F O U N D A T I O N S O F C O M P U T I N G A N D D E C I S I O N S C I E N C E S
Vol. 48 (2023) No. 2
ISSN 0867-6356
sciendo DOI: 10.2478/fcds-2023-0010 e-ISSN 2300-3405

Adaptive differential evolution algorithm with a pheromone-based learning strategy for global continuous optimization

Pirapong Singsathid, Pikul Puphasuk, Jeerayut Wetweerapong *

Abstract. Differential evolution algorithm (DE) is a well-known population-based method for solving continuous optimization problems. It has a simple structure and is easy to adapt to a wide range of applications. However, with suitable population sizes, its performance depends on the two main control parameters: scaling factor (F) and crossover rate (CR). The classical DE method can achieve high performance by a time-consuming tunning process or a sophisticated adaptive control implementation. We propose in this paper an adaptive differential evolution algorithm with a pheromone-based learning strategy (ADE-PS) inspired by ant colony optimization (ACO). The ADE-PS embeds a pheromone-based mechanism that manages the probabilities associated with the partition values of F and CR. It also introduces a resetting strategy to reset the pheromone at a specific time to unlearn and relearn the progressing search. The preliminary experiments find a suitable number of subintervals (ns) for partitioning the control parameter ranges and the reset period (rs) for resetting the pheromone. Then the comparison experiments evaluate ADE-PS using the suitable ns and rs against some adaptive DE methods in the literature. The results show that ADE-PS is more reliable and outperforms several well-known methods in the literature.

Keywords: Differential evolution algorithm, ant colony optimization,pheromone strategy, optimization

1. Introduction

Challenging real-world optimization problems arise in many fields of applied sciences, economics, and engineering [27, 26]. Tackling these problems often requires well-

*Deparment of Mathematics, Faculty of Science, Khon Kaen University, Khon Kaen, 40002, Thailand, pirapongs@kkumail.com, ppikul@kku.ac.th, wjeera@kku.ac.th

designed efficiency optimization methods to overcome complicated objective functions. Their multimodality, high dimensionality, and non-linearity make the derivative methods inefficient in solving them. Therefore, evolutionary optimization methods such as genetic algorithm (GA), evolutionary algorithm (EA), ant colony optimization (ACO), particle swarm optimization (PSO), differential evolution (DE), and other nature-inspired optimization methods have been designed and applied to solve the problems.

In this research, we aim to improve the performance of the DE algorithm for solving continuous optimization problems. Like GA and other evolutionary methods, DE simulates the basic genetic operations: initialization, mutation, crossover, and selection. Its performance significantly depends on two main control parameters: scaling factor F and crossover rate CR for mutation and crossover [19]. During the early development of DE, researchers attempted to find a suitable parameter setting for DE by tuning. The process is time-consuming and usually gives a fixed parameter setting for a specific problem. The obtained parameter values may lead to inferior performance for other problems [20]. In recent years, modifications and enhancements of DE have been suggested to deal with high-complexity problems. Many parameter adaptation techniques to enable the solvability for various types of objective functions have been developed [7, 8].

The control parameter adaptations of evolutionary methods can be classified into three types: deterministic, adaptive, and self-adaptive [11]. The deterministic parameter control modifies the parameter values using the predefined rules without using feedback from the search. In contrast, the adaptive parameter control uses feedback from the search to adjust the parameter values. The self-adaptive parameter control, a higher-level scheme, directly encodes parameters into the population and evolves them with genetic operations. However, the adaptive algorithm will incur some new control parameters for handling the adaptation mechanism.

In this work, we propose a new adaptive DE with a pheromone-based learning strategy called ADE-PS. The pheromone strategy is derived from the ACO method introduced by Dorigo in 1992 [9]. The ADE-PS divides the parameter ranges of F and CR into partition points and applies the pheromone on these points in the selection process. The associated pheromone amounts are used to calculate probabilities for selecting F and CR values in mutation and crossover operations. ADE-PS also introduces the resetting strategy to reduce the effect of the dominant pheromone.

The remainder of this paper is organized as follows. Section 2 briefly describes the classic DE algorithm, reviews several related adaptive DE algorithms, and presents the pheromone strategy. Section 3 gives details of the proposed ADE-PS algorithm. Then, the preliminary experiment for finding suitable parameters of ADE-PS and the comparison experiments are conducted in Section 4. The results and discussion are presented in Section 5. Finally, Section 6 concludes this paper and offers suggestions and future work development.

2. Literature review

This section describes the classic DE algorithm and reviews some related adaptive DE algorithms and the pheromone strategy.

2.1. The classic DE algorithm

Differential evolution (DE) is a population-based stochastic optimization method introduced by Storn and Price during 1995 - 1997 for solving continuous optimization problems [19, 18, 15]. The algorithm consists of three genetic operators: mutation, crossover, and selection. The initial population vectors x_i, $i = 1, \ldots, NP$ are generated randomly from the feasible D-dimensional solution space, then the best one among them is indicated. At each generation, the mutation operation constructs a mutant vector v_i for each target vector x_i using three distinct random vectors $x_{r_1}, x_{r_2}, x_{r_3}$ different from x_i as follows:

$$v_i = x_{r_1} + F \cdot (x_{r_2} - x_{r_3}) \tag{1}$$

where F is the scaling factor in $(0, 1]$. Next, the crossover operation combines v_i and x_i to create the trial vector u_i as follows:

$$u_{i,j} = \begin{cases} v_{i,j}\,; & s_j \leq CR \text{ or } j = I_{rand} \\ x_{i,j}\,; & otherwise \end{cases} \tag{2}$$

where $j = 1, 2, ..., D$; s_j is a uniform random number in $(0, 1)$ for each j and CR is the crossover rate in $[0, 1]$ which controls the number of trial vector components to be replaced by the values from the mutant vector components. I_{rand} is a fixed random integer from 1 to D for each target vector x_i which guarantees that the trial vector will differ from its corresponding target vector by at least one component. Then, the selection operation compares the objective function values of x_i and u_i to select the better vector for the next generation. The best vector and its value are updated. The algorithm repeats these three operations until reaching a termination condition.

The classic DE algorithms with fixed F and CR values are not suitable for general problems. Therefore, many adaptive DE algorithms have been proposed.

2.2. Adaptive DE Algorithms

An adaptive DE algorithm adjusts the F and CR values by a parameter adaptation mechanism during the optimization process [1]. We review some well-known adaptive DE algorithms by providing their characteristics and performances.

In 2006, Tvrdík [24] presented the competitive differential evolution (cDE) which adaptively selects a combination of F and CR values from the predefined parameter pools: $F_{pool} = \{0.5, 0.8, 1\}$ and $CR_{pool} = \{0, 0.5, 1\}$. In the beginning, all possible combinations of F and CR values are chosen randomly with equal probability. The

associated probabilities are changed periodically according to the success counters in the selection process. If some probability is less than the threshold, the algorithm reset all probabilities to equal value. The cDE is more reliable and less time-consuming than the classic DE.

In 2007, Brest et al. [2, 3] introduced a self-adaptive DE called jDE. It adjusts the control parameters F_i and CR_i at the individual level. The initial values of F_i and CR_i are set to 0.5 and 0.9, respectively. For each generation, the values F_i and CR_i are updated with probability 0.1 by using random numbers from $[0.1, 0.9]$ and $[0, 1]$, respectively; otherwise, the current parent's values are used. The jDE outperforms the classic DE and some evolutionary algorithms when considering the quality of the solutions obtained.

In 2009, Qin et al. [16] proposed the SaDE, a differential evolution algorithm with a strategy adaptation. The algorithm adaptively uses four mutation strategies: DE/rand/1/bin, DE/rand-to-best/1/bin, DE/rand/2/bin, and DE/current-to-rand/1. It computes the probabilities associating with mutation strategies from the number of successful trial vectors in the selection process. The values F_i and CR_i are initialized to 0.5. Later, they are generated from normal distributions with fixed means and standard deviations. They are updated every 25 generations based on the median of the successful values. SaDE outperforms jDE and classic DE with some static values of F and CR.

In the same year, 2009, Zhang and Sanderson [30] presented an adaptive DE, called JADE, with two main features: external archive and new mutation strategy DE/current-to-pbest/1/bin. The algorithm keeps the target vectors which are worse than trial vectors in the archive A and uses them to increase population diversity. The mutation operation utilizes some top $p\%$ best individuals in the current population P and some individuals from $A \cup P$. The F_i and CR_i values are generated from Cauchy and normal distributions, respectively. Their means are updated based on the set that stores successful F_i and CR_i values in the selection process. The results show that JADE is better than the classic DE and some adaptive DE algorithms.

