Research Article

Estimation of Distribution Algorithm Using Correlation between Binary Elements: A New Binary-Code Metaheuristic

Nantiwat Pholdee and Sujin Bureerat

*Sustainable and Infrastructure Research and Development Center, Department of Mechanical Engineering,
Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand*

Correspondence should be addressed to Sujin Bureerat; sujbur@kku.ac.th

Received 29 April 2017; Revised 28 July 2017; Accepted 2 August 2017; Published 13 September 2017

Academic Editor: Benjamin Ivorra

Copyright © 2017 Nantiwat Pholdee and Sujin Bureerat. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A new metaheuristic called estimation of distribution algorithm using correlation between binary elements (EDACE) is proposed. The method searches for optima using a binary string to represent a design solution. A matrix for correlation between binary elements of a design solution is used to represent a binary population. Optimisation search is achieved by iteratively updating such a matrix. The performance assessment is conducted by comparing the new algorithm with existing binary-code metaheuristics including a genetic algorithm, a univariate marginal distribution algorithm, population-based incremental learning, binary particle swarm optimisation, and binary simulated annealing by using the test problems of CEC2015 competition and one real-world application which is an optimal flight control problem. The comparative results show that the new algorithm is competitive with other established binary-code metaheuristics.

1. Introduction

Nowadays in the economic-competitive world, optimisation has become increasingly popular for real applications as it is a powerful mathematical tool for solving a wide range of engineering design types. Once an optimisation problem is posed, one of the most important elements in the optimisation process is an optimisation method or an optimiser used to find the optimum solution. Optimisers can be categorised as the methods with and without using function derivatives. The former are traditionally called mathematical programming or gradient-based optimisers whereas the latter have various subcategories. One of them is a metaheuristic (MH). The term metaheuristics can cover nature-inspired optimisers [1–10], swarm intelligent algorithms [11–20], and evolutionary algorithms [21–24]. Most of them are based on using a set of design solutions, often called a population, for searching an optimum. The main operator usually consists of the reproduction and selection stages. The advantages of such an optimiser are simplicity to use, global optimisation capability, and flexibility to apply as it is derivative-free. However, it still has a slow convergence rate and search consistency. These issues

have made researchers and engineers around the globe investigate how to improve the search performance of MHs.

A genetic algorithm (GA) [21] is probably the best known MH while other popular methods are differential evolution (DE) [22] and particle swarm optimisation (PSO) [17]. Among MH algorithms, they can be categorised as the methods using real, binary, or integer codes. The mix of those types of design variables and some other types can also be made. This makes MHs considerably appealing for use with real-world applications particularly for those design problems that function derivatives are not available or impossible to calculate. Most MHs are based on continuous design variables or real codes. For single objective optimisation, there have been numerous real-code MHs being developed. At the early stage, methods like evolutionary programming [25] and evolution strategies [26] were proposed. Then, DE and PSO were introduced. Until recently, there have been probably over a hundred new real-code MHs in the literature. Some recent algorithms include, for example, a sine-cosine algorithm [27], a grey wolf optimiser [20], teaching-learning-based optimisation [2], and Jaya algorithm [28]. Meanwhile, powerful existing algorithms such as PSO and DE have been upgraded by

integrating into them some types of self-adaptive schemes, for example, adaptive differential evolution with optional external archive (JADE) [29], Success-History Based Parameter Adaptation for Differential Evolution (SHADE) [30], SHADE Using Linear Population Size Reduction (LSHADE) [31], and adaptive PSO [32–34]. MHs are even more popular when they can be used to find a Pareto front of a multiobjective optimisation problem within one optimisation run. Such a type of algorithm is usually called multiobjective evolutionary algorithms (MOEAs) where some of the best known algorithms are nondominated sorting genetic algorithm (NSGA-I, NSGA-II, and NSGA-III) [35–37], multiobjective particle swarm optimisation [38], strength Pareto evolutionary algorithm [39], multiobjective grey wolf optimisation [40], multiobjective teaching-learning-based optimisation [41], multiobjective evolutionary algorithm based on decomposition [42], multiobjective ant colony optimisation [43], multiobjective differential evolution [44], and so forth. One of the most challenging issues in MHs is to improve their ability for tackling many-objective optimisation (a problem with more than three objectives). Some recently proposed algorithms are knee point-driven evolutionary algorithm [45], an improved two-archive algorithm [46], preference-inspired coevolutionary algorithms [47], and so forth.

