Bulletin of Electrical Engineering and Informatics
Vol. 13, No. 1, February 2024, pp. 559∼571
ISSN: 2302-9285, DOI: 10.11591/eei.v13i1.6590

Self-adaptive differential evolution algorithm with dynamic fitness-ranking mutation and pheromone strategy

Pirapong Singsathid, Jeerayut Wetweerapong, Pikul Puphasuk
Department of Mathematics, Faculty of Science, Khon Kaen University, Khon Kaen, Thailand

Article Info

Article history:

Received Apr 27, 2023
Revised Jul 11, 2023
Accepted Aug 30, 2023

Keywords:

Continuous optimization
Differential evolution
Mutation strategy
Pheromone strategy
Self-adaptation

ABSTRACT

Differential evolution (DE) is a population-based optimization algorithm widely used to solve a variety of continuous optimization problems. The self-adaptive DE algorithm improves the DE by encoding individual parameters to produce and propagate better solutions. This paper proposes a self-adaptive differential evolution algorithm with dynamic fitness-ranking mutation and pheromone strategy (SDE-FMP). The algorithm introduces the dynamical mutation operation using the fitness rank of the individuals to divide the population into three groups and then select groups and their vectors with adaptive probabilities to create a mutant vector. Mutation and crossover operations use the encoded scaling factor and the crossover rate values in a target vector to generate the corresponding trial vector. The values are changed according to the pheromone when the trial vector is inferior in the selection, whereas the pheromone is increased when the trial vector is superior. In addition, the algorithm also employs the resetting operation to unlearn and relearn the dominant pheromone values in the progressing search. The proposed SDE-FMP algorithm using the suitable resetting periods is compared with the well-known adaptive DE algorithms on several test problems. The results show that SDE-FMP can give high-precision solutions and outperforms the compared methods.

Corresponding Author:

Pikul Puphasuk
Deparment of Mathematics, Faculty of Science, Khon Kaen University
Khon Kaen, 40002, Thailand
Email: ppikul@kku.ac.th

1. INTRODUCTION

Solving multimodal, high-dimensional, and non-linear real-world optimization problems requires well-designed efficiency optimization methods [1]–[5]. To address this challenge, many researchers have proposed evolutionary algorithms such as genetic algorithm (GA) [6], ant colony optimization (ACO) [7], particle swarm optimization (PSO) [8], [9], artificial bee colony algorithm (ABC) [10], and differential evolution (DE) [11], [12] for these problems.

DE is a population-based global search algorithm introduced by Storn and Price in 1997 for continuous optimization. Its operations consist of mutation, crossover, and selection [11]. The performance of DE depends on the control parameters: scaling factor F and crossover rate CR. The scaling factor controls the step size of the mutation operation, and the crossover rate indicates the probability of exchanging elements between the mutant and target vectors. These control parameter values significantly affect the algorithm's performance, and DE with the fixed parameters F and CR are only suitable for specific problems. Thus, many control parameter adaptation techniques have been proposed to improve the algorithm's performance.

The adaptive parameter technique uses the overall feedback from the search to adjust the parameter values, and the self-adaptive approach encodes the parameter values to the individuals and propagates them to others. These adaptation techniques can also modify the mutation operation. The DE algorithms with adaptive parameters or improved mutation operations can accelerate the search process. However, they may face challenges in achieving high-precision solutions, such as getting stuck in local optimum or losing population diversity. Therefore, further improvements and strategies are necessary to enhance the adaptive approaches.

This paper proposes a self-adaptive differential evolution algorithm with dynamic fitness-ranking mutation and pheromone strategy called SDE-FMP. The algorithm introduces a fitness-ranking mutation strategy that dynamically divides the population into three groups according to fitness rank. Then, it selects groups and their vectors with adaptive probabilities to create a mutant vector. The SDE-FMP also employs the self-adaptive control parameter adaptation by encoding the pre-assigned F and CR values to the target vector and adjusting these values with the pheromone information. This self-adaptive process learns to find the appropriate parameters naturally. In addition, the algorithm incorporates the resetting operation to manage the dominant pheromone and enhance the efficiency of probabilities. The main contributions of our proposed algorithm are the ability to achieve high-precision solutions and superior performance to the compared methods.

The remainder of this paper is organized as follows: i) section 2 briefly reviews some related work; ii) section 3 gives details of the proposed SDE-FMP algorithm; iii) the experiment sets are given in section 4; iv) section 5 discusses experimental results; v) the results and discussion are presented in section 6; and vi) the conclusion is given in section 7.

2. LITERATURE REVIEW

This section reviews the basic DE algorithm, self-adaptive DE algorithms, adaptive DE algorithms, and continuous ACO algorithms that inspire our proposed algorithm.

The basic DE algorithm has three iterative operations: mutation, crossover, and selection. The algorithm generates NP population vectors from feasible solution space and indicates the best vector. For $j = 1, \ldots, NP$, the mutation operation creates the mutant vector v_j by randomly choosing three distinct vectors different from the target vector x_j as (1):

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

where F is the scaling factor.

Next, the crossover operation constructs the trial vector u_j by exchanging the component of x_j and v_j with the crossover rate CR. Finally, the selection operation updates the target vector with the trial vector when the fitness value of u_j is better than that of x_j.

Since the basic mutation equation with the fixed parameters F and CR cannot solve a wide range of problems, the parameter adaptation and enhanced mutation strategy have been proposed to improve the performance of basic DE.

2.1. Self-adaptive DE algorithms

The self-adaptive DE algorithms encode the control parameter values to each target vector, adapt them based on the feedback of the search, and propagate the better ones to the next generation.

The jDE by Brest *et al.* [13] is the self-adaptive DE algorithm encoding F_i and CR_i to ith target vector with initial values 0.5 and 0.9, respectively. Mutation and crossover operations use these values from the target vector with the probabilities of 0.9; otherwise, it generates anew from $[0.5, 0.9]$ and $[0, 1]$, respectively. The algorithm does not use feedback from the selection operation. The jDE outperforms the basic DE with $F = 0.5$ and $CR = 0.9$ on several benchmark functions. Cheng *et al.* [14] proposed DE with FDDE strategy that uses the combined fitness and diversity rankings to position the random vectors in a mutation operator where the diversity ranking computes from the difference of the median fitness and individual fitness values. The strategy improves the performance of jDE in both low-dimensional and high-dimensional problems. This work indicates the role of position selection for vectors in the mutation operator. Qin *et al.* [15] proposed a differential evolution algorithm with strategy adaptation called SaDE. The algorithm uses four mutation operations and encodes their indices to each target vector. It assigns corresponding parameters F and CR from

Normal distributions and changes encoded information based on the probability of success and failure in the selection operation. The experimental results show that SaDE outperforms basic DE and jDE. The EPSDE algorithm by Mallipeddia *et al.* [16] uses an ensemble of parameters and mutation strategies. This algorithm initially encodes a mutation strategy for a target vector and assigns corresponding parameters F and CR. If the generated trial vector is not better than its target vector, the algorithm changes the target's encoded information to the new one. EPSDE outperforms the classic DE and three adaptive DE algorithms: jDE, SaDE, and JADE.

