symmetry

MDPI

Article

A Modified jSO Algorithm for Solving Constrained Engineering Problems

Yong Shen, Ziyuan Liang, Hongwei Kang *, Xingping Sun and Qingyi Chen

School of Software, Yunnan University, Kunming 650000, China; sheny@ynu.edu.cn (Y.S.); tsbshs@mail.ynu.edu.cn (Z.L.); sunxp@ynu.edu.cn (X.S.); devas9@ynu.edu.cn (Q.C.)
* Correspondence: hwkang@ynu.edu.cn

Abstract: Proposing new strategies to improve the optimization performance of differential evolution (DE) is an important research study. The jSO algorithm was the announced winner of the Congress on Evolutionary Computation (CEC) 2017 competition on numerical optimization, and is the state-of-the-art algorithm in the SHADE (Success-History based Adaptive Differential Evolution) algorithm series. However, the jSO algorithm converges prematurely in the search space with different dimensions and is prone to falling into local optimum during evolution, as well as the problem of decreasing population diversity. In this paper, a modified jSO algorithm (MjSO) is proposed which is based on cosine similarity with parameter adaptation and a novel opposition-based learning restart mechanism incorporated with symmetry to address the above problems, respectively. Moreover, it is well known that parameter setting has a significant impact on the performance of the algorithm and the search process can be divided into two symmetrical parts. Hence, a parameter control strategy based on a symmetric search process is introduced in the MjSO. The effectiveness of these designs is supported by presenting a population clustering analysis, along with a population diversity measure to evaluate the performance of the proposed algorithm, three state-of-the-art DE variant algorithms (EBLSHADE, ELSHADE-SPACMA, and SALSHADE-cnEPSin) and two original algorithms (jSO and LSHADE) are compared with it, for solving 30 CEC'17 benchmark functions and three classical engineering design problems. The experimental results and analysis reveal that the proposed algorithm can outperform other competitions in terms of the convergence speed and the quality of solutions. Promisingly, the proposed method can be treated as an effective and efficient auxiliary tool for more complex optimization models and scenarios.

Keywords: differential evolution; cosine similarity; opposition-based learning; parameter adaptive; symmetric search; constrained engineering problems

check for updates

Citation: Shen, Y.; Liang, Z.; Kang, H.; Sun, X.; Chen, Q. A Modified jSO Algorithm for Solving Constrained Engineering Problems. *Symmetry* **2021**, *13*, 63. https://doi.org/10.3390/sym13010063

Received: 14 December 2020
Accepted: 29 December 2020
Published: 31 December 2020

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In the real world, there are many engineering optimization problems that need to be solved. The design cost problems in engineering have some practical constraints and one or more objective functions [1]. These engineering problems often demand methods to find optimum from a large number of available solutions, without wasting efforts in searching sub-optimal regions. Because of the large number of these problems and their practical application value, it has become a research hotspot in recent years. These problems are characterized mainly by their large scale and high difficulty. When the traditional optimization algorithm solves such a large-scale problem, it has certain inherent drawbacks, such as local optimal stagnation and derivation of the search space. Therefore, much attention has been paid to stochastic optimization methods [2] in recent decades [3,4]. Stochastic optimization algorithms can avoid the derivation of the mathematical model in that they only adjust the inputs and monitor the outputs of a given system for objective outputs and treat the optimization task as a black box [5]. In addition, stochastic optimization algorithms can solve optimization tasks randomly [6], which means that they have intrinsic superiority

over conventional optimization algorithms in the capability of local optimal avoidance. Based on the number of solutions generated at each iteration optimization during the whole process, stochastic optimization algorithms can be divided into population-based algorithms, which generate many random solutions and promote them during the full optimization, and individual-based algorithms, which generate only one candidate solution throughout optimization.

Numerous stochastic optimization algorithms have been proposed based on evolutionary phenomena, swarm-based intelligence techniques, physical rules, and human-related concepts. Evolutionary algorithms include genetic algorithm (GA) [7], particle swarm optimization (PSO) [8], differential evolution (DE) [9], and evolutionary strategy (ES) [10]. Swarm-based intelligence algorithms usually simulate living beings' behavior of social hierarchy, cooperation, and collective, and then mathematically abstract optimization models for problems to be optimized.

The differential evolution (DE) algorithm was proposed by Storn and Price [9] in 1995, and this laid the foundation for a series of successful continuous optimization algorithms. Literature [11] summarizes the extensive research fields focused on DE in recent years. The algorithms based on DE [12–26] have gained very high rankings in various competitions held annually at the IEEE (Institute of Electrical and Electronics Engineers) Congress on Evolutionary Computation (CEC). Therefore, DE-based algorithms are generally considered to be an effective and popular population-based evolutionary algorithm for single-objective continuous optimization problems [27,28].

In recent years, many modified algorithms based on DE algorithms have been derived [29]. In addition to combining DE with other algorithms, the main research modifications of DE include the exploitation of mutation strategies to control the movement of individuals and the adaptive modification of the DE control parameters during operation. SHADE is an adaptive DE that combines parameter adaptation based on the success history with one of the most advanced DE algorithms [30]. LSHADE further expands SHADE through linear population size reduction (LPSR), thus continuously reducing the population size using linear functions [31]. The jSO algorithm, which mainly has a new weighted version of the mutation strategy and some parameter settings, is an improved variant of the LSHADE algorithm [32]. ELSHADE-SPACMA is an improved method based on LSHADE-SPACMA. In LSHADE-SPACMA, the value of p used to control the degree of the greed of the mutant strategy is constant, while in ELSHADE-SPACMA, the value of p is dynamic. The performance of ELSHADE-SPACMA was further improved by integrating another directed mutation strategy into the hybridization framework [33,34]. The improved SALSHADE algorithm with global optimization linearly reduces the scaling factor and exponentially reduces the crossover rate and final Levy distribution step size to adapt to the improvement of the frequency component [35]. These classical algorithms will be reviewed in Section 2.

Furthermore, one of the disadvantages of differential evolutionary algorithms is the loss of population diversity. Therefore, the algorithm may prematurely converge to the local optimum. A good algorithm needs to have high enough population diversity in the early stage of the evolutionary process while reducing the population diversity to improve the convergence rate in the late stage. In order to solve the problems of premature convergence and population stagnation, Ming Yang et al. [36] study the adaptability of the population diversity in dimensions and a mechanism called automatically enhanced population diversity (AEPD) was proposed to automatically enhance the population diversity. Cheng Hongtan et al. [37] focus on controlling the value of the search parameter P. In his proposal, after he normalized the diversity of the population, each individual would select its unique search area according to the diversity conditions. Jing-zhong Wang et al. [38] proposed a variation strategy for a differential evolutionary algorithm (DE) with a fuzzy inference system (FIS), which enables the FIS to have well-controlled population diversity performance by controlling the ratio of the variation with respect to the globally optimal individual. All previously mentioned algorithms share the same idea of balancing between

exploration and exploitation abilities, and they try to achieve it in various ways, but there is a general issue that they cannot maintain a long exploration phase or the ability to jump out of local optimum.

Nevertheless, many improved algorithms based on DE algorithms have made great progress in engineering problems. Xiaoyu He et al. [39] proposed a novel DE variant with covariance matrix self-adaptation, which produces a good solution for solving three classic constrained engineering design problems (pressure vessel design problem, tension/compression spring design problem and welded beam design problem). Noor H. Awad et al. [40] introduced an effective surrogate model to assist the differential evolution algorithm to generate competitive solutions during the search process and solved the two engineering design problems of welded beams and pressure vessels. At the same time, the improved algorithm fused with the DE algorithm has also achieved good performance in solving engineering problems. K.R. Subhashini et al. [41] integrated the basic animal migration optimization (AMO) algorithm with the DE algorithm and applied it to engineering problems. The result is significantly better than the original algorithm AMO.

Therefore, this study proposes a modified jSO algorithm to solve engineering problem. The MjSO algorithm uses a novel parameter adaptive mechanism based on cosine similarity instead of the original scaling factor and crossover rate with the weighted mean adaptive mechanism and adopts a parameter control strategy based on a symmetric search process. Thus, it can maintain a high population diversity and a longer exploration phase. In addition, in order to jump out of the local optimum, a novel opposition-based learning restart mechanism is applied. Finally, abundant experiments on 30 unconstrained benchmark functions [42], and 3 constrained engineering problems are carried out to demonstrate the superiority of MjSO algorithm. The experimental results show that the proposed MjSO algorithm can obtain more persuasive optima results than other optimization algorithms on unconstrained benchmark functions and achieve less cost engineering design results than other algorithms on constrained engineering problems.

The rest of this article is arranged as follows: in Section 2, several of the more famous DE variants are reviewed. The MjSO algorithm is described in detail in Section 3. Section 4 describes the experimental setup. Section 5 presents the analysis of the experimental results. The proposed MjSO algorithm for engineering problems is conducted in Section 6. Finally, the conclusion is given in Section 7.

2. Related Work

In this section, several of the state-of-the-art DE variants will be reviewed, including JADE [43], SHADE [30], LSHADE [31], iLSHADE [44], and jSO [32], because our proposed MjSO algorithm is a further extension of these algorithms. In addition, these algorithms provide a clear idea for the exploitation of DE. By introducing these state-of-the-art DE variants, researchers can better understand how the MjSO algorithm works.

2.1. JADE

JADE, proposed by Zhang and Sanderson, was the first algorithm to propose a new experimental vector generation strategy with an optional external archive called "DE/current-to-*p*best/1/bin". By integrating external archives, a good balance was achieved between the population diversity and convergence rate. In addition to the mutation strategy, the control parameters F and Cr in JADE obey a Gaussian distribution and Cauchy distribution, respectively, and the control parameters PS remain constant throughout the whole evolution process. In terms of the adaptation of control parameters, JADE further extends the innovation of jDE that "better control parameters lead to better individuals and therefore these control parameters should be retained in the next generation" by using these better control parameters to update their distribution.

2.2. SHADE

The SHADE algorithm was modified based on JADE; therefore, they have many of the same mechanisms, and their main differences are historical memory M_F and M_{CR} and their updating mechanisms. SHADE is very representative of the state-of-the-art DE variants, and how it works is described in detail below. The basic steps for SHADE are as follows:

Step 1. The initial population P consists of randomly generated solution vectors. The solution vector X is generated according to the lower and upper bounds of the solution space:

$$P = (x_1, \cdots, x_i, \cdots, x_n); i = 1, \cdots, NP \tag{1}$$

$$x_{i,j} = rand[x_{min}, x_{max}]; \forall i = 1, \cdots, NP; \forall j = 1, \cdots, D \tag{2}$$

where i represents the individual, j represents the dimension, and $rand()$ is uniformly random. D denotes the dimension of the problem, NP is the population size, and $X_{min} = \left(x^1_{min}, \ldots, x^D_{min}\right)$ and $X_{max} = \left(x^1_{max}, \ldots, x^D_{max}\right)$ are the lower and upper bounds of the search range, respectively.

Step 2. There are two additional components in SHADE, the historical memory and the external archive, which also need to be initialized. The control parameter scaling factor and crossover rate stored in the historical memory are initialized to 0.5.

$$M_{CR,i} = M_{F,i} = 0.5; \forall i = 1, \cdots, H \tag{3}$$

where H is the capacity of the user-defined historical memory. The index K used for historical memory updates is initialized to 1. In addition, the deposit out solution of the external archive A is initialized to null, $A = \varnothing$.

Step 3. The variation strategy of "current-to-$p$best/1" is used by SHADE. The variation formula is shown in (4):

$$v_{i,G} = x_{i,G} + F_i \cdot \left(x_{pbest,G} - x_{i,G}\right) + F_i \cdot (x_{r1,G} - x_{r2,G}) \tag{4}$$

$$p_i = rand[p_{min}, 0.2] \tag{5}$$

$$p_{min} = \frac{2}{NP} \tag{6}$$

where $\boldsymbol{x}_{i,G}$ is the current individual and $\boldsymbol{x}_{pbest,G}$ is randomly selected from the former $NP \times p_i$ (where $p_i \in [0, 1]$) optimal individuals of thegeneration population. The vectors $\boldsymbol{x}_{r1,G}$ are randomly selected individuals from thegeneration population, $\boldsymbol{x}_{r2,G}$ are randomly selected individuals from the combination of thegeneration population and the external archive A, $r_1 \neq r_2 \neq r_3 \neq i$, F_i is the scaling factor, NP is the population size, $rand()$ is the uniform random distribution, G is the current iteration, and $\boldsymbol{v}_{i,G}$ is the generated variation vector. Regarding "current-to- $p$best/1", the greed of the mutation strategy depends on the control parameters of p_i, and the calculation formula for p_i is as shown in (5) and (6). It balances exploration and exploitation (a small P is greedier). The scaling factor F_i is generated by the following formula:

$$F_i = randc_i(M_{F,ri}, 0.1) \tag{7}$$

$randc_i$ () is a Cauchy distribution, $M_{F,ri}$ is randomly selected from the historical memory M_F, and r_i is a uniformly distributed random value of [1, H]. If the generated $F_i > 1$, then $F_i = 1$; and if $F_i \leq 0$, Equation (7) is repeated to generate effective values.

If a one-dimensional $v_{j,i,G}$ of the variation vector is outside the search range $[x_{min}, x_{max}]$, the following correction is performed, as shown in Formula (8):

$$v_{j,i,G} = \begin{cases} \frac{x_{min} + x_{j,i,G}}{2} \; if \; v_{j,i,G} < x_{min} \\ \frac{x_{max} + x_{j,i,G}}{2} \; if \; v_{j,i,G} > x_{max} \end{cases} \tag{8}$$

where i represents the individual and j represents the dimension.

Step 4. Conduct the SHADE of the crossover operation using binomial crossover to generate trial vector u. The crossover operation formula is as follows:

$$u_{j,i,G} = \begin{cases} v_{i,i,G} \; if \; rand[0,1) \leq CR_i \; or \; j = j_{rand} \\ x_{j,i,G} \qquad\qquad otherwise \end{cases} \tag{9}$$

where i represents the individual, j represents the dimension, *rand* () is a uniform random distribution, and *jrand* is the decision variable index selected from the uniform random distribution of [1, D]. When *rand* [0, 1] is less than or equal to the crossover rate CR_i or $j = jrand$, the dimension of the test vector u inherits the dimension of the variation vector v; otherwise, the dimension of the original vector x is inherited. The purpose of setting $j = jrand$ is to provide a protection mechanism so that at least one dimension of the test vector is inherited from the variation vector.

