Available online at www.HighTechJournal.org

HighTech and Innovation Journal

Vol. 4, No. 2, June, 2023

ISSN: 2723-9535

An Improved Differential Evolution Algorithm for Numerical Optimization Problems

Irfan Farda [1], Arit Thammano [1]*

[1] Computational Intelligence Laboratory, School of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand.

Received 28 January 2023; Revised 03 May 2023; Accepted 11 May 2023; Published 01 June 2023

Abstract

The differential evolution algorithm has gained popularity for solving complex optimization problems because of its simplicity and efficiency. However, it has several drawbacks, such as a slow convergence rate, high sensitivity to the values of control parameters, and the ease of getting trapped in local optima. In order to overcome these drawbacks, this paper integrates three novel strategies into the original differential evolution. First, a population improvement strategy based on a multi-level sampling mechanism is used to accelerate convergence and increase the diversity of the population. Second, a new self-adaptive mutation strategy balances the exploration and exploitation abilities of the algorithm by dynamically determining an appropriate value of the mutation parameters; this improves the search ability and helps the algorithm escape from local optima when it gets stuck. Third, a new selection strategy guides the search to avoid local optima. Twelve benchmark functions of different characteristics are used to validate the performance of the proposed algorithm. The experimental results show that the proposed algorithm performs significantly better than the original DE in terms of the ability to locate the global optimum, convergence speed, and scalability. In addition, the proposed algorithm is able to find the global optimal solutions on 8 out of 12 benchmark functions, while 7 other well-established metaheuristic algorithms, namely NBOLDE, ODE, DE, SaDE, JADE, PSO, and GA, can obtain only 6, 2, 1, 1, 1, 1, and 1 functions, respectively.

Keywords: Optimization; Differential Evolution; Self-adaptive; Metaheuristic.

1. Introduction

Most of our real-life problems are complex optimization problems [1] due to the high dimensionality, nonlinearity, discontinuity, and multimodality of the problems. Metaheuristic techniques have been used effectively to solve complex optimization problems, replacing the mathematical programming techniques that have limited success in solving the incrasingly complex optimization problems due to several drawbacks such as limited global strength, dependency on gradient information, and considerable computation time [2]. Metaheuristic techniques can be classified into two categories [3]. The first is a neighborhood-based algorithm, often known as a local search algorithm. Two well-known neighborhood-based algorithms are simulated annealing and tabu search [4]. The second category is the population-based algorithm, most of which is inspired by natural evolution.

Examples of population-based algorithms are the genetic algorithm introduced by Holland [5], the differential evolution by Storn and Price [6], the particle swarm optimization by Kennedy and Eberhart [7], the ant colony optimization by Dorigo et al. [8], the firefly algorithm by Yang [9], and the artificial bee colony algorithm proposed by Karaboga and Basturk [10]. Nowadays, researchers are working on further improving the performance of population-based algorithms. Several improved population-based optimization algorithms are reviewed and discussed in the following paragraphs.

* Corresponding author: arit@it.kmitl.ac.th

http://dx.doi.org/10.28991/HIJ-2023-04-02-014

Sheng et al. [11] presented a particle swarm optimizer (PSO) with multi-level population sampling and a dynamic p-learning mechanism for solving large-scale optimization problems. The particle learning strategy used in the original PSO had limited capability to archive a balanced evolutionary search; thus, a multi-level population sampling mechanism was proposed to solve this problem. The mechanism divided the population into L levels based on the fitness values before evolution. The higher level corresponded to a smaller index and contained particles with higher fitness, while the lower level associated with a larger index contained particles with lower fitness. Then a sub-swarm was dynamically selected from the particles at different levels. During the early evolution stage, particles from lower levels had a higher probability of being selected; this increased the exploration ability. However, in the later evolution stage, the exploitation ability was increased by selecting particles from higher levels. Finally, the dynamic p-learning mechanism was adopted to support efficient search while preserving swarm diversity.

Ali et al. [12] designed three different strategies to improve the real-coded genetic algorithm (GA). In the first strategy, a new multi-parent crossover based on a differential evolution algorithm, called DEx, was used to improve the crossover operator in a genetic algorithm. In this strategy, the new differential evolution crossover increased the population diversity of the real-code genetic algorithm to avoid premature convergence and stagnation stage. The second strategy was GA-DEx$_{SPS}$, which provided an alternative way to select a parent during the differential evolution crossover process, DEx. The last strategy, called GA-αDEx$_{SPS}$, used an adaptive parameter setting scheme to set the value of α in the proposed DEx crossover. The parameter setting was adaptively based on the aging mechanism and the history of successful parent selection to increase the search's effectiveness during the optimization process.

Chu et al. [13] proposed an adaptive heterogeneous competition for solving global optimization problems based on the artificial bee colony (ABC) algorithm, which had slow convergence and poor generalization drawbacks. The proposed ABC-AHC algorithm divided the population into two bee swarms and conducted two heterogeneous searching approaches, a superior tracking strategy (STS) and a sub-gradient strategy (SGS) in each swarm. STS enhanced the exploration ability of the algorithm, while SGS was used to improve the convergence speed and the local exploitation. During the optimization process, an adaptive competition and migration mechanism (ACM) was adopted to dynamically balance the heterogeneous processes of the two bee swarms.

She et al. [14] presented a self-adaptive and gradient-based cuckoo search algorithm (HAGCS) for solving global optimization problems. The original cuckoo search (CS) algorithm was inspired by the reproduction strategy of cuckoos, which laid their eggs in other birds' nests. Having fewer control parameters and an efficient exploration ability became the CS algorithm's advantages. However, the CS algorithm was not sufficiently effective in solving multimodal optimization problems. To address this issue, the authors provided a method to increase convergence speed and enhance the capability of the CS algorithm to solve a wide range of high-dimensional problems. The HAGCS took advantage of three variants of the CS algorithm, i.e., the gradient-based cuckoo search (GBCS) algorithm, the hybrid self-adaptive cuckoo search (HSACS) algorithm, and the gradient-based local optimization (GBLO) algorithm. Additionally, self-adaptation and diversity promotion schemes were adopted to prevent the premature convergence effect caused by the gradient method.

Wu et al. [15] improved the exploitation and exploration abilities of the firefly algorithm (FA) with a logarithmic-spiral path and an adaptive switch. In the proposed adaptive logarithmic spiral-Levy FA (AD-IFA), a Levy-flight FA (LF-FA), one of many variants of FA, was used as the core algorithm. LF-FA, which used a levy distribution approach to strengthen its exploration ability, had poor exploitation ability. Therefore, the authors adopted the logarithmic spiral approach to improve the exploitation ability of LF-FA. Furthermore, to balance the exploration mode of the Levy flight approach and the exploitation mode of the logarithmic spiral approach, an adaptive switch was used to determine which approach should be applied in the next iteration. If the best fitness value of the current iteration was significantly greater than that of the previous iteration, the exploitation mode would be chosen for the next iteration. However, if the best fitness value of the current iteration was significantly worse than that of the previous iteration, the exploration mode would be chosen for the next iteration.

Chen and Pi [16] proposed a flower pollination algorithm based on cloud mutation (CMFPA) to solve the continuous optimization problems. The original flower pollination algorithm (FPA), inspired by the natural pollination phenomenon of flowering plants, had poor exploitation ability and slow convergence speed. To overcome these drawbacks, CMFPA divided the evolution into two stages: global exploration and local mining. First, the new global search equation, designed to direct each individual solution toward the current population optimal solution and each individual's own historical optimal solution, was introduced to improve the search ability of each individual in the population and to expand the population's in-depth search of the problem space. Next, the cloud mutation deeply mined the current solution information to increase the chance of finding a better solution within the allowed number of iterations.

Sun et al. [17] proposed an adaptive differential evolution algorithm with two mutation strategies for global numerical optimization, called CSDE. First, a new dynamic adjustment strategy was used to balance the global and local search abilities of DE/current-to-pbest/1. The value of the scaling factor was adaptively adjusted, relying on the individual-independence macro-control parameter in the early stage and on the individual-dependence function in the later stage.

Second, to boost the exploitation ability around the promising pbest, a new mutation strategy called DE/pbest-to-rand/1, was introduced. In this mutation strategy, the value of the scaling factor relied on the individual-independence macro-control parameter in the early stage but depended on the modulo-based periodic parameter in later stages. Finally, the historical success rate of each mutation strategy was used to determine which one of the two mutation strategies would be selected to generate the mutant vector.

Deng et al. [18] presented the differential evolution algorithm with neighborhood mutation operators and opposition-based learning, namely NBOLDE. To overcome the limitation of DE/current-to-pbest/1, the new mutation strategy DE/neighbor-to-neighbor/1, which aimed to improve the convergence speed and accuracy, was presented. In contrast to DE/current-to-pbest/1, this new strategy selected individuals only from the local neighborhood. Furthermore, opposition-based learning was used to optimize the quality of the random initial population. The opposition-based learning selected the better solutions from current and reverse solutions and filtered out individuals with poor performance. It provided more opportunities to reach global optimal solutions, corrected the convergence direction, and increased stability in high-dimensional problems.