2.2. Adaptive DE algorithms

The adaptive DE algorithm adjusts the control parameter with the overall feedback from the search. Then, the newly generated parameter will be biased according to the search feedback.

Zhang and Sanderson [17] presented the JADE algorithm that generates parameters F and CR from the Normal and Cauchy distributions, respectively. The algorithm implements an external archive for keeping the inferior vectors from selection and uses some top best vectors and the archived vectors in mutation.The results show that JADE is better than the classic DE and some adaptive DE algorithms. Tanabe and Fukunaga [18] introduced success-history-based parameter adaptation for differential evolution (SHADE), which improves the JADE algorithm with historical memories for each individual to update the mean of distributions. Experimental results show that SHADE is competitive with EPSDE, JADE, and CoDE. Next, Wang *et al.* [19] introduced DE with composite trial vector generation strategies and control parameters called CoDE. It generates three trial vectors with three different mutation operations and chooses the best one to compete with the target vector. The results show that CoDE is better than jDE, JADE, SaDE, and EPSDE algorithms. Zou *et al.* [20] presented the CUSDE algorithm that uses a new mutation strategy by selecting the vectors with the probability calculated from the number of consecutive unsuccessful updates and removing individuals with those large numbers. The basic DE with this approach outperforms basic DE and some adaptive DEs.

2.3. Ant colony optimization

ACO is the population-based algorithm using the pheromone strategy to guide the ant population to locate optimal solutions for discrete optimization [21]. The ACO for continuous optimization requires dividing the initial spaces into discrete subspaces for formulating the pheromone structure. During the search process, the algorithm constructs a new solution vector with the components corresponding to subspaces according to the pheromone gathered from better solutions [22].

The ACO$_R$ algorithm is the first ACO algorithm for continuous optimization introduced by Dorigo and Socha [23]. The algorithm uses a pheromone structure to store the best solution components in an archive table and generates a new candidate solution with the corresponding distributions. At the end of the generation, it updates the pheromone by adding better candidate solutions and removing the worst archive solutions in the table. The performance of ACO$_R$ is competitive with other probability learning methods.

Xiao and Li [24] presented the hybrid of ACO$_R$ and DE algorithms called HACO. It uses DE to generate new candidate solutions for the ACO$_R$ algorithm. The experimental results show that HACO performs better than ACO$_R$ algorithms. Singsathid and Wetweerapong [25] introduced the ACO with the domain partitioning technique called PACO. It generates solution components from partition points of adaptive search subspace that cover the best solutions and updates the pheromone according to the newly obtained best solution. The experiments show that PACO outperforms some well-known continuous ACO algorithms.

3. THE PROPOSED SDE-FMP ALGORITHM

We propose a self-adaptive differential evolution algorithm with dynamic fitness-ranking mutation and pheromone strategy, called SDE-FMP, which improve classic DE mutation by using the dynamic mutation and self-adaptive control parameters controlled with pheromone strategy and resetting operation. The details of SDE-FMP are as follows.

3.1. New dynamic mutation strategy for SDE-FMP

SDE-FMP sorts the population vectors at the beginning of each generation according to the fitness values and divides them into three groups with the same size, denoted by G_1, G_2, G_3 where they represent the top best individuals, intermediate individuals, and worst individuals, respectively.

To create a mutant vector for each target vector $x_i, i = 1, 2, \ldots, NP$, the algorithm chooses a group g_k (from G_1, G_2, or G_3) for each position $k = 1, 2, 3$ of the mutation equation (g_2 and g_3 must be different for diversity) with probability vectors $PropR_k$ calculated from pheromone vectors $PheromoneR_k$. Note that SDE-FMP initializes all components of pheromone vectors to be 1, ensuring equal probability.

Next, the algorithm chooses three distinct random vectors $x_{R_1}, x_{R_2}, x_{R_3}$ which differ from x_i by uniformly selecting random vectors in the corresponding selected g_k groups to generate a mutant vector as (2):

$$v_i = x_{R_1} + F(i) \cdot (x_{R_2} - x_{R_3}) \tag{2}$$

where the scaling factor $F(i)$ corresponds to target vector x_i.

The mutant vector enters the crossover operation to generate the trial vector u_i as (3):

$$u_{i,j} = \begin{cases} v_{i,j} & ; s_j \leq CR(i) \ \ or \ \ j = I_{rand} \\ x_{i,j} & ; \text{otherwise} \end{cases} \tag{3}$$

where $j = 1, \ldots, D$; s_j is a uniform random number in $(0, 1)$ and I_{rand} is a randomly fixed integer from 1 to D.

At the selection operation, if u_i is better than x_i, the algorithm updates the pheromone at the lth position of $g_k = G_l$ by adding one to the associated pheromone vector position to reinforce the pheromone information related to a successful solution as (4):

$$PheromoneR_k(l) = PheromoneR_k(l) + 1 \tag{4}$$

At the end of each generation, the algorithm normalizes the $PheromoneR_k$ to the probability vector $PropR_k$ as (5):

$$PropR_k(l) = \frac{PheromoneR_k(l)}{\sum_{m=1}^{3} PheromoneR_k(m)} \tag{5}$$

for $l = 1, 2, 3$.

3.2. Self-adaptive control parameters of F and CR

We use scaling factor values $F = 0.5, 0.7, 0.9$ to control the step size for the mutant vectors and crossover rate values $CR = 0.1, 0.9$ to balance between intensifying and diversifying the search. $CR = 0.1$ is suitable for a local search, while $CR = 0.9$ is suitable for a global search. So, we have six combinations of these values.

At initialization, the algorithm sets all components of the pheromone vector $PheromoneFCR$ to be one and encodes a random pair of $F(i)$ and $CR(i)$ for each target vector x_i. The algorithm uses $F(i)$ and $CR(i)$ from x_i to generate a trial vector u_i from mutation and crossover operations.

