dimensions space. The number of levels is 3, and the number of groups is 4. According to Eqs. (14) and (15), we have $D_1 = (x_1,x_2)$, $D_2 = (x_3,x_4)$, $D_3 = (x_5,x_6)$, *and* $D_4 = (x_7,x_8,x_9)$. By Eq. (16), we can obtain 3 levels in each group as follows: $D_{1,1}$ = (1,3), $D_{1,2}$ = (3,3.5), $D_{1,3}$ = (5,4), $D_{2,1}$ = (0,1), $D_{2,2}$ = (3,4.5), $D_{2,3}$ = (6,8), $D_{1,1}$ = (1,3), $D_{1,2}$ = (3,3.5), $D_{1,3}$ = (5,4), $D_{3,1}$ = (6,0), $D_{3,2}$ = (6.5,2), $D_{3,3}$ = (7,4), $D_{4,1}$ = (2,3,2), $D_{4,2}$ = (5.5,4.5,2.5), and $D_{4,3}$ = (9,6,3). Based on these data, we can apply orthogonal experimental design to generate new individuals. All individuals are given in Table 1.

Table 1 New individual generated by OED

N_1	(1,3,0,1,6,0,2,3,2)
N_2	(1,3,3,4.5,6.5,2,5.5,4.5,2.5)
N_3	(1,3,6,8,7,4,9,6,3)
N_4	(3,3.5,0,1,6.5,2,9,6,3)
N_5	(3,3.5,3,4.5,7,4,2,3,2)
N_6	(3,3.5,6,8,6,0,5.5,4.5,2.5)
N_7	(5,4,0,1,7,4,5.5,4.5,2.5)
N_8	(5,4,3,4.5,6,0,9,6,3)
N_9	(5,4,6,8,6.5,2,2,3,2)

3.5 Description of SMO^3 *Algorithm*

The original SMO process consists of six phases: Local Leader Phase (LLP), Global Leader Phase (GLP), Global Leader Learning Phase (GLLP), Local Leader Learning Phase (LLLP), Local Leader Decision Phase (LLDP), and Global Leader Decision Phase (GLDP). The differences between the original SMO algorithm and the SMO^3 algorithm that integration of OBL and orthogonal experiment design include: (1) In addition to the six phases of traditional SMO, it adds OBL and orthogonal experiment design stages. The implementation of these two stages is after LLP and GLP but before GLLP and LLLP. The addition of these two stages allows the group to generate new individuals in a variety of ways, thereby ensuring the diversity of the group. (2) The LLP and GLP stages of the algorithm in this paper are different from the traditional SMO algorithm. It adopts the local leader and global leader algorithms based on the historically optimal domains and particle swarms proposed in Sections 3.1 and 3.2. The flowchart of the SMO^3 algorithm based on the fusion of OBL and orthogonal experimental design is shown in Fig. 1. The time complexity of the SMO^3 algorithm is $\mathbf{O}\ ((n^3 + n^2 + n)\ N_{gen} + n^2)$, where N_{gen} is the number of generations of the algorithm.

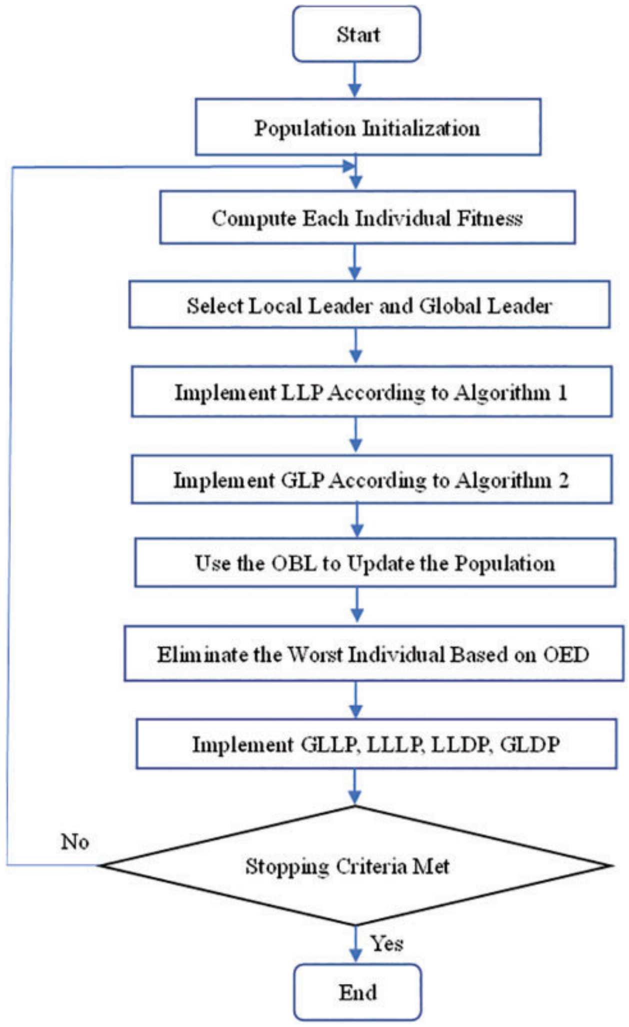
Start
Population Initialization
Compute Each Individual Fitness
Select Local Leader and Global Leader
Implement LLP According to Algorithm 1
Implement GLP According to Algorithm 2
Use the OBL to Update the Population
Eliminate the Worst Individual Based on OED
Implement GLLP, LLLP, LLDP, GLDP
No
Stopping Criteria Met
Yes
End

Figure 1: Flowchart of SMO^3

4 SMO Algorithm for Constrained Optimization Problem

Different from the unconstrained optimization problem, the solution for the constrained optimization problems may not be the feasible solution, and the SMO^3 algorithm cannot be used directly to solve the constrained optimization problem. For this reason, we extend the SMO^3 algorithm to propose a spider monkey algorithm for dealing with constrained function optimization problems, namely CSMO^3.

4.1 Evaluation of Individual's Pros and Cons

It is well known that the pros and cons of the two individuals X_1 and X_2 in the search space are usually evaluated in the algorithm. For unconstrained optimization problems, if $f(X_1) < f(X_2)$ X_1 is better than X_2. Otherwise, X_2 is better than X_1. In a constrained optimization problem, an individual in a search space may not be the feasible solution. Therefore, the evaluation of the individual's pros and cons must be based on whether the individual is a feasible solution. We use the following rules to evaluate the individual's pros and cons.

Rule 1: In case both individuals X_1 and X_2 are feasible solutions, if $f(X_1) < f(X_2)$, then the individual X_1 is better than X_2, else X_2 is better than X_1.

Rule 2: If the individual X_1 (X_2) is a feasible solution, but X_2 (X_1) is not a feasible solution, the individual X_1 (X_2) is better than X_2 (X_1).

Rule 3: If both individuals X_1 and X_2 are not feasible solutions, the individual who violates fewer constraints is better than the individual who violates more constraints.

The judgment rule of Rule 3 is as follows: Let cn_1 and cn_2 be the numbers of individuals X_1 and X_2 that do not meet the constraints respectively, $value(X_1) < value(X_2)$ be the violation constraint value of X_1 and X_2. If $cn_1 < cn_2$, the individual X_1 is better than X_2. If $cn_2 < cn_1$, the individual X_2 is better than X_1. When $cn_1 = cn_2$, if $value(X_1) < value(X_2)$, the individual X_1 is better than X_2. Otherwise, X_2 is better than X_1. Different from the rule in [18] that only evaluates the pros and cons of individuals based on the violation constraint value, we give priority to individuals with a small number of violation constraints, and the pros and cons of the individual are determined by the violation constraint value only when cn_1 is equal to cn_2.