Zeng et al. [19] introduced a new selection operator to enhance the performance of the differential evolution algorithm. The authors focused on the drawbacks of the commonly used greedy selection operator. When the best solution was not trapped in a local optimum, a new selection operator acted the same as the greedy selection operator, for which the better vector between the trial and parent vectors was chosen to survive to the next generation. However, when the algorithm was trapped in a local optimum or in a stagnation state, three candidate vectors were selected. The best and second-best vectors of all discarded trial vectors were the first and second candidates, respectively. The third candidate was randomly selected from the successfully updated solutions. If none of the above three candidate vectors were able to guide the algorithm to escape from the stagnation state, the current value of the parent vector was replaced by the best value in the history of the parent vector.

Meng & Yang [20] solved real-parameter optimization problems using a two-stage differential evolution (TDE). TDE consisted of two stages; each stage employed different mutation strategies. A historical-solution-based mutation strategy that had better perception of the landscape of the objective function was used in the earlier stage of the evolution, while an inferior-solution-based mutation strategy that had the ability to balance the diversity of trial vector candidates and convergence speed was used in the later stage of the evolution. The TDE algorithm also included a population enhancement technique to solve the stagnation problem. The authors conducted experiments using a test suite containing 88 benchmarks from CEC2013, CEC2014, and CEC2017. The performance of the TDE algorithm was compared with several state-of-the-art differential evolution variants. The results showed that the TDE algorithm outperformed the other methods on most of the benchmarks tested.

Kumar et al. [21] identified a gap in the literature regarding the less explored initialization and selection operators of DE compared to mutation and crossover operators. To address this gap, they proposed a comprehensive approach that includes an orthogonal-array-based initialization, an ensemble of four mutation strategies, a parameter adaptation technique, and a conservative selection scheme. The authors conducted experiments to analyze the influence of the proposed initialization and selection schemes on several DE variants. They also compared the performance of their approach with that of other state-of-the-art approaches. The results showed that their proposed approach significantly improved the searchability and convergence speed of the DE.

Houssein et al. [22] proposed a modified version of the Adaptive Guided Differential Evolution (AGDE), called mAGDE. They integrated three mutation mechanisms and adapted control parameters into the original AGDE algorithm to get rid of its weaknesses–premature convergence and failing to maintain diversity in evolutionary processes. The effectiveness of the proposed algorithm was tested using CEC2020 benchmark problems. The results of the mAGDE algorithm demonstrated its effectiveness and robustness in solving complex engineering problems.

Deng et al. [23] proposed an improved adaptive differential evolution algorithm, called ACDE/F. Their main objective was to overcome the issues of premature convergence and local optimization that were common in traditional DE algorithms. The authors introduced three strategies—belief space strategy, generalized opposition-based learning strategy, and parameter adjustment strategy—to enhance the performance of the differential evolution algorithm. The experimental results showed that ACDE/F outperformed other state-of-the-art algorithms in terms of convergence speed and solution quality on a set of benchmark functions. In practical applicability, ACDE/F also effectively solved the real-world problem of airport gate allocation.

Yi et al. [24] presented a novel algorithm known as EJADE (Adaptive Differential Evolution with Ensembling Populations), designed to tackle continuous optimization problems. EJADE, built upon the JADE algorithm, employed two sets of mutation and crossover operators to achieve a better balance between exploration and exploitation capabilities. Additionally, an adaptive parameter control strategy was utilized to dynamically adjust the algorithm's parameter settings. Experimental results demonstrated that EJADE achieved competitive performance against several state-of-the-art DE algorithms on benchmark functions and a real-world wireless sensor localization application. It exhibited strong global search ability and rapid convergence speed.

According to the above-reviewed research, metaheuristic techniques typically suffer from two main problems: slow convergence speed and getting stuck in local optima. Researchers managed to overcome these two problems by proposing methods to increase population diversity and balance exploration and exploitation in metaheuristic searches. As the optimization problems get more complex, the current metaheuristic algorithms are not as effective as they used to be. In this paper, we focus on improving the performance of the differential evolution algorithm, which is known as one of the best metaheuristic algorithms. For the past 14 years, various DE variants have emerged as the top three best-performing optimizers in most Congress of Evolutionary Computation (CEC) competitions [2, 25]; they even ranked first in more than half of those competitions. Our proposed DE introduces three strategies to improve the performance of differential evolution. The first is the population improvement strategy, which adopts a multi-level sampling mechanism. This strategy aims to improve convergence speed and reduce the chance of being trapped in local optima. The new self-adaptive mutation strategy is introduced as the second strategy, which helps determine an appropriate value of the mutation parameters. Third, a new selection strategy with an external archive guides the search to avoid local optima.

The rest of this paper is organized as follows. Section II describes the original differential evolution algorithm. Section III explains our proposed algorithm. The experiments and discussion are reported in Section IV. In the end, Section V contains the conclusion.

2. Differential Evolution

Differential evolution (DE) is known as one of the best metaheuristic algorithms. Executing DE to solve optimization problems usually needs four steps: Population initialization, Mutation, Crossover, and Selection. A brief explanation of the four steps is explained below.

2.1. Population Initialization

In the first step, the DE algorithm randomly generates the initial population which comprises *NP* target vectors $X_i^G = (x_{i,1}, x_{i,2}, \dots, x_{i,j}, \dots, x_{i,d})$, where *G* is the number of generations, *i* = 1, 2, …, *NP*; *NP* is the number of target vectors in the population, and *d* is the number of dimensions of the problem. The initial target vectors are generated within the bound of the search space by using Equation 1:

$$x_{i,j} = S_j^{\min} + rand[0,1] \times (S_j^{\max} - S_j^{\min}) \tag{1}$$

where S_j^{max} and S_j^{min} is the maximum and minimum values of the search space on the jth dimension respectively.

2.2. Mutation

After initializing the candidate solution within a specific dimension and search space, the second step performs the mutation operation. This step generates mutant vectors $V_i^G = (v_{i,1}, v_{i,2}, \dots, v_{i,j}, \dots, v_{i,d})$ from the target vectors. Many of the mutation strategies are known for their efficient performance. Some of them are described as follows:

a) DE/best/1

$$V_i^G = X_{best}^G + F(X_{r1}^G - X_{r2}^G) \tag{2}$$

b) DE/best/2

$$V_i^G = X_{best}^G + F(X_{r1}^G - X_{r2}^G) + F(X_{r3}^G - X_{r4}^G) \tag{3}$$

c) DE/current-to-best/1

$$V_i^G = X_i^G + F(X_{best}^G - X_{r1}^G) + F(X_{r2}^G - X_{r3}^G) \tag{4}$$

d) DE/current-to-pbest/1

$$V_i^G = X_i^G + F(X_{pbest}^G - X_i^G) + F(X_{r1}^G - X_{r2}^G) \tag{5}$$

e) DE/rand/1

$$V_i^G = X_{r1}^G + F(X_{r2}^G - X_{r3}^G) \tag{6}$$

f) DE/rand/2

$$V_i^G = X_{r1}^G + F(X_{r2}^G - X_{r3}^G) + F(X_{r4}^G - X_{r5}^G) \tag{7}$$

where X_{best}^G is the best target vector in the current population while X_{r1}^G,X_{r2}^G,X_{r3}^G,X_{r4}^G and X_{r5}^G are randomly chosen target vectors from the current population. It is important to note that the randomly chosen target vector must not be X_i^G. F, a scale factor used to control the step size of the mutation, is a real number in the range (0,1]. As can be observed from the above mutation strategies, all of them consist of two types of components. The first type is a base vector used as the center of the search area, and the second type is a differential variation between two target vectors used to determine the search direction. For example, in the DE/best/1 strategy, the first term is the base vector X_{best}^G and the second term is the differential variation between the target vectors X_{r1}^G and X_{r2}^G.

2.3. Crossover

The third step generates the trial vector $U_i^G = (u_{i,1}, u_{i,2}, \dots, u_{i,j}, \dots, u_{i,d})$ by using the crossover operator to combine the target vector with the mutant vector. The most popular crossover strategy for DE algorithm is the binary crossover, described as follows:

$$u_{i,j} = \begin{cases} v_{i,j}; if\,(rand[0,1] \leq CR \; or \; j = j_{rand}) \\ x_{i,j}; otherwise \end{cases} \tag{8}$$

where the crossover parameter *CR* is a random real number in the range (0,1]; it controls the number of elements in the trial vector chosen from the mutant vector. j_{rand}, a random integer number in the range [1, *d*], is used to ensure that the generated trial vector is different from the target vector.

