biomimetics

MDPI

Article

SaMDE: A Self Adaptive Choice of DNDE and SPIDE Algorithms with MRLDE

Pravesh Kumar [1,†] and Musrrat Ali [2,*,†]

1 REC Bijnor, Chandpur 246725, India; praveshtomariitr@gmail.com
2 Department of Basic Sciences, PYD, King Faisal University, Al Ahsa 31982, Saudi Arabia
* Correspondence: mkasim@kfu.edu.sa
† These authors contributed equally to this work.

Abstract: Differential evolution (DE) is a proficient optimizer and has been broadly implemented in real life applications of various fields. Several mutation based adaptive approaches have been suggested to improve the algorithm efficiency in recent years. In this paper, a novel self-adaptive method called SaMDE has been designed and implemented on the mutation-based modified DE variants such as modified randomized localization-based DE (MRLDE), donor mutation based DE (DNDE), and sequential parabolic interpolation based DE (SPIDE), which were proposed by the authors in previous research. Using the proposed adaptive technique, an appropriate mutation strategy from DNDE and SPIDE can be selected automatically for the MRLDE algorithm. The experimental results on 50 benchmark problems taken of various test suits and a real-world application of minimization of the potential molecular energy problem validate the superiority of SaMDE over other DE variations.

Keywords: differential evolution; optimization; evolutionary algorithms; mutation; self-adaptive

1. Introduction

Optimization problems occur almost in each field of engineering and science branches. In general, these problems may be classified as non-linear, non-convex non-continuous, non-differentiable, or having several local optimum values and therefore solving these problems is beyond the capacity of traditional methods due to their certain limitations. As a result, so many evolutionary algorithms (EAs) like Particle swarm optimization (PSO), Differential evolution (DE), Artificial Bee colony (ABC), Cuckoo Search (CS), Teaching Learning based optimization (TLBO), Gray Wolf optimization (GWO), Reptile Search Algorithm (RSA), Whale Optimization Algorithm (WOA), and Manta Ray Foraging Optimization (MRFO), etc., have emerged during some past years to handle such complicated situations. The prime benefit of EAs over the traditional system is that they only need the objective function, whereas the other calculus properties like differentiability and continuity are not necessarily important.

The Differential evolution algorithm (DE) [1] comes under the EAs categories and has gained a reputation since the last few years as a highly capable and robust optimizer. There are many reasons for its popularity among the researchers such as its compact size, requirements of only few control parameters, easy implementation quality, quick convergence rate, etc. Due to its many advantages, it has been frequently used to deal with large scale, constrained, dynamic, multi-objective and multimodal optimization problems.

Despite many positive attributes, DE also has some deficiencies like population diversity and stagnation problems and consequently it does not work in many situations or gives a slow convergence speed. Alot of research has been carried out in the past two decades to reduce its deficiencies and make it a more efficient algorithm.

Initially Storn & Price [1] and Liu & Lampinen [2] have suggested that control parameters F and Cr should have a value between (0.5, 1) and (0.8, 1), respectively. Later, several

Citation: Kumar, P.; Ali, M. SaMDE: A Self Adaptive Choice of DNDE and SPIDE Algorithms with MRLDE. *Biomimetics* **2023**, *8*, 494. https://doi.org/10.3390/biomimetics8060494

Academic Editor: Huiling Chen

Received: 10 September 2023
Revised: 8 October 2023
Accepted: 12 October 2023
Published: 18 October 2023

studies have been carried out on the selection of suitable parameter settings for DE. A decent literature survey for control parameters may be found in [3]. To keep away from the manual tuning of parameters, researchers have suggested adaptive/ self-adaptive setting of parameters, where the control parameters are changed vigorously based on the response of the search space in place of taking a fixed value. A few works in the improvement of adaptive/self-adaptive methods of control parameters values are suggested in [4–12].

Population initialization methods also play a significant part in enhancing the performance of any population-based algorithm. Teo [13] proposed an exploring dynamic self-adaptive populations method for DE. Rahnamayan et al. [14] suggested an opposition-based method to initialize population for DE. Later, some noticeable works on population initialization methods and population size adaptation have been completed in [15–21].

A crossover and selection operation-based modifications given in [22–24] are also completed to enhance the performance of DE.

Other modifications include development of novel mutation techniques and their adaptive/self-adaptive strategies. Some modified mutation based DE methods are Trigonometric DE (TDE) [25], donor mutation DE (DNDE) [26,27], DE with random localization (DERL) [28], DE with hybrid mutation [29], DE with external archive (JADE) [30], DE with neighborhood mutation (DEGL) [31], Proximity-based mutation operators for DE [32], DE with modified random localization (MRLDE) [33], interpolation rules based mutation [34], ranking based mutation strategy [35], DE with multiple mutation strategies [36], iLSHADE [37], random perturbation based DE [38], IMODE [39], HiPDE [40], and so on.

Several research studies are carried out in the development of adaptive mutation strategies. Some recent research regarding adaptive DE is as follows. Qin and Suganthan [41] proposed SaDE to implement two mutation strategies *DE/rand/1*" and *DE/current-to-best/1* simultaneously. In SaDE the trial vector was created by either success ratios based strategy in the last 50 generations by using adaptive probability. Later, Qin et al. [42] extended this work for four mutation strategies. In their proposed work, both control parameter values and trial vector creation strategies are regularly self-adapted by learning from their prior experiences of the solutions. In SaDE, an adaptive rule is proposed for updating the probability of each variant according to their corresponding success or failure in performance.

Gong et al. [43] proposed a strategy adaptation approach based on the probability matching technique being fed by comparative fitness upgrading. They have also suggested diverse categories of strategy adaptation methods in which a strategy parameter is utilized to manage the selection of different strategies, and two straightforward strategy adaptation approaches are employed to revise the parameter. Later on, Mallipeddi et al. [44] proposed a new variant, EPSDE, with a collection of different mutation strategies and parameters. In this variant each mutation strategy is started randomly, and any mutation strategy is stored to the next generation for which the produced trial vector gives better fitness values than its target vector. Otherwise, it is selected randomly from the preceding winning strategies stored with identical probability. Some other strategy adaptation based DE variants are SspDE [45], composite DE (CoDE) [46], MDE_pBX [47], ISADE [48], Adaptive DE [49], and TS-MSCDE [50]. Various studies on the changes and applications of DE have been conducted periodically. Several of these excellent studies are documented in [51–53].

Apart from modifications in basic operations, a lot of research has also been completed to enhance the local search capability and introduce some additional features. Some of these recently developed variants are LSDE [54], DEGOS [55], CJADE [56], PAIDDE [57], TRADE [58] and so on.

In this study, a self-adaptive technique named as SaMDE has been developed in an effort to continue boosting the performance of DE. This method integrates MRLDE, DNDE, and SPIDE algorithms and takes advantage of all of their benefits in a single location. The selection of these tactics is motivated by their performance. These tactics have demonstrated excellent performance in the past, and we anticipate that their hybridization will further improve the performance of the proposed scheme. SaMDE begins with the provision of a systematic and superior way for selecting individuals for mutation operation by MRLDE, followed by the implementation of DNDE and SPIDE by the proposed self-adapting technique. The advantage of this technique is that if one variant fails, the other variant will automatically activate, and the algorithm will continue to run until the reaches the desired outcomes.

The rest of the paper is organized as follows: In the second section, a brief description of DE, MRLDE, DNDE, and SPIDE is offered. The SaMDE methodology is discussed in Section 3. Section 4 defines the benchmark test tasks and potential energy problem. In Section 5, the numerical and statistical results are analyzed and discussed. Section 6 concludes the entire investigation.

2. Background

2.1. Differential Evolution Algorithm

Similar to other EAs, DE also starts with a uniformly distributed set of solutions called population within a bound domain $[X_{low}, X_{upper}]$. Let at any generation g, the population set is $P^{(g)} = \left\{X_i^{(g)} | i = 1, 2, \cdots, NP\right\}$ where each $X_i^{(g)} = \left\{x_{i,j}^{(g)} : i = 1, 2 \ldots NP; j = 1, 2, \ldots d\right\}$ is a d-dimensional vector and NP is the size of population, then it can be initially generated by Equation (1).

$$X_i^{(g)} = X_{low} + rand(0,1) \times (X_{upper} - X_{low}) \tag{1}$$

Next, the evolution phase starts where new positions for each individual are generated through mutation and crossover operation and then selection operation is applied to choose the best fitted vector to the next generation population. A DE algorithm can be written as *DE/a/b/c*, where a, b and c represent the number of vector differences, mutation and crossover strategy, respectively. There are several mutation techniques for DE, however we have utilized solely the *DE/rand/1/bin* strategy throughout this investigation. The mutation and crossover strategies for *DE/rand/1/bin* are defined as below.

Mutation: For any vector $X_i^{(g)}$, the mutant vector $M_i^{(g+1)}$ is generated through Equation (2).

$$M_i^{(g+1)} = X_r^{(g)} + F \times (X_s^{(g)} - X_t^{(g)}) \tag{2}$$

where $X_r^{(g)}$, $X_s^{(g)}$ and $X_t^{(g)}$ are three mutually different vectors randomly chosen from $P^{(g)}$ different from $X_i^{(g)}$. The vector $X_i^{(g)}$ is called as the base vector and $F \in (0,1]$ is a parameter which is used to control the amplification of the variation $(X_s^{(g)} - X_t^{(g)})$.

Crossover: Crossover operation is required to create a trail vector say $U_i^{(g+1)} = \left\{u_{i,j}^{(g+1)} : j = 1, 2, \ldots d\right\}$ by crossing the components of target vector $X_i^{(g)} = \left\{x_{i,j}^{(g)} : j = 1, 2, \ldots d\right\}$ and mutant vector $M_i^{(g+1)} = \left\{m_{i,j}^{(g+1)} : j = 1, 2, \ldots d\right\}$ by Equation (3).

$$u_{i,j}^{(g+1)} = \begin{cases} m_{i,j}^{(g+1)} & if \quad rand_j(0,1) \le C_R(OR)\ j \in randi(d) \\ x_{i,j}^{(g)} & \quad otherwise \end{cases} \tag{3}$$

where $C_R \in (0, 1)$ known as the crossover parameter and *randi* (d) denotes the random index j from {1, 2,. . .d} which insures that at least one component in the trail vector should be chosen from the mutant vector.

Selection: This procedure selects the optimal vector from the target and trail vectors for the next generation population based on their fitness value as determined by Equation (4).

$$X_i^{(g+1)} = \begin{cases} U_i^{(g+1)} & if \quad fun\,(U_i^{(g+1)}) \le fun(X_i^{(g)}) \\ X_i^{(g)} & else \end{cases} \tag{4}$$

2.2. Mutation Based Modified DE Variants

The primary objective of the mutation operation is to provide a new position for every randomly selected vector by adding it to the weighted difference of two different vectors. The vector that must be perturbed is known as the base vector, while the other two are known as the difference vectors. According to Kaelo and Ali [28], the newly created mutant vector is dependent on the nature of the base vector; hence, a suitable selection of the base vector may help to increase the convergence rate of the algorithm. Inspired by this concept, we have previously presented three new modified DE variants, DNDE [27], MRLDE [33], and SPIDE [34], in which the base vector is taken in an improved manner rather than randomly from the population.

This section will now provide a concise description of these variants.

2.2.1. MRLDE

A modified version of the DERL algorithm was proposed in 2012 [33], and it is known as modified randomized localization-based DE (MRLDE). The choice of random vectors to carry out the mutation is the only distinction between MRLDE and DE. MRLDE divides the entire population into three segments, say, $P_{best}^{(g)}$, $P_{medium}^{(g)}$ and $P_{worst}^{(g)}$ of size λ_1, λ_2 and λ_3 by the fitness values and then select the vectors $X_r^{(g)}$, $X_s^{(g)}$ and $X_t^{(g)}$ from the $P_{best}^{(g)}$, $P_{medium}^{(g)}$ and $P_{worst}^{(g)}$, respectively, to run the mutation operation as defined in the DE algorithm. The effectiveness of the algorithm has been implemented in some real-life problems such as image enhancement [59], economy load dispatch problem [60] and noise source identification [61].

2.2.2. Sequential Parabolic Interpolation Based DE (SPIDE)

The concept of selecting a base vector in this form is inspired by sequential parabolic interpolation (SPI), a root finding approach for the equation $q(x) = 0$. If x_1, x_2 and x_3 are three points with the function values $q(x_1)$, $q(x_2)$) and $q(x_3)$, respectively, then the next root estimation by the SPI method is given by Equation (5).

$$x_4 = x_1 + \frac{1}{2}\frac{(x_1 - x_2)^2\{q(x_1) - q(x_3)\} - (x_1 - x_3)^2\{q(x_1) - q(x_2)\}}{(x_1 - x_2)\{q(x_1) - q(x_3)\} - (x_1 - x_3)\{q(x_1) - q(x_2)\}} \tag{5}$$

In SPIDE, we replace x_1, x_2 and x_3 by $X_r^{(g)}$, $X_s^{(g)}$ and $X_s^{(g)}$, respectively, in Equation (5) and generate a new vector say $X_Q^{(g+1)}$. Next we select the base vector among $X_Q^{(g+1)}$ or $X_{tb}^{(g)}$ by settinga probability (P_s) where $X_{tb}^{(g)}$ denotes the best vector among $X_r^{(g)}$, $X_s^{(g)}$ and $X_s^{(g)}$. As a next estimation root, $X_Q^{(g+1)}$ gives a minimum fitness value than $X_r^{(g)}$, $X_s^{(g)}$ and $X_s^{(g)}$ and hence helps to increase the algorithm's convergence speed. Refer to [34] to understand more about SPIDE and its operation.

2.2.3. Donor Mutation Based DE (DNDE)

Fan et al. [26] suggested that the base vector can be taken as a weighted mean of selected vectors $X_r^{(g)}$, $X_s^{(g)}$ and $X_s^{(g)}$. We have used this idea and selected the base vector from these random vectors based on random localization approach [28] and weighted mean as suggested by [26] and named this variant as 'DNDE'. In this approach, the weighted vector $X_w^{(g+1)}$ is generated by convex combination of three randomly selected vectors $X_r^{(g)}$, $X_s^{(g)}$ and $X_s^{(g)}$ by Equation (6).

$$X_w^{(g+1)} = \nu_1 X_r^{(g)} + \nu_2 X_s^{(g)} + \nu_3 X_t^{(g)} \tag{6}$$

Here ν_i = 1, 2, 3 are uniform random numbers in the range of (0, 1) and should satisfy the condition $\sum_i \nu_i = 1$.

Now similar to SPIDE, the base vector is selected among $X_w^{(g+1)}$ or $X_{tb}^{(g)}$ by setting a probability (P_d). In addition, [27] provides a comprehensive discussion of DNDE and its efficacy. The SPIDE and DNDE pseudo codes are listed in Algorithm 1.