4.2 OBL Strategy of CSMO^3 Algorithm

In the CSMO^3 algorithm, the OBL strategy is used to generate the initial population to improve the quality of the initial population. Firstly, a population is randomly generated, and the OBL strategy is implemented to each decision variable of every individual by (17) with a certain probability.

$$x_j^{new} = L_j + U_j - x_j \tag{17}$$

In addition to the OBL strategy for building the initial population, the population is also updated by the OBL strategy at each generation of the CSMO^3 algorithm.

4.3 Local Leader Phase of CSMO^3 Algorithm

In the local leader phase of the CSMO^3 algorithm, a temporary new position is generated by the position update method of traditional LLP first. Next, the new position is updated by the particle swarm method with a probability of $0.4-cr$ so that another new position is obtained. Finally, the pros and cons of these new positions and the original position are compared by the evaluation rules in

Section 4.1, and the best position is used to update the original position. The main steps of LLP are given in Algorithm 4.

Let N be the population size and D be the dimension size, Algorithm 4 requires updating all components of each two times, and running the SMO algorithm once requires updating N individuals. Thus, the time complexity of Algorithm 4 is $\mathbf{O}(2n^2)$.

4.4 Global Leader Phase of CSMO^3 Algorithm

In the global leader phase of the CSMO^3 algorithm, a temporary new position is generated according to the position update method of traditional GLP. Furthermore, a component of the new position is randomly selected to be updated according to the particle swarm method and obtain another new position. Finally, the pros and cons of these new positions and the original position are compared, and the best position is used to update the original position. The main steps of GLP are given in Algorithm 5. Let N be the population size and D be the dimension size, Algorithm 5 requires updating one component of each two times, and running the SMO algorithm once requires updating N individuals. Thus, the time complexity of Algorithm 5 is $\mathbf{O}(2n)$.

```
Algorithm 4: LLP of the CSMO$^3$ Algorithm
Step 1 Update each component of $SM_i$ according to the traditional SMO algorithm to obtain the position $new\_pos1$.
Step 2 Implement the particle swarm-based update strategy for each component of $SM_i$ with a probability of $0.4-cr$ to obtain another new position $new\_pos2$.
Step 3 According to the evaluation rules in Section 4.1, find the best position in $new\_pos1$, $new\_pos2$, and $SM_i$ to replace $SM_i$.
```

```
Algorithm 5: GLP of the CSMO$^3$ Algorithm
Step 1 If $U(0,1) < prob$, then go to Step 2 to update $SM_i$.
Step 2 Randomly select $j \in \{1, 2, \ldots, D\}$.
Step 3 Implement the traditional update strategy on component $j$ of $SM_i$ to obtain new position $new\_pos1$ according to Eq. (7).
Step 4 Randomly select $j \in \{1, 2, \ldots, D\}$.
Step 5 Implement the particle swarm method on the component $j$ of $SM_i$ to obtain another new position $new\_pos2$
Step 6 According to the evaluation rules in Section 4.1, find the best position among $new\_pos1$, $new\_pos2$, and $SM_i$ to replace $SM_i$.
```

4.5 Description of CSMO^3 Algorithm

The main steps of the CSMO^3 algorithm for handling constrained optimization problems are similar to the main steps of the SMO^3 algorithm in Section 3. The main differences include: (1) the OBL strategy is used to generate the initial population to improve the quality of the initial population in the CSMO^3 algorithm; (2) the pros and cons of unfeasible solution are considered in the CSMO^3 algorithm; (3) the CSMO^3 algorithm only combines traditional position update method and particle swarm update method at LLP and GLP stage. The flowchart of the CSMO^3 algorithm is shown in Fig. 2. The time complexity of the CSMO^3 algorithm is $\mathbf{O}(\,(n^3 + n^2 + n)\,N_{gen} + n^2)$.

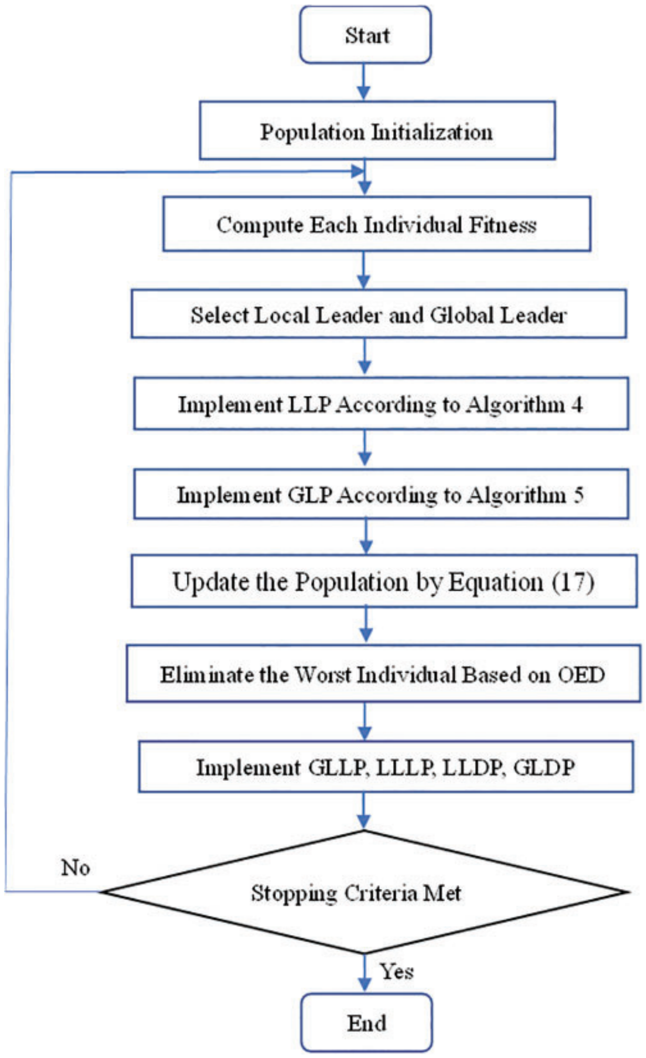
Start
Population Initialization
Compute Each Individual Fitness
Select Local Leader and Global Leader
Implement LLP According to Algorithm 4
Implement GLP According to Algorithm 5
Update the Population by Equation (17)
Eliminate the Worst Individual Based on OED
Implement GLLP, LLLP, LLDP, GLDP
Stopping Criteria Met
No
Yes
End

Figure 2: Flowchart of CSMO^3

5 Numerical Experiments

To verify the effectiveness of the spider monkey optimization algorithm proposed in this paper, the experiment comparison is performed on unconstrained functions, the CEC2006 benchmark set, and engineering examples. The proposed algorithm is coded in Python 3.2 and the experiments are run on a PC with Intel(R) Core(TM) i7-10510U, CPU @1.80 GHz 2.30 GHz, and Windows 10 operating system.

The parameter setting for every SMO algorithm has been adopted as it is mentioned in reference [15]. The parameter setting of SMO algorithms is as follows: the maximum number of generations of algorithms $MIR = 20000$, the population size $N = 50$, the number of groups $MG = 5$, *GlobalLeaderLimit* $= 50$, *LocalLeaderLimi*t $= 1500$, the perturbation rate $pr \in [0.1,0.4]$ with linear increase according to the number of iterations and $pr_{G+1} = pr_G + (0.4-0.1)/MIR$. These parameters are currently recognized as the best combination of parameters for the SMO algorithm.

5.1 Experiments on Unconstrained Optimization Problems

In this section, the effectiveness of the SMO^3 algorithm is investigated. We use the SMO^3 algorithm, original SMO algorithm [13], fitness-based position update in spider monkey optimization algorithm (FPSMO) [15], and adaptive step-size based spider monkey optimization algorithm (AsSMO) [17] to cope with unconstrained functions respectively. The results of SMO^3 are compared with that of SMO, FPSMO, and AsSMO for performance demonstration. The classic test functions used in this section are briefly introduced in Table 2. Among 21 test functions, there are 12 functions with variable dimensions and 9 functions with fixed dimensions.

