Bulletin of Electrical Engineering and Informatics
Vol. 13, No. 5, October 2024, pp. 3503∼3514
ISSN: 2302-9285, DOI: 10.11591/eei.v13i5.6417

Differential evolution with adaptive mutation and crossover strategies for nonlinear regression problems

Watchara Wongsa, Pikul Puphasuk, Jeerayut Wetweerapong
Department of Mathematics, Faculty of Science, Khon Kaen University, Khon Kaen, Thailand

Article Info

Article history:

Received Apr 7, 2023
Revised Mar 1, 2024
Accepted Mar 6, 2024

Keywords:

Controlled random search
Differential evolution
Global optimization
Nonlinear regression
Parameter identification

ABSTRACT

This paper presents the differential evolution algorithm with adaptive mutation and crossover strategies (DEAMC) for solving nonlinear regression problems. The DEAMC algorithm adaptively uses two mutation strategies and two ranges of crossover rate. We evaluate its performance on the National Institute of Standards and Technology (NIST) nonlinear-regression benchmark containing many models of varying levels of difficulty and compare it with classic differential evolution (DE), enhanced differential evolution algorithm with an adaptation of switching crossover strategy (DEASC), and controlled random search methods (CRS4HC, CRS4HCe). We also apply the proposed method to solve parameter identification applications and compare it with enhanced chaotic grasshopper optimization algorithms (ECGOA), self-adaptive differential evolution with dynamic mutation and pheromone strategy (SDE-FMP), and JAYA and its variant methods. The experimental results show that DEAMC is more reliable and gives more accurate results than the compared methods.

Corresponding Author:

Jeerayut Wetweerapong
Department of Mathematics, Faculty of Science, Khon Kaen University
Khon Kaen, 40002, Thailand
Email: wjeera@kku.ac.th

1. INTRODUCTION

Nonlinear regression is a statistical tool used in data analysis [1], [2], modeling [3], [4], and forecasting [5]–[8] to fit a nonlinear model for data and approximate the model's parameters. This technique fits a model function:

$$y = f(x, a) \tag{1}$$

to observed data (x_i, y_i) where $i = 1, 2, ..., n$ and $a = (a_1, a_2, ..., a_k)$ is a vector of k parameters. The goal is to find the vector a that minimizes the residual sum of squares.

$$S(a) = \sum_{i=1}^{n} [y_i - f(x_i, a)]^2 \tag{2}$$

The optimization problem is challenging because of large dataset sizes, complicated models, and many model parameters. The objective functions S are nonlinear, non-separable, and multimodal. One can use gradient methods like Gauss-Newton and Levenberg Marquart algorithms to solve the problems. However, the sequences of approximate solutions generated by those methods may diverge or converge to local minimums when starting with unsuitable points. Therefore, researchers have proposed derivative-free global optimization methods using multiple-point search. These methods generate many points for searching different subregions of a problem space and use heuristic strategies to create better solutions. This approach includes controlled random search (CRS) [9], genetic algorithm (GA) [10], particle swarm optimization (PSO) [11], differential evolution (DE) algorithm [12], and artificial bee colony algorithm (ABC) [13].

DE is a popular and effective population-based method that uses mutation, crossover, and selection operations. It is applied to solve many problems such as severity classification of COVID-19 sickness data [14], estimation of software development effort [15], feedback controller design of six pulses three-phase rectifier [16], power transfer efficiency and power delivered to the load [17], and spam detection of short message service [18]. The algorithm can converge to optimal solutions using appropriate control parameters and mutation strategies for a specific task. However, the adaptive approach is required to find suitable parameter values and techniques for solving various optimization problems.

This research focuses on designing the adaptive DE algorithm for solving various nonlinear regression problems. The algorithm adaptively uses two mutation strategies and low and high crossover rates to generate a candidate solution. The main contribution of our work is an efficient algorithm that can solve problems with reliability and provide high-quality solutions.

The remainder of the paper is organized as: section 2 reviews the related works on solving nonlinear regression problems. Section 3 describes the proposed differential evolution with adaptive mutation and crossover strategies (DEAMC). Section 4 presents a preliminary experiment and results. Section 5 compares the performance of DEAMC with those of other methods. Section 6 applies DEAMC to solve real-world parameter identification problems. Section 7 provides insight discussion. Finally, the conclusion is given in section 8.

2. LITERATURE REVIEW

A designed method for nonlinear regression tasks should perform well on several test problems. The National Institute of Standards and Technology (NIST) nonlinear-regression dataset [19] is a well-known benchmark for evaluating its reliability. This section reviews various methods proposed to solve continuous optimization problems and the NIST nonlinear regression benchmark.

2.1. CRS algorithm

Price [9] proposed a population-based method called a CRS search algorithm for solving minimization problems. The algorithm randomly chooses distinct vectors $x_1, x_2, ..., x_{d+1}$ to generate a new vector x by:

$$x = c + (c - x_{d+1}) \tag{3}$$

where c is the centroid of x_1 to x_d, and d is the problem dimension. The vector x replaces the worst population vector when the function value of x is better. The algorithm repeats this process until reaching the stopping condition. The CRS can find the global minimums on several problems. Krivý and Tvrdík [20] improved the CRS algorithm by using a uniform random number to control the amplification of the differential variation $(c - x_{d+1})$ in (3) as (4):

$$x = c - u_\alpha(x_{d+1} - c) \tag{4}$$

where u_α is a uniform random number in $(0, \alpha)$ and α ranges from 4 to 8. They applied the algorithm for estimating parameters of nonlinear regression models on the benchmark problems from the literature and some NIST datasets. The algorithm shows promising results and provides successful parameter estimations. Tvrdík *et al.* [21] enhanced the CRS algorithm by using two heuristics to generate a better solution for solving nonlinear regression problems. Their algorithms, called CRS4HC and CRS4HCe, use the heuristic from the original CRS and DE algorithms. The improved algorithm selects each heuristic according to its success in creating a better solution during the search process. In addition, the authors added an adaptive mechanism for stopping conditions in CRS4HCe. The algorithms give more reliable results on the NIST dataset when compared with the Levenberg Marquardt method.