Algorithm 1 *SPIDE* and *DNDE*

For i = 1 to *NP do*
    Select three vectors $X_r^{(g)} X_s^{(g)}$ and $X_t^{(g)}$ such that $X_r^{(g)} \neq X_s^{(g)} \neq X_t^{(g)} \neq X_i^{(g)}$
And find best vector $X_{tb}^{(g)}$ between these vectors by their fitness
    /∗∗∗∗∗∗∗base vector by SPIDE ∗∗∗∗∗∗∗/
Obtained $X_Q^{(g+1)}$ by Equation (5)
    if (*rand* (0,1) < P_s)
$X_r^{(g)} = X_Q^{(g+1)}$
    Else
        $X_r^{(g)} = X_{tb}^{(g)}$
end if
/*******************************/
    */******base vector by DNDE ******/*
    Obtained $X_w^{(g+1)}$ by Equation (6)
    if (*rand* (0,1) < P_d)
$X_r^{(g)} = X_w^{(g+1)}$
    Else
        $X_r^{(g)} = X_{tb}^{(g)}$
    end if
/****************************/
    Perform mutation, Crossover and Selection
end for

3. Proposed Self Adaptive Approach (SaMDE)

As noted by numerous researchers, the mutation strategies are significantly reliant on the problems under consideration. A substantial amount of time may be required to solve a single problem by attempting numerous ways. This dilemma motivated us to create a Self-adaptive mutation approach for DE (SaMDE) that can handle difficulties more effectively.

Proposed SaMDE is a fusion of MRLDE, DNDE and SPIDE. As recommended by the MRLDE method, the entire search space is initially partitioned into three sub-regions, and then the DNDE and SPIDE algorithms are utilized adaptively. The proposed rule updates the probability for mutation schemes based on their performance in any generation. Similar to a concept described by Qin et al. [41,42] in SaDE, where probability rules are updated based on the success and failure ratio of the variants, and then any variant is activated based on its probability. Instead of random activation, the variant with the highest likelihood or success rate will be activated in SaMDE.

In SaMDE, probabilities are initially assigned at random to each variant of DNDE and SPIDE, and then the evaluation procedure begins for the variant with the highest probability. This version shall be referred to as active variant.

Now, the active variant is assigned a positive rank if the trial vector created by this technique is picked for the following generation; otherwise, a negative rank is assigned. At the conclusion of a generation, all positive and negative ranks for the active variant are added together. Let p_1 and p_2 represent the probabilities for DNDE and SPIDE, respectively, and let *RP* and *RN* represent the total positive and negative rankings in any generation. The probability is then updated based on the following rules:

Case 1: When $p_1 \geq p_2$, i.e., DNDE is active:

$$p_1 = \frac{RP}{RP + RN}; \quad p_2 = \frac{RN}{RP + RN} \tag{7}$$

Case 2: When $p_2 > p_1$, i.e., SPIDE is active:

$$p_2 = \frac{RP}{RP + RN}; \quad p_1 = \frac{RN}{RP + RN} \tag{8}$$

These criteria assist in updating the probability of a self-adaptive procedure. If the active version performs better in a generation, it will be used in the following generation as well; otherwise, another variant will become the active variant. The updating criteria is immediately influenced by the rejection and acceptance performance (or acceptance rate) of trial vectors into the subsequent generation.

For example, if the acceptance rate for trial vectors generated by the active DE variant is better than its rejection rate then it will imply that the total number of positive ranks is higher than the total negative ranks. Therefore, the corresponding probability for the active variant will be high giving it a chance to continue in the next generation.

Similarly, a higher rejection rate of trial vectors increases the negative ranks decreasing the probability of the current active variant to be continued in the next generation. The operation of SaMDE is also illustrated in Figure 1.

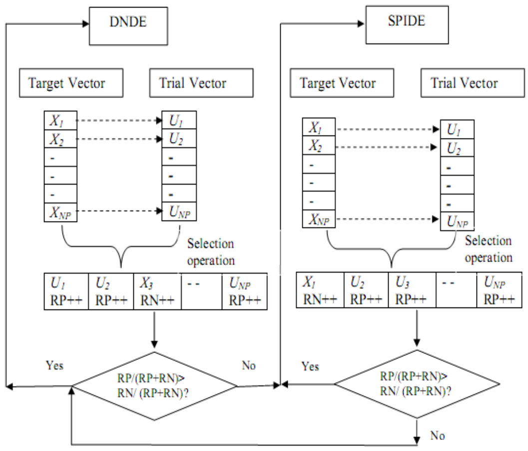


Figure 1. Working structure of proposed Self Adaptive Approach.

It can also be noticed that for any generation, the total number of positive and negative rank should be equal to the population size, i.e., *RP* + *RN* = *NP*

Hence the rules (7) and (8) may be further reduced as follows:

$$Case\ 1: \; p_1 = \frac{RP}{NP}, \; p_2 = 1 - p_1 \tag{9}$$

$$Case\ 2: \; p_2 = \frac{RP}{NP}, \; p_1 = 1 - p_2 \tag{10}$$

Under the aforementioned rules, only the positive ranks of active variations should be considered. This will also reduce the amount of time spent counting negative ranks.

By using SaMDE, the advantage of all three variants MRLDE, DNDE and SPIDE are considered in a single algorithm. First of all, MRLDE provides a strategic method for selecting the individuals for mutation which creates a platform for getting a fast convergence speed. Secondly, DNDE and SPIDE are employed adaptively to make SaMDE more efficient. By using the adaptive rule, the variant which gives successively better performance obtains additional chances to be continued for the next generations. Hence if any variant fails to solve a specific problem the other variant will be automatically activated to solve it. However, a drawback of SaMDE is, if neither of the variants are able to solve a specific problem, performance of SaMDE will naturally deteriorate. Next, the flowchart of the suggested adaptive strategy and pseudo code for SaMDE is depicted in Figure 2 and Algorithm 2, respectively.

Algorithm 2 *SaMDE*

Set NP, F and C_R, p_1 and p_2
Generate population $P^G = \{X_i^{(g)}, i = 1, 2, \ldots, NP\}$.
  Evaluate $f(X_i^{(g)})$ and Sort whole population by their fitness i.e., *Sort* $\{f(X_i^{(g)})\}$
  Generate probability p_1 and p_2 randomly and set positive and negative rank $RP = RN = 0$
  while (Termination criteria is not satisfied) ***do***
    for $i = 1$ to NP ***do***
      Select r, s and t by *MRLDE* for each i
      If $p_1 > p_2$
        execute mutation and crossover by DNDE
        if trial vector selected for the next generation
          Increase a positive rank i.e., $RP = RP + 1$
        else
          Increase a negative rank i.e., $RN = RN + 1$
        end if
      else
        execute mutation and crossover by by SPIDE
        if trial vector selected for the next generation
          Increase a positive rank $RP = RP + 1$
        else
          Increase a negative rank $RN = RN + 1$
        end if
      end if
    end for
    Update the population for the next generation, $P^{g+1} = \{X_i^{(g+1)}, i = 1,2,\ldots NP\}$
    Sort $\{f(X_i^{(g+1)})\}$
    Update p_1 and p_2 using Equations (7) and (8)
  end while

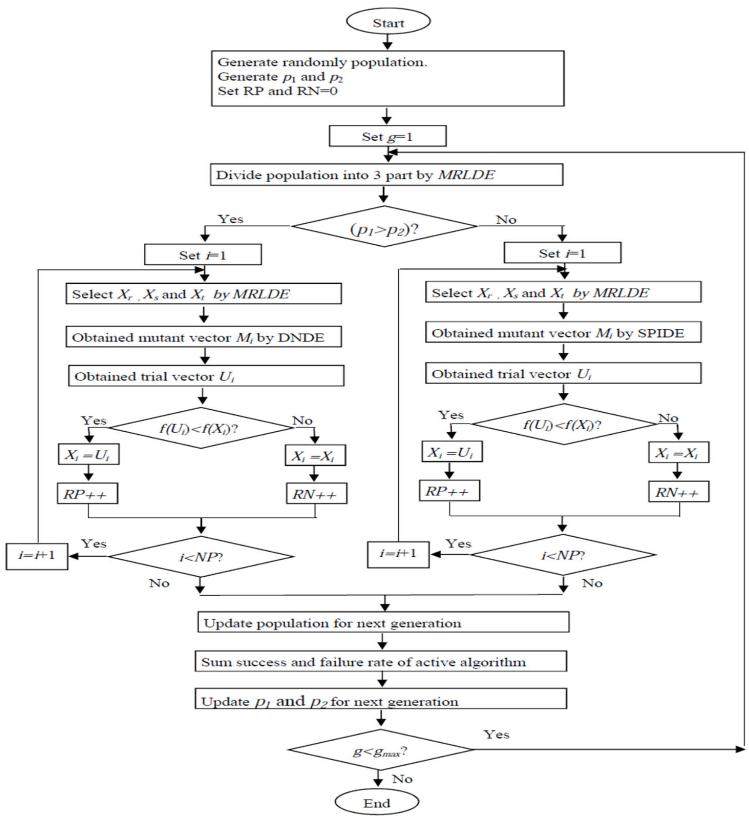


Figure 2. Flow chart of SaMDE.

4. Test Problems and Real-Life Application

4.1. Test Suit-1: Classical Benchmarks Problems

In the first test suite there are 15 traditional benchmark problems selected from various literature [4,14,30,42,43]. All these problems are tested for dimension 30. The test functions f_1–f_5, f_{14} and f_{15} are unimodal, f_6 is a discontinuous function with one minimum, f_7 is a noisy function. The test functions f_8–f_{13} are multimodal functions in which the number of local minima increases exponentially with the problem dimension [4]. According to literature unimodal functions are important to check the exploration ability and convergence rate of algorithms while multimodal functions are important to check the exploitation ability of algorithm.

4.2. Test Suit-2: IEEE CEC2008 Functions

The second set consists of six shifted functions (SF_1–SF_6) selected from the IEEE CEC 2008 test suit [62]. This test suite was particularly planned to test the efficiency and robustness of an algorithm on complex test problems.

A review of classical and CEC2008 benchmark functions is presented in Table 1.

Table 1. Benchmark functions with initial bounds and optimum value.

Table	**Initial Bound**	***f(x*)***	**Test Function**	**Initial Bound**	***f(x*)***
f_1: Sphere	(−100; 100)	0	f_{12}: Generalized Penalized-1	(−50, 50)	0
f_2: Schwefel's-2.22	(−10, 10)	0	f_{13}: Generalized Penalized-2	(−50, 50)	0
f_3: Schwefel's 1.2	(−100; 100)	0	f_{14}: Exponetial	(−1, 1)	0
f_4: Schwefel's-2.21	(−100; 100)	0	f_{15}: Zhakarov	(−5, 10)	0
f_5: Rosenbrock's	(−30, 30)	0	SF_1: Shifted Sphere	(−100; 100)	−450
f_6: Step	(−100; 100)	0	SF_2: Schwefel's 2.21	(−100; 100)	−450
f_7: Noise	(−1.28, 1.28)	0	SF_3: Shifted Rosenbrock	(−100; 100)	390
f_8: Schwefel's2.26	(−500, 500)	0	SF_4: Shifted Rastrigin	(−5.12, 5.12)	−330
f_9: Rastrigin's	(−100; 100)	0	SF_5: Shifted Griewank	(−600, 600)	−180
f_{10}: Ackley's	(−32, 32)	0	SF_6: Shifted Ackley	(−32, 32)	−140
f_{11}: Griewank	(−600, 600)	0			

4.3. Test Suit-3: IEEE CEC 2017 Functions

The IEEE CEC 2017 test suit is renowned as a set of extremely complex benchmark functions. There is a total of 29 benchmark (C_1–C_{30}) problems in the suit where one function C_2 has been removed due to its high unstable nature. These functions can be classified in four categories like Unimodel functions (C_1–C_3), Multimodel functions (C_4–C_{10}), hybrid functions (C_{11}–C_{20}) and composite functions (C_{21}–C_{30}). The bound for the variable for all functions is (−100, 100) and the optimum value is $100 \times i$ where 'i' is function number from [1,30]. A detailed review and specification of CEC 2017 functions can be found in [63].

4.4. Real Life Application: Molecular Potential Energy Problem

The minimization of the potential energy problem of a molecule is a complex and multimodal optimization problem that occurs in the chemical science field [27,64,65]. The most challenging aspect of this problem is that the number of local minimizers grows exponentially as the size of the molecule rises. Consequently, it becomes a difficult and unsolved challenge for scientists and engineers from a variety of disciplines.

The mathematical model of a molecular having a linear chain of n-beads centred at x_1, x_2,...x_n in a 3D domain. The optimization model of molecular potential energy is given as below:

$$\text{Minimize } F = \sum_{i=1}^{4} F_i \tag{11}$$

where F_1, F_2, F_3 and F_4 are the potential forces due to bond length, bond angle, torsion angles and interaction, respectively, and defined as below:

$$\begin{gathered}F_1 = \sum_{(i,j)\in K_1} s^1_{i,j}(\rho_{i,j} - \rho^0_{i,j})^2 \\ F_2 = \sum_{(i,j)\in K_2} s^2_{i,j}(\phi_{i,j} - \phi^0_{i,j})^2 \\ F_3 = \sum_{(i,j)\in K_3} s^3_{i,j}(1 + \cos(3\theta_{i,j} - \theta^0_{i,j})) \\ F_4 = \sum_{(i,j)\in k_3} \left(\frac{(-1)^i}{\rho_{i,j}}\right)\end{gathered} \tag{12}$$

Here $\rho_{i,j}\phi_{i,j}$ and $\theta_{i,j}$ are bond length, bond angle and torsion angle between two, three and four consecutive pairs ofbeads, respectively. $s^1{}_{i,j}$, $s^2{}_{i,j}$, and $s^3{}_{i,j}$ denotes bond stretching, angle bending and the torsion force constant, respectively. K_j, $j = 1, 2, 3$ denotes the set of pair of atoms separated by j-covalent bonds.

As explained in [64], the final optimization function can be defined by Equation (13):

$$Min\ F = \sum_{i=1}^{n-32}\left((1 + \cos(3\theta_{i,i+3})) + \frac{(-1)^i}{\sqrt{10.60099896 - 4.141720682\cos(\theta_{i,i+3})}}\right) \tag{13}$$

The optimization model in Equation (13) is a non-convex function and has several local minimizers even for the small value of n. The number of local minimizers of the function is $2^{\mathrm{n}-3}$, and the doamin bound for $\theta_{i,j}$; will be restricted in (0, 5).

5. Result Analysis and Discussion

5.1. Experimental Settings:

Experiments are conducted using a 64-bit equipped laptop of Dell company with a 2.6-GHz Intel Core i3 processor, 8 GB RAM, and Windows 10 operating system. Other parameters settings were taken as suggested in various literature [27,33,34,42,43]:

- NP = 100; D = 30,50; F = 0.5; Cr =0.9.
- For MRLDE, $\lambda_1 = 20$, $\lambda_2 = 40$ and $\lambda_3 = 40$ [33].
- For SPIDE and DNDE, $P_s = 0.1$, $P_d = 0.1$ [27,34].
- Max NFE = 10,000 D for all functions.
- Total Run = 50.