Table 2: Unconstrained function

Identifier	Function name	Dimension	Domain	Tolerance error
F01	Michalewicz	variable	[0,p]	1.00E−05
F02	Step function	variable	[−100,100]	1.00E−05
F03	Levy montalvo 1	variable	[−10,10]	1.00E−05
F04	Levy montalvo 2	variable	[−5,5]	1.00E−05
F05	Ellipsoidal	variable	[−D, D]	1.00E−05
F06	Beale	2	[−1000,1000]	1.00E−05
F07	Kowalik	4	[−5,5]	1.00E−05
F08	2D Tripod	2	[−100,100]	1.00E−05
F09	Shifted Rosenbrock	variable	[−100,100]	1.00E−05
F10	Shifted Sphere	variable	[−100,100]	1.00E−05
F11	Shifted Rastrigin	variable	[−5,5]	1.00E−05
F12	Shifted Schwefel	variable	[−100,100]	1.00E−05
F13	Shifted Griewank	variable	[−600,600]	1.00E−05
F14	Shifted Ackley	variable	[−32,32]	1.00E−05
F15	Goldstein-Price	2	[−2,2]	1.00E−14
F16	Easom’s function	2	[−10,10]	1.00E−13
F17	Dekkers and Aarts	2	[−20,20]	5.00E−01
F18	McCormick	2	[−3,3]	1.00E−04
F19	Meyer and Roth	3	[−10,10]	1.00E−04
F20	Shuber	2	[−10,10]	1.00E−02
F21	Sinusoidal	variable	[0,p]	1.00E−02

The accuracy of the results, the number of iterations, and the success rate of the four algorithms are compared in this section. Each algorithm runs 50 times on each function. Table 3 shows the experimental results which include the mean deviation (MD), standard deviation (SD), average number of iterations (AIR), and success rate (SR) of these 50 results. The success rate is the proportion of the results obtained from 50 runs of the algorithm within the tolerance error range. Among the 21 test functions, the success rate of the proposed method in this paper is higher than or equal to that of the other three algorithms. The success rate of the SMO^3 algorithm is 100% except for function F15. F15 is the only function whose success rate of the four algorithms cannot reach 100%, where the success rates of SMO^3, SMO, AsSMO, and FPSMO were 80%, 54%, 78%, and 0%, respectively. The mean deviation of the SMO^3 algorithm on 14 functions is better than or equal to the other three algorithms, and SMO, AsSMO, and FPSMO algorithms have 4 functions, 5 functions, and 1 function, respectively. The standard deviation of the SMO^3 algorithm on 13 functions is better than or equal to the other three algorithms, while SMO, AsSMO, and FPSMO algorithms have 5 functions, 5 functions, and 1 function respectively. The average number of iterations of the SMO^3 algorithm on 9 functions is better than that of the other three algorithms, while SMO, AsSMO, and FPSMO algorithms only have 7 functions, 1 function, and 4 functions, respectively.

Table 3: Experimental results of unconstrained functions

Func_D	Algorithm	MD	SD	AIR	SR(%)
F01_10	SMO	**2.60E−06**	**3.07E−06**	581.5	**100**
	AsSMO	2.98E−06	3.40E−06	560.9	**100**
	FPSMO	6.67E−01	8.20E−01	2.00E+04	0
	SMO^3	2.92E−06	3.46E−06	**285.4**	**100**
F02_60	SMO	**0.00E+00**	**0.00E+00**	149.1	**100**
	AsSMO	**0.00E+00**	**0.00E+00**	148.4	**100**
	FPSMO	**0.00E+00**	**0.00E+00**	**85.7**	**100**
	SMO^3	**0.00E+00**	**0.00E+00**	140.2	**100**
F03_60	SMO	6.55E−07	7.87E−07	191.9	**100**
	AsSMO	5.78E−07	7.42E−07	**188.8**	**100**
	FPSMO	1.15E−03	1.74E−03	15619.4	22
	SMO^3	**2.45E−07**	**3.36E−07**	375.1	**100**
F04_60	SMO	6.01E−07	7.49E−07	**197**	**100**
	AsSMO	4.60E−07	5.98E−07	200	**100**
	FPSMO	1.19E−02	2.52E−02	16418.8	18
	SMO^3	**1.03E−07**	**1.35E−07**	607.6	**100**
F05_100	SMO	1.54E−06	1.82E−06	2105.1	**100**
	AsSMO	1.00E−06	1.28E−06	2230.2	**100**
	FPSMO	6.82E+03	8.67E+03	2.00E+04	0
	SMO^3	**1.05E−07**	**1.47E−07**	**1671.3**	**100**
F06_2	SMO	2.50E−06	2.91E−06	**25.9**	**100**
	AsSMO	**2.33E−06**	**2.76E−06**	30.1	**100**
	FPSMO	3.65E+01	4.64E+01	18666.5	6
	SMO^3	2.59E−06	3.00E−06	36.7	**100**
F07_4	SMO	1.72E−06	2.05E−06	**323.5**	**100**
	AsSMO	**1.45E−06**	**1.75E−06**	408.2	**100**
	FPSMO	6.23E−03	8.76E−03	2.00E+04	0
	SMO^3	1.63E−06	2.14E−06	348.7	**100**
F08_2	SMO	3.10E−01	4.77E−01	12996.5	86
	AsSMO	2.69E−01	3.67E−01	3886.8	84
	FPSMO	7.90E−01	8.89E−01	19599.4	2
	SMO^3	**2.39E−06**	**2.78E−06**	**4104.3**	**100**

(Continued)

Table 3 (continued)

Func_D	Algorithm	MD	SD	AIR	SR(%)
F09_20	SMO	1.27E−06	1.65E−06	12809	**100**
	AsSMO	1.87E−06	2.29E−06	8438.3	**100**
	FPSMO	8.53E+00	8.86E+00	19295.2	4
	SMO^3	**7.16E−07**	**9.44E−07**	**2322.3**	**100**
F10_600	SMO	**2.01E−07**	**2.71E−07**	8383.4	**100**
	AsSMO	2.62E−07	3.53E−07	6925.3	**100**
	FPSMO	4.17E−07	6.60E−07	1802.2	**100**
	SMO^3	5.40E−07	6.50E−07	**584.5**	**100**
F11_100	SMO	8.27E−07	1.30E−06	4177.2	**100**
	AsSMO	1.20E−06	1.67E−06	4260.2	**100**
	FPSMO	3.48E−06	3.93E−06	**437.8**	**100**
	SMO^3	**4.60E−07**	**5.95E−07**	1215.2	**100**
F12_10	SMO	1.44E−06	1.83E−06	**171.6**	**100**
	AsSMO	1.60E−06	1.98E−06	183.4	**100**
	FPSMO	2.90E−06	3.44E−06	1167.6	**100**
	SMO^3	**1.19E−06**	**1.35E−06**	300.4	**100**
F13_10	SMO	2.10E−06	2.72E−06	371.3	**100**
	AsSMO	**4.87E−07**	**6.61E−07**	558.9	**100**
	FPSMO	2.33E−06	2.88E−06	**317**	**100**
	SMO^3	2.13E−06	2.54E−06	358.3	**100**
F14_30	SMO	4.83E−07	6.11E−07	910	**100**
	AsSMO	5.68E−07	6.66E−07	1109.5	**100**
	FPSMO	3.81E−06	4.03E−06	**148.1**	**100**
	SMO^3	**4.75E−07**	**5.94E−07**	434	**100**
F15_2	SMO	4.86E−14	4.88E−14	9226.6	54
	AsSMO	3.41E−14	4.07E−14	4449.3	78
	FPSMO	2.34E+00	3.27E+00	2.00E+04	0
	SMO^3	**3.14E−14**	**3.96E−14**	**4041**	**80**
F16_2	SMO	2.87E−14	3.25E−14	**66.9**	**100**
	AsSMO	2.51E−14	2.93E−14	68.4	**100**
	FPSMO	1.42E−01	2.62E−01	2.00E+04	0
	SMO^3	**2.45E−14**	**2.85E−14**	76.7	**100**
F17_2	SMO	5.08E−03	5.77E−03	**11**	**100**
	AsSMO	**4.16E−03**	**4.90E−03**	12.2	**100**
	FPSMO	3.98E+02	1.01E+03	16401.5	18
	SMO^3	4.42E−03	5.14E−03	14.9	**100**
F18_2	SMO	5.27E−06	6.02E−06	6.1	**100**
	AsSMO	5.24E−06	6.25E−06	6.3	**100**
	FPSMO	5.17E−03	1.14E−02	14400	16
	SMO^3	**4.27E−06**	**5.36E−06**	**5.2**	**100**