In 2011, Mallipeddia et al. [13] introduced a DE algorithm with an ensemble of parameters and mutation strategies (EPSDE). In the beginning, the algorithm randomly assigns a mutation strategy from the three mutation pools and parameters F_i, CR_i from the parameter pools. In the selection process, the replacing trial vector carries the mutation strategy and parameter values to the next generation; otherwise, the target vector is randomly assigned a new mutation strategy and new parameter values. The successful mutation strategy and parameter values are stored in the pools. EPSDE performs better than the classic DE and three adaptive DE algorithms: jDE, SaDE, and JADE. In the same year, Wang et al. [25] proposed DE with composite trial vector generation strategies and control parameters, called CoDE. The algorithm uses three mutation strategies (DE/rand/1/bin, DE/rand/2/bin, and DE/current-to-best/1) to generate three trial vectors for a target vector. Then, the best one replaces the target and survives for the next generation. The experimental results show that CoDE slightly outperforms jDE, JADE, SaDE, and EPSDE.

In 2013, Tanabe and Fukunaga presented a success-history-based adaptive DE algorithm (SHADE) [22] that employs the mutation strategy and external archive

similar to JADE. For each vector x_i, the algorithm uses the successful F_i and CR_i values to update its historical memories and uses their means to modify the F_i and CR_i values at every specified period. The algorithm is competitive to JADE, CoDE, and EPSDE. In addition, the SHADE parameter adaptation scheme was improved into LSHADE [23] in 2014, iL-SHADE algorithm [4] in 2016 and jSO algorithm [5] in 2017.

In 2016, Leon and Xiang introduced an adaptive DE called GADE [12]. The algorithm uses a greedy mechanism to dynamically update the F and CR values during the search. It initially sets the centers $F^* = 0.5, CR^* = 0.5$ and step size $d_1 = d_2 = 0.01$. In each generation, the values of F_i are randomly selected from the set $\{F^* - d_1, F^*, F^* + d_1\}$ and the value of CR_i is generated from the Cauchy distribution with mean μ_{CR} which randomly selected from the set $\{CR^* - d_2, CR^*, CR^* + d_2\}$. The best values are kept and used to update the new centers. GADE performs better than classic DE, and it is the second-best when compared with SaDE, JADE, and SHADE.

In 2018, Wu et al. proposed an ensemble of differential evolution variants (EDEV) [28] which consists of three DE algorithms: JADE, CoDE, and EPSDE. The algorithm divides the population into four subpopulations. Three subpopulations are for each DE algorithm and the other one as a reward subpopulation. After every predefined number of generations, the best-performing DE algorithm is determined. Then, the population partition operation is invoked to reward the best DE variant by merging its subpopulation with the reward one. The experiment shows that EDEV overall outperforms jDE, JADE, SaDE, CoDE, and EPSDE.

In 2019, Meng et al. [14] presented a parameter adaptive DE (PaDE), in which F_i is generated and updated as in JADE, and CR_i is managed the same way as in SHADE. Moreover, PaDE employs a parabolic population size reduction by using the ratio of the number of function evaluations and the maximum number of function evaluations. The algorithm uses a timestamp-based mutation strategy to select some individuals for inserting into the external archive. PaDE is competitive with JADE and SHADE. However, the setting of PaDE depends on the maximum number of function evaluations.

Recently in 2021, Cheng et al. [6] have proposed the fitness and diversity ranking-based mutation operator for DE, called FDDE. An individual rank is calculated from its fitness and diversity rank. Then, the ranking numbers assign the sorted positions for individuals in the mutation operation. The FDDE operator is applied to jDE with "DE/rand/1" and SHADE with "DE/rand-to-pbest/1". Experimental results show that FDDE improves the convergence performance of jDE and SHADE in both low-dimensional and high-dimensional problems.

In summary, there are two main approaches to generating parameter values in adaptive DE algorithms: using some statistical distributions and using the discrete values from the predefined sets. They both use feedback during the search to adjust the parameter values. Next, we introduce the pheromone strategy from ant colony optimization. This strategy will be modified and used as a parameter control strategy of the proposed ADE-PS algorithm.

2.3. Pheromone strategy

In 2008, Dorigo and Socha [10] introduced the first ant colony optimization for a continuous function, called ACO_R. It indirectly formulates the pheromone system by keeping the solution vector components and their objective function values in the archive table. The algorithm generates a new solution using the normal distributions with means from component values of the best solution and the standard deviations computed from all component values. The better solutions are added to the archive, then the same number of worst solutions are removed. The performance of ACO_R is not significantly worse than other probability learning methods.

In 2011, Xiao and Li [29] presented a hybrid ant colony optimization (HACO). It divides the population into two halves. The first half uses the ACO_R method to generate a new solution. The second half applies DE to create and update the means and the standard deviations of normal distributions. The solution archive is updated similarly to the ACO_R. The experimental results show that HACO performs better than ACO_R algorithms.

In 2018, Singsathid and Wetweerapong [17] proposed a continuous ACO with domain partitioning technique, called PACO. It uses partition points on each component bound to generate solution components and constructs a matrix to represent the pheromone associated with those points. The pheromone is updated according to the obtained best solutions during the search periods. PACO shrinks or extends the domain range based on the components of the current best solution and then repartitions the domain. The experiments show that PACO outperforms some well-known continuous ACO algorithms.

Continuous ACO methods either apply the pheromone to the solution's components directly or use their archive indirectly as the pheromone mechanism. The methods update the pheromone by giving rewards to better solutions or add better solutions to an archive. Our proposed ADE-PS applies the pheromone to the F and CR partitioned values and uses the search feedback to reward the better control parameters.

3. The proposed ADE-PS algorithm

We present an adaptive differential evolution algorithm with a pheromone-based learning strategy called ADE-PS. It partitions F, and CR ranges into finite partition points and applies pheromone on these points to formulate the probability for choosing the associated F_i and CR_i values in mutation and crossover operations. In the selection operation, the algorithm adds pheromone to partition points that generate a better trial vector. Then, the associated probabilities are dynamically adapted. During the search process, the pheromone tends to accumulate on some partition points. Few points may obtain a large amount of pheromone and dominate the others. Consequently, the algorithm needs to unlearn the old pheromone at some specific generations by resetting pheromone values to the initial values. The ADE-PS algorithm for minimization is described as follows.

Step 1: Set the following parameters:
f: objective function to be minimized
D: problem dimension
LB, UB: lower and upper bounds of the problem
NP: population size
LF, UF: lower and upper bounds of scaling factor F
LC, UC: lower and upper bounds of crossover rate CR
VTR: value to reach
$maxnf$: maximum number of function evaluations
ns: number of subintervals for partitioning parameter ranges F and CR into partition points
rs: generation period to reset pheromone
Step 2: Randomly initialize the population $P = [x_{ij}]$ with real numbers between LB and UB where $i = 1, 2, \ldots, NP$ and $j = 1, 2, \ldots, D$. Compute the fitness for each x_i. Then, find the best solution x_{best} and its function value f_{best}.
Step 3: Divide the ranges of F and CR into $ns + 1$ partition points.

$$\begin{aligned} F_i &= LF + (i-1)(\frac{UF - LF}{ns}) \\ CR_i &= LC + (i-1)(\frac{UC - LC}{ns}) \end{aligned} \tag{3}$$

for $i = 1, 2, \ldots, ns + 1$ where F_i and CR_i are the ith partition point of scaling factor and crossover rate, respectively.
Step 4: Create two vectors $PheromoneF$ and $PheromoneCR$ of size $ns + 1$ where $PheromoneF(i)$ and $PheromoneCR(i)$ represent the amount of pheromone associated with the point F_i and CR_i, respectively. Set $PheromoneF(i) = 1$ and $PheromoneCR(i) = 1$ for all i.
Step 5: Compute probability vectors $ProbF$ and $ProbCR$ by normalizing $PheromoneF$ and $PheromoneCR$. For each i,

$$\begin{aligned} ProbF(i) &= \frac{PheromoneF(i)}{\sum_{i-1}^{ns+1} PheromoneF(i)} \\ ProbCR(i) &= \frac{PheromoneCR(i)}{\sum_{i-1}^{ns+1} PheromoneCR(i)}. \end{aligned} \tag{4}$$

Step 6: (Mutation) For each target vector x_i, random three distinct vectors x_{r1}, x_{r2}, x_{r3} different from x_i to generate a mutant vector v_i as follows.

$$v_i = x_{r1} + F_i \cdot (x_{r2} - x_{r3}) \tag{5}$$

where the scaling factor F_i is probabilistically chosen according to $ProbF$.
Step 7: (Crossover) Construct a trial vector u_i by

$$u_{i,j} = \begin{cases} v_{i,j}\,; & s_j \le CR_i \ \ or \ \ j = I_i \\ x_{i,j}\,; & otherwise \end{cases} \tag{6}$$

where s_j is a random number in $(0, 1)$, CR_i is the crossover rate chosen according to $ProbCR$ and I_i is a randomly fixed index from 1 to D.
Step 8: (Selection) Compare the function values of u_i and x_i. If $f(u_i) < f(x_i)$ then replace x_i with u_i and update pheromone of F_i and CR_i by adding 1 to $PheromoneF(i)$ and $PheromoneCR(i)$. If the value is also better than f_{best} then update f_{best} and x_{best}.
Step 9: Reset the pheromone vectors every rs generations by setting all components of $PheromoneF$ and $PheromoneCR$ to 1.
Step 10: Repeat steps 5-9 until f_{best} is less than VTR or $maxnf$ is reached. Then report x_{best} and f_{best}.