At the selection operation, if u_i is better than x_i, the target vector retains its current $F(i)$ and $CR(i)$ values and the associated pheromone vector value $PheromoneFCR(t)$ is incremented by one.

$$PheromoneFCR(t) = PheromoneFCR(t) + 1 \tag{6}$$

where t is the corresponding index of that combination. Otherwise, the target vector is re-encoded with a new random pair of $F(i)$ and $CR(i)$ based on the probability vector $PropFCR$, allowing the target vector to explore new parameter values. Note that $PropFCR$ is calculated from $PheromoneFCR$ for each generation as (7):

$$PropFCR(t) = \frac{PheromoneFCR(t)}{\sum_{m=1}^{6} PheromoneFCR(m)} \tag{7}$$

for $t = 1, 2, \ldots, 6$.

3.3. The pheromone resetting

At the end of each generation, SDE-FMP determines whether the pheromone vectors have reached the specified thresholds to prevent the dominance of certain pheromone values and promote fair competition among choices. The two parameters r_g and r_p represent the thresholds for $PheromoneR_k$ and $PheromoneFCR$, respectively. If the sum of any pheromone vector is greater than the corresponding threshold, the algorithm resets all elements of those pheromone vectors back to 1. The pseudo code of SDE-FMP is presented in Algorithm 1.

Algorithm 1 SDE-FMP algorithm

Initialize the population of NP individuals
Find the best vector x_{best} and its best function value f_{best}
Encode the $F(i), CR(i)$ values to each target vector $x_i, i = 1, \dots, NP$
Set all elements of $PheromoneR_k, k = 1, 2, 3$ and $PheromoneFCR$ to 1
Set all elements of $ProbR_k$ to be equal for $k = 1, 2, 3$
Set all elements of $ProbFCR$ to be equal
Set number of function evaluations $nf = 0$
Set the VTR or the maximum number of function evaluations $maxnf$
while stopping condition is not satisfied **do**
    Sort the population individuals according to fitness ranking and divide them into G_1, G_2, G_3 groups
    for $i = 1 : NP$ **do**
        Choose distinct x_{R_k} vectors with the probabilities $ProbR_k, k = 1, 2, 3$
        Generate a mutant vector using eq. (2)
        Apply the crossover operation eq. (3) to get a trial vector u_i
        Evaluate $f(u_i)$ and $nf \leftarrow nf + 1$
        if $f(u_i) < f(x_i)$ **then**
            Replace x_i with u_i
            Update each $PheromoneR_k$ using eq. (4)
            Update $PheromoneFCR$ using eq. (6)
            if $f(u_i) < f_{best}$ **then**
                Replace x_{best} with u_i
            end if
        else
            Re-encode the new random $F(i), CR(i)$ for x_i according to $ProbFCR$
        end if
    end for
    if $\sum PheromoneR_k \geq r_g$ for some k **then**
        $PheromoneR_k \leftarrow 1, k = 1, 2, 3$
    end if
    if $\sum PheromoneFCR \geq r_p$ **then**
        $PheromoneFCR \leftarrow 1$
    end if
    Normalize $PheromoneR_k$ to be $ProbR_k$ using eq.(5)
    Normalize $PheromoneFCR$ to be $ProbFCR$ using eq.(7)
end while
Report the obtained x_{best}, f_{best} and nf

4. EXPERIMENTAL DESIGN

The performance of SDE-FMP is tested on eight selected benchmark functions that cover four main types: uni-modal, multi-modal, separable, and non-separable. Their formulae and search ranges are presented in Table 1. First, we design a preliminary experiment for finding the suitable values r_g and r_p for all types of problems. Then, we conduct two comparison experiments to evaluate the performance of SDE-FMP against other adaptive DE algorithms. The details of each experiment are given in the following subsections.

Table 1. Test functions

Function	Formula	Search range
Sphere	$F_1(x) = \sum_{i=1}^{D}(x_i)^2$	$[-100, 100]$
Schwefel 1.2	$F_2(x) = \sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$	$[-100, 100]$
Rosenbrock	$F_3(x) = \sum_{i=1}^{D-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$[-100, 100]$
Griewank	$F_4(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]$
Rastrigin	$F_5(x) = \sum_{i=1}^{D}[x_i^2 - 10cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$
Ackley	$F_6(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2})$	$[-32.32]$
	$-\exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)) + 20 + e$	
Schwefel	$F_7(x) = 418.98288727243369 \cdot D - \sum_{i=1}^{D}(x_i \sin(\sqrt{\lvert x_i \rvert}))$	$[-500, 500]$
Schwefel 2.22	$F_8(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-100, 100]$

4.1. Finding the suitable values r_g and r_p for SDE-FMP

To find the suitable values for resetting periods r_g and r_p, the dimensions of test functions are $D = 10, 30$. The number of population $NP = 30$, $maxnf = 20000D$ and $VTR = 10^{-10}$ are used. The parameters r_g and r_p are varied as $r_g = 200, 500$ and $r_p = 200, 300, 400$. The algorithm performs 50 independent runs for each configuration. We report the number of successful runs (NS), the mean number of function evaluations (meanNF), and the percentage of the standard deviation of function evaluations (%SD).

4.2. Comparing the performance of SDE-FMP with other adaptive DE algorithms using VTR

We use the obtained values r_g and r_p to compare the performance of SDE-FMP with some well-known adaptive DE algorithms: JADE [17], CoDE [19], jDE [13], and SaDE [15]. The SDE-FMP uses the same setting as the first experiment, while the compared algorithms use the parameter settings as in their original papers. The dimensions are varied as $D = 10, 30, 50$. All algorithms perform 100 independent runs for each configuration. The MATLAB source codes of JADE, CoDE, jDE, and SaDE are available from Zhang's homepage: http://dces.essex.ac.uk/staff/qzhang.

4.3. Comparing the performance of SDE-FMP with other adaptive DE algorithms using maxnf on CEC 2005 benchmark functions

We compare the performance of SDE-FMP using suitable r_g and r_p with SaDE [15], FDDE_F [14], and CUSDE [20] on 30-dimensional benchmark functions of CEC 2005 [26]. The experiment reports the mean of optimal values and standard deviation using $maxnf = 10000D$ over 50 independent runs. The best values of the compared algorithms are from their original papers. We use the t-test at the significance level of 0.05 to compare their performances. The symbols $+, 0, -$ represent that the mean of the optimal value of the SDE-FMP is superior to, equal to, and inferior to the compared algorithm, respectively.