5.2. Performance Evaluation of SaMDE over DE, SPIDE, DNDE, MRLDE

5.2.1. Results Analysis in Terms of Average Error and Standard Deviation:

The experimental results for traditional functions (f_1–f_{15}) and the shifted function (SF_1–SF_6) are presented in Table 2. When many algorithms can obtain the global optima, then only intermediate solutions for the function are presented.

From Table 2, it is clear that all modified variants give superior performance over basic DE for all functions, except DNDE in case of function f_4. SaMDE has obtained best results in case of all functions except 03 functions f_2, f_5 and f_8 where MRLDE is in the leading position. MRLDE obtained the second best position on 09 functions while SPIDE obtained the second position in the case of 03 functions f_4, f_7, f_9. Similarly, DNDE also obtained the second position in the case of 03 functions f_{12}, f_{13} and SF_4. In case SF_1, both SaMDE and DNDE gives equal accuracy and obtains the first position. Similarly, SaMDE, MRLDE and gives DNDE equal performance on SF_6. In the case of SF_5, all four variants give equal performances.

A statistical analysis based on the mean difference of samples is also conducted in Table 2. The symbols '+', '−' and '=' show the performance of SaMDE as significantly better, worse or equal, respectively, with its competitor. As per the last column of table, the win/loss/tie performance of SaMDE is 21/0/0, 20/0/1, 18/0/3 and 17/2/2/ with respect to DE, SPIDE, DNDE and MRLDE, respectively, $\times 10^{-}/\times 10^{+}$.

Table 2. Comparison of SaMDE with DE, SPIDE, DNDE and MRLDE in terms of best, mean values and standard deviation (SD) at D = 30.

Fun	Max-NFE	Error	Numerical Results					Statistical Significance			
			DE	**SPIDE**	**DNDE**	**MRLDE**	**SaMDE**	**5/1**	**5/2**	**5/3**	**5/4**
		Best	1.17×10^{-14}	4.37×10^{-36}	1.56×10^{-43}	1.13×10^{-43}	2.11×10^{-57}				
f_1	150 k	Mean	4.03×10^{-14}	4.64×10^{-35}	4.42×10^{-42}	1.17×10^{-42}	4.75×10^{-57}	+	+	+	+
		SD	2.71×10^{-14}	5.50×10^{-35}	3.58×10^{-42}	1.33×10^{-42}	2.23×10^{-57}				
		Best	9.21×10^{-11}	1.21×10^{-24}	6.83×10^{-32}	1.41×10^{-41}	3.56×10^{-38}				
f_2	200 k	Mean	5.18×10^{-10}	6.75×10^{-24}	7.05×10^{-31}	2.68×10^{-41}	6.39×10^{-38}	+	+	+	−
		SD	3.46×10^{-10}	4.52×10^{-24}	2.81×10^{-31}	1.19×10^{-41}	3.52×10^{-38}				
		Best	9.82×10^{-12}	1.56×10^{-36}	1.09×10^{-31}	8.75×10^{-38}	1.71×10^{-63}				
f_3	500 k	Mean	1.11×10^{-11}	2.30×10^{-35}	6.89×10^{-30}	2.72×10^{-37}	5.43×10^{-63}	+	+	+	+
		SD	3.36×10^{-12}	3.28×10^{-36}	6.61×10^{-30}	1.91×10^{-37}	4.07×10^{-61}				
		Best	2.49×10^{-09}	1.22×10^{-37}	9.87×10^{-02}	1.21×10^{-28}	2.67×10^{-64}				
f_4	500 k	Mean	3.18×10^{-01}	2.61×10^{-37}	6.97×10^{-01}	3.21×10^{-27}	4.30×10^{-64}	+	+	+	+
		SD	6.25×10^{-01}	2.99×10^{-37}	6.09×10^{-01}	2.34×10^{-27}	8.70×10^{-64}				
		Best	1.16×10^{-13}	4.99×10^{-28}	1.26×10^{-20}	$0.00 \times 10^{+00}$	1.83×10^{-30}				
f_5	500 k	Mean	3.32×10^{-12}	1.72×10^{-28}	5.07×10^{-17}	1.65×10^{-30}	2.13×10^{-30}	+	+	+	−
		SD	2.89×10^{-12}	1.22×10^{-29}	1.01×10^{-16}	8.15×10^{-30}	$1.34 \times 10_{-30}$				
		Best	$1.69 \times 10^{+03}$	$4.80 \times 10^{+01}$	$7.50 \times 10^{+00}$	$1.00 \times 10^{+01}$	$1.00 \times 10^{+00}$				
f_6	10 k	Mean	$2.17 \times 10^{+03}$	$6.41 \times 10^{+01}$	$9.50 \times 10^{+00}$	$1.08 \times 10^{+01}$	$1.10 \times 10^{+00}$	+	+	+	+
		SD	$2.49 \times 10^{+02}$	$1.32 \times 10^{+01}$	1.21×10^{-01}	4.00×10^{-01}	4.89×10^{-01}				
		Best	2.89×10^{-03}	1.29×10^{-03}	1.12×10^{-03}	1.30×10^{-03}	1.81×10^{-04}				
f_7	300 k	Mean	5.81×10^{-03}	1.47×10^{-03}	5.13×10^{-03}	1.56×10^{-03}	3.08×10^{-04}	+	+	+	+
		SD	6.69×10^{-03}	3.51×10^{-04}	2.80×10^{-04}	1.89×10^{-04}	1.70×10^{-04}				
		Best	$5.61 \times 10^{+03}$	$5.72 \times 10^{+02}$	$3.35 \times 10^{+02}$	$2.17 \times 10^{+02}$	$2.36 \times 10^{+02}$				
f_8	500 k	Mean	$6.23 \times 10^{+03}$	$2.85 \times 10^{+03}$	$6.98 \times 10^{+02}$	$1.82 \times 10^{+03}$	$6.36 \times 10^{+02}$	+	+	+	+
		SD	$4.44 \times 10^{+02}$	$1.74 \times 10^{+02}$	$3.05 \times 10^{+02}$	$1.44 \times 10^{+03}$	$3.31 \times 10^{+02}$				
		Best	$8.59 \times 10^{+01}$	8.67×10^{-18}	1.04×10^{-14}	$1.39 \times 10^{+01}$	$0.00 \times 10^{+00}$				
f_9	500 k	Mean	$9.32 \times 10^{+01}$	9.54×10^{-18}	3.97×10^{-14}	$1.61 \times 10^{+01}$	$0.00 \times 10^{+00}$	+	+	+	+
		SD	$5.39 \times 10^{+00}$	9.50×10^{-19}	7.95×10^{-14}	$2.31 \times 10^{+00}$	$0.00 \times 10^{+00}$				

Table 2. *Cont.*

Fun	Max-NFE	Error	Numerical Results					Statistical Significance			
			DE	SPIDE	DNDE	MRLDE	SaMDE	5/1	5/2	5/3	5/4
f_{10}	50 k	Best Mean SD	1.83×10^{-02} 9.16×10^{-02} 2.81×10^{-02}	1.02×10^{-05} 1.45×10^{-05} 3.54×10^{-05}	5.16×10^{-06} 6.19×10^{-06} 1.02×10^{-06}	7.21×10^{-07} 8.02×10^{-07} 1.12×10^{-07}	1.02×10^{-10} 2.71×10^{-10} 5.53×10^{-09}	+	+	+	+
f_{11}	50 k	Best Mean SD	2.21×10^{-02} 5.35×10^{-02} 1.94×10^{-02}	1.18×10^{-09} 8.78×10^{-09} 4.74×10^{-09}	9.75×10^{-11} 1.75×10^{-11} 4.62×10^{-11}	1.76×10^{-12} 3.38×10^{-12} 2.91×10^{-11}	7.94×10^{-17} 2.94×10^{-16} 1.81×10^{-16}	+	+	+	+
f_{12}	50 k	Best Mean SD	2.11×10^{-03} 3.73×10^{-03} 1.36×10^{-03}	1.45×10^{-09} 3.85×10^{-09} 2.51×10^{-09}	4.74×10^{-13} 4.64×10^{-13} 1.40×10^{-13}	3.84×10^{-13} 8.23×10^{-13} 6.72×10^{-13}	1.18×10^{-17} 2.01×10^{-17} 2.16×10^{-17}	+	+	+	+
f_{13}	50 k	Best Mean SD	1.75×10^{-02} 3.68×10^{-02} 1.91×10^{-02}	6.93×10^{-08} 1.93×10^{-07} 7.73×10^{-07}	3.53×10^{-13} 4.08×10^{-12} 1.22×10^{-12}	7.33×10^{-12} 1.41×10^{-11} 1.11×10^{-11}	8.79×10^{-17} 1.29×10^{-16} 2.67×10^{-16}	+	+	+	+
f_{14}	50 k	Best Mean SD	2.27×10^{-06} 6.68×10^{-06} 5.98×10^{-06}	2.38×10^{-14} 4.67×10^{-14} 1.33×10^{-14}	2.49×10^{-16} 3.55×10^{-16} 1.21×10^{-16}	1.08×10^{-16} 2.24×10^{-16} 1.24×10^{-16}	2.16×10^{-19} 2.16×10^{-19} $0.00 \times 10^{+00}$	+	+	+	+
f_{15}	150 k	Best Mean SD	2.74×10^{-14} 3.33×10^{-14} 1.41×10^{-14}	1.12×10^{-35} 2.02×10^{-35} 1.07×10^{-35}	8.45×10^{-43} 1.65×10^{-42} 1.27×10^{-42}	3.58×10^{-44} 4.51×10^{-43} 3.68×10^{-43}	1.85×10^{-57} 2.68×10^{-57} 6.52×10^{-57}	+	+	+	+
SF_1	150 k	Best Avg SD	5.68×10^{-14} 4.43×10^{-13} 3.84×10^{-13}	$0.00 \times 10^{+00}$ 2.27×10^{-14} 2.78×10^{-14}	$0.00 \times 10^{+00}$ $0.00 \times 10^{+00}$ $0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$ 1.13×10^{-14} 2.27×10^{-14}	$0.00 \times 10^{+00}$ $0.00 \times 10^{+00}$ $0.00 \times 10^{+00}$	+	+	=	+
SF_2	150 k	Best Avg SD	2.16×10^{-01} $9.22 \times 10^{+00}$ $1.09 \times 10^{+00}$	5.60×10^{-02} 2.39×10^{-01} 2.81×10^{-01}	$2.65 \times 10^{+00}$ $4.55 \times 10^{+00}$ $1.64 \times 10^{+00}$	3.89×10^{-08} 4.45×10^{-07} 8.27×10^{-07}	2.81×10^{-08} 9.76×10^{-08} 8.78×10^{-08}	+	+	+	+
SF_3	150 k	Best Avg SD	$1.89 \times 10^{+01}$ $1.95 \times 10^{+01}$ $1.08 \times 10^{+00}$	$7.48 \times 10^{+00}$ $1.13 \times 10^{+01}$ $2.66 \times 10^{+00}$	2.91×10^{-01} $9.80 \times 10^{+00}$ $7.51 \times 10^{+00}$	1.13×10^{-08} 4.34×10^{-06} 6.64×10^{-06}	3.24×10^{-12} 5.23×10^{-10} 4.22×10^{-10}	+	+	+	+
SF_4	150 k	Best Avg SD	$1.64 \times 10^{+02}$ $1.73 \times 10^{+02}$ $7.22 \times 10^{+00}$	$1.41 \times 10^{+02}$ $1.60 \times 10^{+02}$ $1.18 \times 10^{+01}$	$1.42 \times 10^{+01}$ $2.83 \times 10^{+01}$ $2.15 \times 10^{+01}$	$9.01 \times 10^{+01}$ $1.13 \times 10^{+02}$ $2.81 \times 10^{+01}$	$1.29 \times 10^{+01}$ $1.71 \times 10^{+01}$ $4.09 \times 10^{+00}$	+	+	+	+

Table 2. *Cont.*

Fun	Max-NFE	Error	Numerical Results					Statistical Significance			
			DE	**SPIDE**	**DNDE**	**MRLDE**	**SaMDE**	**5/1**	**5/2**	**5/3**	**5/4**
SF_5	150 k	Best Avg SD	8.52×10^{-14} 4.03×10^{-13} 2.79×10^{-13}	$0.00 \times 10^{+00}$ $0.00 \times 10^{+00}$ $0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$ $0.00 \times 10^{+00}$ $0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$ $0.00 \times 10^{+00}$ $0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$ $0.00 \times 10^{+00}$ $0.00 \times 10^{+00}$	+	=	=	=
SF_6	150 k	Best Avg SD	9.12×10^{-08} 1.01×10^{-07} 2.42×10^{-08}	5.68×10^{-14} 9.09×10^{-14} 3.31×10^{-14}	2.84×10^{-14} 2.84×10^{-14} $0.00 \times 10^{+00}$	2.84×10^{-14} 2.84×10^{-14} $0.00 \times 10^{+00}$	2.84×10^{-14} 2.84×10^{-14} $0.00 \times 10^{+00}$	+	+	=	=
+/−/= *p-value* *Significance (at 5%)*								21/0/0 0.000 Yes	20/0/1 0.000 Yes	18/0/3 0.000 Yes	17/2/2 0.001 Yes

'+', '−' and '=' represent the proposed scheme is significantly better, worse or equal, respectively, when compared with the competitor.

5.2.2. Result Analysis by Non-Parametric Statistical Tests

Three non-parametric statistical tests 'Wilcoxon rank sum test', 'Friedman rank test' and 'Bonferroni-Dunn test' [66] are used to check the significant difference between the performances of the proposed variants. These results are tabulated in Tables 3 and 4.

Table 3. Wilcoxon Rank Sum Test Results for SaMDE versus DE, SPIDE, DNDE and MRLDE.

Algorithms		ΣR^+	ΣR^-	z-Value	*p*-Value	Sig at $\alpha = 0.05$
SaMDE vs.	DE	231	0	4.015	<0.001	+
	SPIDE	210	0	3.920	<0.001	+
	DNDE	171	0	3.724	<0.001	+
	MRLDE	182	8	3.501	<0.001	+

Table 4. Friedman's Ranks and Critical difference (CD) calculated by Bonferroni–Dunn's Method.

Rank	DE	SPIDE	DNDE	MRLDE	SaMDE	CD ($\alpha = 0.1$)	CD ($\alpha = 0.05$)
D = 30	4.95	3.50	2.90	2.40	1.24	1.0935	1.2189

The Wilcoxon rank sum test results are given in Table 3. ΣR^+ and ΣR^- represent the sum of ranks for positive and negative differences, respectively. A higher positive rank sum shows the SaMDE over other algorithms. The z-value and corresponding *p*-value are also given in Table 3. The significant level of difference is taken as $\alpha = 0.05$. From this table, it can be noticed that all variants obtained significantly better results in comparison to DE. MRLDE gives better results than SPIDE whereas the performances of MRLDE and DNDE are significantly equal. SaMDE gives the best performance in comparison to other variants.

Attending the results given in Table 4, the Friedman's rank test and Bonferroni–Dunn's test are used to detect significant differences for the control algorithm SaMDE and the results are presented in Table 4.