The flowchart of ADE-PS is illustrated in Figure 1. Note that the ADE-PS does not require the evaporation process of the pheromone. Some F and CR values will dominate the others in a short period of generations. Thus, the algorithm needs a pheromone resetting process to stop the dominant status of the pheromone. After the resetting, all F and CR values will enter a new competition cycle, which allows the algorithm to search and use the new suitable F and CR values that could be different from the previous cycles.

4. Experimental design

ADE-PS induces two important parameters ns and rs in addition to F and CR. We design two preliminary experiments to find the suitable ns and rs. Then we conduct three comparison experiments to evaluate the performance of ADE-PS against those of classic DE and some adaptive DE algorithms. Table 1 lists the test functions with their formulations, global optima, and search ranges for the first four experiments. Table 2 lists the shifted and rotated test functions for the last experiment. The following settings are used for ADE-PS in the preliminary and the first two comparison experiments: $NP = 30$, $LF = 0.5, UF = 1$, $LC = 0.1$ and $UC = 1$. The dimensions $D = 5, 10, 30, 50$ are varied. The $maxnf = 20000D$ and $VTR = 10^{-10}$ are set. The algorithm with each configuration is tested for 100 independent runs and each run is considered successful when x_{best} gives the function value less than VTR. We record the number of successful runs (NS), the mean of function evaluations (mean NF), and the percentage of standard deviation(%SD) of all successful runs. The %SD is computed by

$$\%\text{SD} = \frac{\text{SD}}{\text{mean NF}} \times 100$$

where SD is standard deviation of function evaluations. When reporting the experimental results, the best values are indicated in bold.

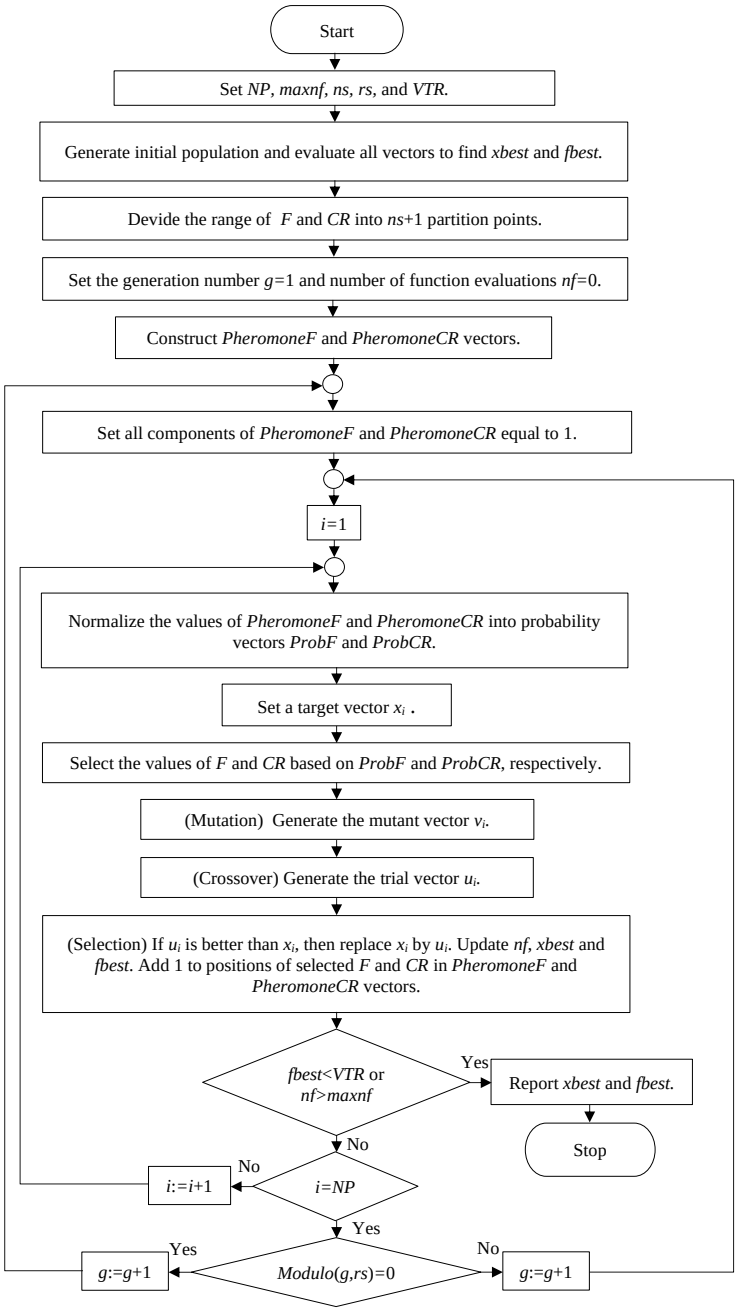

Figure 1. Flowchart of the proposed ADE-PS algorithm.

4.1. Finding the suitable number of subintervals for partitioning parameter ranges

The ADE-PS collects pheromone to partition points of the parameter control domains. Experiment 1 finds a suitable number of subintervals (ns) for partitioning the parameter F and CR ranges into partition points. The ns values are varied as $ns = 10, 20, 30$. The reset period $rs = 20D$ is fixed.

4.2. Finding the suitable resetting period

The resetting strategy resets the pheromone to the initial value for every period of rs generations. By using the suitable ns from Experiment 1, we conduct Experiment 2 to find a suitable reset period (rs) by considering two types of rs: the fixed $rs = 100, 500$ and the deterministic $rs = 10D, 20D, 30D$. We also compare ADE-PS using the suitable reset period rs and ADE-PS without resetting.

Table 1. Test functions.

Function	Formula	Global optimum	Search range
Sphere	$f(x) = \sum_{i=1}^{D} (x_i)^2$	$\bar{0}$	$[-100, 100]^D$
Schwefel 1.2	$f(x) = \sum_{i=1}^{D} \sum_{j=1}^{i} (x_j)^2$	$\bar{0}$	$[-100, 100]^D$
Rosenbrock	$f(x) = \sum_{i=1}^{D} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$\bar{1}$	$[-100, 100]^D$
Schwefel 2.22	$f(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$\bar{0}$	$[-100, 100]^D$
Rastrigin	$f(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$\bar{0}$	$[-5.12, 5.12]^D$
Schwefel	$f(x) = 418.98288727243369 \cdot D - \sum_{i=1}^{D} (x_i \sin(\sqrt{\lvert x_i \rvert}))$	$\overline{420.96}$	$[-500, 500]^D$
Ackley	$f(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)) + 20 + e$	$\bar{0}$	$[-32.32]^D$
Griewank	$f(x) = \frac{1}{4000}\sum_{i=1}^{D} (x_i)^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$	$\bar{0}$	$[-600, 600]^D$

4.3. Comparing the performance of ADE-PS with classic DE

Experiment 3 is carried out to compare the performance of the proposed ADE-PS algorithm with those of the classic DE algorithms. The ADE-PS uses the suitable parameters ns and rs from Experiments 1 and 2 while the classic DE uses the param-

eters $F = 0.5, CR = 0.1$ and $F = 0.5, CR = 0.9$. The population sizes $NP = 50, 100$ are considered. The DE algorithms with $(F, CR) = (0.5, 0.1)$ and $(F, CR) = (0.5, 0.9)$ are recommended for solving multimodal functions and nonseparable functions, respectively [20].

4.4. Comparing the performance of ADE-PS with ADE-PS, SaDE, CoDE, jDE and JADE algorithms

Experiment 4 compares the proposed ADE-PS using the suitable parameters with 4 well-known adaptive DE algorithms: ADE-PS, SaDE, CoDE, jDE and JADE algorithms. The compared methods use the same parameter settings in their original papers[2, 16, 30, 25]. The MATLAB source codes of JADE, CoDE, jDE, and SaDE have been downloaded from Q. Zhang's webpage at https://dces.essex.ac.uk/staff/qzhang/. The statistical t-test at the significance level of 0.05 is also used to compare their performances. The symbols "+", "=", and "–" denote that a compared method performs "better than", "similarly to", and "worse than" the ADE-PS, respectively.

4.5. Comparing the performance of ADE-PS with EDEV and FDDE algorithms on shifted and rotated test functions

Experiment 5 compares the proposed ADE-PS using the suitable parameters with recently proposed EDEV [28] and FDDE [6] algorithms on shifted and rotated test functions from CEC2005 [21]. Their names and search ranges are listed in Table 2. The settings $D = 30$, $maxnf = 300000$, and 25 independent runs are used according to the original papers. We record the mean of function values (Mean) and the standard deviation (SD). The statistical test and notations are the same as in Experiment 4.