2.2. Differential evolution algorithm

Storn and Price [12] presented a DE for minimizing continuous functions. DE is a population-based method with three main operations: mutation, crossover, and selection. It has three control parameters: population size, scaling factor, and crossover rate. The algorithm generates the initial population vectors $x_i = [x_{ij}]$ where $i = 1, 2, 3, ..., NP$ and $j = 1, 2, 3, ..., D$ with random real values between lower and upper bounds. It also finds and records the best vector xb and the best value fb. In mutation operation, the mutant vector xm for the target vector x_i is generated by:

$$xm = x_{r_1} + F(x_{r_2} - x_{r_3}) \tag{5}$$

where r_1, r_2, and r_3 are random distinct indices between 1 to NP and also different from i and F is a scaling factor. Next, the crossover operation creates the trial vector xc using the crossover rate CR as (6):

$$xc_j = \begin{cases} xm_j & ; rand_j < CR \text{ or } j = IC \\ x_{ij} & ; \text{otherwise} \end{cases} \tag{6}$$

where $rand_j$ is a uniform random number in $[0, 1]$ for each $j = 1, 2, 3, ..., D$ and IC is a randomly fixed index from 1 to D. Then, the objective functions of the trial vector xc and the target vector x_i are compared in the selection operation. The vector xc replaces x_i when $f(xc) < f(x_i)$. It also updates xb and fb when $f(xc) < fb$. The algorithm repeats mutation, crossover, and selection operations until reaching the stopping condition.

2.3. Adaptive differential evolution algorithms

Since the performance of the classic DE depends on the control parameters F and CR, many adaptive DE algorithms and improvement variants have been proposed. Tvrdík [22] proposed two algorithms, DEBR18 and DEBR18rl, which improve DE algorithms by using nine combinations of F and CR values and two mutation strategies for solving nonlinear regression problems. DEBR18 and DEBR18rl select each heuristic according to its success in creating a better solution during the search process. These two algorithms give reliability results close to the CRS4HCe algorithm on the NIST benchmark. Puphasuk and Wetweerapong [23] improved the DE algorithm using a crossover strategy for solving continuous optimization problems. The algorithm, called DE algorithm with an adaptation of switching crossover strategy (DEASC), uses the scaling factor in the range of $[0.5, 0.7]$ and crossover rates in the ranges of $[0, 0.1]$ and $[0.9, 1]$. The DEASC chooses each crossover range according to its success in creating a better solution in the selection process. Experimental results show that it can solve benchmark problems of various types and difficulties. Cheng *et al.* [24] used the fitness and diversity rankings to assign the ranks for all vectors in a DE algorithm. The rank numbers determine the positions of vectors in the mutation operation. This operator improves the performance of the jDE, SHADE, and L-SHADE algorithms on the CEC 2013 and CEC 2014 benchmarks.

2.4. Other population-based algorithms

Pan *et al.* [25] used a GA for parameter estimation of nonlinear regression models. Test problems include six models from the early literature. The numerical results show that the GA can provide solutions close to those obtained by the compared methods. Tvrdík *et al.* [26] proposed an evolutionary algorithm with eleven heuristic strategies for solving nonlinear regression problems. Each heuristic is selected according to its success during the search process. The algorithm can solve eight nonlinear regression models from NIST datasets. Kapanoglu and Erdoğmuş [27] examined GAs for parameter estimation of twenty NIST nonlinear regression models using seven crossover operators, four crossover probability values, and four mutation probability values. The results show that these factors affect the performance of GAs. Özsoy and Örkçü [28] used the PSO algorithm for solving nonlinear regression problems consisting of fifteen NIST datasets and thirteen other datasets. The PSO can provide solutions close to eight NIST reference solutions with four decimal places. Michailidis [29] proposed JAYA algorithm for parameter estimation in nonlinear regression models. The algorithm uses the best and the worst population vectors to generate a new solution. The algorithm gives higher success rates and accuracy solutions than the PSO algorithm on fourteen NIST datasets. Alkreem and Kalaf [30] solved nonlinear regression problems using a gravitational search algorithm (GSA). The algorithm applies the laws of gravity and motion to create a new solution. Two models of NIST datasets are used to evaluate its performance. The numerical results show that the GSA outperforms the maximum likelihood method.

2.5. Parameter identification applications

Photovoltaic (PV) models are real-world parameter identification problems essential in developing PV devices for generating electricity from solar energy. They represent the systems using mathematical equations with input data and unknown parameters. To obtain the calculated electrical output values that fit the measured data, researchers have proposed algorithms to estimate accurate parameters. Yu *et al.* [31] introduced the PG-JAYA method that improves the JAYA method using self-adaptive evolution and chaotic perturbation strategies. The algorithm outperforms nine compared methods. Zhou *et al.* [32] proposed a dynamic opposite learning strategy (DOLADE) algorithm that improves the JADE method with a dynamic opposite learning strategy. DOLADE provides more accurate solutions than the compared algorithms. Singsathid *et al.* [33] presented a self-adaptive DE algorithm called self-adaptive differential evolution with dynamic mutation and pheromone strategy (SDE-FMP). The method uses adaptive probabilities for mutation, self-adaptive control parameters, and the resetting operation. The SDE-FMP outperforms several compared methods.

3. DIFFERENTIAL EVOLUTION ALGORITHM WITH ADAPTIVE MUTATION AND CROSSOVER STRATEGIES

We present an improved DEAMC for solving nonlinear regression problems. The method uses two mutation strategies: the classic mutation (CM) and sorting mutation (SM). It also uses low and high ranges of CR values: $[0, 0.1]$ and $[0.9, 1]$. DEAMC chooses each mutation strategy and CR value according to their success in the selection. The description of the proposed algorithm is as:

(1) Inputs and control parameters:
Objective function to be minimized: f
Problem dimension: D
Lower and upper bounds for each problem dimension: L_j, U_j
Population size: $NP = 10D$
Stopping condition: $\log(\frac{fw}{fb}) < \epsilon$ where fw and fb are the worst and best values, and $\epsilon = 10^{-10}$
Maximum number of function evaluations: $maxnf = 40000D$
Scaling factor: F in the range of $[0.5, 0.7]$
Crossover rate: CR in the range of $[0, 0.1]$ or $[0.9, 1]$
The initial probabilities for using CM and SM: $pm1 = pm2 = 0.5$
The initial probabilities for using low and high crossover rates: $pc1 = pc2 = 0.5$
The initial counters corresponding to $pm1$ and $pm2$: $nm1 = nm2 = 0$
The initial counters corresponding to $pc1$ and $pc2$: $nc1 = nc2 = 0$

(2) Initialization:
Generate the initial population vectors $x_i = [x_{ij}]$ where $i = 1, 2, 3, ..., NP$ and $j = 1, 2, 3, ..., D$. Each component x_{ij} is a random real value between L_j and U_j. Find and record the best vector xb and the best value fb.