2.4. Selection

The last step selects the better one between the target and the trial vectors to fill up the population for the next generation. The selection strategy can be expressed as follows:

$$X_i^{G+1} = \begin{cases} U_i^G; if\; f(U_i^G) \leq f(X_i^G) \\ X_i^G; otherwise \end{cases} \tag{9}$$

where $f(U_i^G)$ is the objective value of the trial vector i and $f(X_i^G)$ denotes the objective value of the target vector i. The goal of this strategy is to ensure that the population always gets better or at least maintains the same state, but never becomes worse in every evolution process.

3. Proposed Algorithm

Although DE offers many advantages over other metaheuristic algorithm, it still has 2 major disadvantages. First, it has the possibility of being trapped in local optima that leads to premature convergence. Second, the performance of DE is very sensitive to the values of control parameters. The bad choice of control parameters causes an imbalance between exploration and exploitation of the algorithm, resulting in the inability to converge and premature convergence. Our proposed algorithm introduces three new strategies aimed to further increase the convergence speed of the DE and to overcome the above 2 disadvantages.

This section is divided into 4 subsections. Details of our three proposed strategies are described in the first three subsections. The first strategy is the population improvement strategy. The second strategy is the adaptive mutation strategy, and the third is the new selection strategy. The fourth subsection presents the step-by-step process of the proposed algorithm.

3.1. Population Improvement Strategy

In this subsection, a population improvement strategy is proposed. The purpose of this strategy is to increase the convergence speed of the DE algorithm and to reduce the chance of being trapped in local optima. This strategy employs a multi-level sampling mechanism. In the beginning of each iteration, before the mutation process, the second population is generated by using Equation 10.

$$X_{2,i}^G = X_i^G \times up \tag{10}$$

where *up* is a random number in the range (0, 1). Then, the second population is combined with the current population. The combined population of size 2NP is then sorted by fitness value from best to worst. Next, the sorted population is divided into *L* levels, L_0 to L_{L-1}. Next, we calculate the sampling probability of each level as follows:

$$PL_k = \frac{G}{G_{\max}}\left(PL_{k,Final} - PL_{k,Initial}\right) + PL_{k,Initial} \tag{11}$$

where *G* and G_{max} is the current generation and the maximum number of generations, respectively. $PL_{k,Initial}$ denotes the initial sampling probability of L_k and $PL_{k,Final}$ is the final sampling probability of L_k. They can be calculated by using Equations 12 and 13.

$$PL_{k,Initial} = \frac{k}{L-1} \tag{12}$$

$$PL_{k,Final} = 1 - PL_{k,Initial} \tag{13}$$

After that, a new population of size N is generated by randomly selecting SL_k vectors from each level k, $k = 0, 1, 2, \ldots, L-1$. The number of vectors selected from the level k is determined by Equation 14:

$$SL_k = \lfloor PL_k \times L \times 2 \rfloor \tag{14}$$

By using this strategy, in the early generation, the probability of the lower index level will be higher than the probability of the higher index level. Meaning that, we select more vectors with high fitness to accelerate the algorithm's speed in the early evolution of generation. In contrast, in the later generation, the probability of the higher index level will be higher than the probability of the lower index level; this causes the algorithm to select more vectors with low fitness to increase the diversity of the population. In other words, this strategy focuses on the exploitation ability to increase the algorithm's convergence in the early generation. The later generations focus on exploration ability to find a more promising solution and avoid local optima.

3.2. Adaptive Mutation Strategy

A mutation process is one of the most critical processes in the differential evolution algorithm. Finding appropriate values for the mutation parameters, which is time-consuming and very difficult, is essential to the convergence of the DE. Instead of using a trial-and-error method, the most common method to determine parameter values, a self-adaptive strategy is introduced in this research. The proposed self-adaptive strategy is an improved version of our preliminary research [26], a modified DE/current-to-best/1 strategy.

$$V_i^G = X_i^G + \lambda(X_{best}^G - X_i^G) + F(X_{r1}^G - X_{r2}^G) \tag{15}$$

where X_{best}^G is the target vector with the best fitness value in the generation G. λ and F denote the first and second mutation parameters, which is used to scale the difference between the best and the target vectors and the difference between two random target vectors, respectively. In each generation during evolution, the values of λ and F are self-adapted as follows:

$$\lambda^{G+1} = \begin{cases} \lambda^G + (\lambda^G \times C_1);\ if\ stagnation\ happens \\ \lambda^G - (\lambda^G \times C_1);\ otherwise \end{cases} \tag{16}$$

$$F^{G+1} = \begin{cases} F^G - (F^G \times C_2);\ if\ stagnation\ happens \\ F^G + (F^G \times C_2);\ otherwise \end{cases} \tag{17}$$

where C_1 and C_2 are adapting rate of the mutation parameters λ and F, respectively. λ and F have to be in the range of 0 to 1. If they are out of the range after self-adaptation, they are reset back to 0.5. For the first generation, the values of C_1 and C_2 are set to:

$$C_1 = C_2 = |N(\mu, \sigma)| \tag{18}$$

where $N(\mu, \sigma)$ generates a random number from the normal distribution with mean of μ and standard deviation of σ. For subsequent generations, however, the values of C_1 and C_2 are dynamically adapted according to the percentage of improvement from the previous generation.

$$C_1 = C_2 = \eta \left| \frac{f(X_{best}^G) - f(X_{best}^{G-1})}{f(X_{best}^{G-1}) + \varepsilon} \right| \tag{19}$$

where η is a scaling factor in the range [0, 1]. $f(X_{best}^G)$ is a fitness value of the best target vector in the current generation and $f(X_i^{G-1})$ denotes the fitness value of the best target vector in the previous generation. ε is a very small number to avoid division by zero.

3.3. Selection with External Archive Strategy

The differential evolution typically uses a greedy selection operator to generate a new population for the next generation. That means a better one between the trial vector and the target vector is selected to fill up the new population. The issue is that when the algorithm falls into a local optimum, the generated trial vectors are most likely not better than their counterpart target vectors. The target vectors are then selected to fill up the new population. As a result, the algorithm is not able to escape from the local optimum.

Our new selection strategy addresses this problem by introducing an external archive to store promising solutions obtained during the search. That is, during each generation, all mutant vectors as well as the target vectors and trial vectors not selected to be in the next generation population are added to the external archive. When the algorithm does not select the trial vector to fill up the new population, the algorithm will select the vector with the best fitness value from the external archive to fill up the new population. In this way, the diversity of the population is increased and the chance of being trapped in local optima is reduced.

3.4. Differential Evolution with Population Improvement, Adaptive Mutation, and New Selection Strategies

Our proposed algorithm combines the above three strategies to the DE algorithm. The goals of these three strategies are to increase the algorithm's performance in terms of convergence speed, exploration and exploitation abilities, and avoid falling too quickly in the local optimum. The flowchart of the PASDE algorithm is displayed in Figure 1, and the pseudocode of the PASDE algorithm is shown in Algorithm 1 and described below:

Step 1: Define the values of the parameters, such as the number of population (*NP*), the number of dimension (*d*), the boundary of the search space, the external archive size, and the maximum number of generations (G_{max}).

Step 2: Randomly generate the initial population by using (1).

Step 3: Perform the population improvement strategy as follows:

- Generate the second population by using (10).
- Combine the second population with the current population. Then the combined population is sorted by fitness value from best to worst.
- Divide the sorted population into *L* levels. Then calculate the sampling probability of each level by using (11).
- Randomly select vectors from each of the *L* levels to form a new population.

Step 4: Perform the mutation operation to generate the mutant vector by using (15).

Step 5: Add the mutant vector to the external archive and check the size of the external archive. If the size of the external archive is bigger than the allowed size, randomly delete one vector from the external archive to maintain the size of the external archive to the allowed size.

Step 6: Perform the crossover operation to generate a trial vector by using (8).

Step 7: Perform the selection operation as follows:

- Compare the fitness of the target vector with that of the trial vector. If the trial vector is better than the target vector, the trial vector is added to the next generation population while the target vector is stored in the external archive. However, if the target vector is better than the trail vector, store both the trial vector and the target vector in the external archive. Then the best vector in the external archive is added to the next generation population.
- Check the size of the external archive. If the size of the external archive is bigger than the allowed size, randomly delete one vector from the external archive to maintain the size of the external archive to the allowed size.

Step 8: Update the values of the mutation parameters, λ and F, by using (16) and (17).