Table 2. Shifted and rotated test functions.

Function	Search range
F1: Shifted Sphere Function	[-100,100]
F2: Shifted Schwefel's Problem 1.2	[-100,100]
F3: Shifted Rotated High Conditioned Elliptic Function	[-100,100]
F4: Shifted Schwefel's Problem 1.2 with Noise in Fitness	[-100,100]
F5: Schwefel's Problem 2.6 with Global Optimum on Bounds	[-100,100]
F6: Shifted Rosenbrock's Function	[-100,100]
F7: Shifted Rotated Griewank's Function without Bounds	–
F8: Shifted Rotated Ackley's Function with Global Optimum on Bounds	[-32,32]
F9: Shifted Rastrigin's Function	[-5,5]
F10: Shifted Rotated Rastrigin's Function	[-5,5]
F11: Shifted Rotated Weierstrass Function	[-0.5,0.5]
F12: Schwefel's Problem 2.13	[-π,π]

5. Experimental results

This section presents the results of Experiments 1 - 5 as shown in Table 3, Tables 4-5, Table 6, Table 7, and Table 8 respectively.

5.1. Suitable number of subintervals for partitioning parameter ranges

Table 3 shows the performances of ADE-PS with different numbers of subintervals ns where $ns = 10, 20, 30$. The numbers of cases that give the success rate of 100% (NS=100) and the lowest mean NF are summarized in the two bottom rows. Out of all 32 cases, the ADE-PS algorithms with $ns = 10, 20, 30$ give 22, **32**, and 27 cases of 100% success rate, respectively. Their corresponding numbers of cases that give the lowest mean NF are 10, **19**, and 3, respectively. Therefore, $ns = 20$ is the suitable value. This value will be used for ADE-PS algorithms in the remaining experiments.

5.2. Suitable pheromone resetting period

Table 4 shows the performances of ADE-PS with different pheromone reset periods $rs = 10D, 20D, 30D$ and $rs = 100, 500$ generations. It summarizes the cases that give the success rate of 100% and the lowest mean NF in the two bottom rows. ADE-PS with $rs = 20D$ achieves the 100% success rate for all 32 cases while the other reset periods give 28 cases, except $rs = 10D$ which gives 13 cases. The setting $rs = 20D$ also provides the most cases (15 cases) of the lowest mean NF while the other reset periods do not give more than 5 cases. Therefore, $rs = 20D$ is the suitable reset period. This value will be used for ADE-PS algorithm in the comparison experiments. In addition, Table 5 compares the performances of ADE-PS with the suitable $rs = 20D$ and without resetting. The ADE-PS without pheromone resetting gives only 22 cases of the 100% success rate and 6 cases of the lowest mean NF. This result shows that the resetting strategy significantly enhances the performance of ADE-PS.

5.3. Performance comparison of ADE-PS and classic DE

The performance of ADE-PS with the suitable parameters $ns = 20$ and $rs = 20D$, and those of the classic DE algorithms with $F = 0.5, CR = 0.1$ and $F = 0.5, CR = 0.9$ are shown in Table 6. ADE-PS achieves the 100% success rate for all 32 cases and obtains the lowest mean NF for 27 cases where the classic DE algorithms give 17-24 cases of 100% success rate and do not give more than 3 cases of the lowest mean NF. This result concludes that the ADE-PS outperforms the classic DE.

Table 3. The performances of ADE-PS with different numbers of subintervals (ns) and the reset period $rs = 20D$.

Function	D	Statistics	$ns = 10$	$ns = 20$	$ns = 30$
Sphere	5	NS	**100**	100	100
		mean NF(%SD)	**5213(6.00)**	5500(4.26)	5557(5.09)
	10	NS	100	**100**	100
		mean NF(%SD)	11855(5.28)	**11754(5.01)**	12040(4.00)
	30	NS	100	**100**	100
		mean NF(%SD)	36632(5.02)	**36083(3.86)**	36696(4.35)
	50	NS	100	**100**	100
		mean NF(%SD)	59561(5.07)	**59179(4.23)**	60858(3.99)
Schwefel 1.2	5	NS	100	**100**	100
		mean NF(%SD)	9636(9.38)	**8962(7.91)**	9329(7.72)
	10	NS	100	**100**	100
		mean NF(%SD)	31198(12.37)	**21713(10.42)**	23496(8.05)
	30	NS	100	100	**100**
		mean NF(%SD)	315342(10.43)	249626(8.17)	**2208082(7.20)**
	50	NS	1	100	**100**
		mean NF(%SD)	943230(-)	836373(7.13)	**747684(6.47)**
Rosenbrock	5	NS	99	**100**	99
		mean NF(%SD)	15172(11.33)	**13416(9.90)**	14220(11.85)
	10	NS	100	**100**	99
		mean NF(%SD)	51913(12.76)	**41170(11.32)**	40742(8.29)
	30	NS	99	100	**100**
		mean NF(%SD)	410992(16.07)	377428(13.06)	**341088(12.50)**
	50	NS	5	**100**	60
		mean NF(%SD)	886476(6.76)	**937679(9.35)**	886479(10.83)
Schwefel 2.22	5	NS	100	**100**	100
		mean NF(%SD)	9596(4.38)	**9591(3.42)**	9646(3.52)
	10	NS	100	**100**	100
		mean NF(%SD)	22299(3.97)	**20212(3.38)**	20645(2.65)
	30	NS	**100**	100	100
		mean NF(%SD)	**59077(3.10)**	61543(2.96)	63096(2.27)
	50	NS	**100**	100	100
		mean NF(%SD)	**100301(3.52)**	102471(3.30)	105115(2.64)
Rastrigin	5	NS	**100**	100	100
		mean NF(%SD)	**7602(6.75)**	8011(5.69)	8059(5.43)
	10	NS	100	**100**	100
		mean NF(%SD)	18342(5.17)	**18211(3.83)**	18900(3.66)
	30	NS	**100**	100	100
		mean NF(%SD)	**68258(5.44)**	68614(5.7)	74146(4.46)
	50	NS	100	**100**	100
		mean NF(%SD)	169698(4.70)	**151624(4.40)**	169366(5.64)
Schwefel	5	NS	100	**100**	100
		mean NF(%SD)	6772(5.51)	**6710(5.26)**	6786(4.70)
	10	NS	99	**100**	100
		mean NF(%SD)	14070(5.00)	**14823(4.05)**	14951(4.04)
	30	NS	99	**100**	100
		mean NF(%SD)	45908(5.46)	**49017(4.78)**	51027(5.01)
	50	NS	99	**100**	99
		mean NF(%SD)	83879(6.18)	**89282(6.27)**	92986(5.12)
Ackley	5	NS	100	**100**	100
		mean NF(%SD)	9515(4.29)	**9454(3.87)**	9526(3.44)
	10	NS	**100**	100	100
		mean NF(%SD)	**18539(3.83)**	19451(3.70)	19867(3.07)
	30	NS	**100**	100	100
		mean NF(%SD)	**53847(4.04)**	56169(3.65)	57457(3.14)
	50	NS	**100**	100	100
		mean NF(%SD)	**87582(4.26)**	90052(3.36)	93563(3.23)
Griewank	5	NS	99	**100**	99
		mean NF(%SD)	17569(10.62)	**16592(7.54)**	17677(6.88)
	10	NS	99	**100**	100
		mean NF(%SD)	27505(11.13)	**28811(9.50)**	30066(9.46)
	30	NS	**100**	100	100
		mean NF(%SD)	**36782(7.33)**	38501(7.55)	39845(6.27)
	50	NS	**100**	100	100
		mean NF(%SD)	**58847(5.11)**	61155(4.87)	62718(4.31)
NS=100			22 cases	**32 cases**	27 cases
Lowest mean NF			10 cases	**19 cases**	3 cases

Table 4. The performances of ADE-PS with different resetting periods (rs) and numbers of subintervals $ns = 20$.