It can be noticed that SaMDE obtained the lowest mean rank among all other variants. In Figure 3, Bonferroni–Dunn's graphic demonstrates the difference between the rankings of each algorithm. The algorithm with the lowest rank is considered as the control algorithm, while the horizontal cut line represents the threshold for the control algorithm. This line is drawn at a distance of the sum of the ranking of the control algorithm and the corresponding CD calculated by the Bonferroni–Dunn method as shown in Table 4 for each $\alpha = 0.1$ and $\alpha = 0.05$. The algorithms for which the rank bar exceeds this line are considered to have a worse performance than the control algorithm. Hence, by using the application of the Bonferroni–Dunn method, it can also be seen that only MRLDE is significantly acceptable when compared to SaMDE.

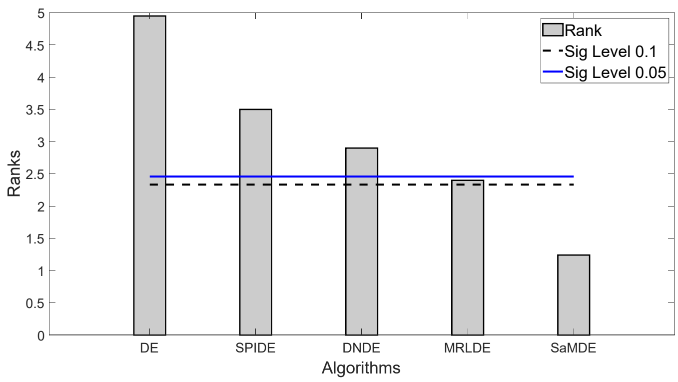
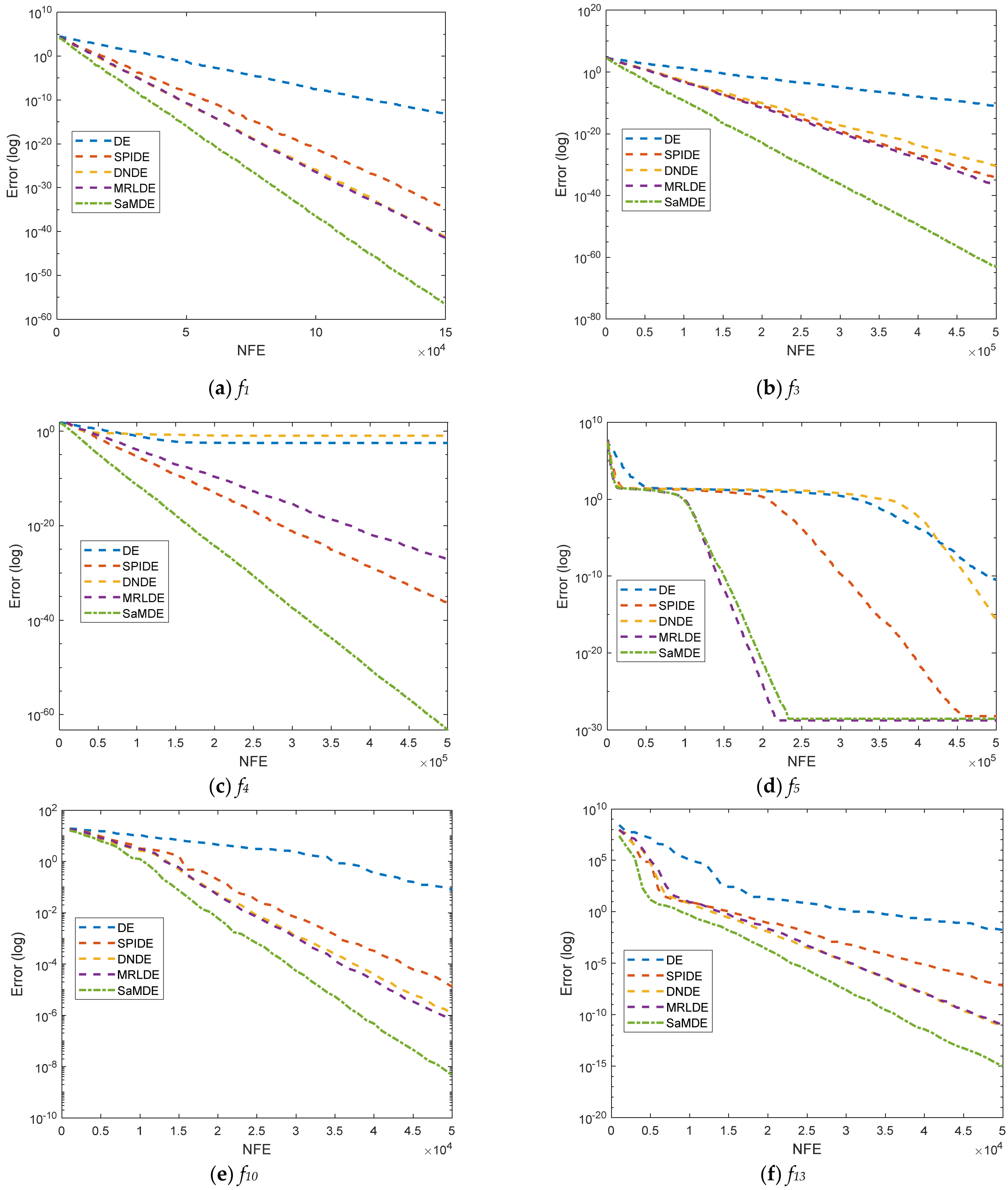


Figure 3. Bonferroni–Dunn bar chart for DE, SPIDE, DNDE, MRLDE and SaMDE. The Bar represents the algorithm's rank and Horizontal cut lines represents the significant levels.

5.2.3. Performance Evaluation of SaMDE by Convergence Curves

Convergence curves for DE, SPIDE, DNDE, MRLDE and SaMDE of selected functions such as $f_1, f_3, f_4, f_5, f_{10}, f_{13}, f_{15}$, SF_2, SF_3 and SF_4 are represented in Figure 4. The X-axis and Y-Axis represents the NFE and its corresponding error value, respectively. By the graphs it can be easily seen that SaMDE performs faster and confirms its robustness over its parent variants.

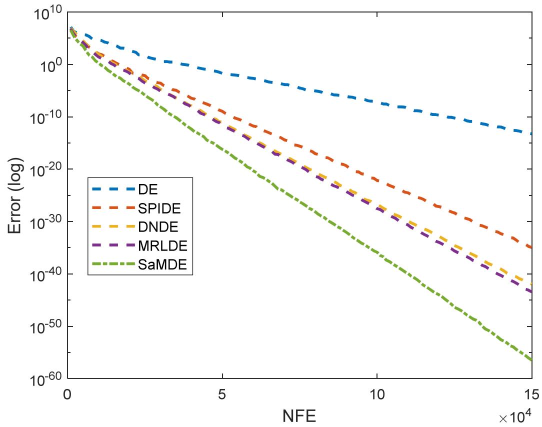


(**a**) f_1 (**b**) f_3

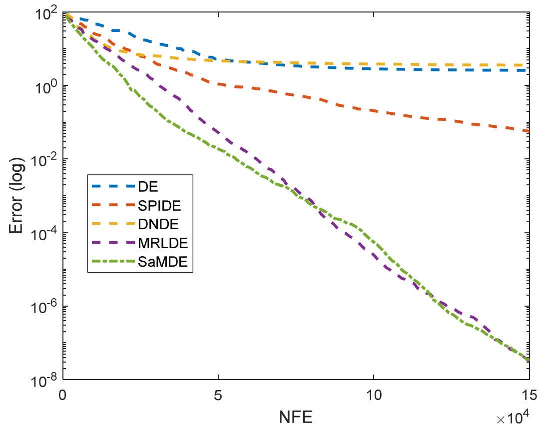
(**c**) f_4 (**d**) f_5

(**e**) f_{10} (**f**) f_{13}

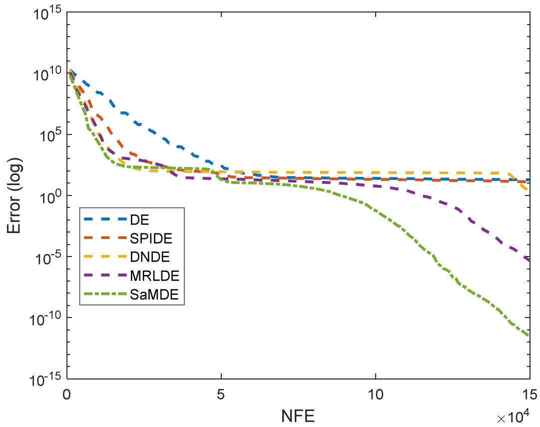
Figure 4. *Cont.*



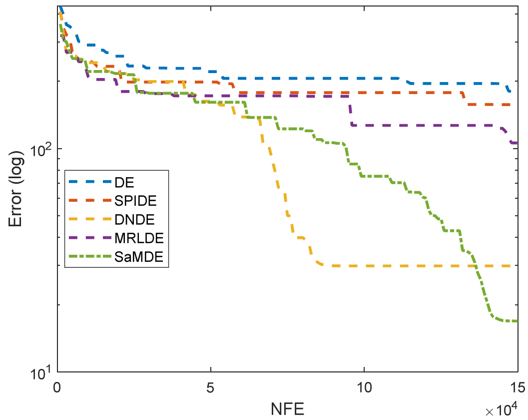
(**g**) *f*15



(**h**) *SF*2



(**i**) *SF*3



(**j**) *SF*4

Figure 4. Convergence graphs for functions $f_1, f_3, f_4, f_5, f_{10}, f_{13}, f_{15}$, SF_2, SF_3 and SF_4.

5.3. Performance Evaluation of SaMDE over Other Enhanced DE Variants

In this section the performance of SaMDE is compared with SHADE [5], JADE [30], SaDE [42], rJADE [43], APadapSS-JADE [43], and DEGOS [55] algorithms. The comparison is taken in terms of average error and standard deviation and then non-parametric tests are applied to check the significant difference between the algorithms. The results for SHADE are taken from [5], for SaDE and JADE are taken from [30], and for rJADE and APadapSS-JADE are taken from [43]. For DEGOS, original results have been obtained by using the code provided on [67]. In order to make a fair comparison, all parameter settings remain similar for all algorithms.

According to Table 5, SaDE, JADE, rJADE, APadapSS-JADE, SHADE, DEGOS and SaMDE have obtained the best results on 0, 0, 1, 5, 2, 0 and 5 functions, respectively. From the win/loss/tie row of the table, we can see that SaMDE surpasses the algorithm SaDE, JADE, rJADE, APadapSS-JADE, SHADE and DEGOS on 12, 10, 11, 6, 7 and 12 cases, respectively.

The Wilcoxon rank sum test presented in Table 6 shows that SaMDE performs significantly equal with respect to APAdapSS-JADE and SHADE while it is significantly better than SaDE, JADE, rJADE and DEGOS.

Table 5. Comparison of SaMDE with SaDE, JADE, rJADE, APadapSS-JADE, SHADE and DEGOS in terms of average error and standard deviation (SD).

F	Max NFE	SaDE	JADE	rJADE	APadapSS-JADE	SHADE	DEGOS	SaMDE
f_1	150k	4.6×10^{-20} + (7.1×10^{-20})	1.9×10^{-60} − (8.3×10^{-60})	1.8×10^{-53} + (1.3×10^{-52})	2.4×10^{-75} − (1.4×10^{-74})	1.1×10^{-70} − (4.6×10^{-70})	3.6×10^{-26} + (3.4×10^{-26})	4.7×10^{-57} (2.2×10^{-57})
f_2	200K	2.0×10^{-14} + (1.2×10^{-14})	1.9×10^{-25} + (9.1×10^{-25})	1.6×10^{-28} + (6.1×10^{-28})	1.8×10^{-44} (1.3×10^{-43})	4.7×10^{-49} − (5.0×10^{-49})	4.5×10^{-19} + (2.6×10^{-19})	6.3×10^{-38} (3.5×10^{-38})
f_3	500k	9.3×10^{-37} + (5.3×10^{-36})	6.0×10^{-61} + (3.0×10^{-60})	$1.6 \times 10^{+00}$ ($3.1 \times 10^{+00}$)	2.50×10^{-61} + (8.35×10^{-61})	5.5×10^{-61} + (3.3×10^{-61})	1.7×10^{-22} + (8.3×10^{-22})	5.6×10^{-63} (4.0×10^{-63})
f_4	500k	7.2×10^{-11} + (2.0×10^{-10})	8.1×10^{-24} + (4.1×10^{-23})	1.1×10^{-15} + (4.8×10^{-16})	5.14×10^{-22} + (5.4×10^{-22})	2.1×10^{-41} + (2.3×10^{-41})	4.4×10^{-01} + (1.09×10^{-01})	4.3×10^{-64} (8.7×10^{-64})
f_5	500k	$2.0 \times 10^{+01}$ + ($8.1 \times 10^{+00}$)	8.1×10^{-02} + (7.2×10^{-01})	2.2×10^{-30} + (4.7×10^{-30})	3.1×10^{-01} + ($1.0 \times 10^{+00}$)	7.9×10^{-02} + (7.7×10^{-02})	3.2×10^{-22} + (3.22×10^{-22})	2.1×10^{-30} (1.3×10^{-30})
f_6	10k	$9.1 \times 10^{+02}$ + ($2.0 \times 10^{+02}$)	$2.9 \times 10^{+00}$ + ($1.1 \times 10^{+00}$)	$1.2 \times 10^{+00}$ + ($1.2 \times 10^{+00}$)	$1.0 \times 10^{+00}$ = ($1.9 \times 10^{+00}$)	$2.6 \times 10^{+00}$ + ($1.1 \times 10^{+00}$)	$5.8 \times 10^{+01}$ + ($1.3 \times 10^{+02}$)	$1.1 \times 10^{+00}$ (4.8×10^{-01})
f_7	300k	5.0×10^{-03} + (1.4×10^{-03})	6.6×10^{-04} + (2.2×10^{-04})	4.8×10^{-04} + (1.4×10^{-04})	5.9×10^{-04} + (1.8×10^{-04})	5.9×10^{-04} + (2.3×10^{-04})	2.1×10^{-03} + (1.2×10^{-04})	3.0×10^{-04} (1.7×10^{-04})
f_8	100k	$4.8 \times 10^{+00}$ − ($3.1 \times 10^{+01}$)	3.0×10^{-05} − (2.1×10^{-05})	4.2×10^{-09} − (4.7×10^{-09})	1.8×10^{-08} − (1.1×10^{-07})	1.5×10^{-03} − (1.6×10^{-03})	$2.1 \times 10^{+02}$ − ($1.0 \times 10^{+02}$)	$6.3 \times 10^{+02}$ ($3.3 \times 10^{+02}$)
f_9	100k	1.6×10^{-03} + (7.0×10^{-04})	1.1×10^{-04} + (6.1×10^{-05})	1.2×10^{-02} + (1.7×10^{-02})	2.9×10^{-01} + (5.6×10^{-01})	1.7×10^{-02} + (7.3×10^{-02})	$1.5 \times 10^{+01}$ + ($1.4 \times 10^{+01}$)	1.0×10^{-09} (1.3×10^{-09})
f_{10}	50k	2.9×10^{-03} + (4.9×10^{-04})	8.9×10^{-10} + (7.0×10^{-10})	3.5×10^{-10} + (2.7×10^{-10})	4.1×10^{-10} + (1.8×10^{-11})	2.6×10^{-10} = (9.3×10^{-10})	5.4×10^{-04} + (2.7×10^{-04})	2.7×10^{-10} (5.5×10^{-09})
f_{11}	50k	7.9×10^{-04} + (1.4×10^{-03})	9.4×10^{-08} + (6.1×10^{-07})	1.1×10^{-06} + (1.2×10^{-06})	$0.0 \times 10^{+00}$ − ($0.0 \times 10^{+00}$)	1.6×10^{-14} + (9.3×10^{-14})	3.1×10^{-05} + (2.5×10^{-05})	2.9×10^{-16} (1.8×10^{-16})
f_{12}	50k	2.0×10^{-05} + (9.4×10^{-06})	4.4×10^{-17} + (2.0×10^{-16})	1.8×10^{-18} − (5.3×10^{-18})	2.2×10^{-22} − (7.7×10^{-22})	3.6×10^{-19} − (6.5×10^{-19})	7.8×10^{-04} + (6.0×10^{-04})	2.0×10^{-17} (2.1×10^{-17})
f_{13}	50k	6.2×10^{-05} + (2.1×10^{-05})	2.1×10^{-16} = (6.6×10^{-16})	1.5×10^{-15} + (4.8×10^{-15})	3.7×10^{-20} − (1.2×10^{-19})	3.8×10^{-18} − (3.4×10^{-18})	5.9×10^{-06} + (1.2×10^{-06})	1.2×10^{-16} (2.6×10^{-16})
SaMDE (*w/l/t*)		12/1/0 p = 0.003+	10/2/1 p = 0.022+	11/2/0 p = 0.022+	6/6/1 p = 1.00=	7/5/1 p = 0.774=	12/1/0 p = 0.003+	–

'+', '=', '−' means significant better, equal or worse, respectively, at $\alpha = 0.05$.