(Continued)

Table 3 (continued)

Func_D	Algorithm	MD	SD	AIR	SR(%)
F19_3	SMO	**1.06E−05**	**1.31E−05**	17.6	**100**
	AsSMO	1.49E−05	1.69E−05	20.1	**100**
	FPSMO	4.51E−03	8.26E−03	2.00E+04	0
	SMO^3	1.41E−05	1.54E−05	**13**	**100**
F20_2	SMO	4.80E−06	5.31E−06	**48.5**	**100**
	AsSMO	4.87E−06	5.44E−06	57.7	**100**
	FPSMO	7.07E+00	1.74E+01	2.00E+04	0
	SMO^3	**4.51E−06**	**5.27E−06**	104.4	**100**
F21_10	SMO	1.56E−03	**1.82E−03**	1347.5	**100**
	AsSMO	1.79E−03	2.11E−03	1465.8	**100**
	FPSMO	9.51E−03	2.22E−02	1241	94
	SMO^3	**1.52E−03**	2.03E−03	**399.5**	**100**

To compare the pros and cons of the above four algorithms comprehensively, the accuracy, average number of iterations, and success rate of each algorithm on 21 test functions are evaluated statistically for the number ranked 1st to 4th according to the experimental results in Table 3. Let x_1, x_2, x_3, and x_4 be the numbers of the algorithm ranked 1st to 4th in the index, and their score calculation method is given in Eq. (18).

$$score = x_1 + 2x_2 + 3x_3 + 4x_4 \tag{18}$$

The experimental results are shown in Table 3 and the comprehensive score of MD, SD, AIR, and SR is shown in Fig. 3. The smaller the comprehensive score, the more the number of rankings at the top. The number of 1st rankings in the MD, SD, AIR, and SR indicators for the SMO^3 algorithm is 14, 13, 9, and 21, respectively; the comprehensive scores are 36, 29, 40, and 21, respectively; the comprehensive rankings of four indexes are all No. 1.

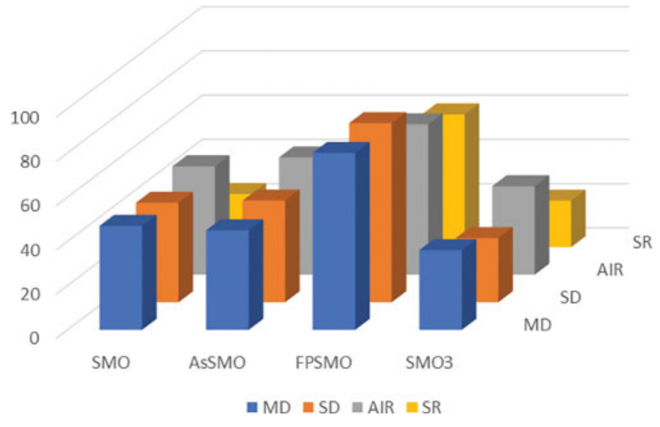

Figure 3: Score of MD, SD, AIR, and SR

The experimental results show that the SMO^3 algorithm is more effective than SMO, AsSMO, and FPSMO in coping with unconstrained optimization problems. That is because the position update method based on PSO can increase the diversity of the population, opposition-based learning can enhance the ability to explore new space, and the method of eliminating the worst individual based on orthogonal experimental design can eliminate the worst individual in time.

5.2 Experiments on Constrained Optimization Problems

To examine the effectiveness of the CSMO^3 algorithm in solving the constrained function optimization problems, the experiment is conducted on the CEC2006 benchmark set [39], which compares with the SMO algorithm, the AsSMO algorithm, and the FPSMO algorithm. These four algorithms use the same individual pros and cons evaluation method, and the main parameter settings are the same as in Section 5.1. Each algorithm runs 30 times. The feasible solution rate (FR), the best solution (BS), the worst solution (WS), the average solution (AV), the average number of iterations (AIR), and the success rate (SR) are respectively examined. The feasible solution rate is the ratio of obtaining the feasible solution in 30 runs, and the success rate is the ratio of obtaining the best-known solution in 30 runs. In the experiments on the CEC2006 benchmark set, the tolerance error is 1E-05 for g01, 1E-03 for g24, 1E-16 for both g08 and g12, and the tolerance errors of other functions are all 1E-04. The experimental comparison data on the CEC2006 benchmark set is shown in Table 4, where the bold data of each index indicates that the corresponding algorithm ranks first in this index, and 'V*n*' means that the number of constraint violations is *n*.

Table 4: Experimental comparison of the CEC2006 benchmark set

Func.	Algorithm	FR (%)	BS	WS	AV	AIR	SR (%)
g01	SMO	**100**	**−14.99999**	−13.828125	14.84374	2939.6	86.7
	AsSMO	**100**	**−14.99999**	−13.82812	−14.8828	2275.5	90
	FPSMO	3.3	−3.71249	V7	V4.7	2.00E+04	0
	CSMO^3	**100**	**−14.99999**	**−14.99999**	−14.99999	**218.4**	**100**
g02	SMO	**100**	−0.80352	−0.59605	−0.73225	18730.6	6.7
	AsSMO	**100**	−0.80352	**−0.67459**	−0.76776	19367	3.3
	FPSMO	**100**	−0.26336	−0.16016	−0.20112	2.00E+04	0
	CSMO^3	**100**	**−0.80353**	−0.66593	**−0.77667**	**16127.5**	**20**
g03	SMO	**100**	−0.82709	−0.28096	−0.60016	2.00E+04	0
	AsSMO	**100**	−0.82211	−0.20072	−0.5079	2.00E+04	0
	FPSMO	90	−0.32654	V1	V0.1	2.00E+04	0
	CSMO^3	**100**	**−0.83291**	**−0.30282**	**−0.62079**	2.00E+04	0
g04	SMO	**100**	−30665.53862	**−30665.53857**	−30665.53858	492.8	**100**
	AsSMO	**100**	−30665.53861	**−30665.53857**	−30665.53858	402.3	**100**
	FPSMO	**100**	−30400.57074	−28152.99479	−29372.52622	2.00E+04	0
	CSMO^3	**100**	**−30665.53864**	**−30665.53857**	**−30665.53859**	**364.8**	**100**
g05	SMO	**100**	**5126.50462**	5704.72014	5265.95289	2.00E+04	0
	AsSMO	**100**	5126.54432	5947.93966	**5233.32023**	2.00E+04	0
	FPSMO	0	V3	V3	V3	2.00E+04	0
	CSMO^3	**100**	5126.52999	**5636.09331**	5261.59181	2.00E+04	0

(Continued)

Table 4 (continued)