Step 9: Repeat step 3-8 until the stopping criteria are met

Algorithm 1. Pseudocode of the PASDE Algorithm

```
1:  Set the values of the parameters;
2:  Randomly generate the initial population by using (1);
3:  $G$ = 1
4:  WHILE ($G < G_{max}$)
5:    Generate the second population by using (10);
6:    Combine the second population with the current population;
7:    Sort the combined population based on fitness value from best to worst;
8:    Split the sorted population into $L$ levels;
9:    Calculate the sampling probability of each level by using (11);
10:   Generate a new population by randomly selecting vectors from each of the $L$ levels;
11:   FOR $i$ = 1: $NP$
12:       Generate the mutant vector $V_i^G$ by using (15);
13:       Store $V_i^G$ in the external archive;
14:       IF the size of the external archive > the allowed size
```

```
15:             Delete one random vector from the external archive;
16:        END IF
17:        Generate the trial vector $U_i^G$ by using (8);
18:        IF $f(U_i^G) \leq f(X_i^G)$
19:             $X_i^{G+1} \leftarrow U_i^G$;
20:             Store $X_i^G$ in the external archive;
21:        ELSE
22:             Store $U_i^G$ and $X_i^G$ in the external archive;
23:             $X_i^{G+1} \leftarrow$ the best vector in the external archive;
24:        END IF
25:        IF the size of the external archive > the allowed size
26:             Delete one random vector from the external archive;
27:        END IF
28:    END FOR
29:    Update the values of $\lambda$ and $F$ by using (16) and (17);
30:    $G = G + 1$;
31: END WHILE
32: Output the best fitness of the population.
```

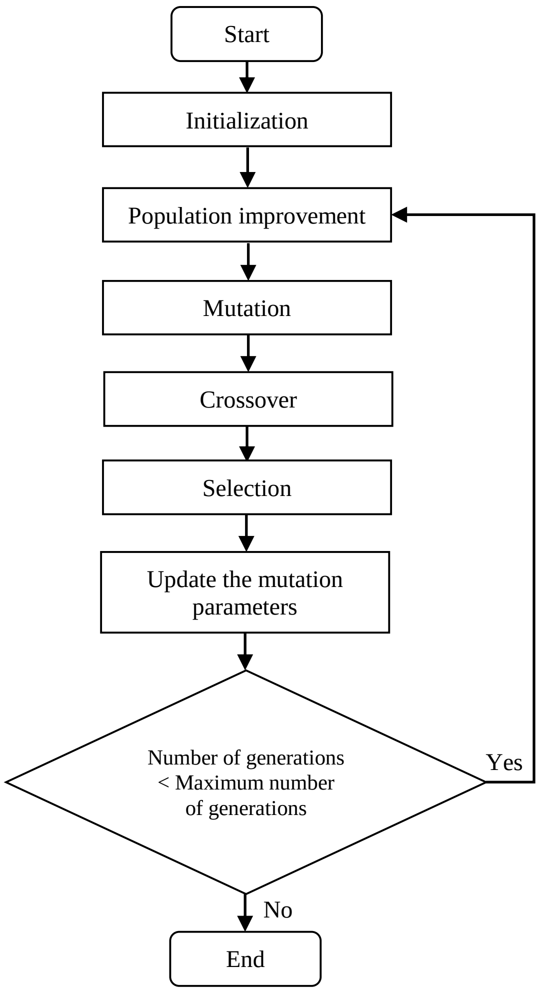

Start
Initialization
Population improvement
Mutation
Crossover
Selection
Update the mutation parameters
Number of generations < Maximum number of generations
Yes
No
End

Figure 1. Flowchart of PASDE Algorithm

4. Experiments and Discussion

In order to validate the performance of the proposed algorithm, experiments have been carried out in three parts. The first part validated the effects of the three proposed strategies. In the second part, the performance of the proposed algorithm was compared with that of other DE variants and that of recent state-of-the-art algorithms. The last part evaluated the scalability of the proposed algorithm. Twelve benchmark functions shown in Table 1 were used to validate the performance of the proposed algorithm. Functions F1 – F4, F9 and F12 are unimodal functions while functions F5 – F8, F10 and F11 are multimodal functions. For each function, thirty experimental repetitions were performed with different initial populations each time. The parameters of each algorithm were set to the values shown in Table 2. For the experiments in the first and second parts, the number of populations is set to 100, and the dimension is set to 30. In the third part, the dimension is set to 100, 200, 500, and 1000. The minimum, maximum, mean, and standard deviation from 30 runs are reported in Tables 3 to 5. The best results obtained among all algorithms are shown in bold.

Table 1. The benchmark functions used for evaluation

	Name	Equation	Search space and Global optimum
F1	Sphere function	$f_1(x) = \sum_{i=1}^{d} x_i^2$	$[-100,100]^d, f_{min} = 0$
F2	Schwefel's problem 2.22 function	$f_2(x) = \sum_{i=1}^{d} \lvert x \rvert + \prod_{i=1}^{d} \lvert x \rvert$	$[-10,10]^d, f_{min} = 0$
F3	Schwefel's problem 1.2 function	$f_3(x) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100,100]^d, f_{min} = 0$
F4	Sum square's function	$f_4(x) = \sum_{i=1}^{d} i x_i^2$	$[-1.28,1.28]^d, f_{min} = 0$
F5	Generalized Rosenbrock's function	$f_5(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_1^2)^2 + (x_i - 1)^2]$	$[-30,30]^d, f_{min} = 0$
F6	Rastrigin's function	$f_6(x) = \sum_{i=1}^{d} (x_i^2 - 10\, coscos(2\pi x_i) + 10)$	$[-5.12,5.12]^d, f_{min} = 0$
F7	Ackley's function	$f_7(x) = -20\, exp\left(-0.2 \frac{\sum_{i=1}^{d} x_i^2}{d}\right) - exp\left(\frac{\sum_{i=1}^{d} cos(2\pi x_i)}{d}\right) + 20 + exp(1)$	$[-32,32]^d, f_{min} = 0$
F8	Generalized Griewank Function	$f_8(x) = \frac{1}{4000} \sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600,600]^d, f_{min} = 0$
F9	Step function	$f_9(x) = \sum_{i=1}^{d} \lfloor x_i + 0.5 \rfloor^2$	$[-100,100]^d, f_{min} = 0$
F10	Quartic function	$f_{10}(x) = \sum_{i=1}^{d} i x_i^4 + Random[0,1)$	$[-1.28,1.28]^d, f_{min} = 0$
F11	Alpine function	$f_{11}(x) = \sum_{i=1}^{d} \lvert x_i\, sin(x_i) + 0.1 x_i \rvert$	$[-10,10]^d, f_{min} = 0$
F12	Sum of different power function	$f_{12}(x) = \sum_{i=1}^{d} \lvert x_i \rvert^{(i+1)}$	$[-1,1]^d, f_{min} = 0$

Table 2. The parameter settings

Algorithm	Parameter Values
PSO	$F = 0.5$; $CR = 0.9$; $C_1 = C_2 = 2$; $w = 1$
GA	$F = 0.3$; $CR = 0.9$
Basic DE	$F = 0.5$; $CR = 0.9$
ODE	$F = 0.5$; $CR = 0.9$; $Jr = 0.3$
JADE	$F = 0.5$; $CR = 0.5$; $P = 0.05$; c = 0.1
SaDE	$F = N(0.5,0.3)$; $CR = 0.5$, $CR \sim N(0.5,0.1)$
NBOLDE	$F_1 = F_2 = 0.4$; $CR = 0.9$; $Jr = 0.3$
PIDE	$F = 0.5$; $CR = 0.9$; $L = 10$
AMDE	$F = \lambda = 0.5$; $CR = 0.9$
NSDE	$F = 0.5$; $CR = 0.9$
PASDE	$F = \lambda = 0.5$; $CR = 0.9$; $L = 10$

Table 3. Comparative results of DE, PIDE, AMDE, NSDE and PASDE on 12 benchmark functions