Table 6. Result of Wilcoxon-rank sum test on the results obtained in Table 7.

Algorithms		Pairwise Rank	ΣR^+	ΣR^-	z Value	p-Value	Sig at $\alpha = 0.05$
SaMDE *vs*	SaDE	(1.08, 1.92)	79	12	2.341	0.019	+
	JADE	(1.15, 1.85)	76	15	2.132	0.033	+
	rJADE	(1.15, 1.85)	74	17	1.992	0.046	+
	APAdapSS-JADE	(1.50, 1.50)	46	45	0.035	0.972	=
	SHADE	(1.46, 1.54)	48	30	0.706	0.480	=
	DEGOS	(1.08, 1.92)	78	13	2.271	0.023	+

'+' means significantly better and '=' means significantly equal.

Table 7 shows the results of the Friedman test and the Bonferroni–Dunn test for the results given in Table 7 in order to check the global significant difference between the algorithms. SaMDE obtained the lowest rank and hence proves its significance over others.

Table 7. Friedman Ranks Critical difference (CD) calculated by Bonferroni–Dunn's procedure on the results obtained in Table 7.

	SaDE	JADE	rJADE	APadapSS-JADE	SHADE	DEGOS	SaMDE	CD ($\alpha = 0.1$)	CD ($\alpha = 0.05$)
Rank	6.17	4.00	3.75	2.71	2.96	6.00	2.42	2.02	2.23

Figure 5 represents the bar graphs of rank of the algorithms. The horizontal control lines are drawn for a significant level at $\alpha = 0.1$ and $\alpha = 0.05$ by taking by taking SaMDE as a control algorithm. The graph shows that the ranks of SaMDE, JADE, rJADE, AdapSS-JADE and SHADE lie under the control lines of the significant level at α = 0.1 and α = 0.05. Hence, these algorithms can be considered as significantly equal with SaMDE and significantly better than SaDE and DEGOS.

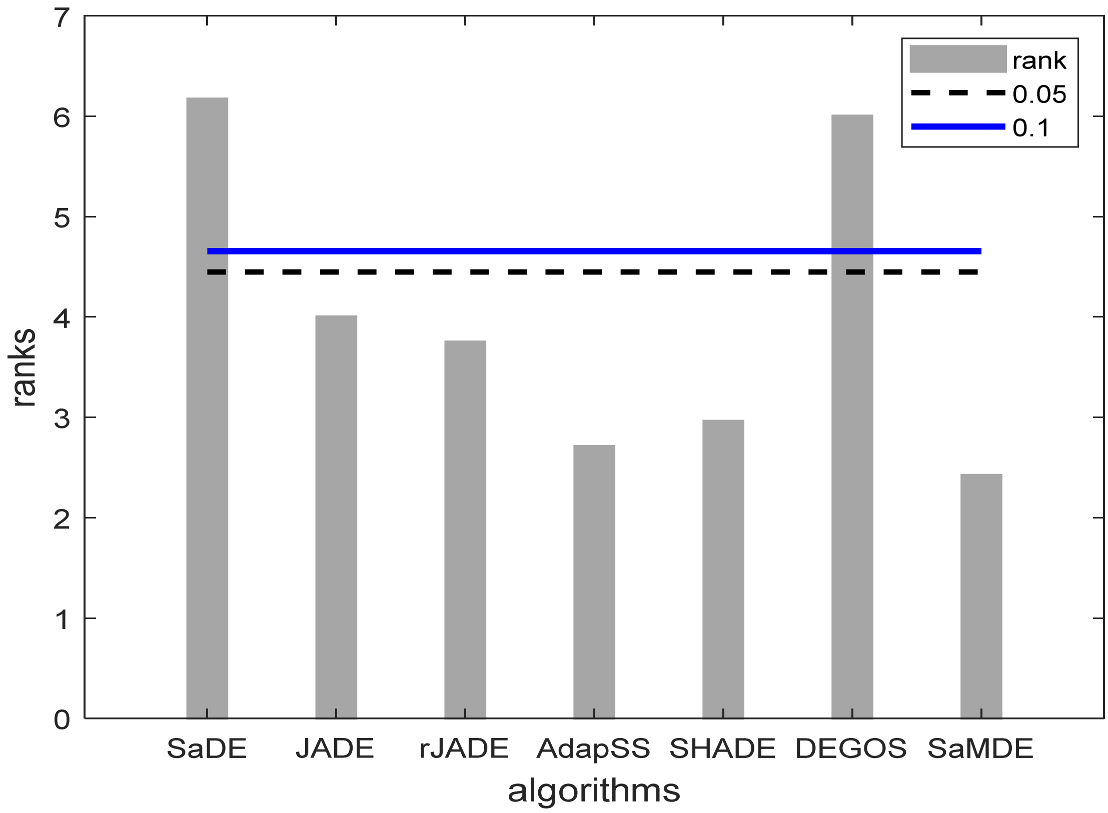


Figure 5. Bonferroni–Dunn bar charts on the results as given in Table 7. The Bar represents the algorithm's rank and horizontal cut lines represent the significant levels.

5.4. Performance Evaluation of SaMDE on CEC2017 Functions

In this section a performance evaluation of SaMDE has been tested on more complex and widely used benchmark functions taken from the IEEE CEC2017 test suit. The performance of SaMDE has been compared with seven recent DE variants as SHADE [5], Deexp [24], iLSHADE [37], DEGOS [55], CJADE [56], PAIDDE [57] and TRADE [58]. We have also compared the performance with another metaheuristic algorithm, HMRFO [68], which is an enhanced variant of the recently developed Manta Ray Foraging Optimization Algorithm. The results for Deexp, iLSHADE are taken from [24], and for PAIDDE, DEGoS, CJADE, IMODE and SHADE are taken from [57]. For TRADE and HMRFO, the original results have been obtained by using the code given in [67]. The population size and maximum NFEs have been taken as 100 and 300,000, respectively, for all algorithms.

In Table 8, all results are collected in terms of average error and standard deviation of 50 runs. According to the table, SaMDE, TRADE, DEexp, iLSHADE, PAIDDE, DEGoS, CJADE, IMODE, SHADE and HMRFO have obtained the best results on 14, 7, 10, 4, 8, 3, 2, 1, 5 and 0 functions, respectively. From the win/loss/tie row of the table, we can see that SaMDE surpasses the algorithm TRADE, DEexp, ILSHADE, PAIDDE, DEGoS, CJADE, IMODE, SHADE and HMRFO on 19, 15, 17, 14, 24, 25, 27, 21 and 27 cases, respectively.

Table 8. Performance evaluation of SaMDE on IEEE CEC2017 for D = 30 in terms of average error and standard deviation.

F	***SaMDE***		***TRADE***		***DEexp***		***iLSHADE***		**PAIDDE**	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
C_1	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$
C_3	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$2.31 \times 10^{+01}$	$4.16 \times 10^{+01}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	5.57×10^{-15}	1.71×10^{-14}	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$
C_4	$5.85 \times 10^{+01}$	$0.00 \times 10^{+00}$	$5.97 \times 10^{+01}$	$2.34 \times 10^{+00}$	$5.90 \times 10^{+01}$	$1.51 \times 10^{+00}$	$5.77 \times 10^{+01}$	$8.41 \times 10^{+00}$	$5.85 \times 10^{+01}$	1.15×10^{-14}
C_5	$1.14 \times 10^{+01}$	$3.85 \times 10^{+00}$	$1.91 \times 10^{+01}$	$4.64 \times 10^{+00}$	$9.83 \times 10^{+00}$	$2.75 \times 10^{+00}$	$7.76 \times 10^{+00}$	$1.70 \times 10^{+00}$	$6.85 \times 10^{+00}$	$1.48 \times 10^{+00}$
C_6	1.13×10^{-11}	6.12×10^{-08}	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	2.60×10^{-09}	1.71×10^{-08}	1.21×10^{-08}	5.04×10^{-08}	4.56×10^{-09}	2.49×10^{-08}
C_7	$4.12 \times 10^{+01}$	$4.23 \times 10^{+00}$	$5.47 \times 10^{+01}$	$9.38 \times 10^{+00}$	$4.19 \times 10^{+01}$	$3.20 \times 10^{+00}$	$3.79 \times 10^{+01}$	$1.47 \times 10^{+00}$	$3.72 \times 10^{+01}$	$1.33 \times 10^{+00}$
C_8	$1.04 \times 10^{+00}$	$8.54 \times 10^{+00}$	$2.01 \times 10^{+01}$	$4.24 \times 10^{+00}$	$1.06 \times 10^{+01}$	$3.12 \times 10^{+01}$	$7.45 \times 10^{+00}$	$1.78 \times 10^{+00}$	$7.10 \times 10^{+00}$	$1.12 \times 10^{+00}$
C_9	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$
C_{10}	$1.06 \times 10^{+03}$	$2.25 \times 10^{+02}$	$7.29 \times 10^{+03}$	$3.11 \times 10^{+02}$	$1.42 \times 10^{+03}$	$2.15 \times 10^{+02}$	$1.74 \times 10^{+03}$	$3.15 \times 10^{+02}$	$1.41 \times 10^{+03}$	$2.53 \times 10^{+02}$
C_{11}	$4.97 \times 10^{+00}$	$2.86 \times 10^{+00}$	$1.29 \times 10^{+01}$	$1.91 \times 10^{+01}$	$3.63 \times 10^{+00}$	$4.40 \times 10^{+00}$	$1.42 \times 10^{+01}$	$2.25 \times 10^{+01}$	$1.91 \times 10^{+01}$	$2.54 \times 10^{+01}$
C_{12}	$1.19 \times 10^{+03}$	$5.54 \times 10^{+03}$	$1.30 \times 10^{+04}$	$8.83 \times 10^{+03}$	$3.87 \times 10^{+02}$	$2.02 \times 10^{+02}$	$8.80 \times 10^{+02}$	$3.91 \times 10^{+02}$	$1.05 \times 10^{+03}$	$3.91 \times 10^{+02}$
C_{13}	$1.79 \times 10^{+01}$	$1.63 \times 10^{+01}$	$2.45 \times 10^{+01}$	$5.61 \times 10^{+00}$	$1.46 \times 10^{+01}$	$6.90 \times 10^{+00}$	$1.85 \times 10^{+01}$	$8.47 \times 10^{+00}$	$1.63 \times 10^{+01}$	$6.14 \times 10^{+00}$
C_{14}	$1.19 \times 10^{+01}$	$6.81 \times 10^{+00}$	$2.41 \times 10^{+01}$	$5.81 \times 10^{+00}$	$1.47 \times 10^{+01}$	$7.49 \times 10^{+00}$	$2.18 \times 10^{+01}$	$1.08 \times 10^{+00}$	$2.07 \times 10^{+01}$	$4.95 \times 10^{+00}$
C_{15}	$3.32 \times 10^{+00}$	$1.72 \times 10^{+00}$	$6.65 \times 10^{+00}$	$2.59 \times 10^{+00}$	$3.41 \times 10^{+00}$	$2.00 \times 10^{+00}$	$3.70 \times 10^{+00}$	$1.94 \times 10^{+00}$	$3.36 \times 10^{+00}$	$1.70 \times 10^{+01}$
C_{16}	$2.69 \times 10^{+01}$	$1.17 \times 10^{+01}$	$1.56 \times 10^{+01}$	$9.27 \times 10^{+00}$	$8.41 \times 10^{+01}$	$8.64 \times 10^{+01}$	$4.88 \times 10^{+01}$	$6.91 \times 10^{+01}$	$6.58 \times 10^{+01}$	$8.32 \times 10^{+01}$
C_{17}	$2.42 \times 10^{+01}$	$4.55 \times 10^{+00}$	$2.72 \times 10^{+01}$	$2.76 \times 10^{+00}$	$2.66 \times 10^{+01}$	$8.90 \times 10^{+00}$	$3.81 \times 10^{+01}$	$5.23 \times 10^{+00}$	$3.34 \times 10^{+01}$	$6.61 \times 10^{+00}$
C_{18}	$2.76 \times 10^{+01}$	$2.56 \times 10^{+00}$	$2.59 \times 10^{+01}$	$1.02 \times 10^{+01}$	$2.10 \times 10^{+01}$	$1.69 \times 10^{+00}$	$2.14 \times 10^{+01}$	8.15×10^{-01}	$2.21 \times 10^{+01}$	$1.28 \times 10^{+00}$
C_{19}	$5.12 \times 10^{+00}$	$3.46 \times 10^{+00}$	$5.54 \times 10^{+00}$	$1.69 \times 10^{+00}$	$5.22 \times 10^{+00}$	$1.40 \times 10^{+00}$	$8.53 \times 10^{+00}$	$1.97 \times 10^{+00}$	$6.52 \times 10^{+00}$	$1.89 \times 10^{+00}$
C_{20}	$1.19 \times 10^{+01}$	$1.36 \times 10^{+00}$	$2.07 \times 10^{+01}$	$6.92 \times 10^{+00}$	$2.88 \times 10^{+00}$	$3.09 \times 10^{+01}$	$4.80 \times 10^{+01}$	$1.86 \times 10^{+01}$	$3.32 \times 10^{+01}$	$6.44 \times 10^{+00}$
C_{21}	$2.22 \times 10^{+02}$	$6.14 \times 10^{+00}$	$2.20 \times 10^{+02}$	$4.90 \times 10^{+00}$	$2.10 \times 10^{+02}$	$3.06 \times 10^{+00}$	$2.08 \times 10^{+01}$	$1.65 \times 10^{+00}$	$2.07 \times 10^{+02}$	$1.48 \times 10^{+00}$
C_{22}	$1.00 \times 10^{+02}$	$0.00 \times 10^{+00}$	$1.00 \times 10^{+02}$	$0.00 \times 10^{+00}$	$1.00 \times 10^{+02}$	1.01×10^{-01}	$1.00 \times 10^{+02}$	1.00×10^{-13}	$1.00 \times 10^{+02}$	$0.00 \times 10^{+00}$
C_{23}	$3.58 \times 10^{+02}$	$7.79 \times 10^{+00}$	$3.62 \times 10^{+02}$	$8.58 \times 10^{+00}$	$3.45 \times 10^{+02}$	$4.40 \times 10^{+00}$	$3.51 \times 10^{+02}$	$4.50 \times 10^{+00}$	$3.48 \times 10^{+02}$	$2.53 \times 10^{+00}$
C_{24}	$2.34 \times 10^{+02}$	$6.41 \times 10^{+00}$	$4.42 \times 10^{+02}$	$5.44 \times 10^{+00}$	$4.22 \times 10^{+02}$	$3.12 \times 10^{+00}$	$4.25 \times 10^{+02}$	$2.70 \times 10^{+00}$	$4.25 \times 10^{+02}$	$1.38 \times 10^{+00}$
C_{25}	$3.86 \times 10^{+02}$	6.33×10^{-02}	$3.87 \times 10^{+02}$	2.69×10^{-02}	$3.86 \times 10^{+02}$	1.29×10^{-02}	$3.87 \times 10^{+02}$	2.40×10^{-02}	$3.86 \times 10^{+02}$	2.73×10^{-02}
C_{26}	$1.15 \times 10^{+03}$	$2.45 \times 10^{+02}$	$9.88 \times 10^{+02}$	$8.03 \times 10^{+01}$	$8.59 \times 10^{+02}$	$4.77 \times 10^{+01}$	$9.08 \times 10^{+02}$	$4.67 \times 10^{+01}$	$9.23 \times 10^{+02}$	$3.32 \times 10^{+01}$
C_{27}	$4.98 \times 10^{+02}$	$2.51 \times 10^{+00}$	$4.93 \times 10^{+02}$	$1.15 \times 10^{+01}$	$5.00 \times 10^{+02}$	$8.00 \times 10^{+00}$	$5.04 \times 10^{+02}$	$6.86 \times 10^{+00}$	$5.02 \times 10^{+02}$	$5.10 \times 10^{+00}$
C_{28}	$3.01 \times 10^{+02}$	$5.63 \times 10^{+01}$	$3.32 \times 10^{+02}$	$5.17 \times 10^{+01}$	$3.27 \times 10^{+02}$	$3.19 \times 10^{+02}$	$4.77 \times 10^{+01}$	$4.12 \times 10^{+01}$	$3.22 \times 10^{+02}$	$4.63 \times 10^{+01}$
C_{29}	$4.58 \times 10^{+02}$	$1.26 \times 10^{+02}$	$4.05 \times 10^{+02}$	$2.63 \times 10^{+01}$	$4.26 \times 10^{+02}$	$1.61 \times 10^{+01}$	$4.46 \times 10^{+02}$	$1.05 \times 10^{+01}$	$4.34 \times 10^{+02}$	$8.34 \times 10^{+00}$
C_{30}	$1.95 \times 10^{+03}$	$8.40 \times 10^{+01}$	$2.06 \times 10^{+03}$	$5.17 \times 10^{+01}$	$2.00 \times 10^{+03}$	$3.73 \times 10^{+01}$	$2.03 \times 10^{+03}$	$5.90 \times 10^{+01}$	$1.99 \times 10^{+03}$	$8.32 \times 10^{+01}$
w/l/t			19/7/3		15/9/5		17/9/3		14/9/6	