5. EXPERIMENTAL RESULTS

5.1. The suitable values r_g and r_p for SDE-FMP

The experiment finds the suitable values r_g and r_p that give the highest number of successful runs and the lowest meanNF for SDE-FMP. Table 2 shows that three combinations of (r_g, r_p) i.e., (200, 400), (500, 200), and (500, 300) give 50 successful runs for all cases. We highlight the lowest meanNF values among these three combinations for each case and obtain $r_g = 500$ and $r_p = 300$ as the best values.

Table 2. The performance comparison of SDE-FMP with different values r_g and r_p over 50 independent runs

F	D	Statistics	r_g =200			r_g =500			$r_g = 1000$		
			$r_p = 200$	$r_p = 300$	r_p =400	r_p =200	r_p =300	$r_p = 400$	$r_p = 200$	$r_p = 300$	$r_p = 400$
F_1	10	NS	50	50	50	50	50	50	50	50	50
		meanNF	9122.62	8966.98	8939.94	8589.55	8569.9	8593.13	8309.1	8301.7	8330.37
		%SD	5.54	5.65	6.66	8.29	7.00	6.32	5.64	9.38	8.78
	30	NS	50	50	50	50	50	50	50	50	50
		meanNF	31727.14	32191.1	32070.8	31035.48	30801.56	31003.3	26165.16	29451.24	29621.88
		%SD	4.24	3.08	3.17	3.87	4.21	4.05	5.54	5.65	6.13
F_2	10	NS	50	50	50	50	50	50	50	50	50
		meanNF	14886.4	14813.6	15007.06	14532.58	14427.32	14364.64	13775.38	14596.9	14374.22
		%SD	6.98	8.68	9.23	11.01	10.5	11.33	7.50	10.68	12.54
	30	NS	50	50	50	50	50	50	50	50	50
		meanNF	141685.6	139038.04	139334.24	133898.42	131200.62	129951.46	107370.16	129738.66	127754.88
		%SD	6.69	8.28	6.92	8.00	9.47	5.56	7.72	8.62	10.38
F_3	10	NS	48	47	50	50	50	46	44	50	48
		meanNF	25334.52	25313.85	25288.82	24499.9	24470.8	24742.02	21613.9	24017.18	23592.12
		%SD	9.77	13.12	10.07	9.77	14.57	11.14	10.03	14.55	17.65
	30	NS	49	49	50	50	50	50	50	47	47
		meanNF	219922.79	227484.93	246704.98	202595.87	207844.62	218021.04	144629.44	192881.38	199274.53
		%SD	10.3	11.3	11.51	9.87	12.76	10.95	8.48	12.23	12.35
F_4	10	NS	50	50	50	50	50	50	50	49	49
		meanNF	23443.48	24055.58	23895.28	22821	23882.2	24615.22	23696.84	23748.12	23536.1
		%SD	17.16	14.51	18.72	19.98	20.50	24.70	19.75	16.59	17.67
	30	NS	50	50	50	50	50	49	50	50	50
		meanNF	35335.16	34497.8	34799.02	33826.88	33284.62	34164.87	28456.36	32186.2	32371.8
		%SD	5.73	5.70	7.00	6.83	5.98	7.32	8.33	6.47	8.56
F_5	10	NS	50	50	50	50	50	50	50	50	50
		meanNF	12863.76	12887.7	12956.24	12571.5	12342.3	12494.54	12598.32	12326.3	12362.02
		%SD	4.95	4.26	4.36	4.96	5.58	5.29	5.35	6.94	6.32
	30	NS	50	50	50	50	50	50	50	50	50
		meanNF	58730.32	57263.4	57201.62	57217.98	56480.04	55874.78	52726.84	55993.94	55891.36
		%SD	4.01	3.22	3.54	4.15	3.92	4.37	5.36	4.18	3.72
F_6	10	NS	50	50	50	50	50	50	50	50	50
		meanNF	15230.08	15207.54	15109.3	14476.26	14143.74	14008.16	13817.44	13791.78	13784.34
		%SD	4.99	5.05	5.38	5.35	4.73	5.63	7.42	8.34	8.84
	30	NS	50	50	50	50	50	50	50	50	50
		meanNF	52043.1	51920.1	51532.06	50354.46	49440.6	50054.94	48582.88	48513.56	48565.88
		%SD	2.30	2.44	2.36	2.92	4.47	2.54	4.28	5.06	4.92
F_7	10	NS	50	50	50	50	50	50	50	50	50
		meanNF	11398.32	11515.02	11405.52	11092.34	10859.18	10954.26	10957.84	10834.18	10900.8
		%SD	4.33	4.89	4.87	5.48	5.72	6.36	5.27	6.55	6.51
	30	NS	50	50	50	50	50	50	50	48	50
		meanNF	41162.04	41313.46	40652.2	40615.06	39669.68	39620.16	35394.56	39113.77	38681.88
		%SD	2.88	2.75	2.79	3.74	4.63	4.70	6.57	5.46	5.39
F_8	10	NS	50	50	50	50	50	50	50	50	50
		meanNF	15752.34	15797.36	15921.62	15143.78	15199.58	15148.26	14683.66	14625.28	14772.74
		%SD	4.09	4.03	4.34	5.04	4.33	5.93	6.75	5.50	6.02
	30	NS	50	50	50	50	50	50	50	50	50
		meanNF	53337.04	53095.36	52594.54	51815.44	51384.32	51184.22	50815.5	50257.46	50110.68
		%SD	2.21	2.28	1.63	2.45	2.69	2.53	3.40	3.16	3.62
NO. 50 successful runs cases			14/16	14/16	16/16	16/16	16/16	14/16	15/16	13/16	13/16

5.2. Performance comparison of SDE-FMP with other adaptive DE algorithms using VTR

We compare the performance of SDE-FMP with SaDE, CoDE, jDE, and JADE using the $VTR = 10^{-10}$. Table 3 presents the number of successful runs and meanNF and highlights the best results that give 100 successful runs with the lowest meanNF.

The results show that SDE-FMP, SaDE, CoDE, jDE, and JADE give 100 successful runs for 24, 12, 16, 21, and 17 cases, respectively. Their lowest mean counts are 18, 0, 1, 0, and 5, respectively. Therefore, SDE-FMP achieves the best performance. Note that all compared algorithms cannot achieve high-quality solutions for the Rosenbrock function F_3.