Func.	Algorithm	FR (%)	BS	WS	AV	AIR	SR (%)
g06	SMO	**100**	**−6961.81386**	**−6961.81378**	−6961.81379	**1485**	**100**
	AsSMO	**100**	**−6961.81386**	**−6961.81378**	**−6961.8138**	1490.2	**100**
	FPSMO	30	−3290.56671	V1	V0.7	2.00E+04	0
	CSMO³	**100**	−6961.81383	**−6961.81378**	−6961.81379	1506.6	**100**
g07	SMO	**100**	24.38381	25.85533	24.95247	2.00E+04	0
	AsSMO	**100**	24.38818	26.39786	24.9758	2.00E+04	0
	FPSMO	43.3	330.42756	V2	V0.63	2.00E+04	0
	CSMO³	**100**	**24.3187**	**24.92294**	**24.47794**	2.00E+04	0
g08	SMO	**100**	**−0.095825**	**−0.095825**	**−0.095825**	**33.7**	**100**
	AsSMO	**100**	**−0.095825**	**−0.095825**	**−0.095825**	34.5	**100**
	FPSMO	**100**	−0.095823	−0.028086	−0.077529	2.00E+04	0
	CSMO³	**100**	**−0.095825**	**−0.095825**	**−0.095825**	39.6	**100**
g09	SMO	**100**	680.63039	680.64343	680.635024	2.00E+04	0
	AsSMO	**100**	680.63025	680.96121	680.64383	2.00E+04	0
	FPSMO	**100**	734.36381	346943.815	18387.43191	2.00E+04	0
	CSMO³	**100**	**680.63022**	**680.63268**	**680.63124**	2.00E+04	0
g10	SMO	**100**	7053.26956	8077.26919	7442.60767	2.00E+04	0
	AsSMO	**100**	3.53E+15	3.54E+15	3.53E+15	2.00E+04	0
	FPSMO	3.3	14801.87224	V2	V1.33	2.00E+04	0
	CSMO³	**100**	**7051.77361**	**7296.01169**	**7153.58994**	2.00E+04	0
g11	SMO	**100**	0.749997	0.99998	0.88055	19766.6	3.33
	AsSMO	**100**	0.75475	0.99879	0.92286	2.00E+04	0
	FPSMO	**100**	0.75614	0.99987	0.86631	2.00E+04	0
	CSMO³	**100**	**0.749994**	**0.877598**	**0.76582**	**17966.9**	**23.3**
g12	SMO	**100**	**−1.00000**	**−1.00000**	**−1.00000**	**49.9**	**100**
	AsSMO	**100**	**−1.00000**	**−1.00000**	**−1.00000**	51	**100**
	FPSMO	**100**	−0.99999	−0.83971	−0.97409	2.00E+04	0
	CSMO³	**100**	**−1.00000**	**−1.00000**	**−1.00000**	51.7	**100**
g13	SMO	96.7	0.36647	V1	V0.033	2.00E+04	0
	AsSMO	93.3	**0.20427**	V1	V0.067	2.00E+04	0
	FPSMO	0	V2	V3	V2.93	2.00E+04	0
	CSMO³	**100**	0.53114	**2.21803**	**0.97086**	2.00E+04	0
g14	SMO	**100**	−45.69618	**−40.75426**	**−43.16372**	2.00E+04	0
	AsSMO	**100**	**−45.96364**	−39.80034	−42.75604	2.00E+04	0
	FPSMO	0	V3	V3	V3	2.00E+04	0
	CSMO³	**100**	−45.04055	−39.46251	−42.35213	2.00E+04	0
g15	SMO	**100**	961.71536	972.00076	964.92695	2.00E+04	0
	AsSMO	**100**	961.74193	**970.71708**	964.26395	2.00E+04	0
	FPSMO	33.3	962.91087	V2	V1.27	2.00E+04	0
	CSMO³	**100**	**961.71521**	971.79005	**964.21167**	2.00E+04	0
	SMO	**100**	−1.90515447	**−1.90515426**	−1.90515432	534.7	**100**

(Continued)

Table 4 (continued)

Func.	Algorithm	FR (%)	BS	WS	AV	AIR	SR (%)
g16	AsSMO	**100**	−1.90515447	**−1.90515426**	−1.90515433	471.4	**100**
	FPSMO	26.7	−1.53373	V1	V0.73	2.00E+04	0
	CSMO³	**100**	**−1.90515464**	**−1.90515426**	**−1.90515436**	**383.6**	**100**
g17	SMO	50	**8871.60597**	**V2**	V0.63	2.00E+04	0
	AsSMO	63.3	8878.58122	**V2**	V0.5	2.00E+04	0
	FPSMO	0	V4	V4	V4	2.00E+04	0
	CSMO³	**93.3**	8892.95779	V3	**V0.2**	2.00E+04	0
g18	SMO	**100**	−0.86601	−0.67194	−0.78878	2.00E+04	16.7
	AsSMO	**100**	**−0.86602**	−0.67347	−0.82717	2.00E+04	**30**
	FPSMO	6.7	−0.53208	V8	V5.43	2.00E+04	0
	CSMO³	**100**	**−0.86602**	**−0.67463**	**−0.85793**	2.00E+04	**30**
g19	SMO	**100**	33.84292	38.87246	35.26715	2.00E+04	0
	AsSMO	**100**	33.68172	37.19281	35.08186	2.00E+04	0
	FPSMO	**100**	1402.22812	5091.41985	2749.60765	2.00E+04	0
	CSMO³	**100**	**33.38158**	**36.38671**	**34.60199**	2.00E+04	0
g20	SMO	0	V10	V18	V13.3	2.00E+04	0
	AsSMO	0	**V9**	**V18**	**V13.1**	2.00E+04	0
	FPSMO	0	V19	V20	V19.9	2.00E+04	0
	CSMO³	0	V12	V19	V16.8	2.00E+04	0
g21	SMO	0	V1	**V4**	V2.27	2.00E+04	0
	AsSMO	0	V1	**V4**	V2.73	2.00E+04	0
	FPSMO	0	V3	V5	V4.6	2.00E+04	0
	CSMO³	**10**	**290.75339**	**V4**	**V1.53**	2.00E+04	0
g22	SMO	0	V15	**V19**	V17.43	2.00E+04	0
	AsSMO	0	V16	**V19**	V18.1	2.00E+04	0
	FPSMO	0	V19	**V19**	V19	2.00E+04	0
	CSMO³	0	**V8**	**V19**	**V12.97**	2.00E+04	0
g23	SMO	0	V1	**V4**	V2.73	2.00E+04	0
	AsSMO	0	V2	**V4**	V2.97	2.00E+04	0
	FPSMO	0	V2	V5	V3.77	2.00E+04	0
	CSMO³	**23.3**	**−68.57482**	**V4**	**V1.40**	2.00E+04	0
g24	SMO	**100**	**−5.50801**	**−5.50801**	**−5.50801**	**183.8**	**100**
	AsSMO	**100**	**−5.50801**	**−5.50801**	**−5.50801**	193.7	**100**
	FPSMO	**100**	−5.47611	−4.66801	−5.0939	2.00E+04	0
	CSMO³	**100**	**−5.50801**	**−5.50801**	**−5.50801**	187.3	**100**

For function g01, the algorithm proposed in this paper can obtain the known optimal solution in each solution and its success rate is 100%. The success rates of SMO, AsSMO, and FPSMO algorithms are 86.7%, 90%, and 0%, respectively.

For function g02, the feasible solution rate of the four algorithms is 100%, but the success rate of the CSMO^3 algorithm is higher than that of the other three algorithms.

The known optimal solution of g03 is -1.00050. However, none of these four algorithms can obtain the known optimal solution. The feasible solution rate of CSMO^3 algorithms is 100% and the BS, WS, and AV of CSMO^3 algorithms are better than that of the other three algorithms.

For g04, the success rate of the CSMO^3 algorithm is 100% and the BS, WS, AV, and AIR of CSMO^3 algorithms are better than that of the other three algorithms.