F	***DEGoS***		***CJADE***		***IMODE***		***SHADE***		***HMRFO***	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
C_1	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	9.11×10^{-11}	1.50×10^{-03}	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	1.01×10^{-11}	$2.46 \times 10^{+03}$
C_3	1.82×10^{-01}	9.97×10^{-01}	1.22×10^{-04}	$1.80 \times 10^{+04}$	1.91×10^{-07}	8.30×10^{-09}	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$2.70 \times 10^{+02}$	$6.90 \times 10^{+05}$
C_4	$4.98 \times 10^{+01}$	$2.23 \times 10^{+02}$	$4.78 \times 10^{+01}$	$2.37 \times 10^{+02}$	$2.21 \times 10^{+01}$	$2.87 \times 10^{+02}$	$4.09 \times 10^{+01}$	$2.75 \times 10^{+02}$	$4.76 \times 10^{+01}$	$3.61 \times 10^{+01}$
C_5	$3.60 \times 10^{+01}$	$3.65 \times 10^{+02}$	$2.67 \times 10^{+01}$	$4.80 \times 10^{+00}$	$2.63 \times 10^{+02}$	$4.17 \times 10^{+00}$	$1.79 \times 10^{+01}$	$2.30 \times 10^{+00}$	$4.49 \times 10^{+01}$	$3.63 \times 10^{+01}$
C_6	5.52×10^{-06}	1.85×10^{-05}	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$5.82 \times 10^{+01}$	$6.39 \times 10^{+00}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$6.47 \times 10^{+00}$	$2.07 \times 10^{+01}$
C_7	$1.64 \times 10^{+02}$	$5.55 \times 10^{+01}$	$5.46 \times 10^{+01}$	$3.41 \times 10^{+00}$	$9.22 \times 10^{+02}$	$3.14 \times 10^{+02}$	$4.83 \times 10^{+01}$	$3.09 \times 10^{+00}$	$1.56 \times 10^{+02}$	$1.01 \times 10^{+00}$
C_8	$7.68 \times 10^{+01}$	$7.25 \times 10^{+01}$	$2.71 \times 10^{+01}$	$4.64 \times 10^{+00}$	$2.19 \times 10^{+01}$	$4.00 \times 10^{+01}$	$1.86 \times 10^{+01}$	$3.04 \times 10^{+00}$	$2.31 \times 10^{+01}$	$1.78 \times 10^{+01}$
C_9	7.57×10^{-02}	3.23×10^{-01}	5.96×10^{-03}	2.27×10^{-03}	$5.59 \times 10^{+03}$	$1.52 \times 10^{+03}$	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$	$6.21 \times 10^{+02}$	$4.86 \times 10^{+01}$
C_{10}	$5.41 \times 10^{+03}$	$2.26 \times 10^{+03}$	$1.91 \times 10^{+03}$	$2.34 \times 10^{+01}$	$3.81 \times 10^{+03}$	$4.76 \times 10^{+02}$	$1.94 \times 10^{+03}$	$2.18 \times 10^{+02}$	$2.35 \times 10^{+03}$	$6.44 \times 10^{+02}$
C_{11}	$1.73 \times 10^{+01}$	$2.05 \times 10^{+01}$	$2.84 \times 10^{+01}$	$2.05 \times 10^{+01}$	$1.96 \times 10^{+02}$	$4.83 \times 10^{+01}$	$2.68 \times 10^{+01}$	$2.70 \times 10^{+01}$	$3.71 \times 10^{+01}$	$1.21 \times 10^{+01}$
C_{12}	$8.59 \times 10^{+03}$	$8.50 \times 10^{+03}$	$1.26 \times 10^{+03}$	$7.07 \times 10^{+02}$	$1.12 \times 10^{+03}$	$3.75 \times 10^{+02}$	$1.76 \times 10^{+03}$	$1.34 \times 10^{+03}$	$1.87 \times 10^{+03}$	$1.31 \times 10^{+04}$
C_{13}	$3.32 \times 10^{+01}$	$2.37 \times 10^{+01}$	$5.53 \times 10^{+01}$	$3.04 \times 10^{+01}$	$3.98 \times 10^{+02}$	$1.70 \times 10^{+02}$	$3.83 \times 10^{+01}$	$2.10 \times 10^{+01}$	$7.08 \times 10^{+01}$	$1.06 \times 10^{+04}$
C_{14}	$2.49 \times 10^{+01}$	$7.38 \times 10^{+00}$	$4.69 \times 10^{+03}$	$1.26 \times 10^{+04}$	$1.92 \times 10^{+02}$	$5.61 \times 10^{+01}$	$2.86 \times 10^{+01}$	$9.02 \times 10^{+00}$	$5.62 \times 10^{+02}$	$9.00 \times 10^{+02}$
C_{15}	$9.62 \times 10^{+00}$	$5.33 \times 10^{+00}$	$3.22 \times 10^{+01}$	$2.32 \times 10^{+01}$	$2.15 \times 10^{+02}$	$8.73 \times 10^{+01}$	$1.64 \times 10^{+01}$	$1.36 \times 10^{+01}$	$3.58 \times 10^{+01}$	$3.60 \times 10^{+03}$
C_{16}	$1.62 \times 10^{+02}$	$2.99 \times 10^{+02}$	$4.19 \times 10^{+02}$	$1.59 \times 10^{+02}$	$1.48 \times 10^{+03}$	$4.69 \times 10^{+02}$	$3.32 \times 10^{+02}$	$1.28 \times 10^{+02}$	$3.34 \times 10^{+02}$	$2.76 \times 10^{+02}$
C_{17}	$3.82 \times 10^{+01}$	$1.40 \times 10^{+01}$	$6.88 \times 10^{+01}$	$1.77 \times 10^{+01}$	$8.73 \times 10^{+02}$	$2.64 \times 10^{+02}$	$4.52 \times 10^{+01}$	$1.10 \times 10^{+01}$	$1.36 \times 10^{+02}$	$1.35 \times 10^{+02}$
C_{18}	$4.33 \times 10^{+01}$	$5.51 \times 10^{+01}$	$5.68 \times 10^{+03}$	$2.14 \times 10^{+04}$	$1.61 \times 10^{+02}$	$7.49 \times 10^{+01}$	$6.65 \times 10^{+01}$	$5.64 \times 10^{+01}$	$6.81 \times 10^{+02}$	$4.46 \times 10^{+04}$
C_{19}	$6.06 \times 10^{+00}$	$2.51 \times 10^{+00}$	$6.30 \times 10^{+02}$	$2.43 \times 10^{+03}$	$5.96 \times 10^{+02}$	$3.58 \times 10^{+02}$	$8.67 \times 10^{+00}$	$3.31 \times 10^{+00}$	$1.43 \times 10^{+02}$	$2.51 \times 10^{+03}$
C_{20}	$3.31 \times 10^{+01}$	$3.92 \times 10^{+01}$	$1.16 \times 10^{+02}$	$5.88 \times 10^{+01}$	$6.78 \times 10^{+02}$	$1.95 \times 10^{+02}$	$9.65 \times 10^{+01}$	$5.34 \times 10^{+01}$	$1.27 \times 10^{+02}$	$1.24 \times 10^{+02}$
C_{21}	$2.39 \times 10^{+02}$	$4.81 \times 10^{+01}$	$2.25 \times 10^{+02}$	$4.20 \times 10^{+00}$	$4.16 \times 10^{+02}$	$3.21 \times 10^{+01}$	$2.19 \times 10^{+02}$	$3.76 \times 10^{+00}$	$2.40 \times 10^{+02}$	$1.79 \times 10^{+01}$
C_{22}	$1.00 \times 10^{+02}$	8.30×10^{-14}	$1.00 \times 10^{+02}$	5.60×10^{-05}	$1.33 \times 10^{+03}$	$1.96 \times 10^{+03}$	$1.00 \times 10^{+02}$	$0.00 \times 10^{+00}$	$2.37 \times 10^{+02}$	3.07×10^{-13}
C_{23}	$3.75 \times 10^{+02}$	$2.89 \times 10^{+01}$	$3.72 \times 10^{+02}$	$4.78 \times 10^{+00}$	$7.98 \times 10^{+02}$	$8.40 \times 10^{+01}$	$3.65 \times 10^{+02}$	$4.86 \times 10^{+00}$	$4.08 \times 10^{+02}$	$2.37 \times 10^{+01}$
C_{24}	$4.49 \times 10^{+02}$	$2.81 \times 10^{+01}$	$4.40 \times 10^{+02}$	$3.32 \times 10^{+00}$	$9.59 \times 10^{+02}$	$7.34 \times 10^{+01}$	$4.36 \times 10^{+02}$	$3.52 \times 10^{+00}$	$4.91 \times 10^{+02}$	$2.04 \times 10^{+01}$
C_{25}	$3.86 \times 10^{+02}$	6.34×10^{-01}	$3.87 \times 10^{+02}$	1.41×10^{-01}	$3.94 \times 10^{+02}$	$1.82 \times 10^{+01}$	$3.87 \times 10^{+02}$	1.79×10^{-01}	$3.87 \times 10^{+02}$	$1.30 \times 10^{+01}$
C_{26}	$1.20 \times 10^{+03}$	$2.45 \times 10^{+02}$	$1.17 \times 10^{+03}$	$7.74 \times 10^{+01}$	$4.43 \times 10^{+03}$	$1.14 \times 10^{+03}$	$1.10 \times 10^{+03}$	$6.07 \times 10^{+01}$	$1.41 \times 10^{+03}$	$8.08 \times 10^{+02}$
C_{27}	$5.00 \times 10^{+02}$	$1.14 \times 10^{+01}$	$5.03 \times 10^{+02}$	$6.70 \times 10^{+00}$	$7.60 \times 10^{+02}$	$1.25 \times 10^{+02}$	$5.03 \times 10^{+02}$	$6.50 \times 10^{+00}$	$5.31 \times 10^{+02}$	$1.52 \times 10^{+01}$
C_{28}	$3.44 \times 10^{+02}$	$5.71 \times 10^{+01}$	$3.28 \times 10^{+02}$	$4.77 \times 10^{+01}$	$3.36 \times 10^{+02}$	$5.80 \times 10^{+01}$	$3.39 \times 10^{+02}$	$5.31 \times 10^{+01}$	$3.32 \times 10^{+02}$	$5.51 \times 10^{+01}$
C_{29}	$4.13 \times 10^{+02}$	$3.82 \times 10^{+01}$	$4.71 \times 10^{+02}$	$3.57 \times 10^{+01}$	$1.56 \times 10^{+03}$	$4.14 \times 10^{+02}$	$4.65 \times 10^{+02}$	$1.90 \times 10^{+01}$	$5.70 \times 10^{+02}$	$1.68 \times 10^{+02}$
C_{30}	$2.13 \times 10^{+03}$	$1.21 \times 10^{+02}$	$2.15 \times 10^{+03}$	$1.66 \times 10^{+02}$	$4.36 \times 10^{+03}$	$1.42 \times 10^{+03}$	$2.08 \times 10^{+03}$	$1.08 \times 10^{+02}$	$2.32 \times 10^{+03}$	$1.09 \times 10^{+03}$
w/l/t	24/2/3		25/2/2		27/2/0		21/4/4		27/2/0	

The Wilcoxon rank sum test is presented in Table 9. The higher sum of positive ranks shows the superiority of SaMDE over its competitors. However, the p-values show that the performance of DEexp, iLSHADE and PAIDDE will be considered as significantly equal with SaMDE, whereas SaMDE has a significant advantage over the rest of its competitors.