	Algorithms	Min	Max	Mean	S.D.
F1	DE	1.6974E-14	2.2623E-13	7.3014E-14	5.5012E-14
	PIDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	AMDE	7.0950E-18	1.3022E-16	2.9284E-17	2.4714E-17
	NSDE	6.3854E-12	5.9747E-01	3.0579E-02	1.1187E-01
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F2	DE	1.9259E-07	8.1106E-07	4.4049E-07	1.6771E-07
	PIDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	AMDE	3.0319E-15	2.1799E-14	7.0535E-15	4.3455E-15
	NSDE	9.5112E-12	3.2420E-01	1.5694E-02	6.4003E-02
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F3	DE	2.5385E-01	4.8211E+00	9.7540E-01	7.9109E-01
	PIDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	AMDE	9.8445E-14	2.4544E-11	2.8544E-12	4.6750E-12
	NSDE	1.0544E-03	6.4700E+01	2.1942E+00	1.1806E+01
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F4	DE	1.9499E-17	4.4486E-16	1.3307E-16	1.0132E-16
	PIDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	AMDE	7.5002E-17	3.5929E-15	4.6002E-16	6.6421E-16
	NSDE	6.7649E-19	3.4488E-02	2.9517E-03	8.6791E-03
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F5	DE	1.4901E+01	1.8444E+01	1.6928E+01	1.0445E+00
	PIDE	1.6598E+01	**1.8298E+01**	1.7580E+01	**4.6479E-01**
	AMDE	**3.8555E-05**	7.1740E+01	**1.2337E+01**	1.2665E+01
	NSDE	1.6605E+01	1.1737E+04	7.2760E+02	2.1460E+03
	PASDE	1.0440E+01	1.9112E+01	1.4149E+01	2.1704E+00
F6	DE	3.8581E+01	4.8054E+01	4.2358E+01	2.2517E+00
	PIDE	1.8656E+00	2.8607E+00	**1.8990E+00**	**1.8164E-01**
	AMDE	7.9157E+00	2.4284E+01	**1.8990E+00**	4.2033E+00
	NSDE	1.5368E+01	4.3218E+01	2.6939E+01	7.7741E+00
	PASDE	**1.8655E+00**	**2.8605E+00**	1.9485E+00	2.2940E-01
F7	DE	2.9947E-08	1.2770E-07	7.1808E-08	2.5095E-08
	PIDE	**4.4409E-16**	**4.4409E-16**	**4.4409E-16**	**1.5044E-31**
	AMDE	2.0000E+01	2.0299E+01	2.0066E+01	8.9036E-02
	NSDE	2.0000E+01	2.0363E+01	2.0023E+01	7.9240E-02
	PASDE	**4.4409E-16**	**4.4409E-16**	**4.4409E-16**	**1.5044E-31**
F8	DE	4.1189E-19	1.5768E-12	2.9950E-13	3.4158E-13
	PIDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	AMDE	**0.0000E+00**	4.9176E-02	4.9176E-02	1.2613E-02
	NSDE	7.0409E-03	8.3365E-01	2.0886E-01	2.3265E-01
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F9	DE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	PIDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	AMDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	NSDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F10	DE	1.1926E-01	4.2119E-01	2.3608E-01	7.3156E-02
	PIDE	2.9972E-04	6.2865E-02	9.8887E-03	1.2185E-02
	AMDE	1.5260E-03	2.1866E-02	7.4927E-03	5.2448E-03
	NSDE	2.9010E-03	5.9100E-02	1.2370E-02	1.0190E-02
	PASDE	**1.6042E-07**	**1.0086E-02**	**1.9438E-03**	**2.3193E-03**

F11	DE	1.2985E-02	2.9831E-02	2.3668E-02	3.4492E-02
	PIDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	AMDE	1.0677E-15	7.0420E-01	2.3473E-02	1.2857E-01
	NSDE	5.4226E-13	7.8361E-01	9.3256E-02	2.0797E-01
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F12	DE	6.4180E-54	3.2805E-42	1.0935E-43	5.9893E-43
	PIDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	AMDE	1.0677E-15	3.0509E-33	3.0395E-34	6.3549E-34
	NSDE	5.4226E-13	3.5117E-07	3.4692E-08	8.8562E-08
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**

Table 4. Comparative results of PASDE and 7 state-of-the-art algorithms on functions F1 – F12

	Algorithms	Min	Max	Mean	S.D.
F1	DE	1.6974E-14	2.2623E-13	7.3014E-14	5.5012E-14
	ODE	1.2035E-32	3.1663E-30	6.1306E-31	7.1195E-31
	SaDE	6.5747E-39	1.1440E-36	1.1440E-36	2.3291E-37
	JADE	1.8597E-39	9.5777E-59	9.5770E-59	2.4897E-59
	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	PSO	4.9851E-13	6.5667E-03	2.2704E-04	1.1975E-03
	GA	6.7656E-05	1.2830E-03	4.4893E-04	3.4940E-04
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F2	DE	1.9259E-07	8.1106E-07	4.4049E-07	1.6771E-07
	ODE	6.0467E-15	3.6581E-14	1.4927E-14	7.5454E-15
	SaDE	2.3008E-21	1.9501E-20	7.7798E-21	4.6736E-21
	JADE	2.6280E-31	2.6770E-29	4.4648E-30	5.3048E-30
	NBOLDE	1.4989E-257	2.8506E-255	5.3820E-256	**0.0000E+00**
	PSO	2.2130E-07	9.7273E-01	8.2694E-02	1.8724E-01
	GA	3.3296E-03	1.9861E-02	9.6612E-03	3.9225E-03
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F3	DE	2.5385E-01	4.8211E+00	9.7540E-01	7.9109E-01
	ODE	6.1595E-04	4.2961E-02	6.9023E-03	8.3055E-03
	SaDE	3.1726E-03	1.4528E-01	3.2358E-02	3.0733E-02
	JADE	3.0187E-19	8.5610E-15	3.3336E-16	1.5579E-15
	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	PSO	9.0496E+02	1.8472E+04	1.0017E+04	4.2837E+03
	GA	4.5208E-02	7.3656E+00	8.6820E-01	1.3769E+00
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F4	DE	1.9499E-17	4.4486E-16	1.3307E-16	1.0132E-16
	ODE	8.6914E-35	3.9016E-32	2.9731E-33	7.0927E-33
	SaDE	6.0256E-41	8.0002E-39	1.0062E-39	1.6003E-39
	JADE	1.5816E-64	5.4011E-62	1.1997E-62	1.3785E-62
	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	PSO	5.3649E-28	1.1911E-05	4.7184E-07	2.1803E-06
	GA	1.5183E-07	3.1684E-06	9.1843E-07	7.6932E-07
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F5	DE	1.4901E+01	1.8444E+01	1.6928E+01	1.0445E+00
	ODE	2.4395E+01	2.7761E+01	2.6489E+01	7.0182E-01
	SaDE	9.5510E+00	7.6368E+01	2.5173E+01	1.0113E+01
	JADE	**5.5512E-16**	**3.9866E+00**	**6.6444E-01**	1.5111E+00
	NBOLDE	2.8584E+01	2.8836E+01	2.8728E+01	**7.2436E-02**
	PSO	2.2267E+00	2.4017E+02	6.2576E+01	4.9405E+01
	GA	3.1192E-03	5.2238E+01	1.6178E+01	1.5627E+01
	PASDE	1.0440E+01	1.9112E+01	1.4149E+01	2.1704E+00

F6	DE	3.8581E+01	4.8054E+01	4.2358E+01	2.2517E+00
	ODE	1.9227E+01	3.9403E+01	3.1637E+01	5.1056E+00
	SaDE	1.8823E+00	1.9149E+00	1.8956E+00	9.1411E-03
	JADE	1.8661E+00	1.8681E+00	1.8668E+00	5.5391E-04
	NBOLDE	1.7895E+01	3.8793E+01	3.2122E+01	4.8062E+00
	PSO	4.3668E+00	3.9020E+01	1.2684E+01	6.2202E+00
	GA	1.8656E+00	**1.8657E+00**	**1.8656E+00**	**3.7206E-05**
	PASDE	**1.8655E+00**	2.8605E+00	1.9485E+00	2.2940E-01
F7	DE	2.9947E-08	1.2770E-07	7.1808E-08	2.5095E-08
	ODE	4.4409E-15	7.9936E-15	6.5725E-15	1.7702E-15
	SaDE	4.4409E-15	9.3130E-01	3.1043E-02	1.7003E-01
	JADE	4.4409E-15	1.1551E+00	1.0805E-01	3.3144E-01
	NBOLDE	8.8818E-16	4.4409E-15	4.3225E-15	6.4863E-16
	PSO	2.0000E+01	2.0085E+01	2.0003E+01	1.5524E-02
	GA	2.4175E-05	2.0000E+01	1.0667E+01	1.0148E+01
	PASDE	**4.4409E-16**	**4.4409E-16**	**4.4409E-16**	**1.5044E-31**
F8	DE	4.1189E-19	1.5768E-12	2.9950E-13	3.4158E-13
	ODE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	SaDE	**0.0000E+00**	7.3960E-03	4.9307E-04	1.8764E-03
	JADE	**0.0000E+00**	1.9690E-02	3.5312E-03	5.7369E-03
	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	PSO	0.0000E+00	1.1778E-01	3.7167E-02	3.7461E-02
	GA	1.0532E-04	3.3516E-03	9.8277E-04	8.3058E-04
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F9	DE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	ODE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	SaDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	JADE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	PSO	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	GA	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F10	DE	1.1926E-01	4.2119E-01	2.3608E-01	7.3156E-02
	ODE	9.6422E-02	3.5449E-01	2.3909-01	5.8055E-02
	SaDE	1.1583E-03	**7.5251E-03**	3.6490E-03	**1.4981E-03**
	JADE	2.7253E-02	1.0342E+00	1.7611E-01	2.2069E-01
	NBOLDE	2.8253E-04	1.9839E-02	5.4950E-03	5.8967E-03
	PSO	4.2142E-02	1.5618E-01	8.9729E-02	3.1233E-02
	GA	1.2134E-02	4.0849E-02	2.5947E-02	8.5751E-03
	PASDE	**1.6042E-07**	1.0086E-02	**1.9438E-03**	2.3193E-03
F11	DE	1.2985E-02	2.9831E-02	2.3668E-02	3.4492E-02
	ODE	1.0738E-02	2.1359E-02	1.7502E-02	2.1971E-03
	SaDE	2.0005E-07	1.1145E-04	2.1164E-05	2.2453E-05
	JADE	4.4374E-32	6.1062E-16	2.2945E-16	2.6552E-16
	NBOLDE	2.4301E-258	2.1230E-256	3.5999E-257	**0.0000E+00**
	PSO	5.9425E-08	6.4402E+00	8.6268E-01	1.8420E+00
	GA	6.0349E-05	7.1342E-04	3.9119E-04	1.7319E-04
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F12	DE	6.4180E-54	3.2805E-42	1.0935E-43	5.9893E-43
	ODE	1.6380E-118	3.7777E-112	4.0795E-113	9.4811E-113
	SaDE	5.5304E-79	2.3642E-54	9.5760E-56	4.3786E-55
	JADE	1.1946E-140	1.7699E-128	1.0134E-129	3.7803E-129
	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	PSO	2.2237E-37	6.4402E+00	8.6268E-01	1.8420E+00
	GA	6.0349E-05	7.1342E-04	3.9119E-04	1.7319E-04
	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**