The crossover rate CR_i is generated using the following formula:

$$CR_i = randn_i(M_{CR,ri}, 0.1) \tag{10}$$

where $randn_i$ () is a Gaussian distribution, $M_{CR,ri}$ is randomly selected from historical memory M_{CR}, and r_i is a uniformly distributed random value of [1, H]. If the generated $CR_i > 1$, then let $CR_i = 1$; and if $CR_i < 0$, then let $CR_i = 0$.

Step 5. SHADE follows some selection steps to ensure that the optimization will be in the direction of a better solution because it only allows better objective function values or values that at least equal to the next generation of the surviving individual. The selection operation formula is as follows:

$$x_{i,G+1} = \begin{cases} u_{i,G} \;\; if \; f(u_{i,G}) \leq f(x_{i,G}) \\ x_{i,G} \qquad\qquad otherwise \end{cases} \tag{11}$$

where f () represents the fitness function, G represents the current generation, and $G + 1$ represents the next generation.

SHADE also needs to update the external archive during the selection process. If a trial individual better than the original individual is generated, the original individual $\boldsymbol{x}_{i,G}$ will be stored in the external archive of elimination solution A. If the external archive exceeds its capacity, one of them will be deleted randomly to make room for the subsequent elimination solution.

Step 6. The renewal of historical memory. The historical memories M_F and M_{CR} are initialized according to Formula (3), but the contents inside will change as the algorithm iterates. These memories store the scaling factor F and crossover rate CR of "success", where "success" means that the trial vector u rather than the original vector x is selected in the selection process to be part of the next generation. In each iteration, these "successful" F and CR values are first stored in the arrays S_F and S_{CR}, respectively. After each iteration, the historical memories of M_F and M_{CR} are updated by one unit. The updated cell is represented by K. It is initialized as 1, 1 is added after each iteration, and it is reset to 1 when K exceeds the memory capacity H. The K_{th} unit of historical memory is updated using the following formula:

$$M_{CR,k,G+1} = \begin{cases} mean_{WA}(S_{CR}) \;\; if \; S_{CR} \neq \varnothing \\ M_{CR,k,G} \qquad\quad otherwise \end{cases} \tag{12}$$

$$M_{F,k,G+1} = \begin{cases} mean_{WL}(S_F) \;\; if \; S_F \neq \varnothing \\ M_{F,k,G} \qquad\quad otherwise \end{cases} \tag{13}$$

When all the individuals in iteration G fail to make a better trial vector than the original one, that is, when $S_F = S_{CR} = \varnothing$, the historical memory does not update. The weighted

average W_A and the weighted Lehmer average W_L are respectively calculated by the following formulas:

$$mean_{WA}(S_{CR}) = \sum_{k=1}^{|S_{CR}|} w_k \cdot S_{CR,k} \tag{14}$$

$$mean_{WL}(S_F) = \frac{\sum_{k=1}^{|S_F|} w_k \cdot S_{F,k}^2}{\sum_{k=1}^{|S_F|} w_k \cdot S_{F,k}} \tag{15}$$

In order to improve the adaptability of parameters, the weight vector w is calculated based on the absolute value of the difference of the objective function value between the trial individual and the original individual in the current generation G, as shown below:

$$\begin{cases} w_k = \frac{\Delta f_k}{\sum_{k=1}^{|S_{CR}|} \Delta f_k} \\ \Delta f_k = \left| f(u_{k,G}) - f(x_{k,G}) \right| \end{cases} \tag{16}$$

Step 7. Repeat Steps 2 to 6 until a stopping criterion is met.

2.3. LSHADE

LSHADE, proposed by Tanabe and Fukunaga, is an improved version of the SHADE algorithm that introduces linear population size reduction (LPSR). In LSHADE, the new population size is calculated after each iteration, as shown in Formula (18). When the new population size NP_{new} is smaller than the current population size NP, the population is sorted according to the value of the objective function, and the worst NP-NP_{new} individuals are discarded. The size of external archive A also decreases as the population size increases.

$$NP_{new} = round\left(NP_{init} - \frac{FES}{MAXFES} * \left(NP_{init} - NP_f \right) \right) \tag{17}$$

NP_{init} is the initial population size, and NP_f is the final population size. FES is the current number of fitness function evaluations, $MAXFES$ is the maximum number of the fitness function evaluations, and $round()$ is the integral function.

2.4. iLSHADE

The iLSHADE algorithm is an improvement on the LSHADE algorithm, and it includes the following modifications:

a. iLSHADE uses a larger M_F = 0.8 in the evolution of the initialization phase and a smaller population size NP_{init}= 12·D.
b. In the iLSHADE algorithm, the last entry in the H-entry pool records constant control parameter pairs, which are M_F = 0.9 and M_{CR} = 0.9, respectively. These two parameters remain unchanged throughout evolution.
c. At different stages of the evolution, the F and CR of each individual are set to different fixed values, as shown in Equations (18) and (19).
d. The value of the degree of greed control parameter P of the variation strategy in iLSHADE increases linearly as the number of fitness function evaluations increases (see Equation (20)).

$$F_i = \begin{cases} min(F_i, 0.7) & if\ FES < 0.25 * MAXFES \\ min(F_i, 0.8) & if\ FES < 0.5 * MAXFES \\ min(F_i, 0.9) & if\ FES < 0.75 * MAXFES \end{cases} \tag{18}$$

$$Cr_i = \begin{cases} min(Cr_i, 0.5) & if\ FES < 0.25 * MAXFES \\ min(Cr_i, 0.25) & if\ FES < 0.5 * MAXFES \end{cases} \tag{19}$$

$$p = p_{min} + \frac{FES}{MAXFES}(p_{max} - p_{min}) \tag{20}$$

2.5. jSO

The jSO algorithm is an improved version of the iLSHADE algorithm, and it won the CEC 2017 single objective real parameter optimization competition [42]. In the jSO algorithm, p_{max} = 0.25, $p_{min} = p_{max}/2$, the initial population size $NP_{init} = 25\sqrt{D}\log D$, and the historical memory capacity H = 5. In addition, all parameter values in the historical memory M_F and M_{CR} are set to 0.3 and 0.8, respectively, and the weighted current mutation strategy current-to-*p*Best-w/1 was used.

$$v_{i,G} = x_{i,G} + F_w \cdot \left(x_{pbest,G} - x_{i,G}\right) + F_i \cdot (x_{r1,G} - x_{r2,G}) \tag{21}$$

F_w is calculated by the following formula:

$$F_w = \begin{cases} 0.7F_i & FES < 0.2MAXFES \\ 0.8F_i & FES < 0.4MAXFES \\ 1.2F_i & otherwise \end{cases} \tag{22}$$

The distribution of control parameters F and CR is the same as that of iLSHADE. Unlike iLSHADE, jSO's control parameters F and CR need to be adjusted. These adjustments are shown in Equations (23) and (24).

$$F_i = \begin{cases} F_i & if\ F_i < 0.7 \& FES < 0.6 * MAXFES \\ 0.7, & otherwise \end{cases} \tag{23}$$

$$Cr_i = \begin{cases} 0.7 & if\ Cr_i < 0.7 \& FES < 0.25 * MAXFES \\ 0.6 & if\ Cr_i < 0.6 \& FES \in [0.25, 0.5) * MAXFES \\ Cr_i & otherwise \end{cases} \tag{24}$$

3. MjSO

In this section, a modified algorithm called MjSO is proposed, in which three modifications are made: (1) a parameter control strategy based on a symmetric search process, (2) a novel parameter adaptive mechanism based on cosine similarity, and (3) a novel opposition-based learning restart mechanism. The MjSO pseudocode is shown in Algorithm 2.

3.1. A Parameter Control Strategy Based on a Symmetric Search Process

It is well known that the search process of an effective evolutionary algorithm can be divided into two symmetrical parts: exploration and exploitation. In the process of exploration, individuals face an unknown area to search, but also to avoid falling into the local optimal. On the other hand, in the exploitation process, individuals should search around the current optimal solution to find a better solution. Each problem has its own appropriate parameter values used to better balance exploration and exploitation [45]. Therefore, the symmetric parameter control strategy to the jSO algorithm is added, and the parameter F will be randomly generated using a uniform distribution within a specific range.

The optimization process of the algorithm into two symmetrical stages is divided empirically. When $FES < MAXFES/2$, the scaling factor (F) is generated using Equation (25). When the $FES > MAXFES/2$, the scaling factor (F) uses Equation (7) the Cauchy distribution to generate, where FES is the current number of fitness function evaluations, $MAXFES$ is the largest number of the fitness function evaluations and *rand* () is a random number from 0∼1.

$$F_i = 0.45 + 0.1 * rand() \tag{25}$$

3.2. A Novel Parameter Adaptive Mechanism Based on Cosine Similarity

The original adaptive mechanism for the scaling factor and crossover rate values uses the weighted form of the means in Equations (15) and (16), where the weights are based on the improvement of the objective function value from Equation (17). This approach pro-

motes exploitation rather than exploration; therefore, it can lead to premature convergence, which can be a problem, especially in higher dimensions.

The idea behind the proposed approach is simple. Regardless of whether one wants to improve the search ability of the algorithm or to maintain the search ability of the algorithm in a longer period of time, it is necessary to make individuals explore the search space more intensively and maintain a high population diversity. To do this through any kind of adaptation, it would be beneficial to find a parameter setting that enables an individual to achieve the desired behavior.

By replacing the fitness difference with the cosine similarity, in the original weight calculation in Equation (17), the difference between the objective function values can reach a higher value, especially in the high-dimensional search space [46,47]. However, the cosine similarity method in Equation (26) only considers the direction of the individual within the search space, which greatly reduces the complexity of the calculation.

In this modification, the individual-related scaling factor and crossover ratio values with the greatest directional differences will have the highest weights. This occurs because in n-dimensional space, any vector can be viewed as a directed line segment pointing in different directions from $[0,\ 0,\ 0, \ldots, 0]$, and there are a definitely included angle and corresponding cosine between any two directed line segments. Cosine similarity comparison is a mathematical method used to measure the difference between vectors by using the cosine of the angle between line segments. Cosine similarity measures the angle between vectors in space and is more about the difference in direction than the difference in position, especially when solving higher-dimensional problems. As shown in Figure 1, if the position of point A is kept unchanged and point B is away from the origin of the coordinate axis in the original direction, then the cosine similarity remains unchanged (because the included angle does not change) while the distance between points A and B changes.

$$\begin{cases} w_k = \dfrac{\sum_{j=1}^{D} cos\left(u_{k,j,G}, x_{k,j,G}\right)}{\sum_{m=1}^{|S_{CR}|} cos\left(u_{m,j,G}, x_{m,j,G}\right)} \\ cos(\theta) = \dfrac{\sum_{j=1}^{D} u_{k,j,G} * x_{k,j,G}}{\sqrt{\sum_{j=1}^{D} u_{k,j,G}{}^2} * \sqrt{\sum_{j=1}^{D} x_{k,j,G}{}^2}} \end{cases} \tag{26}$$

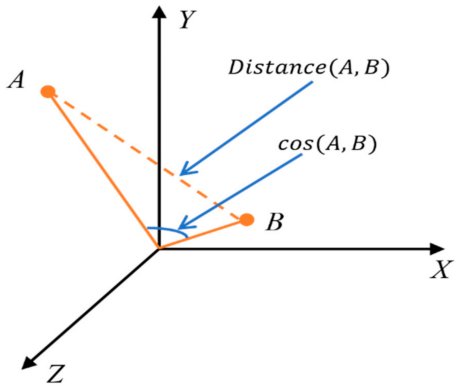


Figure 1. Cosine similarity.

As a result, the ability to explore is rewarded by avoiding premature convergence in higher-dimensional target spaces.

3.3. A Novel Opposition-Based Learning Restart Mechanism

At the time of evolution, individuals in the population may lack diversity and therefore do not update the results of the algorithm. Therefore, if certain conditions are met, the novel opposition-based learning (OBL) restart mechanism is implemented. The pseudocode is shown in Algorithm 1.

Algorithm 1: A novel OBL restart mechanism

1: **if** λ = ξ
2: **for** $i = 1 : NP$ **do**
3: Generate the opposite vector OP_i' using Equation (27)
4: Calculate the fitness value OP_i';
5: $FES++$
6: Replace P_i with a fitter one between P_i and OP_i'
7: **end for**
8: **end if**

In general, the evolutionary algorithm first obtains the feasible solution and then searches for the optimal solution in the solution space according to the fitness of the feasible solution. Various evolutionary algorithms search in different ways, but they all essentially have a random factor. In recent years, opposition-based learning has been widely applied in the evolutionary algorithm [48,49]. The main principle of opposition-based learning (OBL) is to replace random search with symmetric search, which can significantly improve the searching ability of the algorithm [50–53].

The inventors of OBL argue that given a random number, its opposite is more likely to be near the solution than a random point in the search space. Therefore, by comparing a number with its opposite number, the convergence time for finding the best solution can be reduced. For example, if the current solution X is -1 and the globally optimal value is 2, the X^* solution that is symmetric about the origin of the coordinate axis is 1, and X is 3 away from the globally optimal value. However, the global optimal value of X^* is only 1. Thus, the solution X^* is closer to the global optimal, as shown in Figure 2.

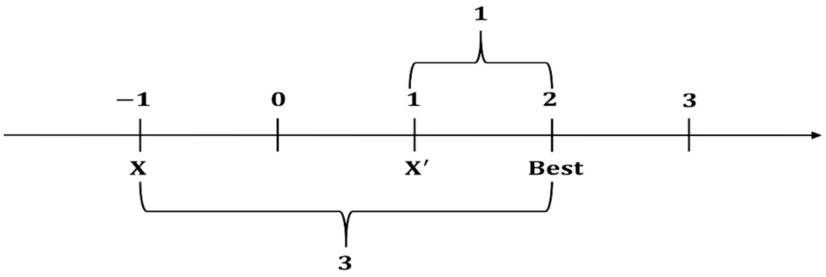


Figure 2. The opposition-based learning (OBL) principle.