The results of the Friedman test and the Bonferroni–Dunn test on the CEC 2017 performance are given in Table 10. SaMDE obtained lowest rank and hence proves its significance over others.

Table 9. Result of Wilcoxon-rank sum test on the results obtained in Table 8.

Algorithms		**Pairwise Rank**	**ΣR^+**	**ΣR^-**	**z Value**	***p*-Value**	**Sig at α = 0.05**
	TRADE	(1.29, 1.71)	271	080	2.426	0.015	+
	DE*exp*	(1.40, 1.60)	168	132	0.514	0.607	=
	iLSHADE	(1.34, 1.66)	211	114	1.305	0.192	=
	PAIDDE	(1.41, 1.59)	170	105	0.989	0.323	=
SaMDE *vs.*	DEGoS	(1.12, 1.88)	326	025	3.821	<0.001	+
	CJADE	(1.11, 1.89)	344	007	4.280	<0.001	+
	IMODE	(1.07, 1.93)	421	014	4.400	<0.001	+
	SHADE	(1.21, 1.79)	289	036	3.404	<0.001	+
	HMRFO	(1.07, 1.93)	421	014	4.400	<0.001	+

'+' means significantly better and '=' means significantly equal.

Table 10. Friedman Ranks Critical difference (CD) calculated by Bonferroni–Dunn's procedure on the results obtained in Table 8.

	SaMDE	**TRADE**	**DEexp**	**iLSHADE**	**PAIDDE**	**DEGoS**	**CJADE**	**IMODE**	**SHADE**	**HMRFO**	**CD (α = 0.1)**	**CD (α = 0.05)**
Rank	3.00	4.89	3.05	4.02	3.36	6.46	6.98	9.04	5.82	8.38	2.018	2.204

Figure 6 represents the bar graphs of rank of the algorithms by taking SaMDE as a control algorithm. Graph shows that the ranks of SaMDE, TRADE, DEexp, iLSHADE and PAIDDE lie under the control lines of a significant level at α = 0.1 and α = 0.05 and hence these algorithms will be considered as significantly equal with each other and also significantly better than rest of the algorithms.

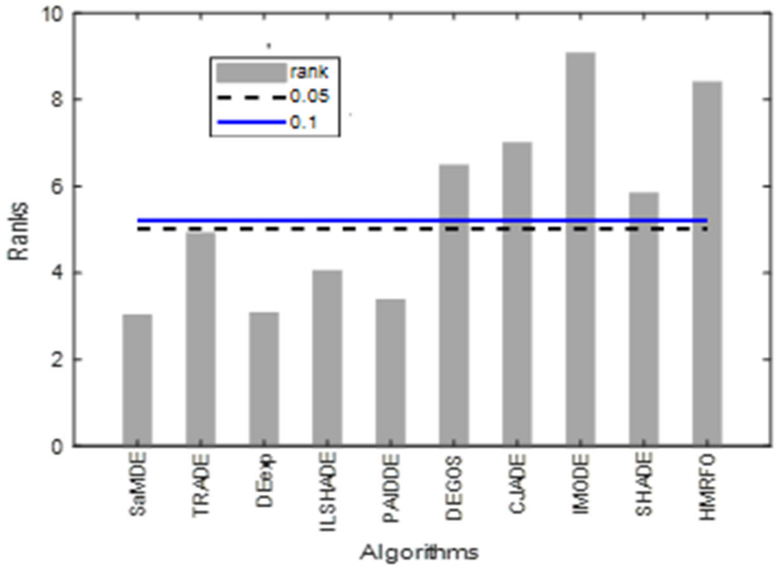
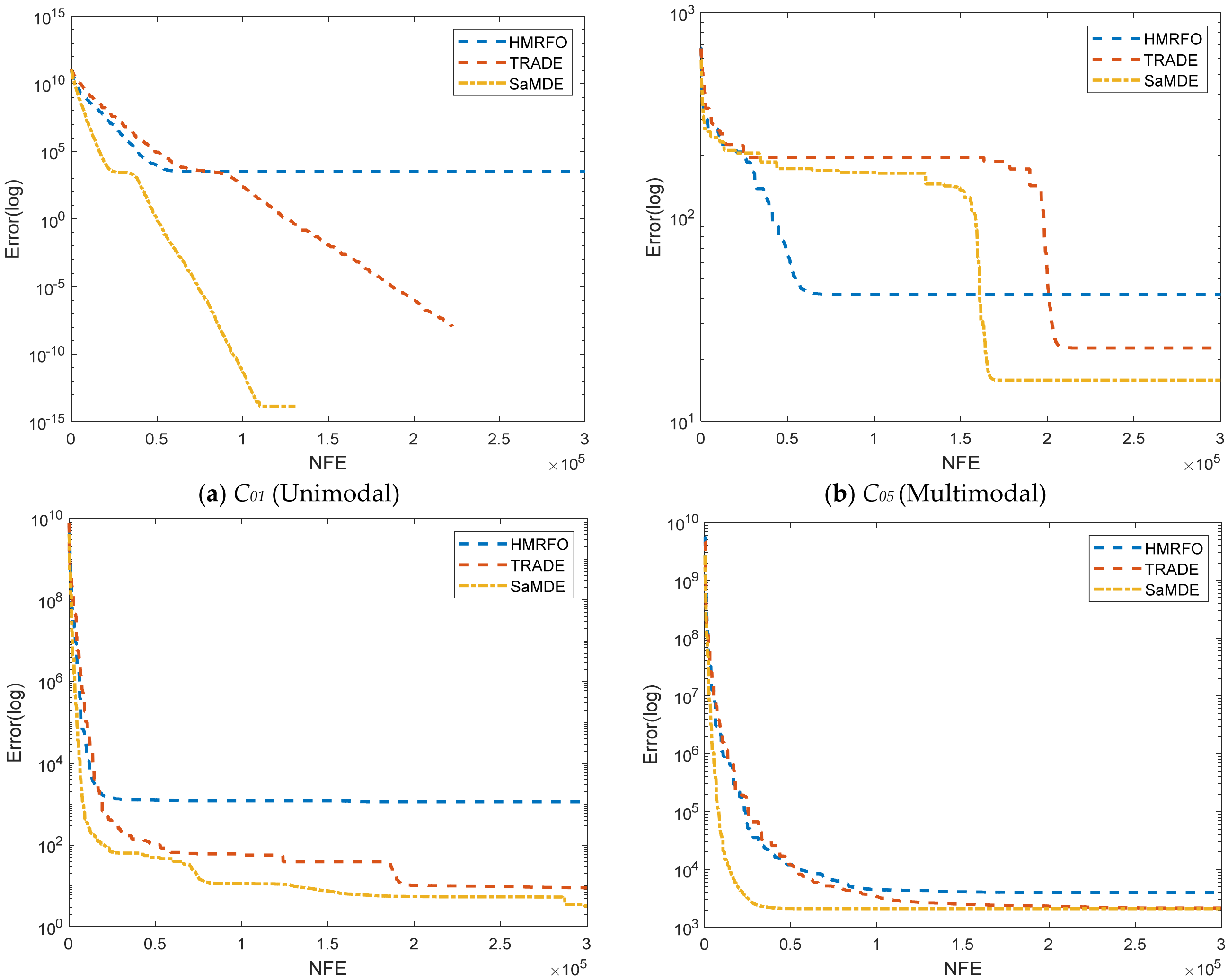


Figure 6. Bonferroni–Dunn bar charts on the results as given in Table 10. The Bar represents the algorithm's rank and horizontal cut lines represent the significant levels.

To analyze the convergence behavior of SaMDE, the convergence graphs of SaMDE with TRADE and HMRFO on selected functions C_1, C_5, C_{15} and C_{30} are represented in Figure 7. Graphs confirm the robustness of SaMDE to achieve the accuracy rapidly.



(**a**) C_{01} (Unimodal) (**b**) C_{05} (Multimodal)

(**c**) C_{15} (Hybrid) (**d**) C_{30} (Composite)

Figure 7. Convergence graphs for CEC-2017 functions: (**a**) C_{01} (**b**) C_{05} (**c**) C_{15} *and* (**d**) C_{30}.

5.5. Performance Evaluation of SaMDE on Molicular Potential Energy Problem

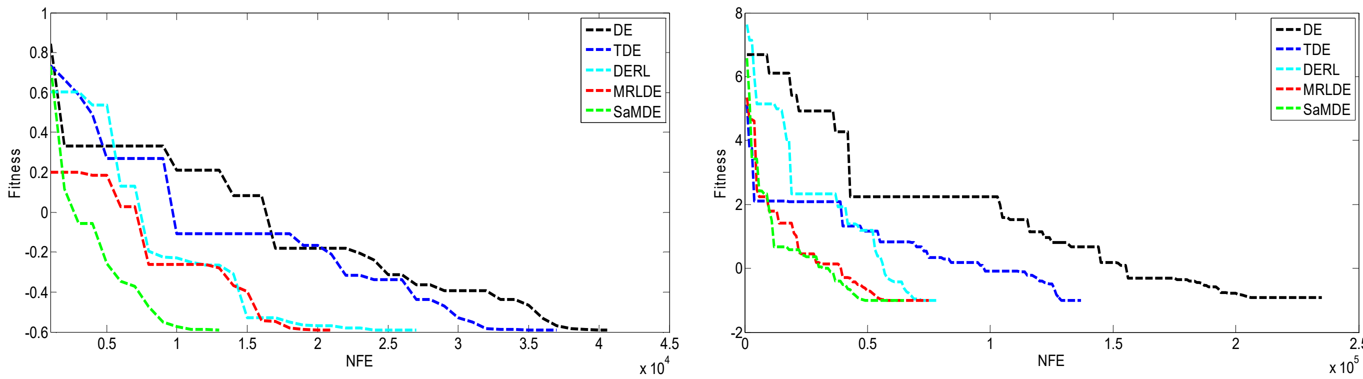
The experimental results for DE, TDE, DERL, MRLDE and SaMDE are given in Table 11. In order to make a fair comparison, similar, all parameter settings are used for each algorithm as given in previous section. The experiments are executed for n = 10, 15, 20 and 25 where n indicates the number of beads. The results are obtained in terms of best, worst and mean fitness value and standard deviation (SD) on 50 runs.

It can be seen that for n = 10, the fitness value obtained by SaMDE is −0.589389 which is best among those obtained by other algorithms. It can also be noticed that all algorithms perform similarly in terms of best fitness value. MRLDE gives the second-best performance for n = 10. Similarly, it can be seen that SaMDE gives the best results for n = 15, 20 and 25. MRLDE gives the second best performance while DERL and TDE take third and fourth place, respectively, for each n = 15, 20 and 25.

Figure 8 represents the convergence graphs' potential molecular energy problem at n = 10 and 20. The X-axis and Y-axis represent the NFE and its corresponding error value, respectively. By the graphs it can be easily seen that SaMDE performs faster and confirms its robustness over other variants.

Table 11. Experimental results for molecular potential energy problem in terms of fitness value.

No. of Beads	**Max−NFE**	**Fitness**	**DE**	**TDE**	**DERL**	**MRLDE**	**SaMDE**
$n = 10$	50,000	Best	−0.589389	−0.589389	−0.589389	−0.589389	−0.589389
		Worst	−0.507152	−0.507152	−0.576621	−0.589338	−0.589389
		Mean	−0.523599	−0.556494	−0.581728	−0.589368	−0.589389
		SD	3.28×10^{-02}	4.02×10^{-02}	6.25×10^{-03}	2.47×10^{-05}	0.00×10^{-00}
$n = 15$	150,000	Best	−0.493301	−0.493409	−0.493420	−0.493420	−0.493420
		Worst	−0.328946	−0.258712	−0.247861	−0.347986	−0.411183
		Mean	−0.263733	−0.296743	−0.329177	−0.365739	−0.476932
		SD	1.08×10^{-01}	6.55×10^{-02}	5.76×10^{-02}	5.57×10^{-02}	3.28×10^{-02}
$n = 20$	200,000	Best	−0.673352	−0.995013	−1.009910	−1.000530	−1.000570
		Worst	−0.344406	−0.508879	−0.426067	−0.673352	−0.836098
		Mean	−0.606527	−0.623435	−0.705211	−0.853121	−0.918450
		SD	1.31×10^{-01}	8.36×10^{-02}	1.52×10^{-01}	8.33×10^{-02}	7.35×10^{-02}
$n = 25$	250,000	Best	−0.830098	−0.904371	−0.904489	−0.904500	−0.904501
		Worst	−0.165625	−0.247286	−0.395887	−0.330674	−0.494571
		Mean	−0.248202	−0.362993	−0.412335	−0.462137	−0.576693
		SD	1.03×10^{-01}	8.38×10^{-02}	1.59×10^{-01}	1.33×10^{-01}	1.66×10^{-01}



(**a**) $n = 10$ (**b**) $n = 20$

Figure 8. Convergence graphs of molecular potential energy problem for (**a**) $n = 10$, (**b**) $n = 20$.

6. Conclusions

In this study, a novel self-adaptive method called SaMDE has been developed and used to MRLDE, a mutation-based enhanced DE variant. This method applies the MRLDE algorithm after selecting an appropriate mutation strategy from DNDE and SPIDE. The performance of the proposed approach has been investigated on 50 benchmark problems and on a real-life application in the potential molecular energy problem. The summary of the paper is given below:

- During the initial series of trials, the proposed SaMDE is compared against DE and its parent versions MRLDE, DNDE, and SPIDE. SaMDE provides the best performance among all versions, whereas MRLDE provides the second-best performance.
- In the second series of studies, the performance of SaMDE is compared to five other enhanced DE variations, namely SaDE, JADE, rJADE, APadapSS-JADE, SHADE and DEGOS. The comparison is made in terms of average error and standard deviation. The numerical results demonstrate that SaMDE outperforms every other algorithm. However, the non-parametric tests show that APadapSS-JADE and SHADE performs significantly equal to the SaMDE.
- The third series of experiments includes the performance evaluation of SaMDE on more complicated benchmark functions taken from the CEC 2017 test suit. The results

of SaMDE have been compared with 07 other recent DE variants named SHADE, DEexp, iLSHADE, DEGOS, CJADE, PAIDDE, TRADE and one metaheuristic named HMRFO. The numerical and statistical results demonstrated that SaMDE was superior to all of the algorithms except DEexp, iLSHADE and PAIDDE whose performances are significantly equal to the SaMDE.