The known optimal solution of g05 is 5126.49671. The best solution of SMO, AsSMO, and CSMO^3 are 5126.50462, 5126.54432, and 5126.52999 respectively. These solutions are close to the known optimal solution. For the average solution, AsSMO is better than CSMO^3. The BS of SMO is better than CSMO^3. The WS of CSMO^3 is better than that of SMO, AsSMO, and FPSMO.

For g06, the FR, BS, WS, AV, and SR of SMO, AsSMO, and CSMO^3 are almost equal. The AIR of SMO is better than AsSMO and CSMO^3.

The known optimal solution of g07 is 24.30621. Except for FPSMO, the FR of the other three algorithms is 100%. The best solution of the CSMO^3 algorithm is 24.31870, and the difference between this value and the known optimal solution is only 0.01249. The results show that CSMO^3 is better than the other three algorithms.

For g08, the experiment results of SMO, AsSMO, and CSMO^3 are equal except for the AIR. The AIR of SMO is better than AsSMO and CSMO^3.

The known optimal solution of g09 is 680.63006. The SMO, AsSMO, and CSMO^3 can find the solution, which is almost equal to the known optimal solution, where the best solution of the CSMO^3 algorithm is 680.63022. The BS, WS, and AV of CSMO^3 are better than that of the other three algorithms.

The known optimal solution of g10 is 7049.248021. The best solution of SMO, AsSMO, FPSMO, and CSMO^3 is 7053.26956, 3.52952E+15, 14801.87224, and 7051.77361, respectively, where the BS of the CSMO^3 algorithm is less than the value of other algorithms. In addition, the worst solution of CSMO^3 is 7296.01169, which is better than that of the other algorithms.

For g11, all the algorithms can find a feasible solution in each solution. However, the SR of these algorithms is the smallest. The SR of CSMO^3 is 23.3% and it is better than the BS of the other algorithms.

For g12, the experiment results of SMO, AsSMO, and CSMO^3 algorithms are equal except for the AIR. The AIR of SMO, AsSMO, and CSMO^3 are 49.9, 51, and 51.7, respectively.

The known optimal solution of g13 is 0.053942. None of these four algorithms can obtain the known optimal solution. The feasible solution rate of SMO, AsSMO, FPSMO, and CSMO^3 is 96.7%, 93.3%, 0%, and 100%, respectively. CSMO^3 is the only algorithm that can find a feasible solution in each run.

The known optimal solution of g14 is -47.76489. The feasible solution rates of SMO, As SMO, FPSMO, and CSMO^3 are 100%, 100%, 0%, and 100%, respectively, but these four algorithms cannot obtain the known optimal solution. The BS, WS, and AV of CSMO^3 are worse than those of SMO and AsSMO.

The known optimal solution of g15 is 961.71502. The best solution of CSMO^3 is 961.71521, and the difference between this value and the known optimal solution is 0.00019. The WS of AsSMO is better than the WS of CSMO^3, but the BS and AV of CSMO^3 are better than the other algorithms.

For function g16, the success rate of CSMO^3 is 100% and the BS, WS, AV, and AIR of CSMO^3 are better than that of the other three algorithms.

The known optimal solution of g17 is 8853.53967. None of these four algorithms can obtain the known optimal solution. The feasible solution rates of SMO, AsSMO, FPSMO, and CSMO^3 algorithms are 50%, 63.3%, 0%, and 93.3%, respectively. The average number of constraint violations of CSMO^3 is 0.2, which is better than the other algorithms. The BS of the SMO algorithm is 8871.60597. This value is better than the BS of CSMO^3.

For g18, the FR, BS, WS, AV, and SR of CSMO^3 are equal to or better than the other algorithms.

The known optimal solution of g19 is 32.65559. None of these four algorithms can obtain the known optimal solution, but these algorithms can all find a feasible solution. The BS, WS, and AV of the CSMO^3 algorithm are better than SMO, AsSMO, and FPSMO.

A feasible solution for g20 is not found so far. AsSMO can find solutions that violate 9 constraints and the average number of constraint violations of AsSMO is 13.1. In this function, the performance of AsSMO is better than SMO, FPSMO, and CSMO^3.

The known optimal solution of g21 is 193.72451. SMO, AsSMO, and FPSMO cannot find feasible solutions. The FR of CSMO^3 is 10% and the best solution of CSMO^3 is 290.75339. The performance of the CSMO^3 algorithm is better than that of the other algorithms.

The known optimal solution of g22 is 236.43098. None of these four algorithms can find a feasible solution. CSMO^3 algorithm can find solutions that violate 8 constraints, and the average number of constraint violations is 12.97. In this function, the performance of CSMO^3 is better than the other algorithms.

The known optimal solution of g23 is −400.05510. SMO, AsSMO, and FPSMO cannot find feasible solutions. The FR of CSMO^3 is 23.3% and the best solution of CSMO^3 is −68.57482. The performance of CSMO^3 is better than other algorithms in function g23.

For function g24, the experiment results of SMO, AsSMO, and CSMO^3 are equal except for the AIR. The AIR of SMO, AsSMO, and CSMO^3 are 183.8, 193.7, and 187.3, respectively.

Experiment results show that the proposed algorithm in this paper is better than SMO, AsSMO, and FPSMO. The FR of CSMO^3 in 24 test functions is equal to or better than the other three algorithms. It is noted that the FR of CSMO^3 in g13, g17, g21 and g23 are 100%, 93.3%, 10%, and 23.3%, respectively. These data are better than the FR of SMO, AsSMO, and FPSMO. Moreover, the SR of the CSMO^3 algorithm is not worse than the other three algorithms in each test function. For the BS of CSMO^3, the BS of five functions is equal to the other algorithms, the BS of 13 functions is better than the other algorithms and only the BS of six functions is worse than other algorithms. In the WS of CSMO^3, the WS of eight functions is equal to other algorithms, the WS of 10 functions is better than other algorithms, and only the WS of six functions is worse than the other algorithms. For the AV of CSMO^3, the AV of three functions is equal to other algorithms, the AV of 17 functions is better than other algorithms, and only the AV of four functions is worse than other algorithms.

5.3 Experiments on Engineering Optimization Problems

To further verify the effectiveness of the CSMO^3 algorithm, the spring tension design problem and the pressure pipe design problem are considered. The optimal solution of the CSMO^3 algorithm is compared with the solution results of related algorithms in [40,41].

5.3.1 Spring Tension Design

The optimization of spring tension design is to minimize the weight, when the three decision variables of the spring coil diameter, the average diameter of the spring coil, and the number of coils need to meet a set of constraints. The mathematical model of this problem is described by Eq. (19).

$$\min f(x) = x_1^2 x_2 (2 + x_3)$$

$$s.t.\begin{cases} g_1(x) = 1 - \dfrac{x_3 x_2^3}{71785 x_1^4} \le 0 \\ g_2(x) = \dfrac{4x_2^2 - x_1 x_2}{12556\left(x_1^3 x_2 - x_1^4\right)} + \dfrac{1}{5108 x_1^2} - 1 \le 0 \\ g_3(x) = 1 - \dfrac{140.45 x_1}{x_2^3 x_3} \le 0 \\ g_4(x) = \dfrac{x_1 + x_2}{1.5} - 1 \le 0 \\ 0.05 \le x_1 \le 2 \\ 0.25 \le x_2 \le 1.3 \\ 2 \le x_3 \le 15 \end{cases} \tag{19}$$

where the coil diameter is denoted by x_1, the average diameter of coils is denoted by x_2, and the number of coils is denoted by x_3.

The comparison of solution results is shown in Table 5. The results show that the optimal solution of the CSMO^3 algorithm proposed in this paper is 0.012665153339, which is better than the optimal solution of those algorithms used to compare with the flower pollination algorithm based on the gravitational search mechanism (GSFPA) presented in [40]. The result of the CSMO^3 algorithm is only worse than GSFPA.