Although the opposition-based learning method improves the searching capability of the algorithm to a great extent, opposition-based learning is too fixed and the tuning effect of the small space is not good. Therefore, in this paper, an innovative way is proposed to improve the original OBL which adds a random weight factor σ. For each individual p_i, a different individual p_{i1} is randomly selected in the population, where $i \neq i1$ and $i, i1 = 1, 2, ...ps$. Then, P is recombined and obtain a new point. Finally, the opposite operation is performed at the merge point. In this way, the resulting position is not a fixed symmetric position, but a random position between the symmetric position and the central position. This is as shown in Equation (27).

$$OP_{ij} = a_j(t) + b_j(t) - \left(\sigma * P_{ij} + (1 - \sigma) * P_{i1j}\right) \tag{27}$$

Here, OP_{ij} is the opposite of p_{ij}, and p_{ij} is the j_{th} vector of the i_{th} individual in the population. $a_j(t)$ and $b_j(t)$ are dynamic interval boundaries $\left[a_j(t), b_j(t)\right]$ and are respectively the minimum and maximum values of the j_{th} dimension in the current search space. $t = 1, 2, \ldots$. denotes the generation. σ is a random number in the range of $[0, 1]$.

In order to solve the problem of how to judge the search stagnation, the variable coefficient λ is introduced to determine whether the population has low diversity or whether the algorithm converges to the local optimum. The variable coefficient λ is the

variable coefficient of the error of all individual members. The variable coefficient λ is described as follows:

$$\lambda = \frac{f_{std}}{f_{mean}} \tag{28}$$

Here, f_{std} represents the standard deviation of the error value of all individuals and f_{mean} represents to the average value of the error values of all individuals. The restart policy is enacted if the condition in Equation (29) is met.

$$\lambda < \xi \tag{29}$$

where ξ, is the predetermined parameter used to control the restart.

Algorithm 2: MjSO

$g \leftarrow 1$; Archive $A \leftarrow \emptyset$; $FES = 0$;
Initialize population $P_g = (x_{i,g}. \ldots, x_{NP,g})$ randomly
Set all values in M_F to 0.5
Set all values in M_{CR} to 0.5
$k \leftarrow 1$ *//index counter*
while the termination criteria are not meet **do**
  $S_{CR} \leftarrow \emptyset$, $S_F \leftarrow \emptyset$
    for $i = 1$ to NP **do**
      $r_i \leftarrow$ select from $[1, H]$ randomly
      if $r_i = H$ **then**
      $M_{F,ri} \leftarrow 0.9$
        $M_{CR,ri} \leftarrow 0.9$
        end if
        if $M_{CR,ri} < 0$ **then**
      $CR_{i,g} \leftarrow 0$
        else
      $CR_{i,g} \leftarrow N_i(M_{CR,ri}, 0.1)$
        end if
        if $g < 0.25 * MAX_FES$ **then**
       $CR_{ig} \leftarrow \max(CR_{i,g}, 0.7)$
        else if $g < 0.5 * MAX_FES$ **then**
      $CR_{i,g} \leftarrow \max(CR_{i,g}, 0.6)$
      end if
        if $FES < MAX_FES/2$
      $F_{i,g} = 0.45 + 0.1 * rand$
        else
      $F_{i,g} \leftarrow C_i(M_{F,ri}, 0.1)$
        if $g < 0.6 * MAX_FES$ and $F_{i,g} > 0.7$ **then**
         $F_{i,g} \leftarrow 0.7$
        end if
        end if
      $u_{i,j} \leftarrow$ current-to-pBest-w/1/bin using Equation (21)
    end for

    for i = 1 to NP **do**
      if $f\left(u_{i,j}\right) \leq f\left(x_{i,g}\right)$ **then**
      $x_{i,g+1} \leftarrow u_{i,j}$
      else
      $x_{i,g+1} \leftarrow x_{i,g}$
      end if
      if $f\left(u_{i,j}\right) \leq f\left(x_{i,g}\right)$ **then**
        $x_{i,g} \rightarrow A$, $x_{i,g} \rightarrow S_{CR}$, $F_{i,g} \rightarrow S_F$
      end if

      Shrink A, if necessary
      Update M_{CR} and M_F
      Apply **LPSR** strategy*//linear population size reduction*
      Apply **Algorithm 1**
      Update p using Equation (20)
    end for
  $g \leftarrow g + 1$
end while

4. The Experimental Setup

Usually, due to the lack of a theoretical proof, it is difficult to evaluate the degree of "good" of an optimized algorithm; therefore, the benchmark function plays an important role in the performance evaluation of these algorithms. Therefore, the performance of the MjSO algorithm proposed in this paper is evaluated using the CEC 2017 actual parameter single objective optimization competition as the benchmark. The benchmark test includes 30 test functions with various characteristics. D is the dimension of the problem, and the problem is tested using $D = 10$, $D = 30$, $D = 50$, and $D = 100$. Functions 1~3 are unimodal functions, functions 4~10 are multimodal functions, functions 11~20 are mixed functions, and functions 21~30 are composite functions. These unconstrained benchmark functions are shown in Table 1. The termination criteria for the iteration of the algorithm are when the maximum number of fitness function evaluations ($MAXFES$) and the minimum error value (when the error value is less than 10^{-8}, it is considered to have found the optimal value) are $D * 10,000$ and 10^{-8}, respectively. The search range $[x_{min}, x_{max}] = [-100, 100]$, and 51 independent repeated experiments are conducted for each function. In order to statistically compare the quality of the solutions of different algorithms, two nonparametric statistical hypothesis tests were used to analyze the results: (1) the Friedman test was used to sort all the comparison algorithms [54]; (2) all of the comparison algorithms were evaluated using the Wilcoxon's signed rank test with a significance level $\alpha = 0.05$.

Table 1. Congress on Evolutionary Computation (CEC) 2017 unconstrained benchmark functions.

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

Each algorithm has its own parametric optimization method to control the influence of parameters on the algorithm. The algorithm LSHADE uses ParamILS [55] which is a versatile and efficient automatic parameter tuner. The parameters in jSO are kept unchanged according to the parameters setting in the LSHADE, except the following parameters:

p, NP_{init}, H, and M_F. The author of algorithm EBLSHADE thinks that it is difficult to determine the optimal values of the control parameters for a variety of problems with different characteristics at different stages of evolution. Therefore, for achieving good performance, the algorithm LSHADE parameter adaptation method was used in EBLSHADE. The algorithm SALSHADE-cnEPSin introduced the method of adaptive parameters by using Weibull distribution-based scaling factor and exponentially decreasing crossover rate. The algorithm ELSHADE-SPACMA performs semi-parameter adaptation (SPA) for F and Cr, and proposes a semi-parameter adaptation strategy to solve parameter setting problem. The parameter settings of each algorithm are consistent with the recommended values of the original papers [31,32,34,56,57], as shown in Table 2.

Table 2. Parameter settings.

Parameter Setting
MjSO $NP_{init} = 25 \log D$, $NP_{fin} = 4$, $H = 5$, $M_F = 0.5$, $M_{CR} = 0.5$, $\|A\| = NP$, $\xi = 10^{-8}$, $P_{max} = 0.25$ $P_{min} = P_{max}/2$
jSO $NP_{init} = 25 \log D$, $NP_{fin} = 4$, $H = 5$, $M_F = 0.3$, $M_{CR} = 0.8$, $\|A\| = NP$, , $P_{max} = 0.25$ $P_{min} = P_{max}/2$
LSHADE $NP_{init} = 18D$, $NP_{fin} = 4$, $H = 6$, $M_F = 0.5$, $M_{CR} = 0.5$, $\|A\| = NP$, $P = 0.11$
EBLSAHDE $NP_{init} = 18D$, $NP_{fin} = 4$, $H = 5$, $M_F = 0.5$, $M_{CR} = 0.5$, $\|A\| = NP$, $P = 0.11$
ELSHADE-SPACMA $NP_{init} = 18D$, $NP_{fin} = 4$, $H = 5$, F_{cp} =0.5, $c = 0.8$, $P_{init} = 0.3$, $p_{min} = 0.15$
SALSHADE-cnEPSin $NP_{init} = 18D$, $NP_{fin} = 4$, $H = 5$, $M_F = 0.5$, $M_{CR} = 0.5$, $freq = 0.5$, $ps = 0.5$, $pc = 0.4$

4.1. Experimental Environment

All these experiments were conducted on a PC with an Intel(R) Core (TM) i7-9700 CPU @ 3.0 GHz, 32 GB of RAM and the Windows 10 Professional Edition operating system; and all these algorithms were implemented using Matlab 2017b.

4.2. Clustering Analysis

The clustering algorithm selected in this paper is density-based noisy application spatial clustering (DBSCAN) [58]. It is a clustering algorithm based on the clustering density, which can find clusters with arbitrary shapes. The DBSCAN algorithm needs to set two control parameters and a distance measure. The settings are as follows:

1. The core point distance is that Eps = 1% of the decision space. For the CEC2017 benchmark set, Eps = 2.
2. The minimum number of clusters MinPts = 4 (minimum number of individuals with mutations).
3. The distance measurement is equal to Chebyshev distance [59]. If the distance between any corresponding attributes of two individuals is greater than 1% of the decision space, they are not considered to be directly dense-reachable.

4.3. Population Diversity

The population diversity (PD) measure used in this paper was taken from reference [60] and calculated based on the square root of the sum of individual dimensions, as shown in Equation (31), and the deviation from the corresponding mean, as shown in Equation (30):

$$\overline{x_j} = \frac{1}{NP} \sum_{i=1}^{NP} x_{j,i} \tag{30}$$

$$PD = \sqrt{\frac{1}{NP} \sum_{i=1}^{NP} \sum_{j=1}^{D} \left(x_{j,i} - \overline{x_j}\right)} \tag{31}$$

where i is the iteration of the individual population, and j is the iteration of the individual dimension.

5. Experimental Results and Analysis

This section presents the experimental results of this paper. Table 3, Table 4, Table 5, Table 6, Table 7 shows the results of the comparison between MjSO and the latest proposed LSHADE variant algorithms (EBLSAHDE, ELSHADE SPACMA, and SALSHADE-cnEPSin) and their original versions (jSO and LSHADE) on D = 10, D = 30, D = 50, and D = 100. All the experimental results adopt the error value. According to the experimental results, three symbols (" $>$ ", " $<$ ", and " $\approx$ ") are set to compare the advantages and disadvantages of the algorithm. The symbols (" $>$ ", " $<$ ", and " $\approx$ ") indicate that MjSO performed significantly better ($>$), significantly worse ($<$), or not significantly different better or worse ($\approx$) compared to other compared algorithms using the Wilcoxon rank-sum test (significantly, $\alpha = 0.05$). For each benchmark function from F1 to F30, the best value among the six algorithms is shown in bold (or bold if the performance is similar).

As shown in Table 3, the six DE variants compared perform equally well on benchmarks f1, f2, f3, f4, f6, and f9 and the global optimal values for these benchmarks were always found within 51 independent runs. In addition, compared with EBLSHADE, the proposed MjSO algorithm achieved better performance for 14 and similar performance for 13 out of 30 benchmarks. Compared with SALSHADE-cnEPSin, our proposed MjSO algorithm achieved better performance 8 times and similar performance 20 times. Compared to jSO, our algorithm achieved better performance 13 times and similar performance 16 times. Compared to LSHADE, our algorithm achieved better performance 16 times and similar performance 13 times. Compared to ELSHADE-SPACMA, our algorithm achieved better performance 13 times and similar performance 17 times. In summary, the proposed MjSO algorithm achieved the best performance in D = 10 optimization.

It can be seen from Table 4 that the six DE variants compared performed equally well on the benchmarks f1, f2, f3, and f9 and that the global optimal values for these benchmarks were always found within 51 separate runs. The global optimal values of SALSHADE-cnEPSin, ELSHADE-SPACMA and MjSO can be found on f6 each time the algorithms run. In addition, compared with EBLSHADE, the proposed MjSO algorithm achieved better performance in 16 and similar performance in 14 out of 30 benchmarks. Compared with SALSHADE-cnEPSin, our algorithm achieved better performance 19 times and similar performance 10 times. Compared to jSO, our algorithm achieved better performance 20 times and similar performance 10 times. Compared to LSHADE, our algorithm achieved better performance 17 times and similar performance 13 times. Compared to ELSHADE-SPACMA, our algorithm achieved better performance 18 times and similar performance 12 times. From what has been discussed above, the MjSO algorithm achieved the best performance on D = 30 optimization.

Table 3. Algorithm comparison between five powerful differential evolution (DE) variants and our MjSO algorithm on D = 10 optimization under f1–f30 of our test suite.