In practice, GA a metaheuristic using binary strings is arguably the most used method as it is included in engineering software such as MATLAB. Apart from GA, other MHs using a binary string representing a design solution include a univariate marginal distribution algorithm (UMDA) [48], population-based incremental learning (PBIL) [24], binary particle swarm optimisation (BPSO) [49], binary simulated annealing (BSA) [50], binary artificial bee colony algorithm based on genetic operator (GBABC) [51], binary quantum-inspired gravitational search algorithm (BQIGSA) [52], and self-adaptive binary variant of a differential evolution algorithm (SabDE) [53]. With the popularity of GA, a binary-code MH has been rarely developed and proposed while its real-code counterparts have over a hundred different search concepts reported in the literature. That means there are possible more than a thousand real-code MH algorithms being published. It should be noted that real-code MHs can be modified to solve binary-code optimisation by means of binarisation [54].

This paper is therefore devoted to the further development of a binary-code metaheuristic. The method is called estimation of distribution algorithm using correlation between binary elements (EDACE). Performance assessment is made by comparing the proposed optimiser with GA, UMDA, BPSO, BSA, and PBIL by using the CEC2015 test problems. Also, the real-world optimal flight control is used for the assessment. The comparative results are obtained and discussed. It is shown that EDACE is among the top performers.

2. Proposed Method

The simplest but efficient estimation of distribution algorithm is probably population-based incremental learning (PBIL).

Another MH that uses a similar concept is UMDA. Unlike GA which uses a matrix containing the whole binary solutions during the search, PBIL uses the so-called probability vector to represent a binary population. During an optimisation process, the probability vector is updated iteratively until approaching an optimum. In EDACE, a matrix called a correlation between binary elements (CBE) matrix is used to represent a binary population. The matrix can be denoted as $P_{ij} \in [0, 1]$, where the value of the element P_{ij} indicates the correlation between element i and element j of a binary design solution. The higher value of P_{ij} means the higher probability that binary elements i and j will have the same value. The algorithm is developed to deal with a box-constrained optimisation problem:

$$\min f(\mathbf{x}); \quad \mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U, \quad (1)$$

where f is an objective function and $\mathbf{x}$ is a vector containing design variables (a design vector). $\mathbf{x}_L$ and $\mathbf{x}_U$ are the lower and upper bounds of $\mathbf{x}$, respectively. Assuming that a design vector can be represented by a row vector of binary bits size $m \times 1$, the CBE matrix thus has the size of $m \times m$. It should be noted that the details of converting a binary string to be a design vector can be found in [55]. In generating a binary string from the CBE matrix, a reference binary solution (RBS) is needed. It can be a randomly generated solution or the best solution found so far depending on a user preference. Then, a row of the matrix is randomly selected (say the r th row). The r th element of a generated binary solution is set to be the r th element of the reference binary solution. The rest of the created binary elements are based on the value of P_{rj} ; $j \neq r$. The procedure for creating a binary solution sized $m \times 1$ from the $m \times m$ CBE matrix is detailed in Algorithm 1 where $\mathbf{b}$ is a binary design solution, $\mathbf{b}_{\text{REF}}$ is the reference binary solution, n_p is a population size, and $\text{rand} \in [0, 1]$ is a uniform random number. The algorithm spends n_p loops for creating n_p binary solutions. The process for generating a binary solution from the CBE matrix is in steps (3)–(12). For one binary solution, only one randomly selected row of CBE (say row r) is used (step (4)). Then, the r th element of a generated binary solution is set equal to the r th element of the reference binary solution, $\mathbf{b}_{\text{REF}}$. The rest of the elements of the generated binary solution are created in such a way that their values depend on corresponding elements on the r th row of CBE. From the computation steps (5)–(11), the value of P_{rj} determines the probability of a_j to be the same as a_r . The higher value of P_{rj} means the higher correlation between elements r and j and consequently the higher probability that a_j will be set equal to a_r .

The CBE matrix is a square symmetric matrix with equal size to the length of a binary solution whose all diagonal elements are equal to one. For an iteration, the matrix will be updated according to the so far best solution ($\mathbf{b}_{\text{best}}$). The learning rate (L_R) will be used to control the changes in updating P_{ij} as with PBIL. Once P_{ij} is updated, the value of P_{ji} is set to be P_{ij} which means the process requires $m(m-1)/2$

```

Input:  $\mathbf{b}_{\text{REF}}, \mathbf{P}$ 
Output:  $\mathbf{B} = \{\mathbf{b}^i\}$  for  $i = 1, \dots, n_p$ 
Main procedure
(1) Set  $\mathbf{B} = \{\}$ .
(2) For  $i = 1$  to  $n_p$ 
(3)   Set  $\mathbf{a} = \{\}$  a vector used to contain elements of a generated binary string.
(4)   Randomly select a position ( $r$ th row) of  $\mathbf{P}$ .
(5)   Set  $a_r = b_{\text{REF},r}$ . % Set the  $r$ th element of  $\mathbf{a}$  as the  $r$ th element of  $\mathbf{b}_{\text{REF}}$ .
(6)   For  $j = \{1, 2, \dots, m\} - \{r\}$ 
(7)     If  $\text{rand} < P_{rj}$ 
(8)        $a_j = a_r$  %  $a_j$  and  $a_r$  values are equal, which are either "0" or "1".
(9)     Else
(10)       $a_j = 1 - a_r$  % If  $a_r = 1$ ,  $a_j = 0$  or vice versa.
(11)    End
(12)  End
(13)  Set  $\mathbf{B} = \mathbf{B} \cup \mathbf{a}$ .
(14) End