Function	D	resetting periods				
		$10D$	$20D$	$30D$	100	500
sphere	5	100	**100**	100	100	100
		5578(4.46)	**5500(4.26)**	5530(6.31)	5599(5.51)	5578(6.02)
	10	100	100	100	100	**100**
		11947(3.66)	11754(5.01)	11746(5.27)	11903(4.42)	**11501(6.54)**
	30	100	**100**	100	100	100
		36476(3.95)	**36083(3.86)**	36290(4.94)	39032(2.56)	36097(3.95)
	50	100	**100**	100	100	100
		59808(3.49)	**59179(4.23)**	60118(4.40)	68913(2.13)	60000(3.39)
schwefel 1.2	5	100	100	**100**	100	100
		9407(6.83)	8962(7.91)	**8758(10.30)**	8887(8.05)	8788(13.08)
	10	100	**100**	100	100	100
		23020(7.70)	**21713(10.42)**	22853(11.51)	23132(9.02)	22627(15.85)
	30	100	100	100	**100**	100
		213956(7.51)	249626(8.17)	269834(10.53)	**186139(6.12)**	241356(6.85)
	50	100	100	100	**100**	100
		7711093(6.16)	836373(7.13)	930419(7.37)	**547992(5.25)**	708017(6.35)
Rosenbrock	5	100	**100**	98	98	99
		14094(10.00)	**13416(9.90)**	13196(12.95)	13537(11.47)	17294(8.24)
	10	100	**100**	100	99	99
		41961(10.24)	**41170(11.32)**	42133(10.65)	40450(10.46)	44364(16.62)
	30	99	100	100	**100**	100
		345481(12.25)	377428(13.06)	410727(14.39)	**299286(8.15)**	375772(12.69)
	50	93	100	57	**100**	40
		1003323(12.53)	1071341(12.95)	1044035(13.50)	**790427(6.75)**	900669(8.48)
Schwefel 2.22	5	100	100	**100**	100	100
		9680(3.29)	9591(3.42)	**9532(4.10)**	9547(3.80)	9551(5.39)
	10	100	**100**	100	100	100
		20619(2.83)	**20112(3.38)**	20147(3.5)	20575(2.78)	20241(4.37)
	30	100	**100**	100	100	100
		62676(2.58)	**61543(2.96)**	61905(3.84)	68080(1.95)	61576(3.21)
	50	100	100	100	100	100
		104341(2.53)	102471(3.30)	102469(3.55)	121574(1.53)	104604(2.81)
rastrigin	5	**100**	100	100	99	100
		7698(5.49)	8011(5.69)	7773(6.01)	7958(6.21)	8151(6.90)
	10	**100**	100	100	100	100
		17718(5.18)	18211(3.83)	18814(8.72)	17715(5.19)	18810(5.66)
	30	100	**100**	99	100	100
		69567(4.82)	**68614(5.70)**	73728(2.42)	99201(6.12)	68971(3.68)
	50	100	**100**	100	100	100
		170134(5.37)	**151624(4.40)**	156565(5.10)	558229(6.22)	171486(5.34)
Schwefel	5	**100**	100	99	100	100
		6571(5.12)	6710(5.26)	6813(4.42)	6765(5.54)	6863(6.27)
	10	99	**100**	100	100	100
		15009(3.75)	**14823(4.05)**	14930(4.68)	14842(3.67)	15545(5.31)
	30	100	**100**	100	100	100
		48846(3.44)	**49017(4.78)**	49177(4.27)	53464(3.42)	49722(4.74)
	50	**100**	100	100	99	99
		88580(3.93)	89282(6.27)	91126(6.60)	113186(4.56)	88457(3.55)
Ackley	5	100	100	100	100	**100**
		9538(3.24)	9454(3.87)	9362(4.28)	9457(3.55)	**9303(5.60)**
	10	100	100	**100**	100	100
		19794(3.25)	19451(3.70)	**19270(4.85)**	19809(3.04)	19580(5.15)
	30	100	100	100	100	100
		57753(2.99)	56169(3.65)	55811(4.46)	62430(2.00)	56428(3.19)
	50	100	**100**	100	100	100
		93033(2.38)	**90052(3.36)**	90375(4.16)	108559(1.63)	92968(2.29)
Griewank	5	100	100	100	**100**	100
		17709(8.08)	16592(7.54)	16505(6.68)	**16416(6.73)**	19280(11.68)
	10	99	**100**	100	100	100
		30457(10.11)	**28811(9.50)**	29647(9.56)	30177(9.66)	30867(9.74)
	30	100	**100**	100	100	100
		39494(5.35)	**38501(7.55)**	38742(6.01)	42405(5.32)	39284(6.34)
	50	100	100	**100**	100	100
		61371(3.96)	61155(4.87)	**61112(4.85)**	70738(2.72)	61573(3.46)
NS=100		13 cases	**32 cases**	28 cases	28 cases	28 cases
Lowest mean NF		4 cases	**15 cases**	4 cases	5 cases	2 cases

Table 5. The performances of ADE-PS with $rs = 20D$ and without pheromone resetting.

Function	D	Statistics	ADE-PS with resetting	ADE-PS without resetting
Sphere	5	NS	100	**100**
		mean NF(%SD)	5500(4.26)	**5478(6.02)**
	10	NS	100	**100**
		mean NF(%SD)	11754(5.01)	**11501(6.54)**
	30	NS	**100**	100
		mean NF(%SD)	**36083(3.86)**	36121(6.29)
	50	NS	**100**	100
		mean NF(%SD)	**59179(4.23)**	59986(6.05)
Schwefel 1.2	5	NS	100	**100**
		mean NF(%SD)	8962(7.91)	**8830(12.67)**
	10	NS	**100**	100
		mean NF(%SD)	**21713(10.42)**	22308(17.29)
	30	NS	**100**	39
		mean NF(%SD)	**249626(8.17)**	421414(25.09)
	50	NS	**100**	0
		mean NF(%SD)	**836373(7.13)**	–
Rosenbrock	5	NS	**100**	97
		mean NF(%SD)	**13416(9.90)**	19408(16.94)
	10	NS	**100**	100
		mean NF(%SD)	**41170(11.32)**	67478(24.53)
	30	NS	**100**	47
		mean NF(%SD)	**377428(13.06)**	407843(15.84)
	50	NS	**100**	11
		mean NF(%SD)	**937679(9.35)**	825182(8.85)
Schwefel 2.22	5	NS	**100**	100
		mean NF(%SD)	**9591(3.42)**	9611(4.25)
	10	NS	**100**	100
		mean NF(%SD)	**20212(3.38)**	20219(4.90)
	30	NS	**100**	100
		mean NF(%SD)	**61543(2.96)**	63100(5.56)
	50	NS	**100**	100
		mean NF(%SD)	**102471(3.30)**	105467(6.48)
Rastrigin	5	NS	**100**	100
		mean NF(%SD)	**8011(5.69)**	8098(6.64)
	10	NS	**100**	100
		mean NF(%SD)	**18211(3.83)**	19314(7.82)
	30	NS	**100**	100
		mean NF(%SD)	**68614(5.70)**	87959(14.20)
	50	NS	**100**	100
		mean NF(%SD)	**151624(4.40)**	220122(14.66)
Schwefel	5	NS	**100**	100
		mean NF(%SD)	**6710(5.26)**	6924(4.72)
	10	NS	**100**	99
		mean NF(%SD)	**14823(4.05)**	15363(5.66)
	30	NS	**100**	99
		mean NF(%SD)	**49017(4.78)**	53805(6.33)
	50	NS	**100**	99
		mean NF(%SD)	**89282(6.27)**	103050(9.46)
Ackley	5	NS	100	**100**
		mean NF(%SD)	9454(3.87)	**9320(5.95)**
	10	NS	100	**100**
		mean NF(%SD)	19451(3.70)	**19082(5.53)**
	30	NS	100	**100**
		mean NF(%SD)	56169(3.65)	**56013(5.30)**
	50	NS	**100**	100
		mean NF(%SD)	**90052(3.36)**	90869(5.31)
Griewank	5	NS	**100**	100
		mean NF(%SD)	**16592(7.54)**	19435(11.77)
	10	NS	**100**	98
		mean NF(%SD)	**28811(9.50)**	31265(14.84)
	30	NS	**100**	100
		mean NF(%SD)	**38501(7.55)**	38504(7.31)
	50	NS	**100**	100
		mean NF(%SD)	**61155(4.87)**	61581(6.20)
NS=100			**32 cases**	22 cases
Lowest mean NF			**26 cases**	6 cases

Table 6. Performance comparison of ADE-PS and classic DE algorithms.