Table 5: Comparison of optimal solutions for spring tension design

Algorithm	x1(d)	x2(D)	x3(P)	f(x)
CDE	0.051609	0.354714	11.410831	0.0126702
HPSO	0.051706	0.357126	11.265083	0.0126652
AATM	0.051813	0.35969	11.119252	0.0126682
SCA	0.05216	0.368158	10.648442	0.0126692
FPA	0.052277278	0.371203525	10.49087566	0.012673971
GSFPA	0.051413617	0.350365659	11.65914645	0.012659429
CSMO^3	**0.051807**	**0.35957**	**11.1233986**	**0.012665153**

5.3.2 Pressure Pipe Design

Optimization of pressure pipe design is to minimize the costs when four decision variables of cylindrical pipe thickness, hemispherical pipe thickness, cylindrical pipe inner diameter, and cylindrical pipe length must meet a few constraints. The mathematical model of this problem is as follows:

$$\min f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

$$s.t.\begin{cases} g_1(x) = -x_1 + 0.0193x_3 \le 0 \\ g_2(x) = -x_2 + 0.00954x_3 \le 0 \\ g_3(x) = -\pi x_3^2x_4 - \frac{4\pi x_3^3}{3} + 1296000 \le 0 \\ g_4(x) = -240 + x_4 \le 0 \\ 0 \le x_1, x_2 \le 100 \\ 10 \le x_3, x_4 \le 200 \end{cases} \tag{20}$$

where cylindrical pipe thickness is denoted by x_1, the hemispherical pipe thickness is denoted by x_2, the cylindrical pipe inner diameter is denoted by $x_{3,}$ and the cylindrical pipe inner diameter is denoted by x_4.

The comparison of solution results is shown in Table 6. The optimal solution of the CSMO^3 algorithm is 5884.8205, which is 31.6566 smaller than the optimal solution 5916.4771 of the GSFPA algorithm [40] and is better than the optimal solution of those heuristic algorithms used to compare with GSFPA presented in [40]. Thus, it can be found that the CSMO^3 algorithm proposed in this paper is also effective in dealing with engineering optimization problems.

Table 6: Comparison of optimal solutions for pressure pipe design problems

Algorithm	x1(TS)	x2(Th)	x3(R)	x4(L)	f(x)
CPSO	0.8125	0.4375	42.0913	176.7465	6061.0777
GenAS	1.125	0.625	47.7	117.701	8129.8
HPSO	0.8125	0.4375	42.0984	176.6366	6059.7143
SPA	0.8125	0.4375	40.3239	200	6288.7445
FPA	0.7957	0.3976	41.2243	187.8048	5929.6933
GSFPA	0.7863	0.3924	40.7214	194.6374	5916.4771
CSMO^3	**0.7782**	**0.3847**	**40.3212**	**199.9779**	**5884.8205**

5.3.3 Parameter Estimation for Frequency-Modulated (FM) Sound Waves

The mathematical model of this problem is shown in [41]. There are six dimensions to optimize the FM synthesizer parameter. This issue is highly complex and multimodal, with strong episodic nature. In theory, the function value of the optimal solution to this problem is equal to zero.

Inspired by the Gorilla group and their social way of life in nature, a new metaheuristic algorithm called Artificial Gorilla Troops Optimizer (GTO) has been developed [41]. In optimizing the FM synthesizer parameter, a GTO can find high-quality solutions. The optimal solution of GTO is [−1.0000, −5.0000, 1.5000, 4.8000, 2.000, 4.9000], and the function value is 2.2811E−27. By using the CSMO^3 algorithm to solve the problem, another optimal solution [0.9999, 5.0000, 1.5000, −4.7999, −2.0000, 4.8999] can be obtained, corresponding to the function value of 2.3583E−17. The optimization results of engineering problems show that the algorithm proposed in this paper performs also better than those heuristic algorithms used to compare with GTO presented in [41].

6 Conclusions and Future Work

This paper developed an improved method for the spider monkey algorithm. Because the position of a spider monkey determines the solution, how to update the position of the spider monkey plays a crucial role in problem-solving. A new updating method based on historical optimal domains and particle swarm was developed and population diversity can be improved. Also, this paper applied the OBL strategy to the traditional spider monkey algorithm. The proposed method can increase the individual diversity in the iterative process to avoid prematurely falling into the local optima. Furthermore, A method to eliminate the worst individuals in each group of the SMO algorithm based on the orthogonal experimental design is developed. The experiments on the classical unconstrained functions, constrained functions of the CEC2006 benchmark set, and engineering examples show that the optimization ability of the proposed algorithm is significantly better than other SMO and some evolution algorithms.

The method proposed in this paper has good performance in solving continuous function optimization problems. However, the spider monkey position update method cannot effectively describe the change of the solution for a combinatorial optimization problem, so the algorithm cannot effectively solve a combinatorial optimization problem. In the future, one direction worth exploring is how to build a spider monkey position update method suitable for combinatorial optimization problems, and then improve the ability of SMO to solve combinatorial optimization problems.

Acknowledgement: We would like to thank the editor and reviewers for their valuable comments, which are very helpful in improving the quality of the paper.

Funding Statement: This research was supported by the First Batch of Teaching Reform Projects of Zhejiang Higher Education "14th Five-Year Plan" (jg20220434), Special Scientific Research Project for Space Debris and Near-Earth Asteroid Defense (KJSP2020020202), Natural Science Foundation of Zhejiang Province (LGG19F030010), and National Natural Science Foundation of China (61703183).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: W. Liao, X. Xia, H. Zhuang; data collection: X. Jia, X. Zhang; analysis and interpretation of results: X. Jia, S. Shen, H. Zhuang; draft manuscript preparation: X. Xia, S. Shen, H. Zhuang. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data will be made available on request from the corresponding authors.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. Feng, L. Zhao, H. Shi, M. Wang, S. Shen *et al.,* "One-dimensional VGGNet for high-dimensional data," *Applied Soft Computing*, vol. 135, pp. 110035, 2023.

[2] Y. Shen, S. Shen, Z. Wu, H. Zhou and S. Yu, "Signaling game-based availability assessment for edge computing-assisted IoT systems with malware dissemination," *Journal of Information Security and Applications*, vol. 66, pp. 103140, 2022.

[3] L. Kong, G. Li, W. Rafique, S. Shen, Q. He *et al.,* "Time-aware missing healthcare data prediction based on ARIMA model," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Early Access, 2022. https://doi.org/10.1109/TCBB.2022.3205064

[4] G. Wu, H. Wang, H. Zhang, Y. Zhao, S. Yu *et al.,* "Computation offloading method using stochastic games for software defined network-based multi-agent mobile edge computing," *IEEE Internet of Things Journal*, 2023. https://doi.org/10.1109/JIOT.2023.3277541

[5] H. Ma, S. Shen, M. Yu, Z. Yang, M. Fei *et al.,* "Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey," *Swarm and Evolutionary Computation*, vol. 44, pp. 365–387, 2019.

[6] A. Rahman, R. Sokkalingam, O. Mahmod and K. Biswas, "Nature-Inspired metaheuristic techniques for combinatorial optimization problems: Overview and recent advances," *Mathematics*, vol. 9, pp. 2633, 2021.

[7] H. Shi, S. Liu, H. Wu, R. Li, S. Liu *et al.,* "Oscillatory particle swarm optimizer," *Applied Soft Computing*, vol. 73, pp. 316–327, 2018.

[8] J. Liu, Y. Wang, G. Sun and T. Pang, "Multisurrogate-assisted ant colony optimization for expensive optimization problems with continuous and categorical variables," *IEEE Transactions on Cybernetics*, vol. 53, no. 11, pp. 11348–11361, 2022.