(3) Mutation:
Generate a uniform random number u_1 in $[0, 1]$ and random a scaling factor F in $[0.5, 0.7]$. If $u_1 < pm1$, create a mutant vector by using the CM as (7):

$$xm = x_{r_1} + F(x_{r_2} - x_{r_3}). \tag{7}$$

Otherwise, create a mutant vector by using the SM as (8):

$$xm = x^*_{r_1} + F(x^*_{r_2} - x^*_{r_3}) \tag{8}$$

where $f(x^*_{r_1}) \leq f(x^*_{r_2}) \leq f(x^*_{r_3})$. The $r_1, r_2,$ and r_3 numbers are different random indices from 1 to NP and also different from the index i of the target vector. The $x^*_{r_1}, x^*_{r_2}$, and $x^*_{r_3}$ are sorted vectors of x_{r_1}, x_{r_2}, and x_{r_3} by their function values.

(4) Crossover:
Generate a uniform random number u_2 in $[0, 1]$. If $u_2 < pc1$, random a crossover rate CR in $[0, 0.1]$; otherwise, random CR in $[0.9, 1]$. Create the trial vector xc using the crossover rate CR as:

$$xc_j = \begin{cases} xm_j & ; rand_j < CR \text{ or } j = IC \\ x_{ij} & ; \text{otherwise} \end{cases}$$

where $rand_j$ is a uniform random number in $[0, 1]$ for each $j = 1, 2, 3, ..., D$ and IC is a randomly fixed index from 1 to D.

(5) Selection:
Compare the objective function values of the trial vector xc and the target vector x_i. The vector xc replaces x_i when $f(xc) < f(x_i)$. In addition, update xb with xc and update fb with $f(xc)$ when $f(xc) < fb$.

(6) Updating control parameters:
(6.1) update $pm1$ and $pm2$ as follows:
(6.1.1) if a better solution in the selection process is created by the CM, then increase $nm1 := nm1 + 1$; otherwise, increase $nm2 := nm2 + 1$.
(6.1.2) if $nm1 + nm2 \geq 100$, then adjust the counters $nm1 := nm1 + 10$ and $nm2 := nm2 + 10$ to prevent both of them from 0.
(6.1.3) update the probabilities using the weighted formula:

$$pm1 = (0.9)pm1 + \frac{(0.1)nm1}{nm1+nm2},\ pm2 = 1 - pm1$$

(6.1.4) reset the counters $nm1, nm2 = 0$ when the probabilities $pm1$ and $pm2$ are updated.
(6.2) update $pc1, pc2$ and $nc1, nc2$ based on obtaining a better solution using a low or high crossover rate (in the same manner as in 6.1.1-6.1.4).

(7) Stopping condition:
Find the worst function value fw for computing the stopping condition. If the algorithm reaches the stopping condition or $maxnf$, report fb and its accuracy; otherwise, repeat all steps (3) to (6).

4. PRELIMINARY EXPERIMENT AND RESULTS

In this section, we conducted a preliminary experiment to compare the performances of the DEASC algorithms using three different mutations: CM, SM, and adaptive mutation (AM). Each algorithm is tested on selected NIST benchmark problems [19]. They consist of 27 real-world and generated nonlinear regression problems. The reported solution for each problem is the certified value with 11 decimal places. Let fb be the best value obtained from an algorithm and c be the certified value for a problem. The following equation calculates the accuracy λ of fb:

$$\lambda = \begin{cases} 0 & ;\ \frac{|fb-c|}{c} \geq 1 \\ 11 & ;\ \frac{|fb-c|}{c} < 10^{-11} \\ -\log(\frac{|fb-c|}{c}) & ;\ \text{otherwise} \end{cases} \quad (9)$$

The λ value indicates the number of decimal places that fb matches the certified value.

4.1. Effect of CM, SM, and AM on the performance of the DEASC algorithm

We studied the effect of using three different mutations on the DEASC performance: CM, SM, and AM. The AM chooses CMs or SMs according to their success in creating a better solution in the selection process. The experiment uses the population size $NP = 10D$ and stopping condition $\log(\frac{fw}{fb}) < 10^{-10}$ or $maxnf = 40000D$. Each algorithm is run 100 times for each problem. If an algorithm reaches the threshold before exceeding the $maxnf$ and gives $\lambda > 4$, then the successful run, the accuracy λ, and the number of function evaluations nf are recorded. We report the number of successful runs (NS), the mean of the number of function evaluations (mean nf), and the mean of λ value (mean λ).

Table 1 shows that the DEASC with CM, SM, and AM give NS=100 for 6, 7, and 8 out of 8 problems, respectively. The three algorithms obtain almost equally high mean λ values. The algorithm with CM gives lower mean λ on the bennett5 problem. Overall, the mean nf values obtained by AM are less than those obtained by CM but higher than those obtained by SM. These results show that the AM strategy can prevent premature convergence and give good convergence speeds. From the results of this experiment, we obtain the proposed method DEAMC as the DEASC with AM.

Table 1. Performance comparison of DEASC with CM, SM, and AMs

Problem	CM			SM			AM		
	NS	Mean nf	Mean λ	NS	Mean nf	Mean λ	NS	Mean nf	Mean λ
Gauss1	**100**	36,901	10.7	**100**	24,115	10.7	**100**	28,562	10.7
Kirby2	**100**	19,232	10.9	**100**	14,107	11.0	**100**	15,537	11.0
Hahn1	**100**	40,730	10.9	96	26,462	10.9	**100**	30,982	10.9
Enso	**100**	61,281	10.7	**100**	39,525	10.6	**100**	47,462	10.6
Mgh10	4	67,008	11.0	**100**	33,117	11.0	**100**	39,696	11.0
Eckerle4	**100**	3,454	10.7	**100**	2,658	10.7	**100**	2,995	10.7
Bennett5	3	50,192	9.9	**100**	70,918	11.0	**100**	81,797	11.0
Thurber	**100**	34,103	10.8	**100**	21,890	10.9	**100**	25,751	10.9

5. COMPARISON EXPERIMENTS AND RESULTS

5.1. Performance comparison of DEAMC, DE0509, and DEASC

The DEAMC algorithm uses $NP = 10D$, F in $[0.5, 0.7]$, the AM strategy, and adaptive crossover rates CR in $[0, 0.1]$ and $[0.9, 1]$ as described in section 3. This experiment compares DEAMC with DEASC and classic DE algorithm DE0509 using $F = 0.5$ and $CR = 0.9$ [12]. The compared algorithms use the same setting of $NP = 10D$ and the same stopping condition of DEAMC. Each algorithm performs 100 independent runs. The best results are determined by three conditions: NS=100, the maximum mean λ, and the minimum mean nf.