Function	D	Classic DE				ADE-PS
		$F = 0.5, CR = 0.1$		$F = 0.5, CR = 0.9$		
		$NP = 50$	$NP = 100$	$NP = 50$	$NP = 100$	
Sphere	5	100 7742(2.55)	100 15266(2.21)	100 6503(4.04)	100 13230(2.83)	**100** **5500(4.26)**
	10	100 16345(1.82)	100 32279(1.55)	100 13731(2.84)	100 30280(2.56)	**100** **11754(5.01)**
	30	100 47616(0.95)	100 95204(0.85)	100 40076(3.19)	100 108786(2.19)	**100** **36083(3.86)**
	50	100 77171(0.86)	100 154993(0.67)	100 70756(4.25)	100 171678(1.93)	**100** **59179(4.23)**
Schwefel 1.2	5	100 61181(5.27)	0 –	**100** **7924(3.65)**	100 16156(3.18)	100 8962(7.91)
	10	0 –	0 –	100 21912(4.05)	100 50581(2.85)	**100** **21713(10.42)**
	30	0 –	0 –	**100** **223136(6.42)**	100 460644(2.92)	100 249626(8.17)
	50	0 –	0 –	**100** **676840(5.16)**	0 –	100 836373(7.13)
Rosenbrock	5	0 –	0 –	97 21854(70.15)	100 26484(8.65)	**100** **13416(9.90)**
	10	0 –	0 –	92 110829(31.22)	100 76392(4.43)	**100** **41170(11.32)**
	30	0 –	0 –	0 –	100 410196(4.06)	**100** **377428(13.06)**
	50	0 –	0 –	0 –	22 945718(5.05)	**100** **937679(9.35)**
Schwefel 2.22	5	100 23163(8.13)	100 45534(5.65)	88 40791(13.50)	41 95234(4.11)	**100** **9591(3.42)**
	10	100 33583(9.91)	100 65853(7.35)	24 43200(28.68)	42 163273(15.23)	**100** **20212(3.38)**
	30	**100** **51658(2.87)**	100 102943(2.15)	81 41365(3.83)	100 112781(3.56)	100 61543(2.96)
	50	**100** **79267(1.56)**	100 159730(1.26)	70 71870(5.08)	99 172509(2.09)	100 102471(3.30)
Rastrigin	5	100 9197(3.36)	100 18111(2.95)	99 21387(10.73)	100 50406(8.62)	**100** **8011(5.69)**
	10	100 20524(2.58)	100 40838(1.80)	51 85901(32.13)	1 196000(–)	**100** **18211(3.83)**
	30	100 87595(2.69)	100 176909(1.72)	0 –	0 –	**100** **68614(5.70)**
	50	100 293163(3.35)	100 497462(2.08)	0 –	0 –	**100** **151624(4.40)**
Schwefel	5	100 8965(3.32)	100 17662(2.56)	100 11380(10.45)	100 24178(37.5)	**100** **6710(5.26)**
	10	100 19028(2.25)	100 37713(1.66)	100 41551(16.33)	100 105380(13.59)	**100** **14823(4.05)**
	30	100 60053(1.59)	100 120261(1.45)	5 275510(21.13)	0 –	**100** **49017(4.78)**
	50	99 106392(1.85)	100 215736(0.94)	0 –	0 –	**100** **89282(6.27)**
Ackley	5	100 13523(2.07)	100 26825(1.78)	100 111832(2.49)	100 22847(2.13)	**100** **9454(3.87)**
	10	100 27380(1.45)	100 54430(1.03)	100 22900(2.54)	100 51078(1.68)	**100** **19451(3.70)**
	30	100 75980(0.93)	100 152621(0.62)	100 64937(2.39)	100 177285(1.64)	**100** **56169(3.65)**
	50	100 121097(0.67)	100 243894(0.52)	75 114363(3.81)	100 272352(1.68)	**100** **90052(3.36)**
Griewank	5	100 23163(8.13)	100 45534(5.65)	88 40791(13.50)	41 95234(4.11)	**100** **16592(7.54)**
	10	100 33583(9.91)	100 65853(7.35)	24 43200(28.68)	42 163273(15.23)	**100** **28811(9.50)**
	30	100 51658(2.87)	100 102943(2.15)	81 41365(3.83)	100 112781(3.56)	**100** **38501(7.55)**
	50	100 79267(1.56)	100 159730(1.26)	70 71870(5.08)	99 172509(2.09)	**100** **61155(4.87)**
NS=100		24 cases	24 cases	17 cases	22 cases	**32 cases**
Lowest mean NF		2 cases	0 cases	3 cases	0 cases	**27 cases**

Table 7. The performance comparison of ADE-PS, SaDE, CoDE, jDE and JADE algorithms.

Function	D	ADE-PS	SaDE	CoDE	jDE	JADE
Sphere	5	**100**	100(–)	100(–)	100(–)	100(–)
		5500(4.26)	7655(2.95)	10747(3.08)	14821(3.41)	13856(2.65)
	10	**100**	100(–)	100(–)	100(–)	100(–)
		11754(5.01)	12304(2.82)	24199(2.80)	28358(2.57)	23035(2.95)
	30	100	**100**(+)	100(–)	100(–)	100(+)
		36083(3.86)	**31005(3.94)**	61089(2.32)	67349(1.37)	34577(3.07)
	50	100	100(+)	100(–)	100(–)	**100**(+)
		59179(4.23)	56234(3.83)	87384(2.33)	95734(1.87)	**42939(2.52)**
Schwefel 1.2	5	100	**100**(+)	100(–)	100(–)	100(–)
		8962(7.91)	**8889(3.22)**	15440(4.07)	26522(4.83)	20343(3.80)
	10	**100**	100(–)	100(–)	100(–)	100(–)
		21713(10.42)	26321(19.18)	38828(4.08)	65857(4.57)	29031(5.31)
	30	100	100(–)	**100**(+)	100(–)	100(+)
		249626(8.17)	394091(6.68)	**208775(1.00)**	377665(5.12)	87170(4.94)
	50	100	0(–)	9(–)	22(–)	**100**(+)
		836373(7.13)	–	486460(2.19)	970940(2.44)	**218639(4.31)**
Rosenbrock	5	**100**	99(–)	100(–)	100(–)	100(–)
		13416(9.90)	22802(30.02)	25211(7.08)	46081(11.38)	32466(14.64)
	10	**100**	38(–)	100(–)	100(–)	7(–)
		41170(11.32)	14223(25.84)	58911(5.01)	137735(12.69)	53357(12.05)
	30	**100**	0(–)	81(–)	31(–)	10(–)
		377428(13.06)	–	281516(6.59)	529180(4.71)	126790(3.09)
	50	**100**	0(–)	1(–)	4(–)	4 (–)
		937679(9.35)	–	488460(-)	946775(6.24)	225125(3.20)
Schwefel 2.22	5	**100**	100(–)	100(–)	100(–)	100(–)
		9591(3.42)	13733(2.31)	19056(2.37)	25467(1.96)	27602(1.99)
	10	100	100(–)	**100(+)**	100(–)	100(–)
		20212(3.38)	23816(1.84)	**13619(1.87)**	47520(1.72)	54266(3.47)
	30	100	**100(+)**	100(–)	100(–)	100(–)
		61543(2.96)	**53759(2.44)**	122277(1.81)	114079(1.52)	198059(1.48)
	50	100	**100(+)**	100(–)	100(–)	100(–)
		102471(3.3)	**87137(3.90)**	174414(2.19)	165118(1.52)	314970(7.85)
Rastrigin	5	**100**	100(–)	100(–)	100(–)	100(–)
		8011(5.69)	13352(4.86)	14296(4.49)	21345(4.38)	22617(2.56)
	10	**100**	100(–)	100(–)	100(–)	100(–)
		18211(3.83)	24755(4.07)	36493(3.29)	44271(3.47)	47759(1.95)
	30	**100**	97(–)	97(–)	100(–)	100(–)
		68614(5.70)	73422(5.32)	150794(4.22)	123334(2.68)	143686(1.30)
	50	**100**	57(–)	49(–)	100(–)	100(–)
		151624(4.4)	133396(5.16)	290445(5.28)	193196(2.8)	236536(1.4)
Schwefel	5	**100**	100(–)	100(–)	100(–)	100(–)
		6710(5.26)	10907(4.36)	12269(4.00)	17007(3.65)	20131(5.26)
	10	**100**	100(–)	100(–)	100(–)	49(–)
		14823(4.05)	20601(3.93)	28871(3.29)	32980(2.81)	43536(4.79)
	30	**100**	100(–)	99(–)	100(–)	55(–)
		49017(4.78)	53811(3.82)	92755(3.56)	84608(2.64)	138990(1.71)
	50	**100**	91(–)	100(–)	100(–)	16(–)
		89282(6.27)	97754(3.76)	163292(4.12)	126238(2.29)	227293(2.41)
Ackley	5	**100**	100(–)	100(–)	100(–)	100(–)
		9454(3.87)	12713(2.19)	18327(2.48)	25522(2.02)	25332(2.19)
	10	**100**	100(–)	100(–)	100(–)	100(–)
		19451(3.70)	20237(2.00)	40587(2.00)	47096(1.50)	40421(3.12)
	30	100	89(–)	100(–)	100(–)	**100(+)**
		56169(3.65)	68850(1.32)	98994(1.90)	106889(1.67)	**55491(2.99)**
	50	100	12(–)	100(–)	100(–)	**100(+)**
		90052(3.36)	324766(1.08)	138260(1.75)	149720(1.41)	**67224(2.06)**
Griewank	5	**100**	100(–)	100(–)	100(–)	100(–)
		16592(7.54)	32716(12.01)	27843(6.91)	57646(8.79)	47086(5.12)
	10	**100**	100(–)	100(–)	100(–)	100(–)
		28811(9.50)	36324(15.63)	59423(6.46)	65944(10.15)	86969(5.53)
	30	**100**	67(–)	100(–)	100(–)	100(–)
		38501(7.55)	32029(5.01)	65511(7.48)	70432(3.23)	41502(28.77)
	50	**100**	52(–)	95(–)	100(–)	8(–)
		61155(4.87)	56207(3.84)	88634(3.51)	97001(2.43)	43912(2.36)
NS=100		**32 cases**	20 cases	25 cases	29 cases	24 cases
Lowest mean NF		**22 cases**	4 cases	2 cases	0 cases	4 cases
Summary (+/=/–)			5/0/27	2/0/30	0/0/32	6/0/26

5.4. Performance comparison of ADE-PS and ADE-PS, SaDE, CoDE, jDE, and JADE algorithms algorithms

Experiment 4 compares ADE-PS with the following 4 adaptive DE algorithms: SaDE, CoDE, JDE, and JADE. The results are presented in Table 7. The numbers of cases that give a 100% success rate and the lowest mean NF are summarized in the bottom rows. The total of t-test comparison results is in the last row. The ADE-PS obtains all 32 cases of 100% success rate and 21 cases of the lowest mean NF while the remaining 11 cases achieve the second-lowest mean NF. The other methods give 20-29 cases of 100% success rate and do not give more than 4 cases of the lowest mean NF. The statistical test results also indicate that the ADE-PS significantly outperforms the compared methods.