- Finally, the performance of SaMDE is validated to minimize the molecular potential energy. In this problem, the number of local minima increases exponentially with the dimension of the problem. Hence experiments are conducted on various size (n = 10, 15, 20 and 25) of molecular beads. Compared to DE, TDE, DERL, and MRLDE, SaMDE provides more exact results with a faster convergence rate, as shown by the results.

Future work will first concentrate on extending the SaMDE to tackle constrained and multi objective optimization problems. Second, utilizing the self-adaptive technique that has been suggested, it will be fascinating to hybridize different metaheuristics algorithms.

Author Contributions: Conceptualization, P.K. and M.A.; methodology, P.K. and M.A. software, P.K. and M.A.; validation, P.K. and M.A.; formal analysis, P.K.; investigation, P.K. and M.A.; resources, P.K. and M.A.; data curation, P.K.; writing—original draft preparation, P.K. and M.A.; writing—review and editing, P.K. and M.A.; visualization, P.K. and M.A.; supervision, P.K. and M.A.; project administration, P.K.; funding acquisition, M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia (Grant No. 4486).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: All related data is contained within the article.

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

References

1. 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]
2. Liu, J.; Lampinen, J. On setting the control parameter of the differential evolution algorithm. In Proceedings of the 8th International Mendel Conference on Soft Computing, Brno, Czech Republic, 7–9 June 2002; pp. 11–18.
3. Weber, M.; Neri, F.; Tirronen, V. A study on scale factor/crossover interaction in distributed differential evolution. *Artif. Intellegent Rev.* **2013**, *39*, 195–224. [CrossRef]
4. Brest, J.; Greiner, S.; Boskovic, B.; Mernik, M.; Zumer, V. Self adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Trans. Evol. Comput.* **2006**, *10*, 646–657. [CrossRef]
5. Tanabe, R.; Fukunaga, A. Success-history based parameter adaptation for differential evolution. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 71–78.
6. Guo, H.; Li, Y.; Li, J.; Sun, H.; Wang, D.; Chen, X. Differential evolution improved with self-adaptive control parameters based on simulated annealing. *Swarm Evol. Comput.* **2014**, *19*, 52–67. [CrossRef]
7. Meng, Z.; Pan, J.S.; Tseng, K.K. PaDE: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization. *Knowl. Based Syst.* **2019**, *168*, 80–99. [CrossRef]
8. Pan, J.S.; Yang, C.; Meng, F.J.; Chen, Y.X.; Meng, Z.Y. A parameter adaptive DE algorithm on real-parameter optimization. *J. Intell. Fuzzy Syst.* **2020**, *38*, 5775–5786. [CrossRef]
9. Meng, Z.; Yang, C. Two-stage differential evolution with novel parameter control. *Inf. Sci.* **2022**, *596*, 321–342. [CrossRef]
10. Huynh, T.N.; Do, D.T.T.; Lee, J. Q-Learning-based parameter control in differential evolution for structural optimization. *Appl. Soft Comput.* **2021**, *107*, 107464. [CrossRef]
11. Stanovov, V.; Akhmedova, S.; Semenkin, E. Neuro evolution for parameter adaptation in differential evolution. *Algorithms* **2022**, *15*, 122. [CrossRef]
12. Bajer, D. Parameter control for differential evolution by storage of successful values at an individual level. *J. Comput. Sci.* **2023**, *68*, 101985. [CrossRef]
13. Teo, J. Exploring dynamic self-adaptive populations in differential evolution. *Soft Comput.* **2006**, *10*, 673–686. [CrossRef]
14. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M.A. Opposition based differential evolution. *IEEE Trans. Evol. Comput.* **2008**, *12*, 64–79. [CrossRef]
15. De Melo, V.V.; BotazzoDelbem, A.C. Investigating smart sampling as a population initialization method for Differential Evolution in continuous problems. *Inf. Sci.* **2012**, *193*, 36–53. [CrossRef]

16. 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.
17. Zhu, W.; Tang, Y.; Fang, J.A.; Zhang, W. Adaptive population tuning scheme for differential evolution. *Inf. Sci.* **2013**, *223*, 164–191. [CrossRef]
18. Polikolainen, I.; Neri, F.; Carafinni, F. Cluster-Based Population Initialization for differential evolution frameworks. *Inf. Sci.* **2015**, *297*, 216–235. [CrossRef]
19. Ma, Y.; Bai, Y. A multi-population differential evolution with best-random mutation strategy for large scale global optimization. *Appl. Intell.* **2020**, *50*, 1510–1526. [CrossRef]
20. Meng, Z.; Zhong, Y.; Yang, C. CS-DE: Cooperative Strategy based Differential Evolution with population diversity enhancement. *Inf. Sci.* **2021**, *577*, 663–696. [CrossRef]
21. Stanovov, V.; Akhmedova, S.; Semenkin, E. Dual-Population Adaptive Differential Evolution Algorithm L-NTADE. *Mathematics* **2022**, *10*, 4666. [CrossRef]
22. Zeng, Z.; Zhang, M.; Chen, T. A new selection operator for differential evolution algorithm. *Knowl. Based Syst.* **2021**, *226*, 107150. [CrossRef]
23. Kumar, P.; Garg, V. Advanced Selection Operation for Differential Evolution Algorithm. In *Design and Applications of Nature Inspired Optimization. Women in Engineering and Science*; Singh, D., Garg, V., Deep, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2022; pp. 55–74. [CrossRef]
24. Meng, Z.; Chen, Y. Differential Evolution with exponential crossover can be also competitive on numerical optimization. *Appl. Soft Comput.* **2023**, *146*, 110750. [CrossRef]
25. Fan, H.Y.; Lampinen, J. A trigonometric mutation operation to differential evolution. *J. Glob. Optim.* **2003**, *27*, 105–129. [CrossRef]
26. Fan, H.Y.; Lampinen, J.; Dulikravich, G.S. Improvements to mutation donor formulation of differential evolution. In Proceedings of the International Congress on Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems Eurogen, CIMNE, Barcelona, Spain, 2–4 May 2003.
27. Kumar, P.; Pant, M.; Ali, M.; Singh, H.P. Enhanced DE with Weighted Base Vector for Unconstrained Global Optimization. *Indian J. Sci. Technol.* **2017**, *10*, 18. [CrossRef]
28. Kaelo, P.; Ali, M.M. A numerical study of some modified differential evolution algorithms. *Eur. J. Oper. Res.* **2006**, *169*, 1176–1184. [CrossRef]
29. Kaelo, P.; Ali, M.M. Differential evolution algorithms using hybrid mutation. *Comput. Optim. Appl.* **2007**, *37*, 231–246. [CrossRef]
30. Zhang, J.; Sanderson, A. JADE: Adaptive differential evolution with optional external archive. *IEEE Trans EvolComput.* **2009**, *13*, 945–958.
31. Das, S.; Abraham, A.; Chakraborty, U.; Konar, A. Differential evolution using a neighborhood based mutation operator. *IEEE Trans. Evol. Comput.* **2009**, *13*, 526–553. [CrossRef]
32. Epitropakis, M.G.; Tasoulis, D.K.; Pavlidis, N.G.; Plagianakos, V.P.; Vrahatis, M.N. Enhancing differential evolution utilizing proximity-based mutation operators. *IEEE Trans. Evol. Comput.* **2011**, *15*, 99–118. [CrossRef]
33. Kumar, P.; Pant, M. Enhanced mutation strategy for differential evolution. In Proceedings of the 2012 IEEE Congress on Evolutionary Computation (CEC 12), Brisbane, QLD, Australia, 10–15 June 2012; pp. 1–6. [CrossRef]
34. Kumar, P.; Kumar, S.; Pant, M.; Singh, V.P. Interpolation based mutation variants of differential evolution. *Int. J. Appl. Evol. Comput.* **2012**, *3*, 34–50. [CrossRef]
35. Gong, W.; Cai, Z. Differential evolution with ranking based mutation operators. *IEEE Trans. Cybern.* **2013**, *43*, 2066–2081. [CrossRef]
36. Xiang, W.L.; Meng, X.L.; An, M.Q.; Li, Y.Z.; Gao, M.X. An Enhanced Differential Evolution Algorithm Based on Multiple Mutation Strategies. *Comput. Intell. Neurosci.* **2015**, *2015*, 285730. [CrossRef]
37. 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, BC, Canada, 24–29 July 2016; pp. 1188–1195.
38. Wei, Z.; Xie, X.; Bao, T.; Yu, Y. A random perturbation modified differential evolution algorithm for unconstrained optimization problems. *Soft Comput.* **2019**, *23*, 6307–6321. [CrossRef]
39. Sallam, K.M.; Elsayed, S.M.; Chakrabortty, R.K.; Ryan, M.J. Improved multi-operator differential evolution algorithm for solving unconstrained problems. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–8.
40. Meng, Z.; Yang, C. Hip-DE: Historical population based mutation strategy in Differential Evolution with parameter adaptive mechanism. *Inf. Sci.* **2021**, *562*, 44–77. [CrossRef]
41. Qin, A.K.; Suganthan, P.N. Self adaptive differential evolution algorithm for numerical optimization. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; pp. 1785–1791.
42. Qin, A.K.; Huang, V.L.; Suganthan, P.N. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans. Evol. Comput.* **2009**, *13*, 398–417. [CrossRef]
43. Gong, W.; Fialho, A.; Cai, Z.; Li, H. Adaptive strategy selection in differential evolution for numerical optimization: An empirical study. *Inf. Sci.* **2011**, *181*, 5364–5386. [CrossRef]

44. Mallipeddi, R.; Suganthana, P.N.; Pan, Q.K.; Tasgetiren, M.F. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Appl. Soft Comput.* **2011**, *11*, 1679–1696. [CrossRef]
45. Pan, Q.K.; Suganthan, P.N.; Wang, L.; Gao, L.; Mallipeddi, R. A differential evolution algorithm with self-adapting strategy and control parameters. *Comput. Oper. Res.* **2011**, *38*, 394–408. [CrossRef]
46. Wang, Y.; Cai, Z.; Zhang, Q. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Trans. Evol. Comput.* **2011**, *15*, 55–66. [CrossRef]
47. Islam, S.M.; Das, S.; Ghosh, S.; Roy, S.; Suganthan, P.N. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2012**, *42*, 482–500. [CrossRef]
48. Elsayed, S.M.; Sarker, R.A.; Essam, D.L. An improved self-adaptive differential evolution algorithm for optimization problems. *IEEE Trans. Ind. Inform.* **2013**, *9*, 89–99. [CrossRef]
49. Cui, L.; Li, G.; Lin, Q.; Chen, J.; Lu, N. Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations. *Comput. Oper. Res.* **2016**, *67*, 155–173. [CrossRef]
50. Sun, X.; Wang, D.; Kang, H.; Shen, Y.; Chen, Q. A two stage differential evolution algorithm with mutation strategy combination. *Symmetry* **2021**, *13*, 2163. [CrossRef]
51. Eltaeib, T.; Mahmood, A. Differential Evolution: A Survey and Analysis. *Appled. Sci.* **2018**, *8*, 1945. [CrossRef]
52. Opara, K.R.; Arabas, J. Differential Evolution: A survey of theoretical analyses. *Swarm Evol. Comput.* **2019**, *44*, 546–558. [CrossRef]
53. Bilal, P.M.; Pant, M.; Zaheer, H.; Garcia-Hernandez, L.; Abraham, A. Differential evolution: A review of more than two decades of research. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103479. [CrossRef]
54. Noman, N.; Iba, H. Accelerating differential evolution using anadaptive local search. *IEEE Trans. Evol. Comput.* **2008**, *12*, 107–125. [CrossRef]
55. Yu, Y.; Gao, S.; Wang, Y.; Todo, Y. Global optimum-based search differential evolution. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 379–394. [CrossRef]
56. Gao, S.; Yu, Y.; Wang, Y.; Wang, J.; Cheng, J.; Zhou, M. Chaotic local search-based differential evolution algorithms for optimization. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 3954–3967. [CrossRef]
57. Li, X.; Wang, K.; Yang, H.; Tao, S.; Feng, S.; Gao, S. PAIDDE: A Permutation-Archive Information Directed Differential Evolution Algorithm. *IEEE Access* **2022**, *10*, 50384–50402. [CrossRef]
58. Cai, Z.; Yang, X.; Zhou, M.C.; Zhan, Z.H.; Gao, S. Toward explicit control between exploration and exploitation in evolutionary algorithms: A case study on differential evolution. *Inf. Sci.* **2023**, *649*, 119656. [CrossRef]
59. Kumar, S.; Kumar, P.; Sharma, T.K.; Pant, M. Bi-level thresholding using PSO, artificial bee colony and MRLDE embedded with Otsu method. *Memetic Comput.* **2013**, *5*, 323–334. [CrossRef]
60. Kumar, P.; Pant, M.; Singh, V.P. Modified random localization based DE for static economic power dispatch with generator constraints. *Int. J. Bio. Inspired Comput.* **2014**, *6*, 250–261. [CrossRef]
61. Kumar, P.; Pant, M. Recognition of noise source in multi sounds field by modified random localized based DE algorithm. *Int. J. Syst. Assur. Eng. Manag.* **2018**, *9*, 245–261. [CrossRef]
62. Tang, K.; Yao, X.; Suganthan, P.N.; MacNish, C.; Chen, Y.P.; Chen, C.M.; Yang, Z. Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization. Technical Report Nature Inspired Computation and Applications Laboratory, USTC, China. 2007. Available online: https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=0209DA24BB65282AEE466F0215316DDA?doi=10.1.1.515.821&rep=rep1&type=pdf (accessed on 20 July 2023).
63. Awad, N.; Ali, M.; Liang, J.; Qu, B.; Suganthan, P. *Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Bound Constrained Real-Parameter Numerical Optimization*; Technical Report 201311; Computational Intelligence Laboratory, Zhengzhou University: Zhengzhou, China, 2016.
64. Lavor, C.; Maculan, N. A Function to Test Methods Appliedto Global Minimization of Potential Energy of Molecules. *Numer. Algorithms* **2004**, *35*, 287–300. [CrossRef]
65. Deep, K.; Sashi; Katiyar, V.K.; Nagar, A.K. Minimization of molecular potential energy function using newly developed real coded genetic algorithms. *Int. J. Optim. Control. Theor. Appl.* **2012**, *2*, 51–58. [CrossRef]
66. Carrasco, J.; García, S.; Rueda, M.M.; Das, S.; Herrera, F. Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm Evol. Comput.* **2020**, *54*, 100665. [CrossRef]
67. Available online: http://toyamaailab.githhub.io/soucedata.html (accessed on 25 July 2023).
68. Tang, Z.; Wang, K.; Tao, S.; Todo, Y.; Wang, R.L.; Gao, S. Hierarchical manta ray foraging optimization with weighted fitness –distance balance selection. *Int. J. Comput. Intell. Syst.* **2023**, *16*, 114. [CrossRef]