Table 2 shows the performance comparison of DEAMC, DEASC, and DE0509. The best values are highlighted in boldface. The results show that the DEAMC, DEASC, and DE0509 give NS=100 for 26, 20, and 11 out of 27 problems, respectively. Although the DEAMC can not solve the lanczos1 problem successfully, the mean value of the solutions obtained by the DEAMC is 1.315803×10^{-20}, which is close to the reported solution at 1.430787×10^{-25}. The DEAMC, DEASC, and DE0509 give the best results for 22, 1, and 3 out of 27 problems, respectively. It indicates that the DEAMC significantly outperforms the original DEASC and the classic DE.

Table 2. Performance comparison of the DEAMC, classic DE0509, and DEASC algorithms

Problem	DEAMC			DE0509			DEASC		
	NS	Mean nf	Mean λ	NS	Mean nf	Mean λ	NS	Mean nf	Mean λ
Chwirut1	**100**	**5,328**	**11.0**	100	3,809	10.9	100	6,063	11.0
Chwirut2	100	5,272	11.0	**100**	**3,835**	**11.0**	100	6,119	11.0
Danwood	**100**	**2,062**	**11.0**	85	1,360	10.9	100	2,441	11.0
Gauss1	**100**	**28,562**	10.7	100	21,292	10.6	100	37,038	10.7
Gauss2	100	30,592	10.3	**100**	**22,530**	**10.3**	100	40,308	10.3
Lanczos3	100	68,101	10.8	5	290,892	7.3	**100**	**151,481**	**10.9**
Misra1a	**100**	**2,892**	**10.4**	0	5,650	0.0	87	10,120	10.4
Misra1b	**100**	**2,908**	**11.0**	4	1,844	11.0	83	4,994	11.0
Enso	**100**	**47,462**	**10.6**	100	28,920	10.5	100	61,599	10.6
Gauss3	100	34,807	10.5	**100**	**21,264**	**10.6**	100	47,464	10.5
Hahn1	**100**	**28,076**	**10.9**	98	31,472	10.9	100	38,593	10.9
Kirby2	**100**	**15,537**	**11.0**	100	12,574	10.9	100	19,207	10.9
Lanczos1	0	240,000	0.0	0	240,000	0.0	0	240,000	0.0
Lanczos2	**100**	**67,846**	**9.9**	6	266,113	9.9	100	145,116	9.9
Mgh17	**100**	**18,260**	**11.0**	98	102,696	10.5	100	29,084	10.9
Misra1c	**100**	**3,571**	**11.0**	0	5,291	0.0	77	16,743	11.0
Misra1d	**100**	**3,400**	**11.0**	0	5868	0	79	12,499	11.0
Nelson	**100**	**10,605**	**10.8**	0	16,923	0.0	100	24,402	10.8
Roszman1	**100**	**6,734**	**11.0**	100	5,123	10.9	100	8,312	11.0
Bennett5	**100**	**81,797**	**11.0**	0	108,972	0.7	3	50,192	9.9
Boxbod	**100**	**1,333**	**10.4**	99	917	10.5	100	1,517	10.4
Eckerle4	**100**	**2,995**	**10.7**	100	2,381	10.6	100	3,449	10.7
Mgh09	**100**	**15,617**	**11.0**	10	17,377	10.9	100	25,148	11.0
Mgh10	**100**	**39,696**	**11.0**	0	44,609	0.0	4	67,008	11.0
Rat42	**100**	**4,002**	**11.0**	97	2,774	10.9	100	4,843	11.0
Rat43	**100**	**6,228**	**11.0**	100	4,667	10.9	100	7,742	11.0
Thurber	**100**	**25,751**	**10.9**	100	25,734	10.7	100	34,179	10.8

5.2. Performance comparison of DEAMC, CRS4HC, and CRS4HCe

This section compares the performance of DEAMC with CRS4HC and CRS4HCe. The setting and the results of CRS4HC and CRS4HCe are as in the original paper [21]. Those of DEAMC are the same as in section 5.1. Each algorithm performs 100 independent runs.

Table 3 shows that DEAMC, CRS4HC, and CRS4HCe give NS=100 for 26, 22, and 23 out of 27 problems, respectively. Thus, DEAMC is outstanding in providing NS=100 for all cases except the lanczos1 problem. The DEAMC and CRS4HC provide almost equally high mean λ values for 20 cases, and DEAMC gives significantly higher mean λ values for 6 cases. The CRS4HCe shows lower mean λ values for almost all problems. Thus, DEAMC is also outstanding in terms of providing overall high-quality solutions. However, DEAMC requires higher mean nf values than CRS4HC and CRS4HCe. These results show that the DEAMC is more reliable and accurate than the compared methods with the expense of slightly more function evaluations.

Table 3. Performance comparison of the DEAMC algorithm and two variants of the adaptive CRS algorithm