Table 3. The performance comparison of SDE-FMP, SaDE, CoDE, jDE, and JADE over 100 independent runs

F	D	Statistics	SDE-FMP	SaDE	CoDE	jDE	JADE
F_1	10	NS	100	100	100	100	100
		meanNF(%SD)	8549(7.70)	12304(2.82)	24199(2.80)	28358(2.57)	23035(2.95)
	30	NS	100	100	100	100	100
		meanNF(%SD)	30732(4.78)	31005(3.94)	61089(2.32)	67349(1.37)	34577(3.07)
	50	NS	100	100	100	100	100
		meanNF(%SD)	54935(2.62)	56234(3.83)	87384(2.33)	95734(1.87)	42939(2.52)
F_2	10	NS	100	100	100	100	100
		meanNF(%SD)	14372(9.92)	26321(19.18)	38828(4.08)	65857(4.57)	29031(5.31)
	30	NS	100	100	100	100	100
		meanNF(%SD)	131979(7.96)	394091(6.68)	208775(1.00)	377665(5.12)	87170(4.94)
	50	NS	100	0	9	22	100
		meanNF(%SD)	451807(16.85)	0(0.00)	486460(2.19)	970940(2.44)	218639(4.31)
F_3	10	NS	100	38	100	100	7
		meanNF(%SD)	24218(13.20)	14223(25.84)	58911(5.01)	137735(12.69)	53357(12.05)
	30	NS	100	0	81	31	10
		meanNF(%SD)	208507(11.35)	0(0)	281516(6.59)	529180(4.71)	126790(3.09)
	50	NS	100	0	1	4	4
		meanNF(%SD)	769397(11.69)	0(0)	488460(0)	946775(6.24)	225125(3.2)
F_4	10	NS	100	100	100	100	100
		meanNF(%SD)	23283(23.00)	23816(1.84)	13619(1.87)	47520(1.72)	54266(3.47)
	30	NS	100	100	100	100	100
		meanNF(%SD)	33712(6.66)	53759(2.44)	122277(1.81)	114079(1.52)	198059(1.48)
	50	NS	100	100	100	100	100
		meanNF(%SD)	56498(3.67)	87137(3.90)	174414(2.19)	165118(1.52)	314970(7.85)
F_5	10	NS	100	100	100	100	100
		meanNF(%SD)	12460(5.59)	24755(4.07)	36493(3.29)	44271(3.47)	47759(1.95)
	30	NS	100	97	97	100	100
		meanNF(%SD)	56188(4.13)	73422(5.32)	150794(4.22)	123334(2.68)	143686(1.30)
	50	NS	100	57	49	100	100
		meanNF(%SD)	148488(4.75)	133396(5.165)	290445(5.28)	193196(2.80)	236536(1.4)
F_6	10	NS	100	100	100	100	49
		meanNF(%SD)	14197(6.54)	20601(3.93)	28871(3.29)	32980(2.81)	43536(4.79)
	30	NS	100	100	99	100	55
		meanNF(%SD)	49699(3.46)	53811(3.82)	92755(3.56)	84608(2.64)	138990(1.71)
	50	NS	100	91	100	100	16
		meanNF(%SD)	87009(2.22)	97754(3.76)	163292(4.12)	126238(2.29)	227293(2.41)
F_7	10	NS	100	100	100	100	100
		meanNF(%SD)	11002(6.84)	20237(2.00)	40587(2.00)	47096(1.50)	40421(3.12)
	30	NS	100	89	100	100	100
		meanNF(%SD)	40029(3.77)	68850(1.32)	98994(1.90)	106889(1.67)	55491(2.99)
	50	NS	100	12	100	100	100
		meanNF(%SD)	76032(2.61)	324766(1.08)	138260(1.75)	149720(1.41)	67224(2.06)
F_8	10	NS	100	100	100	100	100
		meanNF(%SD)	15243(5.22)	36324(15.63)	59423(6.46)	65944(10.15)	86969(5.53)
	30	NS	100	67	100	100	100
		meanNF(%SD)	51775(2.66)	32029(5.01)	65511(7.48)	70432(3.23)	41502(28.77)
	50	NS	100	52	95	100	8
		meanNF(%SD)	89138(1.60)	56207(3.84)	88634(3.51)	97001(2.43)	43912(2.36)
NO. 100 successful runs cases			24/24	12/24	16/24	21/24	17/24
NO. lowest meanNF cases			18/24	0/24	1/24	0/24	5/24

5.3. Performance comparison of SDE-FMP with other adaptive DE algorithms using maxnf on the CEC 2005 benchmark functions

We compare the performance of SDE-FMP with SaDE, FDDE_F, and CUSDE using the mean of obtained best values. Table 4 shows that the superior cases of SDE-FMP to SaDE, FDDE_F, and CUSDE are 12, 9, and 10, whereas the inferior ones are 5, 6, and 5. Therefore, SDE-FMP overall outperforms the compared methods on CEC 2005 benchmark functions.

Table 4. The performance comparison of SDE-FMP, SADE, FDDE_F, and CUSDE on 30-dimensional CEC2005 benchmark functions