NO	EBLSHADE		SALSHADE-cnEPSin		jSO		LSHADE		ELSHADE-SPACMA		MjSO	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
f1	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f2	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f3	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f4	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f5	>2.52E+00	8.98E-01	>1.99E+00	6.62E-01	>1.76E+00	7.60E–01	>2.57E+00	8.37E-01	>3.87E+00	2.02E+00	**1.35E+00**	9.10E-01
f6	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f7	>1.22E+01	7.09E-01	>1.19E+01	5.67E-01	>1.18E+01	6.07E–01	>1.22E+01	7.05E-01	>1.33E+01	1.75E+00	**1.15E+01**	5.99E-01
f8	>2.23E+00	9.02E-01	>1.99E+00	7.63E-01	>1.95E+00	7.44E–01	>2.52E+00	6.99E-01	>4.10E+00	2.51E+00	**1.37E+00**	5.87E-01
f9	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f10	>2.50E+01	3.99E+01	<7.36E+00	4.49E+01	>3.59E+01	5.55E+01	>3.86E+01	5.50E+01	>2.27E+01	4.84E+01	1.53E+01	3.27E+01
f11	>2.74E-01	5.48E-01	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	>3.65E-01	6.94E-01	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f12	<1.22E+01	3.59E+01	<1.19E+02	7.49E+01	**<2.66E+00**	1.68E+01	>3.89E+01	5.68E+01	>2.85E+01	5.14E+01	2.79E+01	5.02E+01
f13	>3.64E+00	2.23E+00	>4.83E+00	2.30E+00	>2.96E+00	2.35E+00	>3.88E+00	2.37E+00	>3.57E+00	2.21E+00	**2.49E+00**	2.50E+00
f14	>5.38E-01	8.03E-01	**≈0.00E+00**	2.36E-01	>5.85E–02	2.36E–01	>7.73E-01	9.05E-01	>7.80E-02	2.70E-01	**0.00E+00**	0.00E+00
f15	**≈1.44E-01**	2.03E-01	≈2.70E-01	2.03E+00	≈2.21E–01	2.00E–01	≈1.99E-01	2.12E-01	≈2.51E-01	2.17E-01	1.92E-01	2.20E-01
f16	≈4.34E-01	2.23E-01	≈6.25E-01	2.59E-01	≈5.69E–01	2.64E–01	**≈3.83E-01**	1.64E-01	≈5.62E-01	2.55E-01	5.64E-01	2.76E-01
f17	≈1.23E-01	1.51E-01	≈1.77E-01	2.41E-01	≈5.02E–01	3.48E–01	**≈1.06E-01**	1.31E-01	≈1.39E-01	1.44E-01	3.54E-01	3.11E-01
f18	**≈1.79E-01**	1.95E-01	≈4.49E-01	5.43E+00	≈3.08E–01	1.95E–01	≈2.15E-01	1.97E-01	≈7.10E-01	2.80E+00	3.06E-01	1.99E-01
f19	**≈9.06E-03**	1.09E-02	≈1.97E-02	3.01E-02	≈1.07E–02	1.25E–02	≈1.05E-02	1.10E-02	≈1.55E-02	1.14E-02	1.60E-02	2.30E-02
f20	**<1.22E-02**	6.12E-02	≈3.12E-01	4.01E-01	≈3.43E–01	1.29E–01	**<1.22E-02**	6.06E-02	≈1.41E-01	1.57E-01	3.12E-01	0.00E+00
f21	>1.56E+02	5.12E+01	**≈1.00E+02**	5.13E+01	>1.32E+02	4.84E+01	>1.50E+02	5.14E+01	≈1.02E+02	1.48E+01	1.10E+02	3.11E+01
f22	**≈1.00E+02**	1.01E-01	**≈1.00E+02**	6.26E-02	**≈1.00E+02**	0.00E+00	**≈1.00E+02**	4.01E-02	**≈1.00E+02**	1.21E-01	**1.00E+02**	8.17E-14

Table 3. *Cont.*

NO	EBLSHADE		SALSHADE-cnEPSin		jSO		LSHADE		ELSHADE-SPACMA		MjSO	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
f23	>3.03E+02	1.71E+00	>3.01E+02	1.43E+00	>3.01E+02	1.59E+00	>3.03E+02	1.65E+00	>3.04E+02	2.30E+00	**3.00E+02**	9.63E-01
f24	>3.16E+02	5.46E+01	>3.29E+02	7.97E+01	>2.97E+02	7.93E+01	>3.21E+02	4.52E+01	>2.91E+02	9.54E+01	**2.42E+02**	1.14E+02
f25	>4.15E+02	2.24E+01	>4.43E+02	2.22E+01	>4.06E+02	1.75E+01	>4.09E+02	1.95E+01	>4.13E+02	2.18E+01	**3.95E+02**	1.37E+01
f26	**≈3.00E+02**	0.00E+00	**≈3.00E+02**	0.00E+00	**≈3.00E+02**	0.00E+00	**≈3.00E+02**	0.00E+00	**≈3.00E+02**	0.00E+00	**3.00E+02**	0.00E+00
f27	>3.89E+02	1.39E-01	>3.88E+02	1.66E+00	>3.89E+02	2.26E–01	>3.89E+02	1.78E-01	>3.89E+02	1.67E-01	**3.87E+02**	1.63E+00
f28	>3.47E+02	1.10E+02	≈3.00E+02	1.23E+02	>3.39E+02	9.65E+01	>3.58E+02	1.18E+02	>3.25E+02	1.04E+02	**3.00E+02**	0.00E+00
f29	>2.33E+02	2.65E+00	**≈2.28E+02**	1.56E+00	>2.34E+02	2.96E+00	>2.34E+02	2.78E+00	≈2.30E+02	2.26E+00	2.30E+02	2.10E+00
f30	**<3.24E+04**	1.60E+05	≈3.94E+02	9.42E+04	≈3.95E+02	4.50E–02	>4.05E+02	2.08E+01	>4.02E+02	1.77E+01	3.96E+02	9.44E+00

Table 4. Algorithm comparison between five powerful DE variants and our MjSO algorithm on D = 30 optimization under f1–f30 of our test suite.

NO	EBLSHADE		SALSHADE-cnEPSin		jSO		LSHADE		ELSHADE-SPACMA		MjSO	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
f1	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f2	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f3	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f4	≈5.86E+01	3.11E-14	**<4.90E+01**	3.32E+00	≈5.87E+01	7.78E–01	≈5.86E+01	3.22E-14	≈5.86E+01	0.00E+00	5.86E+01	3.66E-14
f5	**≈6.26E+00**	1.29E+00	>1.24E+01	2.39E+00	>8.56E+00	2.10E+00	≈6.41E+00	1.52E+00	>1.86E+01	8.04E+00	7.45E+00	2.20E+00
f6	>6.04E-09	2.71E-08	**≈0.00E+00**	8.66E-08	>6.04E–09	2.71E–08	>2.68E-08	1.52E-07	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f7	≈3.73E+01	1.44E+00	>4.32E+01	2.18E+00	>3.89E+01	1.46E+00	**≈3.71E+01**	1.55E+00	>3.89E+01	3.43E+00	3.75E+01	2.11E+00
f8	**≈6.66E+00**	1.54E+00	>1.36E+01	2.21E+00	>9.09E+00	1.84E+00	≈7.15E+00	1.58E+00	>1.61E+01	7.46E+00	7.98E+00	1.62E+00
f9	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f10	**≈1.42E+03**	2.01E+02	>1.47E+03	2.35E+02	>1.53E+03	2.77E+02	>1.50E+03	1.73E+02	>1.70E+03	4.06E+02	**1.42E+03**	2.73E+02
f11	>2.63E+01	2.82E+01	>3.93E+00	1.76E+01	>3.04E+00	2.65E+00	>3.22E+01	2.85E+01	>7.80E+00	1.42E+01	**1.56E+00**	1.36E+00

Table 4. *Cont.*

NO	EBLSHADE		SALSHADE-cnEPSin		jSO		LSHADE		ELSHADE-SPACMA		MjSO	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
f12	>9.45E+02	3.73E+02	>3.43E+02	2.19E+02	>1.70E+02	1.02E+02	>1.00E+03	3.59E+02	>2.47E+02	1.28E+02	**1.32E+02**	8.57E+01
f13	>1.55E+01	4.88E+00	>1.70E+01	5.24E+00	>1.48E+01	4.83E+00	>1.61E+01	4.97E+00	>1.57E+01	5.03E+00	**1.25E+01**	8.99E+00
f14	**≈2.11E+01**	4.24E+00	>2.20E+01	3.85E+00	>2.18E+01	1.25E+00	≈2.14E+01	2.98E+00	>2.37E+01	5.25E+00	2.16E+01	4.72E+00
f15	>2.67E+00	1.43E+00	>3.65E+00	1.75E+00	**≈1.09E+00**	6.91E–01	>3.22E+00	1.32E+00	≈1.86E+00	1.29E+00	1.80E+00	1.27E+00
f16	>3.84E+01	2.64E+01	>1.88E+01	3.98E+01	>7.89E+01	8.48E+01	>6.52E+01	7.46E+01	>6.68E+01	8.35E+01	**1.55E+01**	5.56E+00
f17	>3.32E+01	5.24E+00	>2.83E+01	5.88E+00	>3.29E+01	8.08E+00	>3.28E+01	6.36E+00	>2.97E+01	6.76E+00	**2.70E+01**	6.12E+00
f18	≈2.07E+01	3.93E+00	≈2.06E+01	9.07E-01	**≈2.04E+01**	2.87E+00	>2.21E+01	9.87E-01	≈2.09E+01	3.03E+00	2.08E+01	3.21E-01
f19	>5.32E+00	1.65E+00	>5.91E+00	1.89E+00	>4.50E+00	1.73E+00	>5.21E+00	1.58E+00	>4.61E+00	1.35E+00	**4.22E+00**	1.32E+00
f20	>3.08E+01	5.80E+00	>3.08E+01	5.96E+00	>2.94E+01	5.85E+00	>3.10E+01	6.54E+00	>2.73E+01	4.56E+00	**2.63E+01**	6.29E+00
f21	>2.11E+02	1.67E+00	>2.13E+02	2.07E+00	>2.09E+02	1.96E+00	≈2.07E+02	1.47E+00	>2.22E+02	6.64E+00	**2.07E+02**	1.72E+00
f22	**≈1.00E+02**	1.00E-13	**≈1.00E+02**	1.00E-13	**≈1.00E+02**	0.00E+00	**≈1.00E+02**	1.00E-13	**≈1.00E+02**	0.00E+00	**1.00E+02**	0.00E+00
f23	>3.48E+02	2.81E+00	>3.54E+02	4.11E+00	>3.51E+02	3.30E+00	>3.50E+02	3.10E+00	>3.69E+02	1.05E+01	**3.45E+02**	3.66E+00
f24	>4.25E+02	1.89E+00	>4.29E+02	2.71E+00	>4.26E+02	2.47E+00	>4.26E+02	1.44E+00	>4.41E+02	7.84E+00	**4.22E+02**	2.90E+00
f25	**≈3.87E+02**	2.71E-02	**≈3.87E+02**	6.82E-03	**≈3.87E+02**	7.68E–03	**≈3.87E+02**	2.47E-02	**≈3.87E+02**	9.60E-03	**3.87E+02**	5.67E-03
f26	>8.97E+02	3.13E+01	>9.51E+02	4.74E+01	>9.20E+02	4.30E+01	>9.51E+02	3.79E+01	>1.08E+03	8.68E+01	**8.91E+02**	3.48E+01
f27	>5.01E+02	5.44E+00	>5.03E+02	4.01E+00	>4.98E+02	7.00E+00	>5.05E+02	4.81E+00	>4.99E+02	6.15E+00	**4.96E+02**	5.69E+00
f28	>3.26E+02	4.66E+01	≈3.05E+02	4.21E+01	>3.09E+02	3.03E+01	>3.33E+02	5.24E+01	≈3.02E+02	1.60E+01	**3.02E+02**	2.49E+01
f29	>4.38E+02	6.17E+00	>4.38E+02	1.05E+01	>4.34E+02	1.36E+01	>4.34E+02	8.45E+00	>4.33E+02	1.56E+01	**4.28E+02**	1.06E+01
f30	≈1.98E+03	3.07E+01	**≈1.97E+03**	4.42E+01	**≈1.97E+03**	1.90E+01	≈1.99E+03	5.24E+01	≈1.98E+03	3.34E+01	**1.97E+03**	1.24E+01

Table 5. Algorithm comparison between five powerful DE variants and our MjSO algorithm on D = 50 optimization under f1–f30 of our test suite.

NO	EBLSHADE		SALSHADE-cnEPSin		jSO		LSHADE		ELSHADE-SPACMA		MjSO	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
f1	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f2	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f3	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f4	>7.00E+01	4.58E+01	>3.69E+01	4.17E+01	>5.62E+01	4.88E+01	>7.07E+01	4.97E+01	>4.36E+01	3.62E+01	**3.00E+01**	2.70E+01
f5	≈1.42E+01	1.78E+00	>2.81E+01	5.30E+00	>1.64E+01	3.46E+00	**≈1.38E+01**	2.95E+00	≈1.39E+01	5.55E+00	1.48E+01	3.28E+00
f6	>6.94E-05	3.32E-04	>9.52E-07	1.40E-06	>1.09E–06	2.62E–06	>6.12E-05	3.09E-04	**<0.00E+00**	0.00E+00	1.83E-08	4.41E-08
f7	≈6.29E+01	1.98E+00	>7.73E+01	5.54E+00	≈6.65E+01	3.47E+00	≈6.30E+01	1.85E+00	**≈6.15E+01**	3.86E+00	6.58E+01	3.26E+00
f8	≈1.22E+01	2.00E+00	>2.64E+01	5.85E+00	>1.70E+01	3.14E+00	**≈1.20E+01**	2.11E+00	>1.79E+01	7.47E+00	1.29E+01	2.17E+00
f9	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f10	>3.17E+03	3.06E+02	>3.36E+03	3.02E+02	>3.14E+03	3.67E+02	>3.15E+03	2.59E+02	>3.69E+03	6.07E+02	**2.99E+03**	3.14E+02
f11	>4.37E+01	7.27E+00	>2.81E+01	1.89E+00	>2.79E+01	3.33E+00	>4.80E+01	8.24E+00	>2.62E+01	3.76E+00	**2.44E+01**	2.96E+00
f12	>2.02E+03	5.00E+02	>1.28E+03	3.64E+02	>1.68E+03	5.23E+02	>2.24E+03	5.22E+02	>1.36E+03	3.42E+02	**8.51E+02**	3.87E+02
f13	>6.40E+01	3.47E+01	>8.68E+01	2.97E+01	>3.06E+01	2.12E+01	>6.41E+01	2.65E+01	>3.68E+01	1.72E+01	**2.56E+01**	1.96E+01
f14	>2.78E+01	2.27E+00	>2.65E+01	2.35E+00	≈2.50E+01	1.87E+00	>2.98E+01	3.01E+00	>3.07E+01	3.95E+00	**2.52E+01**	2.53E+00
f15	>3.41E+01	9.07E+00	>2.63E+01	3.57E+00	>2.39E+01	2.49E+00	>4.01E+01	1.06E+01	>2.28E+01	2.20E+00	**2.11E+01**	1.67E+00
f16	>3.54E+02	1.09E+02	>3.29E+02	1.11E+02	>4.51E+02	1.38E+02	>3.77E+02	1.24E+02	>4.15E+02	1.77E+02	**2.85E+02**	1.22E+02
f17	>2.64E+02	6.33E+01	>2.76E+02	5.33E+01	>2.83E+02	8.61E+01	≈2.51E+02	5.71E+01	**≈2.30E+02**	9.68E+01	2.48E+02	8.68E+01
f18	>3.31E+01	7.86E+00	>2.50E+01	2.09E+00	>2.43E+01	2.02E+00	>3.98E+01	8.64E+00	>2.51E+01	2.56E+00	**2.24E+01**	1.45E+00
f19	>1.93E+01	3.25E+00	>1.81E+01	3.41E+00	>1.41E+01	2.26E+00	>2.32E+01	5.94E+00	>1.44E+01	2.31E+00	**1.26E+01**	2.58E+00
f20	>1.72E+02	6.94E+01	>1.31E+02	2.64E+01	>1.40E+02	7.74E+01	>1.73E+02	7.15E+01	>1.08E+02	7.31E+01	**1.00E+02**	3.35E+01
f21	>2.21E+02	2.55E+00	>2.26E+02	6.04E+00	>2.19E+02	3.77E+00	**≈2.12E+02**	2.25E+00	>2.42E+02	9.52E+00	2.14E+02	3.99E+00
f22	>2.67E+03	1.59E+03	>1.00E+03	1.70E+03	>1.49E+03	1.75E+03	>2.68E+03	1.62E+03	≈7.86E+02	1.64E+03	**7.67E+02**	1.42E+03