Problem	DEAMC			CRS4HC			CRS4HCe		
	NS	Mean nf	Mean λ	NS	Mean nf	Mean λ	NS	Mean nf	Mean λ
Chwirut1	100	5,328	11.0	**100**	**3,008**	**11.0**	100	1,955	8.5
Chwirut2	100	5,272	11.0	**100**	**2,987**	**11.0**	100	1,942	8.3
Danwood	100	2,062	11.0	**100**	**1,620**	**11.0**	100	1,166	8.4
Gauss1	100	28,562	10.7	**100**	**14,137**	**11.0**	100	9,189	7.1
Gauss2	**100**	**30,592**	**10.3**	98	14,726	10.4	98	9,425	7.0
Lanczos3	**100**	**68,101**	**10.8**	100	29,810	6.9	100	30,406	7.0
Misra1a	100	2,892	10.4	**100**	**2,157**	**10.4**	100	1,790	7.8
Misra1b	**100**	**2,908**	**11.0**	100	1,861	10.9	100	1,508	9.0
Enso	**100**	**47,462**	**10.6**	87	19,220	9.7	86	13,454	8.1
Gauss3	100	34,807	10.5	**100**	**15,908**	**11.0**	99	10,340	7.0
Hahn1	**100**	**28,076**	**10.9**	93	16,509	9.9	93	12,217	6.5
Kirby2	100	15,537	11.0	**100**	**8,508**	**11.0**	100	6,551	7.3
Lanczos1	0	240,000	0.0	0	28,361	0.0	0	209,587	2.5
Lanczos2	**100**	**67,846**	**9.9**	55	28,251	4.1	100	30,511	7.1
Mgh17	100	18,260	11.0	**100**	**11,023**	**11.0**	100	9,039	7.5
Misra1c	100	3,571	11.0	**100**	**2,104**	**11.0**	100	1,873	8.3
Misra1d	100	3,400	11.0	**100**	**2,043**	**11.0**	100	1,798	8.4
Nelson	100	10,605	10.8	**100**	**5,904**	**10.9**	100	4,900	9.0
Roszman1	100	6,734	11.0	**100**	**5,301**	**11.0**	100	3,393	7.2
Bennett5	100	81,797	11.0	**100**	**41,335**	**11.0**	100	36,788	5.1
Boxbod	100	1,333	10.4	**100**	**1,308**	**10.4**	100	824	9.7
Eckerle4	100	2,995	10.7	**100**	**2,629**	**10.7**	100	1,709	7.6
Mgh09	100	15,617	11.0	**100**	**10,422**	**11.0**	100	8,859	7.7
Mgh10	**100**	**39,696**	**11.0**	100	20,761	9.0	100	20,969	8.1
Rat42	100	4,002	11.0	**100**	**2,942**	**11.0**	100	1,912	7.4
Rat43	100	6,228	11.0	**100**	**4,807**	**11.0**	100	2,932	7.9
Thurber	100	25,751	10.9	**100**	**13,915**	**11.0**	100	9,741	7.4

6. PERFORMANCE OF DEAMC ON PARAMETER IDENTIFICATION APPLICATIONS

In this section, we apply the DEAMC algorithm for solving parameter identifications on one frequency-modulated (FM) synthesizer model and three PV models and compare DEAMC with the methods in the literature.

Problem 1: the FM synthesizer is a part of FM sound wave synthesis [34], [35]. Its parameter estimation is a six-dimensional optimization problem. The equations of the estimated sound and target sound are as (10) and (11):

$$f(t,a) = a_1\sin(\omega_1 t\theta + a_2\sin(\omega_2 t\theta + a_3 t\theta\sin(\omega_3 t\theta))) \tag{10}$$

$$y_t = (1.0)\sin((5.0)t\theta + (1.5)\sin((4.8)t\theta + (2.0)t\theta\sin((4.9)t\theta))) \tag{11}$$

respectively where $a = (a_1, \omega_1, a_2, \omega_2, a_3, \omega_3)$ is a vector of parameters estimated in the range $[-6.4, 6.35]$ and $\theta = \frac{2\pi}{100}$. The objective function S is the residual sum of squares (RSS) defined by the following:

$$S(a) = \sum_{t=0}^{100}(y_t - f(t,a))^2$$

Problem 2: parameter estimation of PV models consists of three models: single diode, double diode, and PV module models [31]. The datasets, lower and upper bounds, and constant parameters are as in the original paper. In all models, V_L is cell output voltage, I_L is cell output current, and q, K, T are constant parameters.

The equations of these models are the following:

– (2a) the single diode model:

$$f(V_L, I_L, a) = I_{ph} - I_{sd}[e^{(\frac{q(V_L+R_sI_L)}{nkT})} - 1] - \frac{V_L + R_sI_L}{R_{sh}} - I_L \tag{12}$$

where $a = (I_{ph}, I_{sd}, R_s, R_{sh}, n)$ is a vector of parameters to be estimated.

– (2b) the double diode model:

$$f(V_L, I_L, a) = I_{ph} - I_{sd1}[e^{(\frac{q(V_L+R_sI_L)}{n_1kT})} - 1] - I_{sd2}[e^{(\frac{q(V_L+R_sI_L)}{n_2kT})} - 1] - \frac{V_L + R_sI_L}{R_{sh}} - I_L \tag{13}$$

where $a = (I_{ph}, I_{sd1}, I_{sd2}, R_s, R_{sh}, n_1, n_2)$ is a vector of parameters to be estimated.

– (2c) the PV module model:

$$f(V_L, I_L, a) = I_{ph} - I_{sd}[e^{(\frac{q(V_L/N_s + R_s I_L/N_p)}{nkT})} - 1] - \frac{V_L/N_s + R_s I_L/N_p}{R_{sh}} - I_L \tag{14}$$

where $a = (I_{ph}, I_{sd}, R_s, R_{sh}, n)$ is a vector of parameters to be estimated and N_p, N_s are constants.

The objective function for this problem is the root mean square error (RMSE)

$$S(a) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} f(V_L(i), I_L(i), a)^2}$$

where N is the number of experimental data. The dataset for models (2a) and (2b) contains 25 $(V_L(i), I_L(i))$ data, while the dataset for model (2c) contains 26 data.

In the first experiment, DEAMC performs 30 independent runs for each problem using the setting in section 3 without fixing the maximum number of function evaluations to obtain the best solutions. Table 4 shows the best solutions obtained by DEAMC for problem 1. The method gives ten different solutions in 30 runs. The obtained solutions are the exact solution since the objective function values are equal to zero. Table 5 presents the best solutions obtained by DEAMC for problem 2. Our method gives I_{sd}, I_{sd1}, and I_{sd2} parameters value close to zero.