f	SDE-FMP		SaDE [15]			FDDE_F [14]			CUSDE [20]		
	MEAN	std.	MEAN		std.	MEAN		std.	MEAN		std.
f1	0.00E+00	0.00E+00	0.00E+00	=	0.00E+00	1.40E-03	+	1.47E-03	6.73E30	+	5.62E-29
f2	1.48E-14	2.52E-14	2.77E-06	+	8.52E-06	5.20E-13	+	3.36E-13	2.82E-08	+	2.28E-08
f3	3.58E+04	3.07E+04	5.33E+05	+	4.34E+05	1.01E+05	+	8.22E+04	2.32E+05	+	1.09E+05
f4	9.77E-01	6.90E+00	1.93E+02	+	3.22E+02	2.00E-03	-	1.63E-03	1.32E-05	-	1.10E-05
f5	6.83E+02	4.43E+02	3.76E+03	+	6.12E+02	1.80E+02	-	1.47E+02	9.05E-05	-	7.68E-05
f6	1.75E-13	6.75E-13	5.28E+01	+	4.15E+01	6.22E-01	+	5.07E+00	5.47E-09	+	2.60E-08
f7	6.84E-03	9.19E-03	1.65E-02	+	1.58E-02	7.20E-02	+	5.87E-02	0.00E+00	-	0.00E+00
f8	2.09E+01	4.09E-02	2.09E+01	=	1.58E-02	2.09E+01	=	1.70E-02	2.09E+01	=	5.53E-02
f9	0.00E+00	0.00E+00	0.00E+00	=	0.00E+00	5.45E+01	+	4.44E+00	5.23E+01	+	2.13E+01
f10	1.08E+02	1.55E+01	4.76E+01	-	1.26E+01	4.66E+01	-	3.80E+00	1.72E+02	-	8.29E+00
f11	2.92E+01	1.32E+00	1.68E+01	-	1.64E+00	2.79E+01	=	2.27E+00	3.73E+01	+	7.16E-01
f12	3.43E+04	5.37E+03	3.44E+03	-	4.42E+03	3.74E+03	-	5.05E+03	3.59E+03	-	4.99E+03
f13	2.83E+00	2.14E-01	3.84E+00	+	2.66E-01	1.67E+00	-	1.36E+00	1.41E+01	+	8.47E-01
f14	1.29E+01	1.74E-01	1.26E+01	=	2.83E-01	1.29E+01	=	1.05E+00	1.30E+01	=	2.12E-01
f15	2.10E+02	2.67E+01	3.85E+02	+	4.42E+01	4.00E+02	+	3.26E+01	3.33E+02	+	1.11E+02
f16	1.57E+02	5.81E+01	8.65E+01	-	5.65E+01	1.13E+02	+	9.21E+00	2.37E+02	+	8.45E+01
f17	1.88E+02	5.11E+01	8.15E+01	-	3.46E+01	2.64E+02	+	2.15E+01	2.36E+02	+	5.98E+01
f18	9.05E+02	1.53E+01	8.73E+02	=	5.44E+01	9.04E+02	=	2.37E+00	9.03E+02	=	1.18E-01
f19	9.03E+02	2.13E+01	8.74E+02	=	5.44E+01	9.05E+02	=	7.37E+00	9.03E+02	=	1.89E-01
f20	9.07E+02	1.35E+00	8.81E+02	+	5.22E+01	5.54E+02	-	4.54E-01	9.03E+02	=	7.99E-01
f21	5.00E+02	1.83E-13	5.45E+02	+	2.15E+02	5.00E+02	=	0.00E+00	5.00E+02	=	1.62E-13
f22	9.12E+02	1.02E+01	9.21E+02	+	2.66E+01	8.72E+02	=	7.10E-01	8.61E+02	=	3.77E+00
f23	5.34E+02	3.98E-04	5.34E+02	=	8.27E-04	5.34E+02	=	4.35E-02	5.34E+02	=	3.94E-04
f24	2.00E+02	8.66E-13	2.00E+02	=	8.54E-14	2.00E+02	=	0.00E+00	2.00E+02	=	2.94E-14
f25	2.11E+02	7.32E-01	2.14E+02	+	2.35E+00	2.10E+02	=	1.72E-01	2.09E+02	=	2.46E-01
	$(+/=/-)$		12/8/5			9/10/6			10/10/5		

6. DISCUSSION

The SDE-FMP algorithm uses the pheromone strategy to adapt the probabilities for selecting subgroups in mutation where a high pheromone indicates a suitable group for the corresponding position in the mutation equation. Then, the mutant vector has more potential to create a better trial vector during the search. The algorithm also uses the pheromone for self-adaptive control parameters F and CR, where the most successful pair of F and CR has more potential to propagate to the next generations. The pheromone resetting is employed to eliminate the dominance and balance the cycle of gathering the pheromone and using the pheromone to improve the search performance. We obtain the suitable resetting periods $r_g = 500$ and $r_p = 300$ for $PheromoneR_k, k = 1, 2, 3$ and $PheromoneFCR$, respectively.

We further investigate the impact of SDE-FMP's features, including the dynamic mutation strategy, self-adaptive control parameters, and pheromone resetting. We compared the performance of the SDE-FMP with the SDE-FMP without the proposed mutation strategy (using basic DE mutation strategy), the SDE-FMP without self-adaptive control parameters (using fixed $F = 0.5$ and $CR = 0.9$ values), and the SDE-FMP without pheromone resetting on the 30-dimensional Rosenbrock function for 30 independent runs. The results presented in Table 5 demonstrate that the SDE-FMP significantly outperforms the SDE-FMP without each feature. The dynamic mutation strategy and self-adaptive control parameters play crucial roles in improving the convergence speed, while the pheromone resetting further enhances the achievement of high-precision solutions.

Figures 1 and 2 illustrate the convergence graphs of SDE-FMP compared with SaDE, CoDE, jDE, and JADE on the Sphere, Griewank, Ackley, and Schwefel functions for 10 and 30 dimensions. They show that SDE-FMP can solve problems of various types faster than the compared algorithms.

Table 5. Comparison of SDE-FMP and SDE-FMP without dynamic mutation strategy, self-adaptive control parameter, and pheromone resetting on the 30-dimensional Rosenbrock function

Algorithm	SDE-FMP	SDE-FMP without dynamic mutation strategy	SDE-FMP without self-adaptive control parameters	SDE-FMP without pheromone resetting
NS	30	4	8	23
meanNF(%SD)	207681.97(9.52)	412309.50(35.91)	487927.25(20.31)	257962.26(43.18)