Table 5. *Cont.*

NO	EBLSHADE		SALSHADE-cnEPSin		jSO		LSHADE		ELSHADE-SPACMA		MjSO	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
f23	>4.67E+02	4.38E+00	>4.41E+02	7.07E+00	>4.30E+02	6.24E+00	>4.30E+02	4.91E+00	>4.62E+02	1.39E+01	**4.27E+02**	5.86E+00
f24	>5.05E+02	3.51E+00	>5.14E+02	6.05E+00	>5.07E+02	4.13E+00	>5.06E+02	2.55E+00	>5.34E+02	9.14E+00	**4.98E+02**	3.50E+00
f25	>4.88E+02	2.01E+01	>4.88E+02	1.54E+00	≈4.81E+02	2.80E+00	>4.84E+02	1.29E+01	≈4.81E+02	2.80E+00	**4.80E+02**	1.81E-02
f26	>1.13E+03	4.45E+01	>1.25E+03	9.13E+01	>1.13E+03	5.62E+01	>1.14E+03	4.93E+01	>1.34E+03	1.38E+02	**1.05E+03**	4.64E+01
f27	>5.27E+02	1.09E+01	>5.23E+02	8.58E+00	≈5.11E+02	1.11E+01	>5.31E+02	1.67E+01	**≈5.10E+02**	9.52E+00	5.18E+02	1.43E+01
f28	>4.73E+02	2.23E+01	>4.67E+02	6.78E+00	≈4.60E+02	6.84E+00	>4.71E+02	2.15E+01	≈4.60E+02	6.84E+00	**4.59E+02**	2.91E-13
f29	>3.62E+02	1.04E+01	>3.61E+02	1.07E+01	>3.63E+02	1.32E+01	**≈3.50E+02**	1.09E+01	>3.58E+02	1.78E+01	3.53E+02	1.21E+01
f30	>6.54E+05	7.78E+04	>6.48E+05	5.85E+04	≈6.01E+05	2.99E+04	>6.58E+05	8.12E+04	**≈5.97E+05**	2.38E+04	6.02E+05	3.07E+04

Table 6. Algorithm comparison between five powerful DE variants and our MjSO algorithm on D = 100 optimization under f1–f30 of our test suite.

NO	EBLSHADE		SALSHADE-cnEPSin		jSO		LSHADE		ELSHADE-SPACMA		MjSO	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
f1	**≈0.00E+00**	0.00E+00	>1.36E-08	2.95E-08	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f2	**<6.44E+00**	1.54E+01	>1.49E+11	7.72E+11	<8.94E+00	2.42E+01	>8.42E+05	6.01E+06	>7.52E+07	6.01E+07	3.27E+01	5.92E+01
f3	>1.56E-06	1.46E-06	**<0.00E+00**	0.00E+00	>2.39E–06	2.73E–06	>6.90E-06	6.77E-06	≈1.60E-07	4.00E-07	8.81E-07	7.38E-07
f4	**≈1.84E+02**	5.87E+01	>2.01E+02	7.91E+00	≈1.90E+02	2.89E+01	≈1.97E+02	1.58E+01	>2.01E+02	8.67E+00	1.94E+02	7.94E+00
f5	>4.14E+01	3.80E+00	>6.19E+01	1.05E+01	>4.39E+01	5.61E+00	>3.83E+01	4.90E+00	>3.78E+01	5.85E+00	**2.82E+01**	9.53E+00
f6	>1.22E-02	6.81E-03	>5.65E-05	3.51E-05	>2.02E-04	6.20E–04	>5.71E-03	3.43E-03	**<0.00E+00**	1.34E-08	1.05E-06	9.32E-07
f7	>1.40E+02	4.25E+00	>1.71E+02	7.36E+00	>1.45E+02	6.70E+00	>1.41E+02	4.46E+00	>1.51E+02	1.48E+00	**1.34E+02**	6.19E+00
f8	>3.73E+01	6.67E+00	>6.20E+01	9.99E+00	>4.22E+01	5.52E+00	>3.86E+01	4.47E+00	>2.98E+01	1.32E+01	**2.80E+01**	1.00E+01
f9	>6.33E-01	4.60E-01	**≈0.00E+00**	0.00E+00	>4.59E-02	1.15E–01	>4.86E-01	4.83E-01	**≈0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f10	>1.03E+04	4.51E+02	>1.05E+04	5.30E+02	>9.70E+03	6.82E+02	>1.04E+04	5.45E+02	>1.08E+04	9.53E+02	**9.60E+03**	7.03E+02

Table 6. *Cont.*

NO	EBLSHADE		SALSHADE-cnEPSin		jSO		LSHADE		ELSHADE-SPACMA		MjSO	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
f11	>3.71E+02	1.03E+02	>4.54E+01	4.79E+01	>1.13E+02	4.32E+01	>4.52E+02	8.94E+01	>7.34E+01	4.30E+01	**3.01E+01**	4.79E+00
f12	>2.28E+04	5.65E+03	>6.72E+03	9.17E+02	>1.84E+04	8.35E+03	>2.49E+04	9.97E+03	>7.79E+03	2.92E+03	**5.38E+03**	1.32E+03
f13	>2.33E+02	6.04E+01	>1.04E+02	3.72E+01	>1.45E+02	3.80E+01	>5.70E+02	4.14E+02	>1.49E+02	3.83E+01	**5.58E+01**	2.54E+01
f14	>2.36E+02	1.88E+01	>5.12E+01	6.90E+00	>6.43E+01	1.09E+01	>2.51E+02	2.94E+01	>4.75E+01	5.69E+00	**3.84E+01**	4.16E+00
f15	>2.65E+02	3.97E+01	>9.37E+01	3.19E+01	>1.62E+02	3.81E+01	>2.57E+02	4.02E+01	>1.08E+02	4.34E+01	**5.55E+01**	1.54E+01
f16	>1.50E+03	3.54E+02	>1.51E+03	1.90E+02	>1.86E+03	3.49E+02	>1.66E+03	2.78E+02	>1.76E+03	4.88E+02	**1.38E+03**	3.43E+02
f17	>1.13E+03	2.25E+02	>1.14E+02	1.75E+02	>1.28E+03	2.38E+02	>1.16E+03	1.94E+02	>1.27E+03	3.45E+02	**9.17E+02**	2.21E+02
f18	>2.65E+02	4.96E+01	>6.90E+01	1.55E+01	>1.67E+02	3.65E+01	>2.41E+02	5.66E+01	>1.05E+02	2.56E+01	**6.00E+01**	1.36E+01
f19	>1.62E+02	1.78E+01	>5.71E+01	7.301+00	>1.05E+02	2.01E+01	>1.78E+02	2.42E+01	>6.05E+01	7.55E+00	**4.73E+01**	5.00E+00
f20	>1.63E+03	1.86E+02	>1.41E+03	1.88E+02	>1.38E+03	2.43E+02	>1.56E+03	2.04E+02	>1.28E+03	2.54E+02	**1.26E+03**	2.69E+02
f21	>2.59E+02	3.33E+00	>2.88E+02	1.45E+01	>2.64E+02	6.43E+00	>2.59E+02	6.03E+00	>2.96E+02	1.65E+01	**2.51E+02**	7.79E+00
f22	>1.14E+04	5.60E+02	>1.08E+04	5.81E+02	>1.02E+04	2.18E+03	>1.13E+04	5.65E+02	>9.70E+03	1.20E+03	**3.53E+01**	6.04E+00
f23	<5.71E+02	6.68E+00	<5.92E+02	8.64E+00	<5.71E+02	1.07E+01	**<5.66E+02**	9.02E+00	<6.03E+02	2.19E+01	1.13E+03	2.49E+01
f24	>9.01E+02	5.70E+00	>9.19E+02	1.21E+01	>9.02E+02	7.89E+00	>9.20E+02	6.78E+00	>9.32E+02	1.90E+01	**2.95E+02**	2.18E+01
f25	>7.50E+02	2.58E+01	>7.21E+02	4.62E+01	>7.36E+02	3.53E+01	>7.53E+02	2.58E+01	>7.00E+02	3.99E+01	**6.68E+02**	1.80E+01
f26	>3.22E+03	6.88E+01	>3.15E+03	1.73E+02	>3.27E+03	8.02E+01	>3.43E+03	8.34E+01	>3.24E+03	2.19E+02	**3.00E+02**	3.22E-13
f27	<6.19E+02	1.68E+01	<5.88E+02	1.75E+01	**<5.85E+02**	2.17E+01	<6.43E+02	1.70E+01	<5.62E+02	1.75E+01	**1.67E+03**	1.35E+02
f28	>5.32E+02	2.58E+01	>5.16E+02	1.91E+01	>5.27E+02	2.73E+01	>5.27E+02	2.15E+01	>5.21E+02	2.38E+01	**5.04E+02**	1.70E+01
f29	>1.12E+03	1.54E+02	>1.14E+03	1.40E+02	>1.26E+03	1.91E+02	>1.27E+03	1.76E+02	>1.21E+03	1.98E+02	**9.89E+02**	1.84E+02
f30	<2.39E+03	1.39E+02	<2.33E+03	1.66E+02	<2.33E+03	1.19E+02	<2.41E+03	1.52E+02	**<2.25E+03**	1.11E+02	3.32E+03	8.57E+01

Table 7. Summarized statistical testing (Wilcoxon rank-sum test at the 0.05 significance level).

MjSO vs.		D = 10	D = 30	D = 50	D = 100
EBLSHADE	>(better)	14	16	23	24
	≈(no sig)	13	14	7	2
	<(worse)	3	0	0	4
SALSHADE-cnEPSin	>(better)	8	19	26	25
	≈(no sig)	20	10	4	1
	<(worse)	2	1	0	4
jSO	>(better)	13	20	20	24
	≈(no sig)	16	10	10	2
	<(worse)	1	0	0	4
LSHADE	>(better)	16	17	20	25
	≈(no sig)	13	13	10	2
	<(worse)	1	0	0	3
ELSHADE-SPACMA	>(better)	13	18	17	23
	≈(no sig)	17	12	12	3
	<(worse)	0	0	1	4

In Table 5, the six DE variants compared performed equally well on the benchmarks f1, f2, f3, and f9 and the global optimal values for these benchmarks were always found within 51 separate runs. In addition, compared with EBLSHADE, the proposed MjSO algorithm achieved better performance in 23 and similar performance in 7 out of 30 benchmarks. Compared with SALSHADE-cnEPSin, our algorithm achieved better performance 26 times and similar performance 4 times. Compared with jSO, our algorithm achieved better performance 20 times and similar performance 10 times. Compared with LSHADE, our algorithm achieved better performance 20 times and similar performance 10 times. Compared with ELSHADE-SPACMA, our algorithm achieved better performance 7 times and similar performance 12 times. In summary, the MjSO algorithm achieved the best performance on D = 50 optimization.

Table 6 shows the performances of the EBLSHADE, jSO, LSHADE, ELSHADE SPACMA, and MjSO algorithms during each run seeking to find the global optimal value in f1. The global optimal values of SALSHADE-cnEPSin, ELSHADE-SPACMA, and MjSO can be found on f9 every time the algorithms run. In addition, compared with EBLSHADE, the proposed MjSO algorithm achieved better performance in 24 and similar performance in 2 out of 30 benchmarks. Compared with SALSHADE-cnEPSin, our algorithm achieved better performance 25 times and similar performance 1 time. Compared to jSO, our algorithm achieved better performance 24 times and similar performance 2 times. Compared to LSHADE, our algorithm achieved better performance 25 times and similar performance 2 times. Compared to ELSHADE-SPACMA, our algorithm achieved better performance 23 times and similar performance 3 times. In summary, the proposed MjSO algorithm achieved the best performance on D = 100 optimization. From this, the modified algorithm MjSO proposed by us achieved better results on high-dimensional problems compared with the other five algorithms.

In addition, the effectiveness of the proposed MjSO algorithm with respect to convergence speed was also verified. The convergence diagram is shown in Figures 3–10. Figures 3 and 4 show the convergence curves of the partial test functions on D = 10 optimization for the MjSO algorithm and the other five comparison algorithms. Figures 5 and 6 show the convergence curves of the partial test functions on D = 30 optimization for the MjSO algorithm and the other five comparison algorithms. Figures 7 and 8 show the convergence curves of the partial test functions on D = 50 optimization for the MjSO algorithm and the other five comparison algorithms. Figures 9 and 10 show the convergence curves of the partial test functions on D = 100 optimization for the MjSO algorithm and the other five comparison algorithms. In most cases, in the early stage of the optimization process, most MjSO algorithms have similar convergence to the original algorithm, but in

the middle stage of the optimization process, the MjSO algorithm maintains a longer search state and achieves a better objective function value in the middle and late stages of the optimization. The convergence rate of the red curve of the proposed MjSO algorithm is usually slow, but it can achieve a better objective function value.

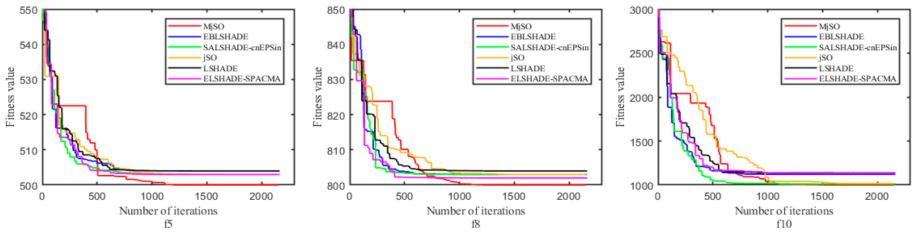


Figure 3. Convergence speed comparison on D = 10 optimization.

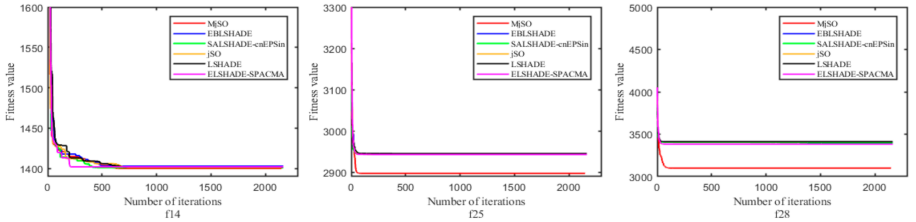


Figure 4. Convergence speed comparison on D = 10 optimization.