Table 4. Different best solutions obtained by the DEAMC algorithm for the FM synthesizer model

	a_1	ω_1	a_2	ω_2	a_3	ω_3	RSS
Solution 1	1.0	5.0	1.5	-4.8	-2.0	4.9	0
Solution 2	-1.0	-5.0	1.5	4.8	-2.0	-4.9	0
Solution 3	1.0	5.0	-1.5	4.8	-2.0	-4.9	0
Solution 4	1.0	5.0	-1.5	4.8	2.0	4.9	0
Solution 5	1.0	5.0	1.5	-4.8	2.0	-4.9	0
Solution 6	-1.0	-5.0	-1.5	-4.8	-2.0	4.9	0
Solution 7	-1.0	-5.0	-1.5	-4.8	2.0	-4.9	0
Solution 8	1.0	5.0	1.5	-4.8	2.0	-4.9	0
Solution 9	1.0	5.0	-1.5	4.8	2.0	4.9	0
Solution 10	-1.0	-5.0	1.5	4.8	2.0	4.9	0

Table 5. The best solutions obtained by the DEAMC algorithm for three PV models

Parameter	Single diode model	Double diode model	PV module model
I_{ph}	0.7608	0.7608	1.03051
I_{sd}	3.23E-7	-	3.4823E-6
I_{sd1}	-	7.494E-7	-
I_{sd2}	-	0.226E-6	-
R_s	0.0364	0.03674	0.8342
R_{sh}	53.7185	55.4855	27.2772
n	1.4812	-	1.3512
n_1	-	1.9999	-
n_2	-	1.4510	-
RMSE	9.8602E-4	9.8248E-4	2.4250E-3

Next, in the second experiment, we compare DEAMC with the enhanced chaotic grasshopper optimization algorithms (ECGOA) [35], SDE-FMP [33], and JAYA [31] methods using $maxnf = 30000$ and $maxnf = 50000$ as in the original papers over 30 independent runs for problems 1 and 2, respectively. Table 6 shows the performance comparison of DEAMC using $NP = 5D$ (DEAMC-5D) and $NP = 10D$ (DEAMC-10D) with ten variants of the ECGOA method. We report the values less than 10^{-10} as zero. Our methods give min, mean, max, and standard deviation (SD) values of fb less than those of ECGOA methods for the FM synthesizer model. Table 7 compares the performances of DEAMC methods with three variants of JAYA and SDE-FMP methods. The min, mean, max, and SD values of fb obtained by DEAMC are less than those of JAYA methods for all PV models. DEAMC methods give the same results as SDE-FMP on single and PV module models, but DEAMC-5D gives a better result on the double diode model.

Table 6. Performance comparison of DEAMC with ten variants of ECGOA on FM synthesizer model

Algorithm	Min	Mean	Max	SD
DEAMC-5D	**0**	**2.439**	**12.535**	**4.805**
DEAMC-10D	0	7.972	15.571	3.876
ECGOA1	14.048	20.748	26.720	2.754
ECGOA2	8.416	20.266	25.560	4.067
ECGOA3	11.407	20.652	27.283	4.335
ECGOA4	1.4E-7	19.863	27.280	5.439
ECGOA5	8.416	20.708	27.354	4.771
ECGOA6	8.416	19.687	26.522	5.183
ECGOA7	10.177	18.830	25.913	4.596
ECGOA8	0	19.108	26.999	5.710
ECGOA9	11.549	20.896	27.123	4.064
ECGOA10	13.393	21.266	27.462	3.890

Table 7. Performance comparison of DEAMC with three variants of JAYA and SDE-FMP methods on three PV models

Model	Algorithm	Min	Mean	Max	SD
Single diode	DEAMC-5D	**9.8602E-4**	**9.8602E-4**	**9.8602E-4**	**3.4172E-17**
	DEAMC-10D	**9.8602E-4**	**9.8602E-4**	**9.8602E-4**	**2.8722E-17**
	SDE-FMP	**9.8602E-4**	**9.8602E-4**	**9.8602E-4**	**2.0997E-17**
	PGJAYA	9.8602E-4	9.8602E-4	9.8603E-4	1.4485E-9
	IJAYA	9.8603E-4	9.9204E-4	1.0622E-3	1.4033E-5
	JAYA	9.8946E-4	1.1617E-3	1.4783E-3	1.8796E-4
Double diode	DEAMC-5D	**9.8248E-4**	**9.8444E-4**	**9.8602E-4**	**1.6701E-6**
	DEAMC-10D	9.8296E-4	9.8666E-4	1.0048E-3	3.967E-6
	SDE-FMP	9.8248E-4	9.8497E-4	9.8735E-4	1.5213E-6
	PGJAYA	9.8263E-4	9.8582E-4	9.9499E-4	2.5375E-6
	IJAYA	9.8293E-4	1.0269E-3	1.4055E-3	9.8325E-5
	JAYA	9.8934E-4	1.1767E-3	1.4793E-3	1.9356E-4
PV module	DEAMC-5D	**2.425075E-3**	**2.425075E-3**	**2.425075E-3**	**1.758857E-17**
	DEAMC-10D	**2.425075E-3**	**2.425075E-3**	**2.425075E-3**	**1.994769E-17**
	SDE-FMP	**2.425075E-3**	**2.425075E-3**	**2.425075E-3**	**2.446400E-17**
	PGJAYA	2.425075E-3	2.425144E-3	2.426764E-3	3.071420E-6
	IJAYA	2.425129E-3	2.428855E-3	2.439269E-3	3.775523E-6
	JAYA	2.427785E-3	2.453710E-3	2.595873E-3	3.456290E-5

7. DISCUSSION

Section 5.1 shows the performance comparison of DEAMC, DE0509, and DEASC. The DE0509 is suitable for easy problems since it uses fixed F and CR values. DEASC can solve problems at various difficulty levels. Thus, using adjustable CR values and F values in the ranges can increase the solving ability of the DE algorithm. However, DEASC requires high mean nf values and can not solve some problems. By integrating SM adaptively, DEAMC can solve almost all NIST problems and give the best results for many cases. Moreover, DEAMC can prevent premature convergence and provides a faster convergence speed than DEASC, as shown in Figure 1.

Section 5.2 performs the comparison of DEAMC and two variants of CRS. CRS4HC and CRS4HCe methods can solve many problems of higher difficulty levels, but these two methods do not give NS=100 in many cases of lower and average difficulty levels. The proposed DEAMC algorithm provides the solving ability with NS=100 for almost all cases. In addition, the solutions obtained from DEAMC are as accurate as those certified values in many problems. In section 6, we applied DEAMC to solve parameter identification applications. The FM synthesizer and PV models are challenging optimization problems. DEAMC can solve all cases very well and provide high-quality solutions. Moreover, the DEAMC outperforms the compared methods in accuracy and reliability.

The DEAMC algorithm can find accurate solutions and speed up convergence using SM that creates search directions from worse to better vectors. The algorithm can solve various problem types using CR values in low and high ranges. By incorporating AM and crossover strategies, DEAMC achieves superior performance with robustness over a wide range of nonlinear regression problems.