```

ALGORITHM 1: Generation of a binary population from a CBE matrix.

updates since P_{ii} is always set to be 1. The updated P_{ij} denoted by P'_{ij} can be calculated from

$$P'_{ij} = (1 - L_R) P_{ij} + L_R (1 - |b_{\text{best},i} - b_{\text{best},j}|), \quad (2)$$

where L_R is the learning rate randomly generated in the interval $[L_{R,L}, L_{R,U}]$. $b_{\text{best},i}$ and $b_{\text{best},j}$ are the i th and j th elements of $\mathbf{b}_{\text{best}}$, respectively. From the updating equation, if the i th and j th elements are similar, it means they are correlated; consequently, the value of P_{ij} (and P_{ji}) is increased. If they are dissimilar or uncorrelated, P_{ij} is then decreased. Nevertheless, the value of P_{ij} must be limited to the predefined interval

$$0 \leq P_L \leq P_{ij} \leq P_U \leq 1, \quad (3)$$

where P_L and P_U are the predefined lower and upper limits of P_{ij} . Equation (3) is used to maintain diversity in optimisation search. In the original PBIL, a mutation operator is used with the same purpose. Therefore, the procedure of EDACE starts with an initial matrix for correlation between binary elements where $P_{ii} = 1$ and $P_{ij} = 0.5$. This implies that when generating a binary solution, its elements have equal probability to be 1 or 0 where its r th element can be 1 or 0, created at random. The procedure for general purpose of EDACE is given in Algorithm 2. The decision on selecting $\mathbf{b}_{\text{REF}}$ for generating a binary solution and $\mathbf{b}_{\text{best}}$ for updating the CBE matrix is dependent on a preference of a user. This means other versions of EDACE can be developed in the future.

An initial binary population is randomly created. The binary solutions are then decoded to be real design variables where function evaluations are performed and $\mathbf{b}_{\text{REF}}$ and $\mathbf{b}_{\text{best}}$ are found. Then, new binary solutions are generated using Algorithm 1 while the greedy selection (steps (6)–(8)) is activated with $\mathbf{b}_{\text{REF}}$ and $\mathbf{b}_{\text{best}}$ being determined. The CBE matrix is updated by using $\mathbf{b}_{\text{best}}$ as detailed in (2)–(3). The search

process is repeated until termination criterion is reached. The generation of a binary design solution of EDACE is, to some extent, similar to those used in binary PSO [49] and binary quantum-inspired gravitational search algorithm (BQIGSA) [52] in the sense that the binary solution is controlled by the probability of being "1" or "0". However, in EDACE, a generated solution relies not only on such probability but also on the reference binary solution $\mathbf{b}_{\text{REF}}$. Apart from that, the update of CBE tends to be similar to the concept employed in PBIL with a learning rate and this is totally different from binary PSO and BQIGSA.

In selecting $\mathbf{b}_{\text{REF}}$ and $\mathbf{b}_{\text{best}}$, if both solutions are the same which is $\mathbf{b}_{\text{best}}$, it could lead to a premature convergence. If both are set to be a solution randomly selected solution from the current binary population, the diversification increases but the convergence rate will be slower. Therefore, the balance between intensification and diversification must be made. In this work, the so far best binary solution is set to be $\mathbf{b}_{\text{REF}}$ to maintain intensification. For updating the CBE matrix, we use the new updating scheme as

$$P'_{ij} = (1 - L_R) P_{ij} + L_R (1 - |b_{\text{best1},i} - b_{\text{best2},j}|). \quad (4)$$

The solutions $\mathbf{b}_{\text{best1}}$ and $\mathbf{b}_{\text{best2}}$ are two types of best solutions. Firstly, n_p best solutions are selected from $\{\mathbf{b}^i\} \cup \{\mathbf{b}_{\text{new}}^i\}$ (see Algorithm 2 for both solution sets), sorted according to their functions, and then saved to a set *Best_sol*. Four $m \times 1$ vectors are created as $\mathbf{b}_1$ the so far best solution, $\mathbf{b}_2$ a solution whose elements are averaged from the elements of the first n_{best} (default = 10) best solutions found so far, $\mathbf{b}_3$ a solution whose elements are averaged from the elements of the members of *Best_sol*, and $\mathbf{b}_4$ a solution whose elements are averaged from the elements of the current binary population. $\mathbf{b}_{\text{best1}}$ is randomly chosen from the aforementioned solutions ($\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$, and $\mathbf{b}_4$) with equal probability while $\mathbf{b}_{\text{best2}}$ is randomly chosen from the members of *Best_sol*. With this idea, the