[9] Y. Xu and X. Wang, "An artificial bee colony algorithm for scheduling call centres with weekend-off fairness," *Applied Soft Computing*, vol. 109, pp. 107542, 2021.

[10] G. Chen, J. Qian, Z. Zhang and S. Li, "Application of modified pigeon-inspired optimization algorithm and constraint-objective sorting rule on multi-objective optimal power flow problem," *Applied Soft Computing*, vol. 92, pp. 106321, 2020.

[11] F. S. Gharehchopogh, A. Ucan, T. Ibrikci, B. Arasteh and G. Isik, "Slime mould algorithm: A comprehensive survey of its variants and applications," *Computational Methods in Engineering*, vol. 30, pp. 2683–2723, 2023.

[12] A. Saxena, "An efficient harmonic estimator design based on augmented crow search algorithm in noisy environment," *Expert Systems with Applications*, vol. 194, pp. 116470, 2022.

[13] J. C. Bansal, H. Sharma, S. S. Jadon and M. Clerc, "Spider monkey optimization algorithm for numerical optimization," *Memetic Computing*, vol. 6, no. 1, pp. 31–47, 2014.

[14] S. Kumar, V. K. Shamar and R. Kumari, "Modified position update in spider monkey optimization algorithm," *International Journal of Emerging Technologies in Computational and Applied Sciences*, vol. 7, no. 2, pp. 198–204, 2014.

[15] S. Kumar, V. Kumar and R. Kumari, "Fitness based position update in spider monkey optimization algorithm," *Procedia Computer Science*, vol. 62, pp. 442–449, 2015.

[16] A. Sharma, A. Sharma, B. K. Panigrahi, D. Kiran and R. Kuma, "Ageist spider monkey optimization algorithm," *Swarm and Evolutionary Computation*, vol. 28, pp. 58–77, 2016.

[17] G. Hazrati, H. Sharma and N. Sharma, "Adaptive step-size based spider monkey optimization," in *2016 IEEE Int. Conf. on Power Electronics, Intelligence Control and Energy Systems*, Delhi, India, pp. 736–740, 2016.

[18] K. Gupta, K. Deep and J. C. Bansal, "Spider monkey optimization algorithm for constrained optimization problems," *Soft Computing: A Fusion of Foundations, Methodologies and Applications*, vol. 21, no. 23, pp. 6933–6962, 2017.

[19] K. Gupta, K. Deep and J. C. Bansal, "Improving the local search ability of spider monkey optimization algorithm using quadratic approximation for unconstrained optimization," *Computing Intelligence*, vol. 33, no. 2, pp. 210–240, 2017.

[20] A. Sharam, H. Sharam, A. Bhargava and N. Sharma, "Power law-based local search in spider monkey optimization for lower order system modeling," *International Journal of System Science*, vol. 48, pp. 150–160, 2017.

[21] X. Xia, W. Liao, Y. Zhang and X. Peng, "A discrete spider monkey optimization for the vehicle routing problem with stochastic demands," *Applied Soft Computing*, vol. 111, pp. 107676, 2021.

[22] V. Agrawal, R. Rstogi and D. C. Tiwari, "Spider monkey optimization: A survey," *International Journal of System Assurance Engineering & Management*, vol. 9, no. 4, pp. 929–941, 2018.

[23] N. Mittal, U. Singh, R. Salgotra and B. S. Sohi, "A boolean spider monkey optimization based energy efficient clustering approach for WSNs," *Wireless Networks*, vol. 24, no. 6, pp. 2093–2109, 2018.

[24] U. Singh and R. Salgotra, "Optimal synthesis of linear antenna arrays using modified spider monkey optimization," *Arabian of Science and Engineering*, vol. 41, no. 8, pp. 2957–2973, 2016.

[25] A. Bhargava, N. Sharma, A. Sharma and H. Sharma, "Optimal design of PIDA controller for induction motor using spider monkey optimization algorithm," *International Journal of Metaheuristics*, vol. 5, no. 3, pp. 278–290, 2016.

[26] R. Cheruku, D. Edla and V. Kuppili, "SM-rule miner: Spider monkey based rule miner using novel fitness function for diabetes classification," *Computers in Biology and Medicine*, vol. 81, no. 4, pp. 79–92, 2017.

[27] J. S. Priya, N. Bhaskar and S. Prabakeran, "Fuzzy with black widow and spider monkey optimization for privacy-preserving-based crowdsourcing system," *Soft Computing*, vol. 25, no. 7, pp. 5831–5846, 2021.

[28] N. Darapureddy, N. Karatapu and T. K. Battula, "Optimal weighted hybrid pattern for content based medical image retrieval using modified spider monkey optimization," *International Journal of Imaging Systems and Technology*, vol. 31, no. 2, pp. 828–853, 2021.

[29] M. R. Sivagar and N. Prabakaran, "Elite opposition based metaheuristic framework for load balancing in LTE network," *Computers, Materials & Continua*, vol. 71, no. 3, pp. 5765–5781, 2022.

[30] N. Rizvi, R. Dharavath and D. R. Edla, "Cost and makespan aware workflow scheduling in IaaS clouds using hybrid spider monkey optimization," *Simulation Modelling Practice and Theory*, vol. 110, pp. 102328, 2021.

[31] R. U. Mageswari, S. A. Althubiti, F. Alenezi, E. L. Lydia, G. P. Joshi *et al.*, "Enhanced metaheuristics-based clustering scheme for wireless multimedia sensor networks," *Computers, Materials & Continua*, vol. 73, no. 2, pp. 4179–4192, 2022.

[32] Y. Shen, S. Shen, Q. Li, H. Zhou, Z. Wu *et al.*, "Evolutionary privacy-preserving learning strategies for edge-based IoT data sharing schemes," *Digital Communications and Networks*, 2022. https://doi.org/10.1016/j.dcan.2022.05.004

[33] P. Sun, S. Shen, Z. Wu, H. Zhou and X. Gao, "Stimulating trust cooperation in edge services: An evolutionary tripartite game," *Engineering Applications of Artificial Intelligence*, vol. 116, pp. 105465, 2022.

[34] G. Wu, Z. Xu, H. Zhang, S. Shen and S. Yu, "Multi-agent DRL for joint completion delay and energy consumption with queuing theory in MEC-based IIoT," *Journal of Parallel and Distributed Computing*, vol. 176, pp. 80–94, 2023.

[35] X. Zhou, Z. Wu, H. Wang, K. Li and H. Zhang, "Elite opposition-based particle swarm optimization," *Chinese Journal of Electronics*, vol. 41, no. 8, pp. 1647–1652, 2013.

[36] H. Wu, "Application of orthogonal experimental design for the automatic software testing," *Applied Mechanics & Materials*, vol. 347, pp. 812–818, 2013.

[37] X. Zhou, Z. Wu and M. Wang, "Artificial bee colony algorithm based on orthogonal experimental design," *Chinese Journal of Software*, vol. 26, no. 9, pp. 2167–2190, 2015.

[38] S. Shen, X. Wu, P. Sun, H. Zhou, Z. Wu *et al.*, "Optimal privacy preservation strategies with signaling Q-learning for edge-computing-based IoT resource grant systems," *Expert Systems with Applications*, vol. 225, pp. 120192, 2023.

[39] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc and K. Deb, "Problem definitions and evaluation criteria for the CEC2006 special session on constrained real-parameter optimization," Technical Report, Singapore: Nanyang Technological University.

[40] H. Xiao, C. Wan, Y. Duan and Q. Tan, "Flower pollination algorithm based on gravity search mechanism," *Acta Automatica Sinica*, vol. 43, no. 4, pp. 576–594, 2017.

[41] B. Abdollahzadeh, F. S. Gharehchopogh and S. Mirjalili, "Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems," *Intelligent Systems*, vol. 36, no. 10, pp. 5887–5958, 2021.