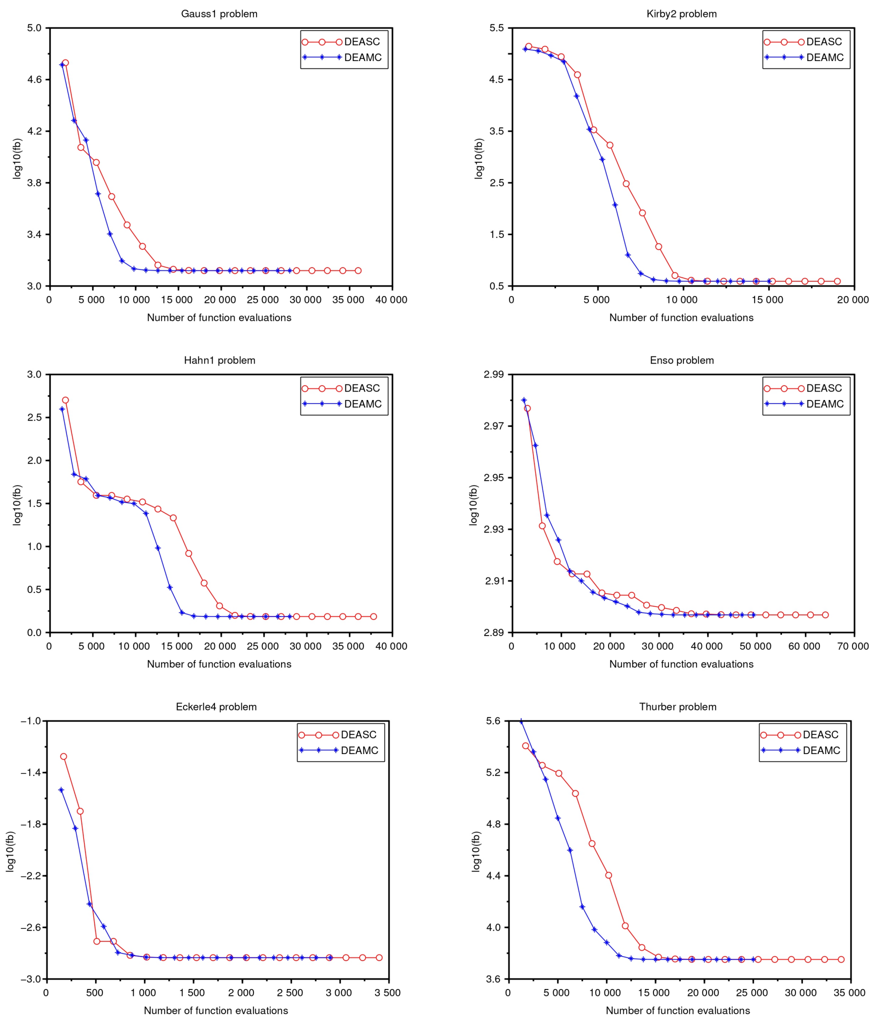

Figure 1. Convergence graphs of DEASC and DEAMC for six problems

8. CONCLUSION

We have presented the differential evolution called DEAMC for solving nonlinear regression problems. The algorithm uses two mutations and crossover rates CR by selecting each mutation strategy and CR value according to their success in the selection process. It balances the local and global searches by adaptively using CR values in low and high ranges and increases convergence speed by integrating sorting and classic mutations. The proposed algorithm is a reliable method that can provide high-quality solutions. It outperforms DE, DEASC, CRS4HC, and CRS4HCe methods on NIST problems. The DEAMC also outperforms ECGOA, SDE-FMP, and JAYA and its variant methods on parameter identification applications.

ACKNOWLEDGEMENT

The research on differential evolution with adaptive mutation and crossover strategies for nonlinear regression problems by Khon Kaen University has received funding support from the National Science Research and Innovation Fund (NSRF).

REFERENCES

[1] C. Lin and F. Yan, "The study on classification and prediction for data mining," in *2015 Seventh International Conference on Measuring Technology and Mechatronics Automation*, Nanchang, China, 2015, pp. 1305-1309, doi: 10.1109/ICMTMA.2015.318.
[2] M. Rastgou, H. Bayat, M. Mansoorizadeh, and A. S. Gregory, "Estimating the soil water retention curve: Comparison of multiple

nonlinear regression approach and random forest data mining technique," *Computers and Electronics in Agriculture*, vol. 174, Jul. 2020, doi: 10.1016/j.compag.2020.105502.

[3] N. Diodato, G. Bellocchi, C. Bertolin, and D. Camuffo, "Mixed nonlinear regression for modelling historical temperatures in Central–Southern Italy," *Theoretical and applied*, vol. 113, pp. 187-196, Oct. 2012, doi: 10.1007/s00704-012-0775-y.

[4] N. Fallah, H. Gu, K. Mohammad, and S. A. Seyyedsalehi, "Nonlinear Poisson regression using neural networks: a simulation study," *Neural Computing and Applications*, vol. 18, pp. 939-943, Jul. 2009, doi: 10.1007/s00521-009-0277-8.

[5] C. W. S. Chen, R. Gerlach, B. B. K. Hwang, and M. McAleer, "Forecasting value-at-risk using nonlinear regression quantiles and the intra-day range," *International Journal of Forecasting*, vol. 28, no. 3, pp. 557-574, Sep. 2012, doi: 10.1016/j.ijforecast.2011.12.004.

[6] W. G. Cobourn, L. Dolcine, M. French, and M. Hubbard, "A comparison of nonlinear regression and neural network models for ground-level ozone forecasting," *Journal of the Air and Waste Managment Association*, vol. 50, no. 11, pp. 1999-2009, Dec. 2011, doi: 10.1080/10473289.2000.10464228.

[7] Z. Ma, H. Zhong, L. Xie, Q. Xia, and C. Kang, "Month ahead average daily electricity price profile forecasting based on a hybrid nonlinear regression and SVM model: an ERCOT case study," *J. Mod. Power Syst. Clean Energy*, vol. 6, no. 2, pp. 281-291, Feb. 2018, doi: 10.1007/s40565-018-0395-3.