5.5. Performance comparison of ADE-PS, EDEV, and FDDE algorithms on shifted and rotated test functions

Table 8 shows the performance comparison of ADE-PS, EDEV, and FDDE algorithms on shifted and rotated test functions from CEC2005. The best values of EDEV and FDDE are taken from the original papers. The results show that ADE-PS has the competitive performance when compare to both EDEV and FDDE.

6. Discussion

ADE-PS uses the pheromone strategy as the control parameter adaptation. The strategy has two parameters: the number of subintervals for partitioning parameter ranges (ns) and the reset period (rs). We obtain suitable $ns = 20$ from Experiment 1 and $rs = 20D$ from Experiment 2 and use this setting in Experiment 4 to make ADE-PS superior to the compared algorithms. The pheromone resetting process is essential for improving the search performance of the algorithm.

Figures 2 and 4 show the effect of pheromone resetting on probabilities associated with partition points of F and CR at every 40 generations for 10-dimensional Rosenbrock and Griewank functions, resepectively. Figures 3 and 5 show probabilities for corresponding cases without pheromone resetting.

ADE-PS without pheromone resetting accumulates all pheromone and use it for the entire run. After a short period of generations, the probabilities of F values are separated into low and high amounts of pheromone. Then, they tend to be stable. For the Griewank function, the pheromone of $F = 0.65$ and $CR = 0.1$ dominate the other values from beginning to end. For the Rosenbrock function, the pheromone of a few F and CR values dominate the others. We can observe that ADE-PS without pheromone resetting cannot change the dominant values during the search.

Table 8. The performance comparison of ADE-PS, EDEV, and FDDE on 30-dimensional shifted and rotated test functions.

Function	Statistics	ADE-PS	EDEV [28]		FDDE [6]	
F1	Mean	0.00E+00	0.00E+00	(=)	0.00E+00	(=)
	SD	0.00E+00	0.00E+00		0.00E+00	
F2	Mean	1.33E–13	3.69E–20	(+)	5.20E–13	(–)
	SD	1.43E–13	6.42E–20		3.36E–13	
F3	Mean	1.13E+05	3.63E+04	(+)	1.01E+05	(=)
	SD	6.60E+04	2.63E+04		8.22E+04	
F4	Mean	6.32E–02	6.26E–01	(–)	2.00E–03	(+)
	SD	5.21E–02	9.12E–01		1.63E–03	
F5	Mean	8.50E+02	1.60E+03	(–)	1.80E+02	(+)
	SD	4.19E+02	8.43E+02		1.47E+02	
F6	Mean	2.30E–06	6.36E–25	(+)	6.22E–01	(–)
	SD	3.99E–06	9.22E–35		5.07E+00	
F7	Mean	3.29E–03	7.86E–03	(–)	7.20E–02	(–)
	SD	5.69E–03	8.65E–03		5.87E–02	
F8	Mean	2.09E+01	2.11E+01	(–)	2.09E+01	(=)
	SD	3.54E–02	2.25E–02		1.70E–02	
F9	Mean	0.00E+00	0.00E+00	(=)	5.45E+01	(–)
	SD	0.00E+00	0.00E+00		4.44E+00	
F10	Mean	1.45E+02	4.05E+02	(–)	4.66E+01	(+)
	SD	2.62E+00	8.33E+00		3.80E+00	
F11	Mean	3.20E+01	4.46E+02	(–)	2.79E+01	(+)
	SD	1.67E+00	4.65E+00		2.27E+00	
F12	Mean	3.36E+04	6.04E+03	(+)	3.74E+03	(+)
	SD	7.65E+03	5.43E+03		5.05E+03	
Summary(+/=/–)			4/2/6		5/3/4	

In contrast, Figures 2 and 4 show the behaviors of ADE-PS with pheromone resetting. The dominant pheromone values occur after the first time of resetting. For both functions, the dominant values of F change after each resetting period. On the other hand, the dominant values of CR tend to be the same for consecutive resetting periods.

ADE-PS with pheromone resetting gradually gathers the feedback from the parameters of better solutions and biases the search toward those values. The resetting mechanism resets the current pheromone back to the equal initial value for eliminating the dominant values. The suitable resetting period balances the times for accumulating the pheromone and intensifying the search by dominant values. Once the resetting is performed, all F and CR values will have the same opportunities to compete for the new dominant values. Therefore, the proposed ADE-PS with a pheromone resetting strategy can archive good performance in solving various continuous optimization problems.

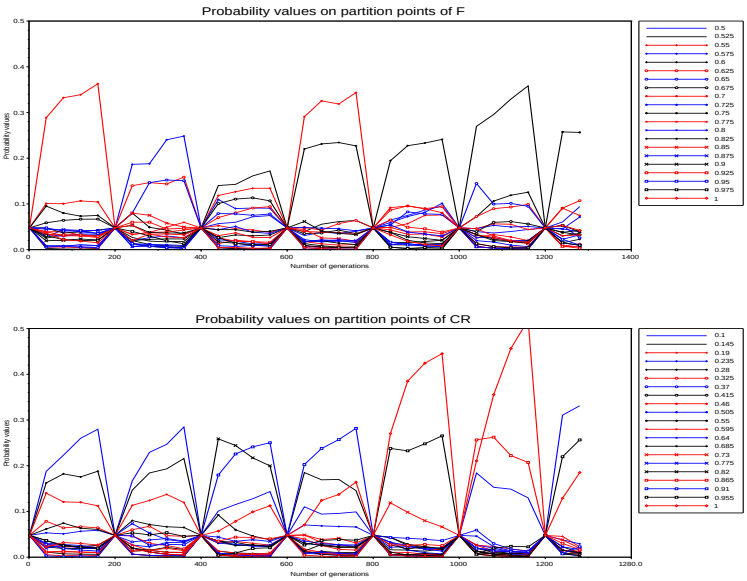

Figure 2. Probabilities on partition points of F and CR at every 40 generations for 10-dimensional Rosenbrock function with pheromone resetting.

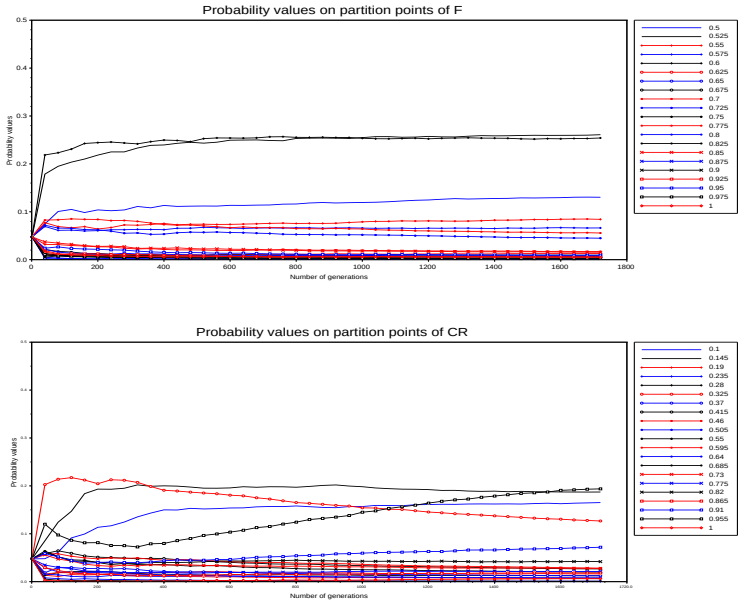

Figure 3. Probabilities on partition points of F and CR at every 40 generations for 10-dimensional Rosenbrock function without pheromone resetting.