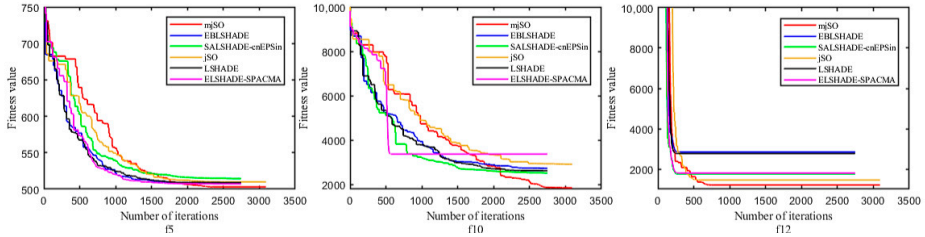


Figure 5. Convergence speed comparison on D = 30 optimization.

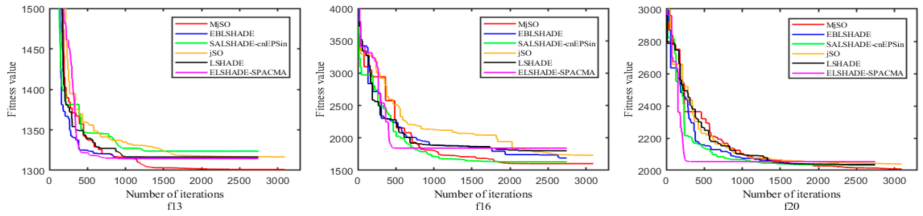


Figure 6. Convergence speed comparison on D = 30 optimization.

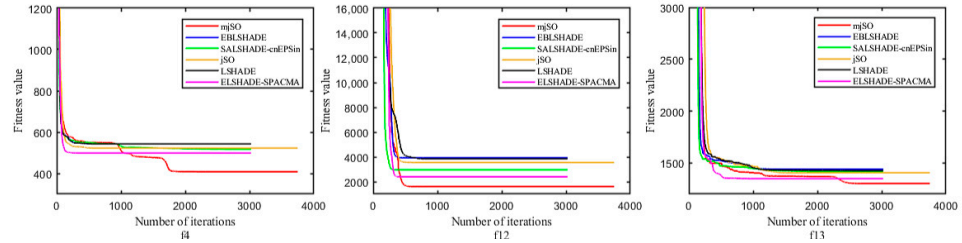


Figure 7. Convergence speed comparison on D = 50 optimization.

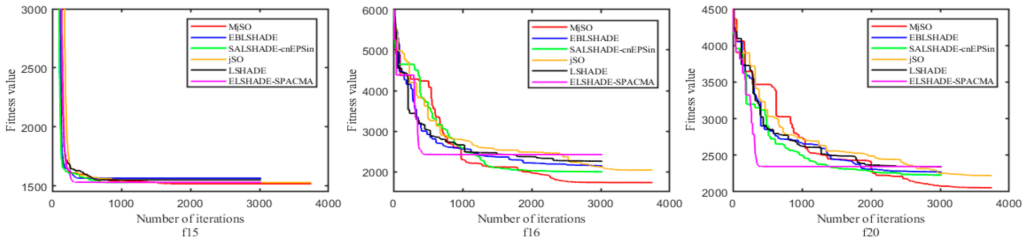


Figure 8. Convergence speed comparison on D = 50 optimization.

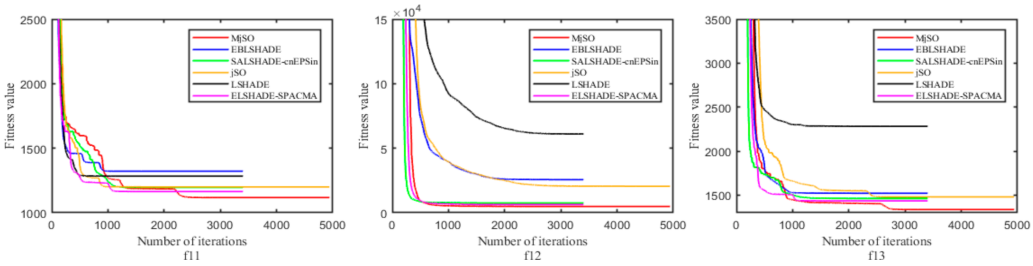


Figure 9. Convergence speed comparison on D = 100 optimization.

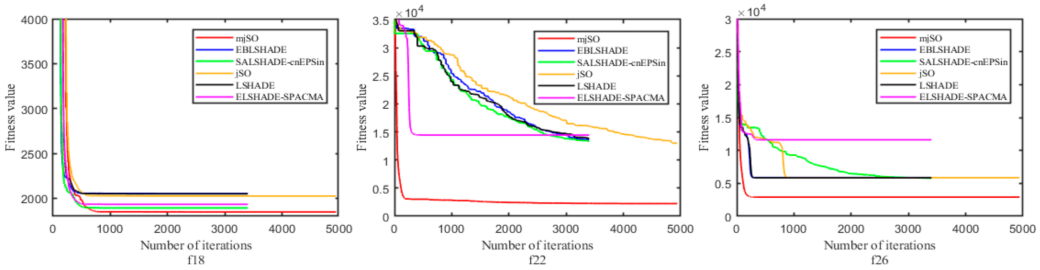


Figure 10. Convergence speed comparison on D = 100 optimization.

Tables A1–A4 show the number of independent experiments (#runs) of the proposed MjSO algorithm and its original version jSO, the mean number of iterations (mean CO) of the first clustering during the independent experiment running period, and the mean population density (mean PD) of these corresponding iterations.

It can be seen from the table that in most cases, the proposed MjSO algorithm had fewer clustering times (#runs), later clustering (mean CO), and greater population density (mean PD), i.e., better population diversity. In summary, in most cases, MjSO maintained the diversity of the population and prolonged the search stage in the optimization process. This also proves the effectiveness of the modifications we introduced.

According to the Friedman rankings in Tables A5–A8, the MjSO algorithm performed the best in all dimensions and was ranked first. Table A9 shows that in all dimensions, the null hypothesis was rejected, which verifies the correctness of the above Friedman rankings. Thus, in the D = 10, D = 30, D = 50, and D = 100 problems, the MjSO proposed by us is superior to the original jSO and LSHADE algorithms. Compared with more advanced algorithms such as EBLSHADE, SALSHADE-cnEPSin and ELSHADE-SPACMA, it has obvious advantages.

6. MjSO for Engineering Problems

In this section, three constrained engineering problems are employed to demonstrate the performance of MjSO, namely, the pressure vessel design problem, tension/compression spring design problem, and welding beam design problem [61,62]. These engineering problems are inspired by real world cases and there exist some real constrained conditions and one objective functions. Therefore, transforming them into constrained optimization problems is the general method to handle them.

6.1. Pressure Vessel Design Problem

The goal of the pressure vessel design problem is to minimize the total cost of the cylindrical pressure vessel [33]. It has four design variables: shell thickness (T_s), ball thickness (T_h), cylindrical shell radius (R), and shell length (L). These four optimization variables $\{T_s, T_h,\ R,\ L\}$ constitute a four-dimensional constrained optimization problem, which is $x = \{x_1,\ x_2, x_3,\ x_4\} = \{T_s, T_h,\ R,\ L\}$. Figure 11 shows the pressure vessel and parameters involved in the design. The mathematical model of the pressure vessel design problem can be expressed as follows:

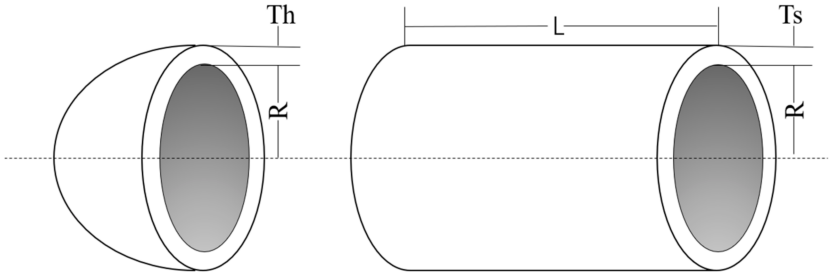


Figure 11. Pressure vessel design problem.

Minimize

$$f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \tag{32}$$

Subject to

$$\begin{cases} g_1(x) = -x_1 + 0.0193x_3 \le 0 \\ g_2(x) = -x_2 + 0.0954x_3 \le 0 \\ g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0 \\ g_4(x) = x_4 - 240 \le 0 \end{cases} \tag{33}$$

where

$$0.0625 \le x_i \le 6.1875,\ i = 1,\ 2;\ 10 \le x_i \le 200,\ i = 3,\ 4; \tag{34}$$

Table 8 shows the seven test algorithms running independently for 51 times to solve the pressure vessel design problem, the optimal value of the optimization results of each algorithm, and the corresponding optimal solution (the optimal design value of the optimization variable). As can be seen from the table, compared with all other algorithms, MjSO achieved better results in the pressure vessel design problem: the MjSO of the objective function value of 5885.522643154713 obtained the best solution.

Table 8. Comparison of results for pressure vessel design problem.

Algorithm	Variable				Target Cost
	T_s	T_h	R	L	
DE	0.8231	0.4453	42.9230	176.7356	6301.5664
LSHADE	0.8168	0.4472	42.1412	177.1231	6138.8931
EBLSHADE	0.7802	0.3856	40.4292	198.4964	5889.3216
ELSHADE-SPACMA	0.8125	0.4375	42.0913	176.7465	6061.0777
SALSHADE-cnEPSin	0.7929	0.3914	41.1773	188.3950	5912.7115
jSO	0.8036	0.3972	41.6392	182.4120	5930.3137
MjSO	0.7782	0.3847	40.3201	199.9975	5885.5226

6.2. Tension/Compression Spring Design Problem

The goal of this design problem is to minimize the weight of the tension/compression spring shown in Figure 12. The design has three design variables: the wire diameter (d), mean coil diameter (D), and number of active coils (N), which are subject to one linear

and three nonlinear inequality constraints on shear stress, surge frequency, and deflection. The mathematical model of tension/compression spring design problem can be expressed as follows:

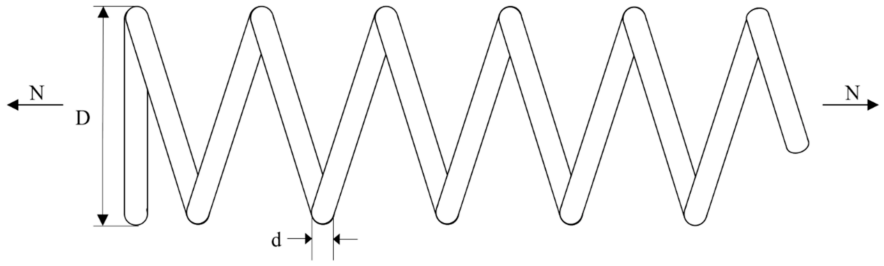


Figure 12. Compression/tension spring design problem.

Consider

$$\vec{x} = [x_1\ x_2\ x_3] = [d\ D\ N] \tag{35}$$

minimize

$$f(x) = (2 + x_1)x_2x_3^2 \tag{36}$$

subject to

$$\begin{cases} g_1(x) = 1 - \frac{x_1x_2^3}{71785x_3^4} \leq 0 \\ g_2(x) = \frac{4x_2^2 - x_2x_3}{12566\left(x_2x_3^3 - x_3^4\right)} + \frac{1}{5108x_3^2} - 1 \leq 0 \\ g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_1} \leq 0 \\ g_4(x) = \frac{x_2 + x_3}{1.5} - 1 \leq 0 \end{cases} \tag{37}$$

where

$$2 \leq x_1 \leq 15, 0.25 \leq x_2 \leq 1.3, 0.05 \leq x_3 \leq 2 \tag{38}$$

As can be seen from Table 9, compared with all other algorithms, MjSO achieved better results in the design of tension/compression springs: the MjSO with the objective function value of 0.012665255009 got the best solution.

Table 9. Comparison of results for compression/tension spring design problem.

Algorithm	Variable			Target Weight
	d	*D*	*N*	
DE	0.0592	0.4983	8.8980	0.0172
LSHADE	0.0524	0.3532	11.6824	0.0133
EBLSHADE	0.0500	0.3171	14.1417	0.0127
ELSHADE-SPACMA	0.0519	0.3487	11.8145	0.0129
SALSHADE-cnEPSin	0.0503	0.3159	14.250	0.0128
jSO	0.0562	0.4754	6.6670	0.0130
MjSO	0.0516	0.3597	11.2880	0.0126

6.3. Welded Beam Design Problem

The purpose of the welded beam design problem is to minimize the cost of the welded beam shown in Figure 13. It contains four design variables: the thickness of the weld (h), length of the clamped bar (l), the height of the bar (t), and thickness of the bar (b), which are subject to two linear and five nonlinear inequality constraints on shear stress, bending stress in the beam, buckling load, and end deflection of the beam. The design is expressed as follows:

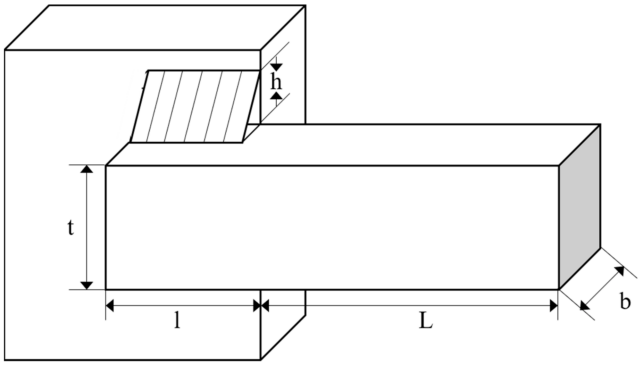


Figure 13. Welding beam design problem.