[8] A. Yasar, M. Bilgili, and E. Simsek, "Water demand forecasting based on stepwise multiple nonlinear regression analysis," *Arabian Journal for Science and Engineering*, vol. 37, no. 8, pp. 2333-2341, May 2012, doi: 10.1007/s13369-012-0309-z.

[9] W. L. Price, "A controlled random search procedure for global optimization," *The Computer Journal*, vol. 24, no. 4, pp. 367-370, Jan. 1977, doi: 10.1093/comjnl/20.4.367.

[10] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, pp. 65-85, Jun. 1994, doi: 10.1007/BF00175354.

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

[12] 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.

[13] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, pp. 459-71, Apr. 2007, doi: 10.1007/s10898-007-9149-x.

[14] H. Setiadi, A. Swandaru, and T. A. Nugroho, "Design feedback controller of six pulse three phase rectifier based on differential evolution algorithm," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 2, pp. 62-69, May 2020, doi: 10.11591/ijeecs.v22.i2.pp670-677.

[15] A. Karimi and T. J. Gandomani, "Software development effort estimation modeling using a combination of fuzzy-neural network and differential evolution algorithm," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 1, pp. 707-715, Feb. 2021, doi:10.11591/ijece.v11i1.pp707-715.

[16] N. A. M. Aseri *et al.*, "Comparison of meta-heuristic algorithms for fuzzy modelling of COVID-19 illness' severity classification," *IAES International Journal of Artificial Intelligence*, vol. 11, no. 1, pp. 50-64, Mar. 2022, doi:10.11591/ijai.v11.i1.pp50-64.

[17] 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 Systems*, vol. 13, no. 3, pp. 1558-1567, Sep. 2022, doi: 10.11591/ijpeds.v13.i3.pp1558-1567.

[18] S. M. Hameed, "Differential evolution detection models for SMS spam," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 1, pp. 596-601, Feb. 2021, doi: 10.11591/ijece.v11i1.pp596-601.

[19] *Statistical reference datasets*, Nonlinear regression, Information Technology Laboratory of the National Institute of Standards and Technology, 2003, doi: 10.18434/T43G6C.

[20] I. krivý and J. Tvrdík, "The controlled random search algorithm in optimizing regression models," *Computational statistics and data analysis*, vol. 20, no. 2, pp. 229-234, Aug. 1995, doi: 10.1016/0167-9473(95)90127-2.

[21] J. Tvrdík, I. Krivý, and L. Misík, "Adaptive population-based search: application to estimation of nonlinear regression parameters," *Computational statistics and data analysis*, vol. 52, no. 2, pp. 713-724, Oct. 2007, doi: 10.1016/j.csda.2006.10.014.

[22] J. Tvrdík, "Adaptive differential evolution: application to nonlinear regression," in *Proceedings of the International Multiconference on Computer Science and Information Technology*, 2007, pp. 193-202.

[23] P. Puphasuk and J. Wetweerapong, "An enhanced differential evolution algorithm with adaptation of switching crossover strategy for continuous optimization," *Foundations of computing and decision sciences*, vol. 45, no. 2, pp. 97-124, Jun. 2020, doi: 10.2478/fcds-2020-0007.

[24] 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, no. 4, Mar. 2021, doi: 10.1016/j.swevo.2020.100816.

[25] Z. Pan, Y. LishanKang, and G. Liu, "Parameter estimation by genetic algorithms for nonlinear regression," in *Proceedings of International Conference on Optimization Techniques and Applications*, 1995, pp. 946-953.

[26] J. Tvrdík, I. Krivý, and L. Misík, "Evolutionary algorithms with competing heuristics in computational statistics," in *Proceedings in Computational Statistics*, Jan. 2002, pp. 349-354, doi: 10.1007/978-3-642-57489-4.

[27] M. Kapanoglu and Ş. Erdoğmuş, "Genetic algorithms in parameter estimation for nonlinear regression models: an experimental approach," *Journal of Statistical Computation and Simulation*, vol. 77, no. 10, pp. 851-867, Sep. 2007, doi: 10.1080/10629360600688244.

[28] V. S. Özsoy and H. H. Örkçü, "Estimating the parameters of nonlinear regression models through particle swarm optimization," *Gazi University Journal of Science*, vol. 29, no. 1, pp. 187-199, 2016.

[29] P. D. Michailidis, "A preliminary performance study on nonlinear regression models using the jaya optimisation algorithm," *IAENG International Journal of Applied Mathematics*, vol. 48, no. 4, pp. 424-428, 2018.

[30] Z. A. A. Alkreem and B. A. Kalaf, "A new algorithm to estimate the parameters of nonlinear regression," *Journal of Physics: Conference Series*, vol. 1879, 2021, doi: 10.1088/1742-6596/1879/3/032042.

[31] K. Yu, B. Qu, C. Yue, S. Ge, X. Chen, and J. Liang, "A performance-guided JAYA algorithm for parameters identification of PV cell and module," *Applied Energy*, vol. 237, pp. 241-257, Mar. 2019, doi: 10.1016/j.apenergy.2019.01.008.

[32] J. Zhou, Y. Zhang, Y. Zhang, W. L. Shang, Z. Yang, and W. Feng, "Parameters identification of PV models using a differential evolution algorithm based on elite and obsolete dynamic learning," *Applied Energy*, vol. 314, pp. 1-20, May 2022, doi:

10.1016/j.apenergy.2022.118877.

[33] P. Singsathid, J. Wetweerapong, and P. Puphasuk, “Parameter estimation of solar PV models using self-adaptive differential evolution with dynamic mutation and pheromone strategy,” *International Journal of Mathematics and Computer Science*, vol. 19, no. 1, pp. 13-21, 2024.

[34] S. Das and P. N. Suganthan, “Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems,” *Jadavpur University, Nanyang Technological University*, Dec. 2010.

[35] A. Saxena, S. Shekhawat, and R. Kumar, “Application and development of enhanced chaotic grasshopper optimization algorithms,” *Modelling and Simulation in Engineering*, vol. 2018, pp. 1-14, May 2018, doi: 10.1155/2018/4945157.

BIOGRAPHIES OF AUTHORS

Watchara Wongsa completed M.Sc. degree in Applied Mathematics from Khon Kaen University, Thailand in 2014. He is a Ph.D. student in Applied Mathematics, Khon Kaen University. His research area is optimization. He can be contacted at email: watchara_w@kkumail.com.

Pikul Puphasuk completed 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.

Jeerayut Wetweerapong completed 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.