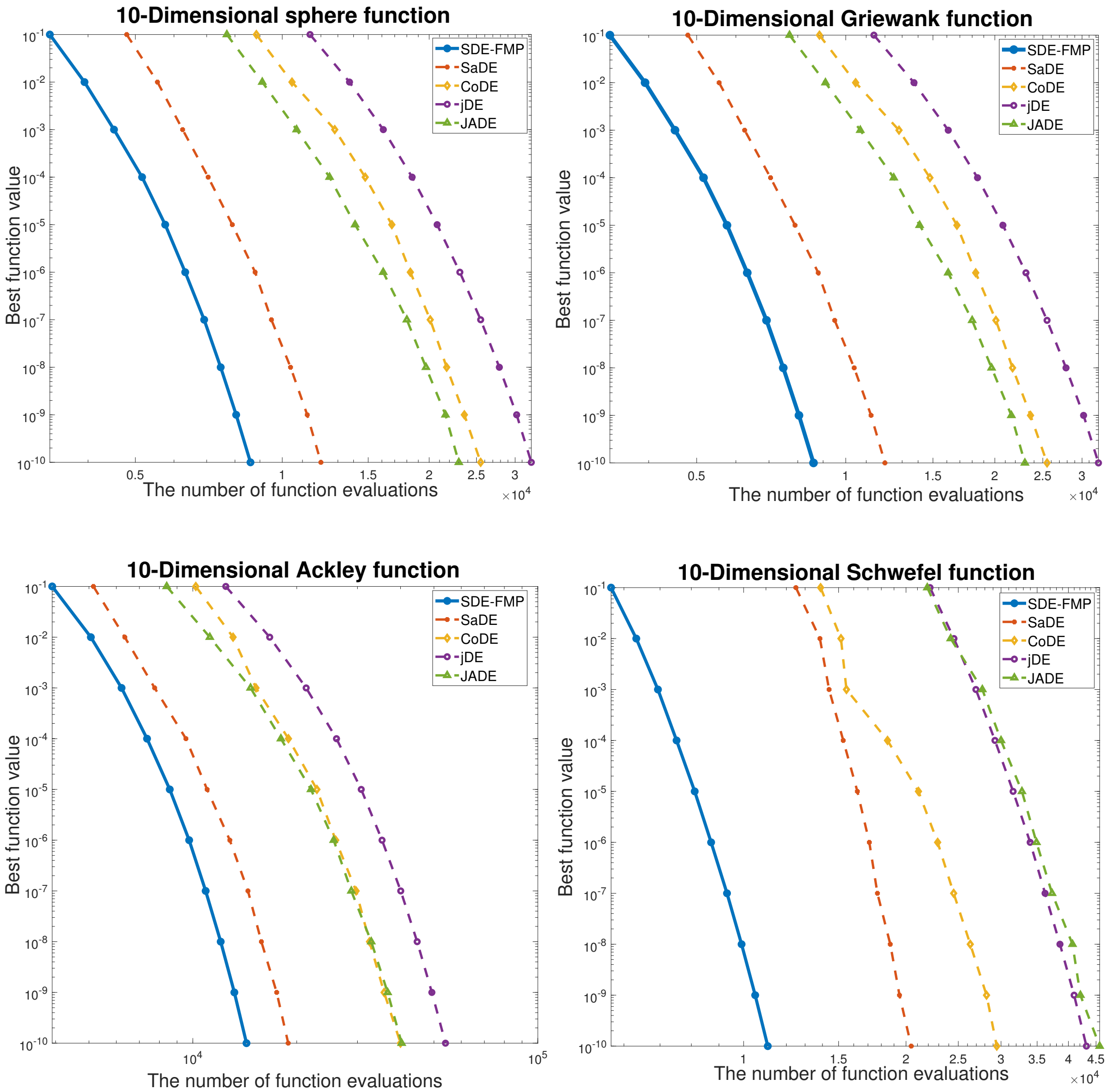

Figure 1. Convergence graphs of SDE-FMP, SaDE, CoDE, jDE, and JADE on 10-dimensional sphere, Griewank, Ackley, and Schwefel functions

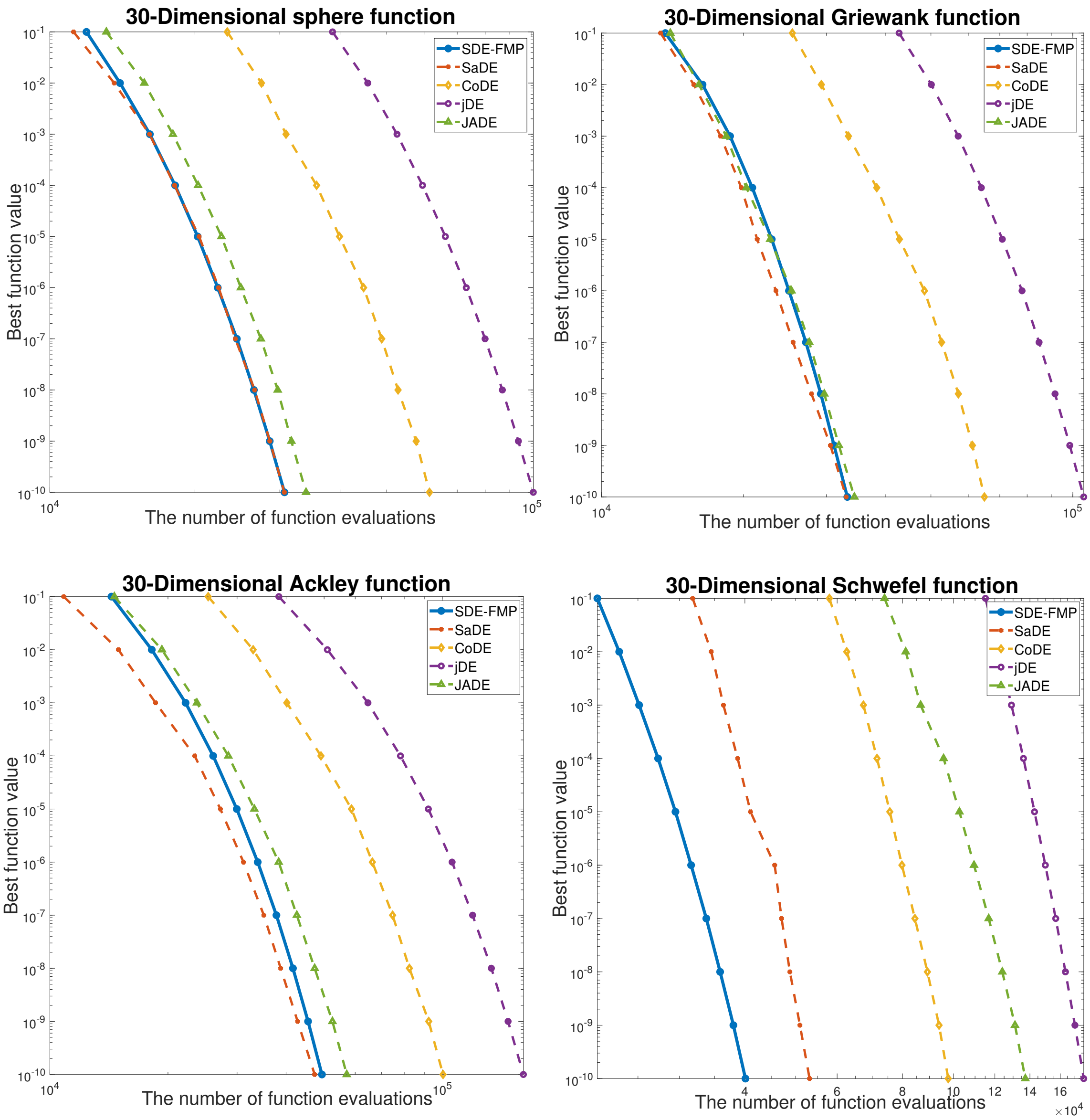

Figure 2. Convergence graphs of SDE-FMP, SaDE, CoDE, jDE, and JADE on 30-dimensional sphere, Griewank, Ackley and Schwefel functions