Table 5. Scalability performance of PASDE and NBOLDE on functions F1 – F12

	d	Algorithms	Min	Max	Mean	S.D.
F1	100	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	100	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	200	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	200	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	500	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	500	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	1000	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	1000	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F2	100	NBOLDE	7.5051E-222	3.9941E-219	2.4768E-220	**0.0000E+00**
	100	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	200	NBOLDE	1.2643E-212	2.6420E-211	6.7907E-212	**0.0000E+00**
	200	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	500	NBOLDE	6.4314E-207	8.3671E-205	1.4241E-205	**0.0000E+00**
	500	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	1000	NBOLDE	-	-	-	-
	1000	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F3	100	NBOLDE	1.7530E-304	2.8343E-299	1.6960E-300	**0.0000E+00**
	100	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	200	NBOLDE	2.7620E-279	1.3134E-273	9.1736E-275	**0.0000E+00**
	200	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	500	NBOLDE	6.0908E-262	1.5434E-255	6.1412E-257	**0.0000E+00**
	500	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	1000	NBOLDE	4.6201E-258	8.7852E-250	7.3256E-251	**0.0000E+00**
	1000	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F4	100	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	100	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	200	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	200	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	500	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	500	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	1000	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	1000	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F5	100	NBOLDE	9.8560E+01	9.8812E+01	9.8699E+01	**6.2063E-02**
	100	PASDE	**8.5791E+01**	**9.8041E+01**	**9.3194E+01**	2.7279E+00
	200	NBOLDE	1.9847E+02	1.9875E+02	1.9864E+02	**6.5760E-02**
	200	PASDE	**1.9040E+02**	**1.9776E+02**	**9.3194E+01**	2.7279E+00
	500	NBOLDE	4.9847E+02	4.9878E+02	4.9863E+02	**6.6567E-02**
	500	PASDE	**4.9259E+02**	**4.9666E+02**	**4.9544E+02**	1.1143E+00
	1000	NBOLDE	9.9852E+02	9.9877E+02	9.9862E+02	**5.8996E-02**
	1000	PASDE	**9.9169E+02**	**9.9546E+02**	**9.9463E+02**	8.7931E-01
F6	100	NBOLDE	1.5091E+02	1.8345E+02	1.6958E+02	8.6816E+00
	100	PASDE	**6.2188E+00**	**2.0690E+01**	**1.0499E+01**	**5.0933E+00**
	200	NBOLDE	3.4141E+02	4.0938E+02	3.7762E+02	**1.8603E+01**
	200	PASDE	**3.8498E+01**	**1.1485E+02**	**7.1384E+01**	2.0981E+01
	500	NBOLDE	9.2901E+02	1.0751E+03	1.0290E+03	**3.2119E+01**
	500	PASDE	**2.4040E+02**	**3.6266E+02**	**2.9226E+02**	3.2214E+01
	1000	NBOLDE	2.0481E+03	2.2551E+03	2.1651E+03	**5.1766E+01**
	1000	PASDE	**7.3618E+02**	**9.9003E+02**	**8.7673E+02**	6.1535E+01

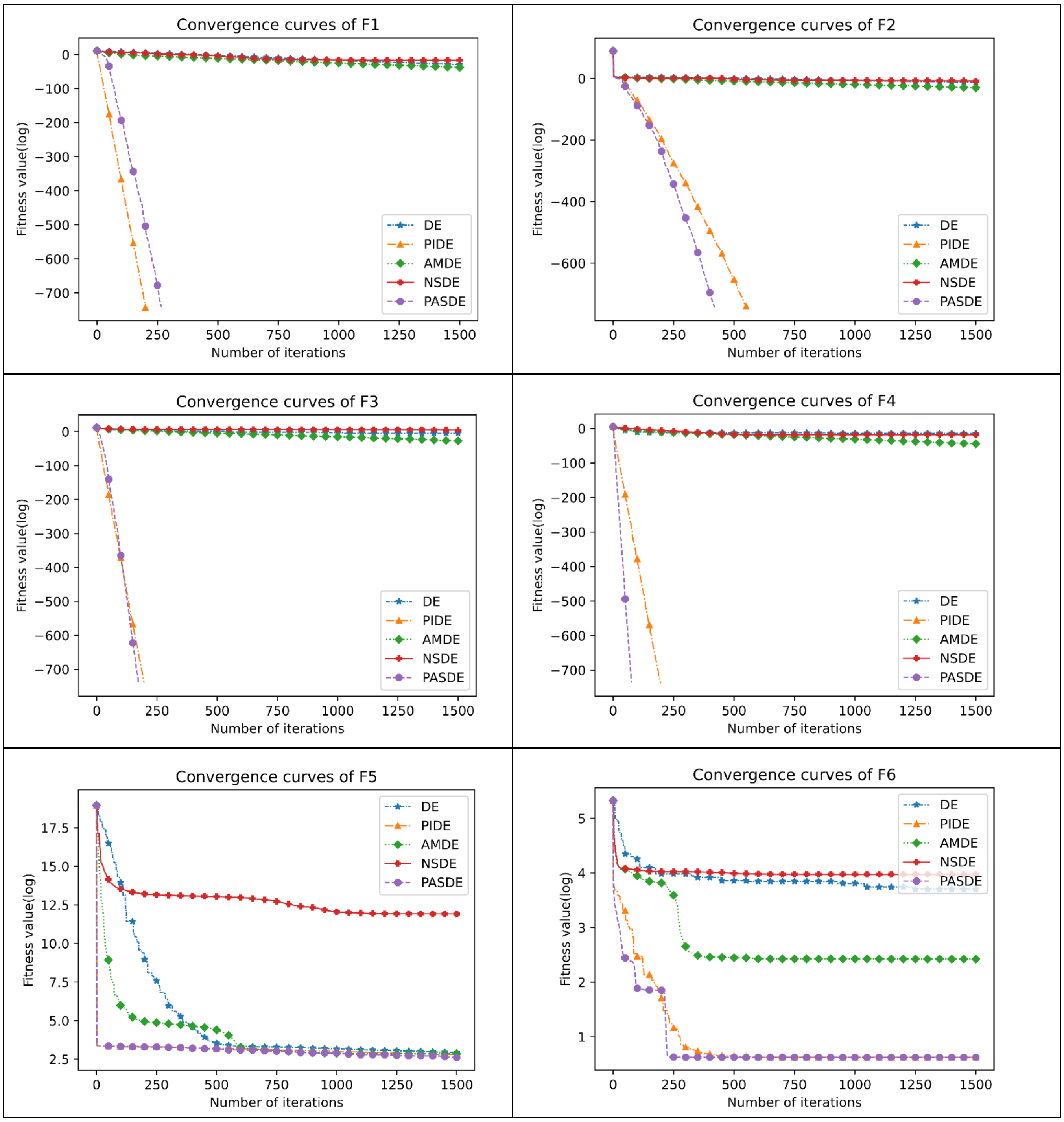
F7	100	NBOLDE	4.4409E-15	4.4409E-15	4.4409E-15	**0.0000E+00**
	100	PASDE	**4.4409E-16**	**4.4409E-16**	**4.4409E-16**	**0.0000E+00**
	200	NBOLDE	4.4409E-15	4.4409E-15	4.4409E-15	**0.0000E+00**
	200	PASDE	**4.4409E-16**	**4.4409E-16**	**4.4409E-16**	**0.0000E+00**
	500	NBOLDE	4.4409E-15	4.4409E-15	4.4409E-15	**0.0000E+00**
	500	PASDE	**4.4409E-16**	**4.4409E-16**	**4.4409E-16**	**0.0000E+00**
	1000	NBOLDE	4.4409E-15	4.4409E-15	4.4409E-15	**0.0000E+00**
	1000	PASDE	**4.4409E-16**	**4.4409E-16**	**4.4409E-16**	**0.0000E+00**
F8	100	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	100	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	200	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	200	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	500	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	500	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	1000	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	1000	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F9	100	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	100	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	200	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	200	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	500	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	500	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	1000	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	1000	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F10	100	NBOLDE	1.8759E-04	4.6191E-02	1.0658E-02	9.9934E-03
	100	PASDE	**2.3012E-05**	**3.0379E-02**	**2.5379E-03**	**6.3269E-04**
	200	NBOLDE	3.4430E-04	3.3818E-02	1.3005E-02	9.8052E-03
	200	PASDE	**1.9758E-05**	**1.5293E-02**	**2.2715E-03**	**3.1804E-03**
	500	NBOLDE	2.7261E-04	3.0671E-02	9.7690E-02	8.1717E-03
	500	PASDE	**1.9057E-05**	**2.1619E-02**	**4.3916E-03**	**5.6523E-03**
	1000	NBOLDE	4.7578E-04	3.0716E-02	8.9893E-03	**7.4480E-03**
	1000	PASDE	**7.9943E-06**	**3.9352E-02**	**3.4372E-03**	7.7102E-03
F11	100	NBOLDE	1.0904E-222	4.3578E-221	1.1465E-221	**0.0000E+00**
	100	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	200	NBOLDE	5.2442E-214	3.2061E-212	8.3541E-213	0.0000E+00
	200	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	500	NBOLDE	6.9040E-208	6.9135E-206	1.4510E-206	0.0000E+00
	500	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	1000	NBOLDE	1.5242E-205	1.0209E-203	2.6189E-204	0.0000E+00
	1000	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F12	100	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	100	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	200	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	200	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	500	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	500	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	1000	NBOLDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	1000	PASDE	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**