Consider

$$\vec{x} = [x_1\ x_2\ x_3\ x_4] = [h\ l\ t\ b] \tag{39}$$

Minimize

$$f(x) = 1.10471x_2 + 0.04811x_3x_4(14.0 + x_2) \tag{40}$$

Subject to

$$\begin{cases} g_1(x) = \tau(x) - 13600 \le 0 \\ g_2(x) = \sigma(x) - 30000 \le 0 \\ g_3(x) = x_1 - x_4 \le 0 \\ g_4(x) = 6.000 - PC(x) \le 0 \\ g_5(x) = 0.1047x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 0.5 \le 0 \\ g_6(x) = \delta(x) - 0.25 \le 0 \\ g_7(x) = 0.125 - x_1 \le 0 \end{cases} \tag{41}$$

where

$$\begin{cases} \tau(x) = \sqrt{(\tau')^2 + (\tau'')^2 + \frac{2\tau'\tau''x_2}{2R}} \\ \tau' = \frac{P}{\sqrt{2}x_1x_2} \\ \tau'' = \frac{MP}{J},\ M = P\left(L + \frac{x_2}{2}\right) \\ R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2} \end{cases} \tag{42}$$

$$\begin{cases} J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\} \\ \sigma(x) = \frac{6PL}{x_4x_3^2},\ \delta(x) = \frac{4PL^3}{x_4x_3^3} \\ PC(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \\ P = 6000lb,\ L = 14\ in,\ E = 30 \times 10^6\ psi,\ G = 12 \times 10^6\ psi \end{cases} \tag{43}$$

$$0.1 \le x_1 \le 2,\ 0.1 \le x_2 \le 10,\ 0.1 \le x_3 \le 10,\ 0.1 \le x_4 \le 2 \tag{44}$$

Table 10 presents the best weights and the optimal values for the decision variables from MjSO and several other algorithms. As can be seen from the table, when the four parameters h, l, t, and b are set as 0.2057, 3.4704, 9.0366, and 0.2057, respectively, the minimum manufacturing cost with MjSO is 1.7248. The solution from MjSO is better than the solutions from the other algorithms.

Table 10. Comparison of results for welded beam design problem.

Algorithm	Variable				Target Cost
	h	*l*	*t*	*b*	
DE	0.2389	3.4067	9.6383	0.2901	2.0701
LSHADE	0.2134	3.5601	8.4629	0.2346	1.8561
EBLSHADE	0.2087	6.7221	9.3673	0.4217	1.7583
ELSHADE-SPACMA	0.1947	3.7831	9.1234	0.2077	1.7796
SALSHADE-cnEPSin	0.2023	3.5442	9.0366	0.2057	1.7280
jSO	0.2147	3.3841	8.8103	0.2195	1.7890
MjSO	0.2057	3.4704	9.0366	0.2057	1.7248

In this section, the proposed algorithm MjSO was used to solve three kinds of engineering problems and compared with the other six algorithms. There is a phenomenon worthy of analysis and explanation, as can be seen from the results: (1) in the three engineering problems, MjSO achieved the best results, which is significantly better than the DE series basic algorithms (DE, LSHADE). (2) For the latest LSHADE variant algorithm, in pressure vessel design problem, the result of MjSO was close to EBLSHADE; in the tension/compression spring design problem and welded beam design problem, the result of MjSO was not much different from the result of SALSHADE-cnEPSin and EBLSHADE. (3) Through Tables 3 and 8–10, it can be found that the performance of MjSO, EBLSHADE, and SALSHADE-cnEPSin on engineering problems was consistent with the performance on D = 10 CEC'17 benchmark functions. This shows the effectiveness of the algorithm. (4) Due to the accuracy of the results generated by the algorithm on the model [40], it can be found that although the numerical results are relatively close, the progress made by MjSO is still significant.

7. Conclusions

This article proposes an improved jSO algorithm (MjSO) for engineering optimization design problems. In our proposed MjSO algorithm, first, a parameter control strategy based on a symmetric search process is applied for the scaling factor to balance exploration and exploitation. In addition, in order to improve the loss of population diversity in the evolutionary process, a parameter adaptive mechanism based on the "cosine similarity" is proposed. And a novel opposition-based learning restart mechanism to the MjSO algorithm to jump out of the local optimum. Through comparative experiments on 30 CEC 2017 benchmark functions, MjSO ranked first in the Friedman test in different dimensions, proving the effectiveness and competitiveness of the proposed algorithm. Finally, through the comparative analysis of the optimization design results of the six comparison algorithms, it is clearly shown that the improved algorithm in this paper has good solution quality and effectiveness in solving engineering optimization design problems.

In future work, we will continue to improve the optimization performance of the algorithm and improve its robustness, apply it to more engineering optimization design problems, and improve its ability to solve problems in engineering practice.

Author Contributions: Conceptualization, H.K.; methodology, H.K.; project administration, Z.L. and Y.S.; software, X.S.; validation, X.S. and Q.C.; visualization, Z.L. and Q.C.; formal analysis, H.K.; investigation, Q.C.; resources, Y.S.; data curation, Z.L.; writing—original draft preparation, Z.L.; writing—review and editing, H.K. and X.S.; supervision, X.S.; funding acquisition, Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China, grant number 61663046, 1876166. This research was funded by Open Foundation of Key Laboratory of Software Engineering of Yunnan Province, grant number 2020SE308, 2020SE309. This research was funded by Science Research Foundation of Yunnan Education Committee of China, grant number 2020Y0004.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data availability Statement: Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Clustering and population diversity of jSO and MjSO on the CEC2017 in D = 10.

No	jSO			MjSO		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
f1	51	5.67E+01	3.93E+01	51	4.58E+01	3.17E+01
f2	51	8.84E+01	3.06E+01	51	6.99E+01	2.92E+01
f3	51	8.64E+01	7.00E+00	51	6.97E+01	7.19E+00
f4	51	6.01E+01	1.18E+01	51	4.70E+01	1.13E+01
f5	48	1.19E+03	3.28E+01	50	1.11E+03	3.11E+01
f6	51	9.35E+01	8.47E+00	51	7.21E+01	8.84E+00
f7	49	1.40E+03	1.05E+01	46	1.32E+03	1.02E+01
f8	50	1.19E+03	3.40E+01	51	1.06E+03	3.32E+01
f9	51	9.00E+01	7.45E+00	51	6.95E+01	7.80E+00
f10	47	1.37E+03	8.31E+01	50	1.29E+03	7.08E+01
f11	51	3.90E+02	2.13E+01	51	3.35E+02	1.98E+01
f12	51	2.76E+02	4.31E+01	51	1.86E+02	3.40E+01
f13	51	8.30E+02	1.35E+01	51	6.52E+02	1.35E+01
f14	51	8.12E+02	2.75E+01	47	6.06E+02	3.25E+01
f15	51	3.26E+02	1.81E+01	51	2.62E+02	1.80E+01
f16	49	8.87E+02	1.77E+01	41	5.95E+02	2.61E+01
f17	1	1.89E+03	1.47E+01	2	1.72E+03	2.55E+01
f18	51	3.00E+02	2.57E+01	51	2.40E+02	2.40E+01
f19	51	5.11E+02	3.04E+01	51	3.77E+02	2.78E+01
f20	47	7.52E+02	3.69E+01	42	6.50E+02	3.44E+01
f21	51	5.70E+02	4.52E+01	51	5.78E+02	5.50E+01
f22	51	5.68E+01	7.83E+00	51	4.50E+01	1.11E+01
f23	51	8.08E+02	2.96E+01	51	6.38E+02	2.44E+01
f24	51	8.42E+02	2.89E+01	51	6.77E+02	3.64E+01
f25	51	7.75E+01	1.85E+01	51	6.37E+01	2.34E+01
f26	51	6.12E+01	6.77E+00	51	5.09E+01	7.10E+00
f27	51	1.05E+02	1.74E+01	51	8.44E+01	1.98E+01
f28	51	8.25E+01	2.42E+01	51	6.89E+01	2.71E+01
f29	24	1.59E+03	4.41E+01	24	1.48E+03	3.85E+01
f30	51	1.60E+02	1.32E+01	51	1.24E+02	1.31E+01

Table A2. Clustering and population diversity of jSO and MjSO on the CEC2017 in D = 30.

No	jSO			MjSO		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
f1	51	1.07E+02	2.37E+01	51	9.13E+01	2.82E+01
f2	51	2.81E+02	1.77E+01	51	2.21E+02	1.77E+01
f3	51	1.90E+02	7.05E+00	51	1.72E+02	6.93E+00
f4	51	1.40E+02	8.45E+00	51	1.11E+02	8.20E+00
f5	47	2.32E+03	4.46E+01	48	2.34E+03	4.15E+01
f6	51	1.82E+02	7.46E+00	51	1.52E+02	7.31E+00
f7	34	2.49E+03	1.47E+01	36	2.54E+03	1.29E+01
f8	48	2.26E+03	5.00E+01	42	3.34E+03	4.45E+01
f9	51	1.77E+02	7.19E+00	51	1.47E+02	7.05E+00
f10	17	2.76E+03	1.42E+02	22	2.59E+03	1.48E+02

Table A2. *Cont.*

No	jSO			MjSO		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
f11	51	1.17E+03	2.70E+01	51	1.22E+03	2.87E+01
f12	51	4.77E+02	1.85E+01	51	5.09E+02	2.00E+01
f13	51	8.72E+02	9.56E+00	51	9.66E+02	9.88E+00
f14	29	2.01E+03	2.38E+01	26	2.88E+03	1.80E+01
f15	51	9.12E+02	1.45E+01	51	1.05E+03	1.71E+01
f16	30	2.86E+03	1.61E+01	29	2.80E+03	1.65E+01
f17	12	2.82E+03	5.36E+01	7	3.74E+03	7.99E+01
f18	20	2.50E+03	6.02E+00	15	3.20E+03	8.67E+00
f19	51	1.41E+03	1.46E+01	51	1.42E+03	1.30E+01
f20	14	2.93E+03	3.33E+01	8	2.85E+03	6.11E+01
f21	48	2.36E+03	4.28E+01	50	2.25E+03	4.27E+01
f22	51	1.15E+02	6.68E+00	51	9.66E+01	6.54E+00
f23	50	2.01E+03	4.25E+01	51	1.86E+03	3.62E+01
f24	51	1.82E+03	4.62E+01	51	1.79E+03	3.94E+01
f25	51	1.27E+02	7.39E+00	51	1.04E+02	7.14E+00
f26	51	1.68E+03	3.07E+01	51	1.15E+03	1.80E+01
f27	51	3.44E+02	1.59E+01	51	2.53E+02	2.03E+01
f28	51	1.58E+02	1.15E+01	51	1.31E+02	9.79E+00
f29	29	2.77E+03	5.36E+01	24	2.72E+03	4.26E+01
f30	51	2.85E+02	1.26E+01	51	2.15E+02	1.34E+01

Table A3. Clustering and population diversity of jSO and MjSO on the CEC2017 in D = 50.

No	jSO			MjSO		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
f1	51	1.20E+02	9.52E+00	51	1.32E+02	1.74E+01
f2	51	3.95E+02	1.28E+01	51	3.68E+02	1.32E+01
f3	51	2.43E+02	7.88E+00	51	2.52E+02	7.55E+00
f4	51	1.84E+02	9.18E+00	51	1.70E+02	8.41E+00
f5	44	2.97E+03	5.62E+01	44	2.83E+03	5.57E+01
f6	51	1.95E+02	7.93E+00	51	1.98E+02	7.67E+00
f7	35	2.99E+03	1.86E+01	34	3.03E+03	1.77E+01
f8	45	3.00E+03	5.46E+01	44	2.90E+03	5.47E+01
f9	51	1.90E+02	7.75E+00	51	1.89E+02	7.50E+00
f10	14	3.49E+03	1.29E+02	17	3.43E+03	1.54E+02
f11	51	1.84E+03	3.05E+01	51	1.85E+03	2.97E+01
f12	51	3.67E+02	1.08E+01	51	4.43E+02	1.25E+01
f13	51	7.42E+02	1.12E+01	51	9.27E+02	1.15E+01
f14	51	1.67E+03	1.33E+01	50	1.84E+03	1.43E+01
f15	51	8.34E+02	1.20E+01	51	1.13E+03	1.65E+01
f16	41	3.47E+03	8.42E+00	37	3.41E+03	1.12E+01
f17	15	3.58E+03	7.67E+00	10	3.52E+03	2.62E+01
f18	51	8.67E+02	9.25E+00	51	1.14E+03	9.77E+00
f19	51	1.46E+03	1.23E+01	51	1.63E+03	1.37E+01
f20	24	3.43E+03	2.04E+01	20	4.44E+03	2.19E+01
f21	47	2.94E+03	5.73E+01	43	2.93E+03	5.60E+01
f22	37	1.64E+03	3.82E+01	36	9.58E+02	3.10E+01
f23	48	2.69E+03	5.35E+01	47	2.64E+03	4.69E+01
f24	51	2.59E+03	4.76E+01	50	2.12E+03	3.38E+01
f25	51	1.94E+02	8.44E+00	51	1.69E+02	7.77E+00
f26	51	2.36E+03	3.69E+01	51	1.40E+03	1.43E+01
f27	51	3.00E+02	1.05E+01	51	2.15E+02	8.77E+00
f28	51	1.71E+02	9.04E+00	51	2.54E+02	1.03E+01
f29	32	3.34E+03	5.63E+01	23	3.34E+03	5.09E+01
f30	51	3.28E+02	1.11E+01	51	4.22E+02	3.08E+01

Table A4. Clustering and population diversity of jSO and MjSO on the CEC2017 in D = 100.