7. CONCLUSION

This paper presents a self-adaptive differential evolution algorithm with dynamic fitness-ranking mutation and pheromone strategy called SDE-FMP. The pheromone strategy manages the adaptive probabilities for a dynamic mutation and self-adaptive control parameters, and the resetting operation helps the search to prevent premature convergence and stagnation. As a result, the proposed algorithm can solve problems of various types. Experiments indicate that SDE-FMP gives high-precision solutions and outperforms the compared methods on benchmark functions.

ACKNOWLEDGEMENT

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] W. Aribowo and B. S. Supari, "A hunger game search algorithm for economic load dispatch," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 2, pp. 632–640, 2022, doi: 10.11591/ijai.v11.i2.pp632-640.

[2] B.Ouacha, H. Bouyghf, M. Nahid, and S. Abenna, "Dea-based on optimization of inductive coupling for powering implantable biomedical devices," *International Journal of Power Electronics and Drive System (IJPEDS)*, vol. 13, no. 3, pp. 1558–1567, 2022, doi: 10.11591/ijpeds.v13.i3.pp1558-1567.

[3] A. R. Khaparde, R. P. Sundarasamy, S. Rajendran, A. Ticku, and A. Palanichamy, "Analysis of new differential evolution variants to solve multi-modal problems," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 12, no. 3, pp. 1352–1359, 2023, doi: 10.11591/ijai.v12.i3.pp1352-1359.

[4] A. Hiendro, I. Yusuf, F. Husin, and K. H. Khwee, "Photovoltaic parameters estimation of poly-crystalline and mono-crystalline modules using an improved population dynamic differential evolution algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 5, pp. 4538–4548, 2022, doi: 10.11591/ijece.v12i5.pp4538-4548.

[5] N. S. K. Gandikota, M. H. Hasan, and J. Jaafar, "An adaptive metaheuristic approach for risk-budgeted portfolio optimization," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 12, no. 1, pp. 305–314, 2023, doi: 10.11591/ijai.v12.i1.pp305-314.

[6] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT press, 1992.

[7] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006, doi: 10.1109/MCI.2006.329691.

[8] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of The Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, IEEE, 1995, doi: 10.1109/MHS.1995.494215.

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.

[10] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing*, vol. 8, no. 1, pp. 687–697, 2008, doi: 10.1016/j.asoc.2007.05.007.

[11] R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997, doi: 10.1023/A:1008202821328.

[12] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016, doi: 10.1016/j.swevo.2016.01.004.

[13] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006, doi: 10.1109/TEVC.2006.872133.

[14] J. Cheng, Z. Pan, H. Liang, Z. Gao, and J. Gao, "Differential evolution algorithm with fitness and diversity ranking-based mutation operator," *Swarm and Evolutionary Computation*, vol. 61, p. 100816, 2021, doi: 10.1016/j.swevo.2020.100816.

[15] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2008, doi: 10.1109/TEVC.2008.927706.

[16] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679–1696, 2011, doi: 10.1016/j.asoc.2010.04.024.

[17] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009, doi: 10.1109/TEVC.2009.2014613.

[18] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *2013 IEEE Congress on Evolutionary Computation*, pp. 71–78, IEEE, 2013, doi: 10.1109/CEC.2013.6557555.

[19] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011, doi: 10.1109/TEVC.2010.2087271.

[20] L. Zou, Z. Pan, Z. Gao, and J. Gao, "Improving the search accuracy of differential evolution by using the number of consecutive unsuccessful updates," *Knowledge-Based Systems*, vol. 250, p. 109005, 2022, doi: 10.1016/j.knosys.2022.109005.

[21] C. Blum, "Ant colony optimization: Introduction and recent trends," *Physics of Life Reviews*, vol. 2, no. 4, pp. 353–373, 2005, doi: 10.1016/j.plrev.2005.10.001.

[22] Z. Chen, S. Zhou, and J. Luo, "A robust ant colony optimization for continuous functions," *Expert Systems with Applications*, vol. 81, pp. 309–320, 2017, doi: 10.1016/j.eswa.2017.03.036.

[23] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155–1173, 2008, doi: 10.1016/j.ejor.2006.06.046.

[24] J. Xiao and L. Li, "A hybrid ant colony optimization for continuous domains," *Expert Systems with Applications*, vol. 38, no. 9, pp. 11072–11077, 2011, doi: 10.1016/j.eswa.2011.02.151.

[25] P. Singsathid and J. Wetweerapong, "Solving continuous optimization problems by ant colony optimization with domain partitioning technique," in *Proceedings of the 23^{rd} Annual Meeting in Mathematics (AMM2018)*, pp. 257–262, 2018.

[26] P. N. Suganthan *et al.*, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," *KanGAL report*, vol. 2005005, no. 2005, p. 2005, 2005.

BIOGRAPHIES OF AUTHORS

Pirapong Singsathid completed his M.Sc. degree in Applied Mathematics from Khon Kaen University, Thailand in 2018. He is a Ph.D. student in Applied Mathematics, Khon Kaen University. His research area is optimization. He can be contacted at email: pirapongs@kkumail.com.

Jeerayut Wetweerapong completed his M.Sc. degree in Mathematics from West Virginia University, US in 1995 and Ph.D. degree in Mathematics from Khon Kaen University, Thailand in 2012. He is an assistant professor at Department of Mathematics, Khon Kaen University. He has been doing research in field of scientific computing and optimization. He can be contacted at email: wjeera@kku.ac.th.

Pikul Puphasuk completed her M.Sc. degree in Mathematics from Khon Kaen University, Thailand in 2002 and Ph.D. degree in Applied Mathematics from Suranaree University of Technology, Thailand in 2009. She is an associate professor at Department of Mathematics, Khon Kaen University. Her research areas include computational sciences, numerical analysis, and optimization. She can be contacted at email: ppikul@kku.ac.th.