‘‘–’’ means that the value exceeds the maximum word length of Matlab 2018b and cannot be displayed.

4.1. Effects of the Proposed Strategies

The purpose of this first part is to validate the effects of the three strategies in improving the DE's performance. To determine this, we compare the original DE with the proposed PASDE algorithm and each of the three proposed DE strategies, i.e., DE with population improvement strategy (PIDE), DE with adaptive mutation strategy (AMDE) and DE with new selection strategy (NSDE).

The results of the experiments are shown in Table 3. The results of the PASDE show that the combination of three proposed strategies significantly improves DE performance. The PASDE is the best performer on all benchmark functions; it is able to find the global optimal solution on 8 benchmark functions, which are F1 – F4, F8, F9, F11 and F12. For F5 – F7 and F10, PASDE and the three proposed DE strategies cannot reach the global optimum. While examining each of the three proposed strategies individually, PIDE is the most promising strategy, followed by AMDE and NSDE. The results in Table 3 show that the first strategy, PIDE, has similar results to the PASDE on most functions. However, from Figure 2, we can see that the PASDE converges faster than PIDE and much faster than DE, AMDE and NSDE. In summary, PASDE is the best among all algorithms in terms of the ability to locate the global optimum and the convergence rate.



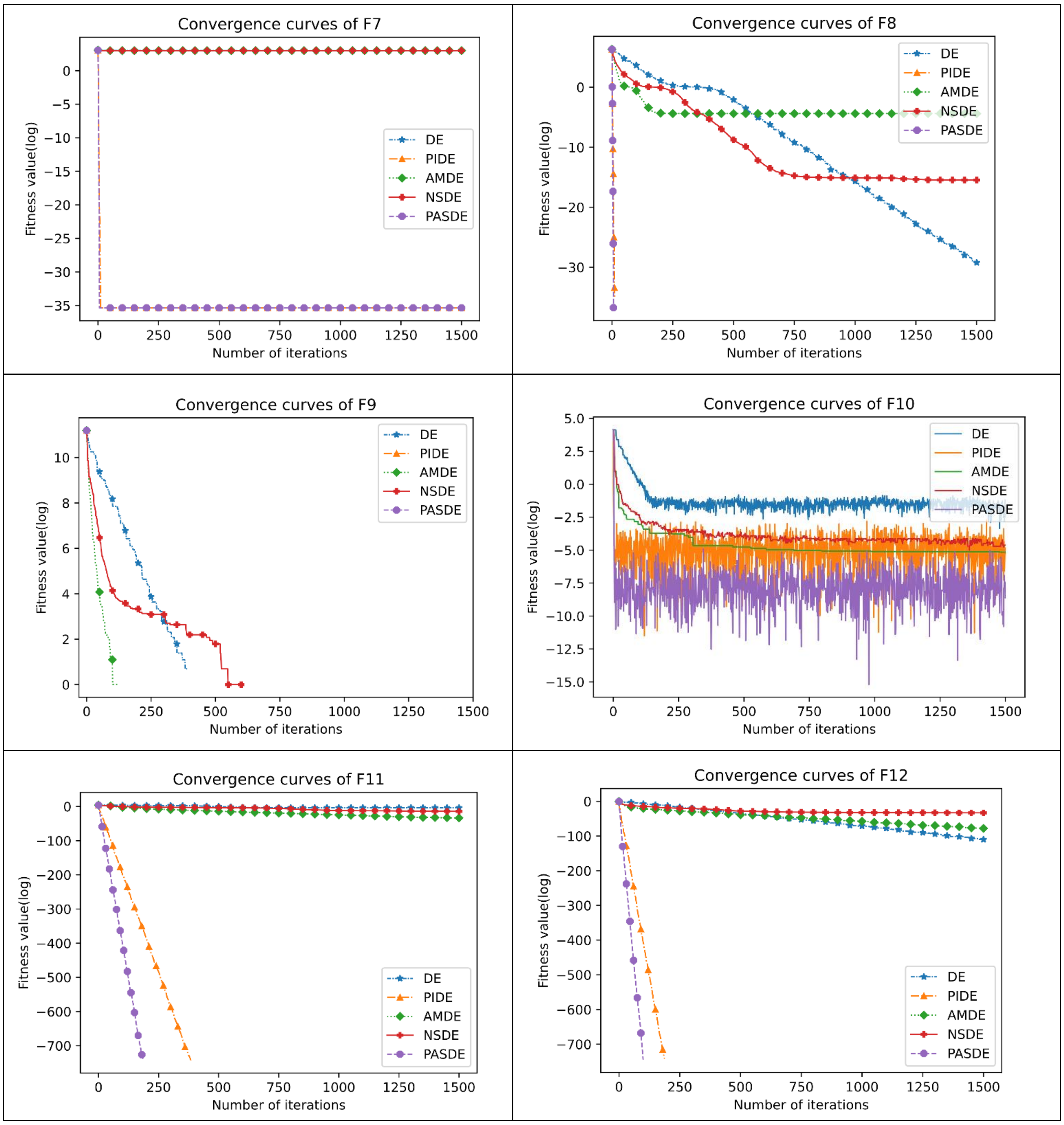


Figure 2. Convergence curves of DE, PIDE, AMDE, NSDE and PASDE on 12 benchmark functions

4.2. Result Comparison with Other Algorithms

In order to validate the optimization performance of PASDE, this second part compares the performance of PASDE with seven other algorithms. They are the original DE, four other popular DE variants, i.e., ODE [27], SaDE [28], JADE [29], NBOLDE [18] and two other meta-heuristic algorithms, i.e., PSO and GA. We choose these algorithms for comparison because they have some similar processes and strategies to our proposed algorithm.

The results which are reported in Table 4 can be summarized as follows. In general, the PASDE has the best performance, closely followed by NBOLDE. PASDE, NBOLDE and ODE obtain the global optimal solutions on 8, 6 and 2 benchmark functions respectively; DE, SaDE, JADE, PSO and GA obtain the global optimal solutions on only 1 function, F9. To be more precise, PASDE performs better than DE, ODE, SaDE, JADE, PSO and GA on 9 benchmark functions, F1-F4, F7-F8 and F10-F12, and performs equally well on F9. In comparison with NBOLDE, PASDE performs better on F2, F5, F6, F7, F10 and F11, and performs equally well on F1, F3, F4, F8, F9 and F12. JADE outperforms PASDE on F5, and GA slightly outperforms PASDE on F6.

Next, we examine how well each algorithm performs on the unimodal and multimodal functions. For all unimodal functions, both PASDE and NBOLDE can achieve the global optimal solutions; they perform equally well and better than the rest of the algorithms. For multimodal functions, PASDE has better performance than other algorithms on F7, F10 and F11; JADE has the best performance on F5 and GA has the best performance on F6; PASDE, NBOLDE and ODE are the best performers on F8.