No	jSO			MjSO		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
f1	51	1.37E+02	9.05E+00	51	1.94E+02	1.15E+01
f2	51	5.10E+02	1.09E+01	51	5.80E+02	1.04E+01
f3	51	3.75E+02	9.86E+00	51	4.08E+02	9.16E+00
f4	51	2.01E+02	9.29E+00	51	2.32E+02	9.04E+00
f5	50	3.93E+03	7.09E+01	49	4.03E+03	5.59E+01
f6	51	2.06E+02	9.48E+00	51	2.79E+02	8.85E+00
f7	46	4.12E+03	2.19E+01	43	4.02E+03	2.07E+01
f8	45	4.04E+03	6.76E+01	51	3.90E+03	6.33E+01
f9	51	2.00E+02	9.04E+00	51	2.62E+02	8.70E+00
f10	13	4.46E+03	3.06E+02	13	4.52E+03	2.13E+02
f11	51	5.27E+02	9.79E+00	51	1.17E+03	1.14E+01
f12	51	3.30E+02	1.02E+01	51	4.53E+02	9.08E+00
f13	51	7.08E+02	1.34E+01	51	9.23E+02	1.39E+01
f14	51	1.01E+03	1.02E+01	51	1.28E+03	1.30E+01
f15	51	4.84E+02	1.03E+01	51	9.30E+02	9.53E+00
f16	49	4.43E+03	1.16E+01	47	4.60E+03	9.92E+00
f17	33	4.66E+03	2.45E+01	49	4.04E+03	7.65E+01
f18	51	5.82E+02	9.80E+00	51	5.88E+02	1.40E+01
f19	51	6.38E+02	1.07E+01	51	1.48E+03	1.20E+01
f20	26	4.64E+03	1.16E+01	26	4.54E+03	7.67E+01
f21	49	3.87E+03	6.44E+01	51	2.41E+02	9.42E+00
f22	14	4.39E+03	2.67E+01	49	3.95E+03	6.22E+01
f23	50	1.72E+03	2.38E+01	51	7.70E+02	9.25E+00
f24	51	1.01E+03	1.18E+01	51	2.12E+02	8.49E+00
f25	51	2.11E+02	9.56E+00	51	2.53E+02	9.72E+00
f26	51	7.98E+02	9.50E+00	51	2.68E+02	8.62E+00
f27	51	3.31E+02	9.28E+00	51	2.93E+02	8.93E+00
f28	51	2.23E+02	9.42E+00	51	2.31E+02	8.99E+00
f29	51	4.38E+03	4.21E+01	51	4.13E+03	4.22E+01
f30	51	5.09E+02	1.08E+01	51	4.55E+02	1.01E+01

Table A5. The Friedman ranks of comparative algorithms on CEC2017 in D = 10.

Rank	Name	F-Rank
0	MjSO	2.62
1	jSO	3.43
2	SALSHADE-cnEPSin	3.55
3	EBLSHADE	3.6
4	ELSHADE-SPACMA	3.82
5	LSHADE	3.98

Table A6. The Friedman ranks of comparative algorithms on CEC2017 in D = 30.

Rank	Name	F-Rank
0	MjSO	2.1
1	jSO	3.4
2	EBLSHADE	3.43
3	SALSHADE-cnEPSin	3.95
4	LSHADE	4.05
5	ELSHADE-SPACMA	4.07

Table A7. The Friedman ranks of comparative algorithms on CEC2017 in D = 50.

Rank	Name	F-Rank
0	MjSO	1.9
1	ELSHADE-SPACMA	3.27
2	jSO	3.4
3	SALSHADE-cnEPSin	4.05
4	LSHADE	4.15
5	EBLSHADE	4.23

Table A8. The Friedman ranks of comparative algorithms on CEC2017 in D = 100.

Rank	Name	F-Rank
0	MjSO	1.87
1	ELSHADE-SPACMA	3.35
2	SALSHADE-cnEPSin	3.4
3	jSO	3.78
4	EBLSHADE	3.93
5	LSHADE	4.67

Table A9. Related statistical values obtained of Friedman test for $\alpha = 0.05$.

D	Chi-sq′	Prob > Chi-sq′(p)	Critical Value
10	13.73819163	1.74E-02	11.07
30	31.07891492	9.04E-06	11.07
50	38.65745856	2.78E-07	11.07
100	38.16356513	3.50E-07	11.07

References

1. Dhiman, G.; Kumar, V. Multi-objective spotted hyena optimizer: A multi-objective optimization algorithm for engineering problems. *Knowl. Based Syst.* **2018**, *150*, 175–197. [CrossRef]
2. Spall, J. *Introduction to Stochastic Search and Optimization*; Wiley Interscience: New York, NY, USA, 2003.
3. Parejo, J.A.; Ruiz-Cortés, A.; Lozano, S.; Fernandez, P. Metaheuristic optimization frameworks: A survey and benchmarking. *Soft Comput.* **2012**, *16*, 527–561. [CrossRef]
4. Zhou, A.; Qu, B.-Y.; Li, H.; Zhao, S.-Z.; Suganthan, P.N.; Zhang, Q. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm Evol. Comput.* **2011**, *1*, 32–49. [CrossRef]
5. Droste, S.; Jansen, T.; Wegener, I. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory Comput. Syst.* **2006**, *39*, 525–544. [CrossRef]
6. Hoos, H.H.; Stützle, T. *Stochastic Local Search: Foundations and Applications*; Elsevier: Amsterdam, The Netherlands, 2004.
7. Holland, J. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Application to Biology*; Control and artificial intelligence, University of Michigan Press: Ann Arbor, MI, USA, 1975; pp. 106–111.
8. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; IEEE; pp. 1942–1948.
9. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
10. Wienholt, W. Minimizing the system error in feedforward neural networks with evolution strategy. In Proceedings of the International Conference on Artificial Neural Networks, Amsterdam, The Netherlands, 13–16 September 1993; Springer: London, UK, 1993; pp. 490–493.
11. Eltaeib, T.; Mahmood, A. Differential evolution: A survey and analysis. *Appl. Sci.* **2018**, *8*, 1945. [CrossRef]
12. Arafa, M.; Sallam, E.A.; Fahmy, M. An enhanced differential evolution optimization algorithm. In Proceedings of the 2014 Fourth International Conference on Digital Information and Communication Technology and Its Applications (DICTAP), Bangkok, Thailand, 6–8 May 2014; pp. 216–225.
13. Liu, X.-F.; Zhan, Z.-H.; Zhang, J. Dichotomy guided based parameter adaptation for differential evolution. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, Madrid, Spain, 11–15 July 2015; pp. 289–296.
14. Sallam, K.M.; Sarker, R.A.; Essam, D.L.; Elsayed, S.M. Neurodynamic differential evolution algorithm and solving CEC2015 competition problems. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015; pp. 1033–1040.

15. Viktorin, A.; Pluhacek, M.; Senkerik, R. Network based linear population size reduction in SHADE. In Proceedings of the 2016 International Conference on Intelligent Networking and Collaborative Systems (INCoS), Ostrawva, Czech Republic, 7–9 September 2016; pp. 86–93.
16. Bujok, P.; Tvrdík, J.; Poláková, R. Evaluating the performance of shade with competing strategies on CEC 2014 single-parameter test suite. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 5002–5009.
17. Viktorin, A.; Pluhacek, M.; Senkerik, R. Success-history based adaptive differential evolution algorithm with multi-chaotic framework for parent selection performance on CEC2014 benchmark set. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 4797–4803.
18. Poláková, R.; Tvrdík, J.; Bujok, P. L-SHADE with competing strategies applied to CEC2015 learning-based test suite. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 4790–4796.
19. Poláková, R.; Tvrdík, J.; Bujok, P. Evaluating the performance of L-SHADE with competing strategies on CEC2014 single parameter-operator test suite. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 1181–1187.
20. Liu, Z.-G.; Ji, X.-H.; Yang, Y. Hierarchical differential evolution algorithm combined with multi-cross operation. *Expert Syst. Appl.* **2019**, *130*, 276–292. [CrossRef]
21. Bujok, P.; Tvrdík, J. Adaptive differential evolution: SHADE with competing crossover strategies. In Proceedings of the International Conference on Artificial Intelligence and Soft Computing, Zakopane, Poland, 14–18 June 2015; pp. 329–339.
22. Viktorin, A.; Senkerik, R.; Pluhacek, M.; Kadavy, T.; Zamuda, A. Distance based parameter adaptation for differential evolution. In Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 27 November–1 December 2017; pp. 1–7.
23. Viktorin, A.; Senkerik, R.; Pluhacek, M.; Kadavy, T. Distance vs. Improvement Based Parameter Adaptation in SHADE. In Proceedings of the Computer Science On-Line Conference, Vsetin, Czech Republic, 25–28 April 2018; 2018; pp. 455–464.
24. Molina, D.; Herrera, F. Applying Memetic algorithm with Improved L-SHADE and Local Search Pool for the 100-digit challenge on Single Objective Numerical Optimization. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 7–13.
25. Awad, N.H.; Ali, M.Z.; Suganthan, P.N. Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June 2017; pp. 372–379.
26. Zhao, F.; He, X.; Yang, G.; Ma, W.; Zhang, C.; Song, H. A hybrid iterated local search algorithm with adaptive perturbation mechanism by success-history based parameter adaptation for differential evolution (SHADE). *Eng. Optim.* **2019**. [CrossRef]
27. Das, S.; Mullick, S.S.; Suganthan, P.N. Recent advances in differential evolution—An updated survey. *Swarm Evol. Comput.* **2016**, *27*, 1–30. [CrossRef]
28. Hansen, N. The CMA evolution strategy: A comparing review. In *Towards a New Evolutionary Computation*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 75–102.
29. Das, S.; Suganthan, P.N. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evol. Comput.* **2010**, *15*, 4–31. [CrossRef]
30. Al-Dabbagh, R.D.; Neri, F.; Idris, N.; Baba, M.S. Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy. *Swarm Evol. Comput.* **2018**, *43*, 284–311. [CrossRef]
31. Tanabe, R.; Fukunaga, A.S. Improving the search performance of SHADE using linear population size reduction. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 1658–1665.
32. Brest, J.; Maučec, M.S.; Bošković, B. Single objective real-parameter optimization: Algorithm jSO. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June 2017; pp. 1311–1318.
33. Mohamed, A.W.; Hadi, A.A.; Fattouh, A.M.; Jambi, K.M. LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June 2017; pp. 145–152.
34. Hadi, A.A.; Wagdy, A.; Jambi, K. *Single-Objective Real-Parameter Optimization: Enhanced LSHADE-SPACMA Algorithm*; King Abdulaziz Univ.: Jeddah, Saudi Arabia, 2018.
35. Salgotra, R.; Singh, U.; Singh, G. Improving the adaptive properties of lshade algorithm for global optimization. In Proceedings of the 2019 International Conference on Automation, Computational and Technology Management (ICACTM), London, UK, 24–26 April 2019; pp. 400–407.
36. Yang, M.; Li, C.; Cai, Z.; Guan, J. Differential evolution with auto-enhanced population diversity. *IEEE Trans. Cybern.* **2014**, *45*, 302–315. [CrossRef] [PubMed]
37. Hongtan, C.; Zhaoguang, L. Improved Differential Evolution with Parameter Adaption Based on Population Diversity. In Proceedings of the 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), Chongqing, China, 27–29 June 2018; pp. 901–905.
38. Wang, J.-Z.; Sun, T.-Y. Control the Diversity of Population with Mutation Strategy and Fuzzy Inference System for Differential Evolution Algorithm. *Int. J. Fuzzy Syst.* **2020**, *22*, 1979–1992. [CrossRef]

39. He, X.; Zhou, Y. Enhancing the performance of differential evolution with covariance matrix self-adaptation. *Appl. Soft Comput.* **2018**, *64*, 227–243. [CrossRef]
40. Awad, N.H.; Ali, M.Z.; Mallipeddi, R.; Suganthan, P.N. An improved differential evolution algorithm using efficient adapted surrogate model for numerical optimization. *Inf. Sci.* **2018**, *451*, 326–347. [CrossRef]
41. Subhashini, K.; Chinta, P. An augmented animal migration optimization algorithm using worst solution elimination approach in the backdrop of differential evolution. *Evol. Intell.* **2019**, *12*, 273–303. [CrossRef]
42. Awad, N.; Ali, M.; Liang, J.; Qu, B.; Suganthan, P.; Definitions, P. Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June 2017.
43. Zhang, J.; Sanderson, A.C. JADE: Adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958. [CrossRef]
44. Brest, J.; Maučec, M.S.; Bošković, B. iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, Canada, 24–29 July 2016; pp. 1188–1195.
45. Fan, Q.; Yan, X. Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies. *IEEE Trans. Cybern.* **2015**, *46*, 219–232. [CrossRef]
46. Wang, D.; Lu, H.; Bo, C. Visual tracking via weighted local cosine similarity. *IEEE Trans. Cybern.* **2014**, *45*, 1838–1850. [CrossRef]
47. Van Dongen, S.; Enright, A.J. Metric distances derived from cosine similarity and Pearson and Spearman correlations. *arXiv* **2012**, arXiv:1208.3145.
48. Luo, J.; He, F.; Yong, J. An efficient and robust bat algorithm with fusion of opposition-based learning and whale optimization algorithm. *Intell. Data Anal.* **2020**, *24*, 581–606. [CrossRef]
49. Shekhawat, S.; Saxena, A. Development and applications of an intelligent crow search algorithm based on opposition based learning. *ISA Trans.* **2020**, *99*, 210–230. [CrossRef] [PubMed]
50. Ergezer, M.; Simon, D. Mathematical and experimental analyses of oppositional algorithms. *IEEE Trans. Cybern.* **2014**, *44*, 2178–2189. [CrossRef] [PubMed]
51. Tizhoosh, H.R. Opposition-based reinforcement learning. *J. Adv. Comput. Intell. Intell. Inform.* **2006**, *10*. [CrossRef]
52. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M. Opposition-based differential evolution. *EEE Trans. Evol. Comput.* **2008**, *12*, 64–79. [CrossRef]
53. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M. Quasi-oppositional differential evolution. In Proceedings of the 2007 IEEE Congress on Evolutionary Computation, San Sebastian, Spain, 5–8 June 2017; pp. 2229–2236.
54. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
55. Hutter, F.; Hoos, H.H.; Leyton-Brown, K.; Stützle, T. ParamILS: An automatic algorithm configuration framework. *J. Artif. Intell. Res.* **2009**, *36*, 267–306. [CrossRef]
56. Mohamed, A.W.; Hadi, A.A.; Jambi, K.M. Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization. *Swarm Evol. Comput.* **2019**, *50*, 100455. [CrossRef]
57. Salgotra, R.; Singh, U.; Saha, S.; Nagar, A. New improved salshade-cnepsin algorithm with adaptive parameters. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 3150–3156.
58. Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Kdd, Menlo Park, CA, USA, August 1996; pp. 226–231.
59. Deza, M.M.; Deza, E. Encyclopedia of distances. In *Encyclopedia of Distances*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–583.
60. Poláková, R.; Tvrdík, J.; Bujok, P.; Matoušek, R. Population-size adaptation through diversity-control mechanism for differential evolution. In Proceedings of the MENDEL, 22th International Conference on Soft Computing, Brno, Czech Republic, 8–10 June 2016; pp. 49–56.
61. Varaee, H.; Ghasemi, M.R. Engineering optimization based on ideal gas molecular movement algorithm. *Eng. Comput.* **2017**, *33*, 71–93. [CrossRef]
62. Askarzadeh, A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Comput. Struct.* **2016**, *169*, 1–12. [CrossRef]