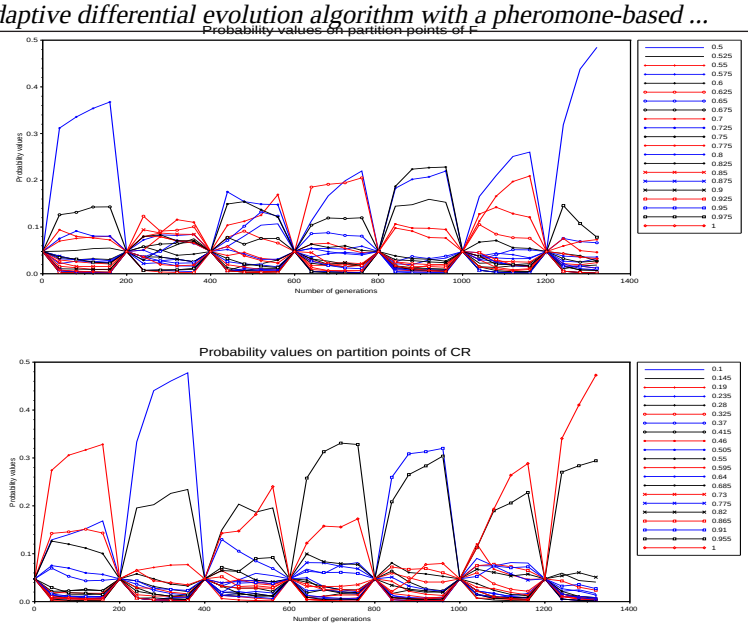

Figure 4. Probabilities on partition points of F and CR at every 40 generations for 10-dimensional Griewank function with pheromone resetting.

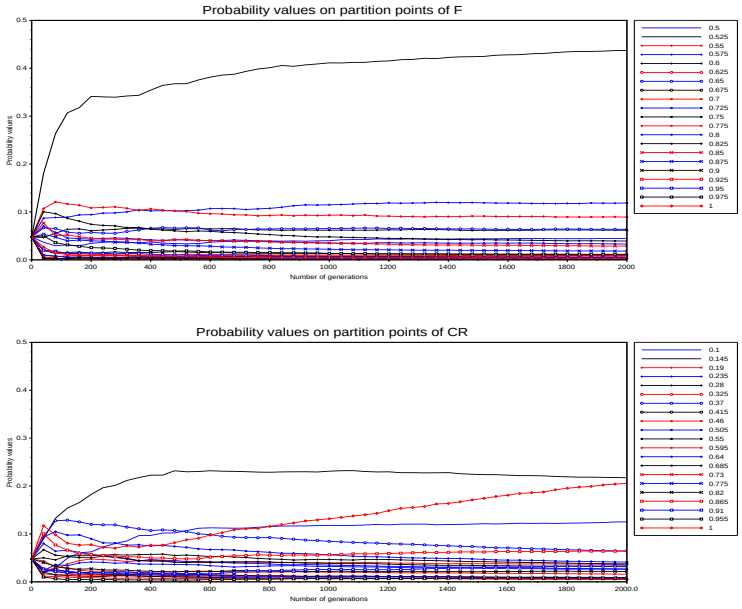

Figure 5. Probabilities on partition points of F and CR at every 40 generations for 10-dimensional Griewank function without pheromone resetting.

7. Conclusion

We have presented an adaptive differential evolution algorithm with a pheromone-based learning strategy (ADE-PS) in this paper. The algorithm applies the pheromone on the partition points of scaling factor F and crossover rate CR values to formulate the probabilities for choosing the associated F_i and CR_i values in mutation and crossover operations. Moreover, ADE-PS also uses the pheromone resetting strategy to unlearn and relearn the dominant values that bias the search. Experimental results show that ADE-PS outperforms the classic DE and several well-known adaptive differential evolution algorithms. This pheromone resetting strategy can be adapted for other optimization methods to enhance their performances.

Acknowledgment

Pirapong Singsathid thanks the Development and Promotion of Science and Technology Talents Project (DPST) for the financial support. Pikul Puphasuk and Jeerayut Wetweerapong would like to thank the Department of Mathematics, Faculty of Science, Khon Kaen University for simulation equipment support.

References

[1] Al-Dabbagh R. D., Neri F., Idris N., Baba M. S., Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy, *Swarm and Evolutionary Computation*, **43**, 2018, 284–311.

[2] 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 Transactions on Evolutionary Computation*, **10**, 2006, 646-–657.

[3] Brest J., Bošković B., Žumer V., An improved self-adaptive differential evolution algorithm in single objective constrained real-parameter optimization, *2010 IEEE Congress on Evolutionary Computation (CEC)*, 2010, 1–8.

[4] Brest J., Maučec M. S., Bošković B., iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization, *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, 1188–1195.

[5] Brest J., Maučec M. S., and Bošković B., Single objective real-parameter optimization: Algorithm jSO, *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, 1311–1318.

[6] Cheng J., Pan Z., Liang H., Gao Z., Gao J., Differential evolution algorithm with fitness and diversity ranking-based mutation operator, *Swarm and Evolutionary Computation*, **61**, 2021, 100816.

[7] Das S., Suganthan P. N., Differential evolution : A survey of the state-of-the-art, *IEEE Transactions on Evolutionary Computation*, **15**, 2011, 4-–31.

[8] Das S., Mullick S. S., Suganthan P. N., Recent advances in differential evolution - An updated survey, *Swarm and Evolutionary Computation*, **27**, 2016, 1-–30.

[9] Dorigo M., Stützle T., *Ant colony optimization*, MIT Press, Cambridge, MA, 2004.

[10] Dorigo M., Socha K., Ant colony optimization for continuous domains, *European Journal of Operational Research*, **185**, 2008, 1155-–1173.

[11] Hinterding R., Michalewicz Z., Eiben A. E., Adaptation in evolutionary computation : A survey, in: *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, 1997, 65–69.

[12] Leon M., Xiong N., Adapting differential evolution algorithms for continuous optimization via greedy adjustment of control parameters, *J. Artif. Intell. Soft Comput. Res.*, **6**, 2016, 103—118.

[13] Mallipeddi R., Suganthan P. N., Pan Q. K., Tasgetiren M. F., Differential evolution algorithm with ensemble of parameters and mutation strategies, *Appl. Soft Comput.*, **11**, 2011, 1679–1696.

[14] Meng Z., Pan J. S., PaDE: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization, *Knowledge-Based Systems*, **168**, 2019, 80-99.

[15] Price K., Storn R., Differential evolution: a simple evolution strategy for fast optimization, *Dr Dobb's. J. Softw. Tools*, **22**, 1997, 18-–24.

[16] Qin A. K., Huang V. L., Suganthan P. N., Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Congress on Evolutionary Computation*, **13**, 2009, 398—417.

[17] Singsathid P., Wetweerapong J., Solving Continuous Optimization Problems by Ant Colony Optimization with Domain Partitioning Technique, in: *Proceedings of the* 23^{rd} *annual meeting in mathematics (AMM2018)*, 2018, 257–262.

[18] Storn R., Price K., *Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces*, Technical Report TR-95-012, ICSI, Berkeley, 1995.

[19] Storn R., Price K., Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.*, **11**, 1997, 341-–359.

[20] Storn R., Differential evolution research-trends and open question, in: U. K. Chakraborty (ed.), *Advances in Differential Evolution*, Springer, Berlin, 2008, 1–31.

[21] Suganthan P. N., Hansen N., Liang J., Deb K., Chen Y., Auger A., Tiwari S., Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, *Natural Computing*, 2005, 341–357.

[22] Tanabe R., Fukunaga A., Success-History based parameter adaptation for differential evolution, *2013 IEEE Congress on Evolutionary Computation (CEC)*, 2013, 71—78.

[23] Tanabe R., Fukunaga A., Improving the search performance of SHADE using linear population size reduction, *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014, 1658–1665.

[24] Tvrdík J., Competitive differential evolution, in: R., Matoušek and P. Ošmera (eds.) *MENDEL 2006, 12th International Conference on Soft Computing*, University of Technology, Brno, 2006, 7–12.

[25] Wang Y., Cai Z., Zhang Q., Differential evolution with composite trial vector generation strategies and control parameters, *IEEE Transactions on Evolutionary Computation*, **15**, 2011, 55 – 66.

[26] Wetweerapong J., Puphasuk P., An improved differential evolution algorithm with a restart technique to solve systems of nonlinear equations, *An International Journal of Optimization and Control: Theories & Applications*, **10**, 2020, 118–136.

[27] Wongpen J., Wetweerapong J., Puphasuk P., Finding a maximum clique in social networks using a modified differential evolution algorithm, *WSEAS Transactions on Systems and Control*, **14**, 2019, 333–342.

[28] Wu G., Shen X., Li H., Chen H., Lin A., Suganthan P. N., Ensemble of differential evolution variants, *Information Sciences*, **423**, 2018, 172—186.

[29] Xiao J., Li L.P., A hybrid ant colony optimization for continuous domains, *Expert Systems with Applications*, **38**, 2011, 11072-–11077.

[30] Zhang J., Sanderson A. C., JADE: adaptive differential evolution with optional external archive, *IEEE Congress on Evolutionary Computation*, **13**, 2009, 945-–958.

Received 08.03.2022, Accepted 15.12.2022