4.3. Scalability Evolution

This last part reports the scalability of PASDE. The experiments are conducted to evaluate the performance of PASDE in solving the above 12 functions with different dimensions (d = 100, 200, 500 and 1000). Table 5 shows the scalability performance of PASDE in comparison with that of NBOLDE. The results in Table 5 show that PASDE has stable performance and can outperform NBOLDE across all dimensions. For F5, F6 and F10, the increase in dimension worsens the performance of PASDE slightly.

In comparison with NBOLDE, PASDE is better on F2, F3, F5 – F7, F10 and F11. On F1, F4, F8, F9 and F12, both PASDE and NBOLDE can obtain the global optimal solutions.

5. Conclusion

An improved differential evolution algorithm with population improvement, adaptive mutation, and new selection strategies (PASDE) is proposed in this paper. A multi-level sampling mechanism is used in the population improvement strategy to accelerate the convergence speed in the early stage of the search and to increase the exploration, which in turn reducing the chance of getting trap in local optima, in the later stage of the search. The adaptive mutation perturbs the solution to enable it to jump out of local optima when trapped. A new selection strategy introduces more diverse solutions to the population to help avoid local optima. Among the three proposed strategies, the population improvement strategy improves the performance of DE the most. The adaptive mutation strategy and the new selection strategy by itself are not very effective in improving the performance of DE. However, when they are combined with the population improvement strategy, the resulted PASDE performs extremely well both in terms of the ability to locate the global optimum and the convergence speed.

The performance of PASDE is compared with three classical metaheuristic algorithms, i.e., the original DE, PSO and GA, and four recent DE variants, i.e., ODE, SaDE, JADE and NBOLDE. The results on 12 benchmark functions show that the overall performance of PASDE is the best among all algorithms. Moreover, the scalability of the PASDE is also very good. On 9 out of 12 functions, PASDE can maintain its excellent performance as the dimension of the search space is increased.

6. Declarations

6.1. Author Contributions

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

6.2. Data Availability Statement

The data presented in this study are available in the article.

6.3. Funding and Acknowledgements

This work was supported by King Mongkut's Institute of Technology Ladkrabang (KMITL), Thailand.

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

7. References

[1] Tansui, D., & Thammano, A. (2020). Hybrid Nature-Inspired Optimization Algorithm: Hydrozoan and Sea Turtle Foraging Algorithms for Solving Continuous Optimization Problems. IEEE Access, 8, 65780–65800. doi:10.1109/ACCESS.2020.2984023.

[2] Ahmad, M. F., Isa, N. A. M., Lim, W. H., & Ang, K. M. (2022). Differential evolution: A recent review based on state-of-the-art works. Alexandria Engineering Journal, 61(5), 3831–3872. doi:10.1016/j.aej.2021.09.013.

[3] Bilal, Pant, M., Zaheer, H., Garcia-Hernandez, L., & Abraham, A. (2020). Differential Evolution: A review of more than two decades of research. Engineering Applications of Artificial Intelligence, 90, 103479. doi:10.1016/j.engappai.2020.103479.

[4] de Werra, D., & Hertz, A. (1989). Tabu search techniques - A tutorial and an application to neural networks. OR Spektrum, 11(3), 131–141. doi:10.1007/BF01720782.

[5] Holland, J. H. (1992). Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, United States. doi:10.7551/mitpress/1090.001.0001.

[6] Storn, R., & Price, K. (1997). Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, 11(4), 341–359. doi:10.1023/A:1008202821328.

[7] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks. doi:10.1109/icnn.1995.488968.

[8] Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE Computational Intelligence Magazine, 1(4), 28–39. doi:10.1109/mci.2006.329691.

[9] Yang, X.-S. (2021). Firefly Algorithms. Nature-Inspired Optimization Algorithms, 123–139. doi:10.1016/b978-0-12-821986-7.00016-0.

[10] Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization, 39(3), 459–471. doi:10.1007/s10898-007-9149-x.

[11] Sheng, M., Wang, Z., Liu, W., Wang, X., Chen, S., & Liu, X. (2022). A particle swarm optimizer with multi-level population sampling and dynamic p-learning mechanisms for large-scale optimization. Knowledge-Based Systems, 242, 108382. doi:10.1016/j.knosys.2022.108382.

[12] Ali, M. Z., Awad, N. H., Suganthan, P. N., Shatnawi, A. M., & Reynolds, R. G. (2018). An improved class of real-coded Genetic Algorithms for numerical optimization. Neurocomputing, 275, 155–166. doi:10.1016/j.neucom.2017.05.054.

[13] Chu, X., Cai, F., Gao, D., Li, L., Cui, J., Xu, S. X., & Qin, Q. (2020). An artificial bee colony algorithm with adaptive heterogeneous competition for global optimization problems. Applied Soft Computing Journal, 93, 106391. doi:10.1016/j.asoc.2020.106391.

[14] She, B., Fournier, A., Yao, M., Wang, Y., & Hu, G. (2022). A self-adaptive and gradient-based cuckoo search algorithm for global optimization [Formula presented]. Applied Soft Computing, 122, 108774. doi:10.1016/j.asoc.2022.108774.

[15] Wu, J., Wang, Y. G., Burrage, K., Tian, Y. C., Lawson, B., & Ding, Z. (2020). An improved firefly algorithm for global continuous optimization problems. Expert Systems with Applications, 149, 113340. doi:10.1016/j.eswa.2020.113340.

[16] Chen, Y., & Pi, D. (2020). An innovative flower pollination algorithm for continuous optimization problem. Applied Mathematical Modelling, 83, 237–265. doi:10.1016/j.apm.2020.02.023.

[17] Sun, G., Yang, B., Yang, Z., & Xu, G. (2020). An adaptive differential evolution with combined strategy for global numerical optimization. Soft Computing, 24(9), 6277–6296. doi:10.1007/s00500-019-03934-3.

[18] Deng, W., Shang, S., Cai, X., Zhao, H., Song, Y., & Xu, J. (2021). An improved differential evolution algorithm and its application in optimization problem. Soft Computing, 25(7), 5277–5298. doi:10.1007/s00500-020-05527-x.

[19] Zeng, Z., Zhang, M., Chen, T., & Hong, Z. (2021). A new selection operator for differential evolution algorithm. Knowledge-Based Systems, 226, 107150. doi:10.1016/j.knosys.2021.107150.

[20] Meng, Z., & Yang, C. (2022). Two-stage differential evolution with novel parameter control. Information Sciences, 596, 321–342. doi:10.1016/j.ins.2022.03.043.

[21] Kumar, A., Biswas, P. P., & Suganthan, P. N. (2022). Differential evolution with orthogonal array ‐ based initialization and a novel selection strategy. Swarm and Evolutionary Computation, 68, 101010. doi:10.1016/j.swevo.2021.101010.

[22] Houssein, E. H., Rezk, H., Fathy, A., Mahdy, M. A., & Nassef, A. M. (2022). A modified adaptive guided differential evolution algorithm applied to engineering applications. Engineering Applications of Artificial Intelligence, 113, 104920. doi:10.1016/j.engappai.2022.104920.

[23] Deng, W., Ni, H., Liu, Y., Chen, H., & Zhao, H. (2022). An adaptive differential evolution algorithm based on belief space and generalized opposition-based learning for resource allocation. Applied Soft Computing, 127, 109419. doi:10.1016/j.asoc.2022.109419.

[24] Yi, W., Chen, Y., Pei, Z., & Lu, J. (2022). Adaptive differential evolution with ensembling operators for continuous optimization problems. Swarm and Evolutionary Computation, 69, 100994. doi:10.1016/j.swevo.2021.100994.

[25] Thanathamathee, P., & Sawangarreerak, S. (2022). Discovering Future Earnings Patterns through FP-Growth and ECLAT Algorithms with Optimized Discretization. Emerging Science Journal, 6(6), 1328-1345. doi:10.28991/ESJ-2022-06-06-07.

[26] Farda, I., & Thammano, A. (2022). A Self-adaptive Differential Evolution Algorithm for Solving Optimization Problems. Lecture Notes in Networks and Systems, 453 LNNS, 68–76. doi:10.1007/978-3-030-99948-3_7.

[27] Rahnamayan, R. S., Tizhoosh, H. R., & Salama, M. M. A. (2008). Opposition-based differential evolution. IEEE Transactions on Evolutionary Computation, 12(1), 64–79. doi:10.1109/TEVC.2007.894200.

[28] Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation, 13(2), 398–417. doi:10.1109/TEVC.2008.927706.

[29] Zhang, J., & Sanderson, A. C. (2009). JADE: Adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation, 13(5), 945–958. doi:10.1109/TEVC.2009.2014613.