Input: number of generation (n_{iter}), population size (n_p), binary length (m)
Output: $\mathbf{b}_{\text{best}}$, f_{best}
Initialisation:
 (0.1) Assign $P_{ij} = 0.5$ and $P_{ii} = 1$, sized $m \times m$.
 (0.2) Randomly generate n_p binary solutions $\mathbf{b}^i$ and decode them to be $\mathbf{x}^i$.
 (0.3) Calculate objective function values $f^i = \text{fun}(\mathbf{x}^i)$ where fun is an objective function evaluation.
 (0.4) Find f_{best} , $\mathbf{b}_{\text{REF}}$, $\mathbf{b}_{\text{best}}$
Main iterations
 (1) For iter = 1 to n_{iter}
 (2) Update $\mathbf{P}$ using Equation (2)
 (3) Generate $\mathbf{b}_{\text{new}}^i$ from $\mathbf{P}$ using Algorithm 1, and decode them to be $\mathbf{x}_{\text{new}}^i$.
 (4) For $i = 1$ to n_p
 (5) Calculate objective function values $f_{\text{new}}^i = \text{fun}(\mathbf{x}_{\text{new}}^i)$.
 (6) If $f_{\text{new}}^i < f_i$
 (7) $f_i = f_{\text{new}}^i$, $\mathbf{b}^i = \mathbf{b}_{\text{new}}^i$, $\mathbf{x}^i = \mathbf{x}_{\text{new}}^i$
 (8) End
 (9) End
 (10) Update f_{best} , $\mathbf{b}_{\text{REF}}$, $\mathbf{b}_{\text{best}}$
 (11) End

ALGORITHM 2: Procedure for EDACE.

Input: $L_{R,L}$, $L_{R,U}$, $\mathbf{P}$, $\mathbf{b}^i$, $\mathbf{b}_{\text{REF}}$, Best_sol , n_{best}
Output: $\mathbf{P}^i$
Main procedure
 Create $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$, $\mathbf{b}_4$
 For $i = 1$ to m
 (1) Assign $P_R = \text{rand}$.
 (2) If $P_R \in [0, 0.25]$, set $b_{\text{best1},i} = b_{1,i}$
 (3) If $P_R \in [0.25, 0.5]$, set $b_{\text{best1},i} = b_{2,i}$
 (4) If $P_R \in [0.5, 0.75]$, set $b_{\text{best1},i} = b_{3,i}$
 (5) Otherwise, set $b_{\text{best1},i} = b_{4,i}$
 (6) Random selected a vector $\mathbf{b}_{\text{best2}}$ from Best_sol .
 For $j = i + 1$ to m
 (7) Generate L_R .
 (8) Update P_{ij} using Equation (4).
 (9) Limit P_{ij} to the interval $[P_L, P_U]$.
 End
 End

ALGORITHM 3: Updating scheme for CBE.

balance between exploration and exploitation is maintained throughout the search process. Algorithm 3 shows the new CBE updating strategy.

3. Experimental Set-Up

To investigate the search performance of the proposed algorithm, fifteen learning-based test problems from CEC2015 and one flight dynamic control optimisation problem are used. The former are used for testing the performance of

EDACE for general types of box-constrained optimisation while the latter is the real-world application.

3.1. CEC2015 Learning-Based Test Problems. The CEC2015 learning-based test problems are box-constrained single objective benchmark functions proposed in [56]. The problems consist of 2 Unimodal Functions, 3 Simple Multimodal Functions, 3 Hybrid Functions, and 7 Composition Functions. The summary of CEC2015 learning-based test problems is shown in Table 1. It should be noted that the details and the codes for the test problems can be downloaded from the website of CEC2015 competition.

3.2. Flight Dynamic Control Optimisation Problem. Flight dynamic control system design is a classical important application for real engineering problems. The motion of an aircraft can be described using the body axes which is herein the stability axes consisting of roll axis (x), pitch axis (y), and yaw axis (z) as shown in Figure 1. The motion of the aircraft is described by Newton's 2nd law or equations of motion for both translational and rotational motions. The dynamical model is nonlinear but can be linearised by applying aerodynamic derivatives. Due to aircraft symmetry with respect to the xz plane, the linearised dynamical model can be decoupled into two groups as longitudinal motion and the lateral/directional motion. For more details of deriving the equations of motion, see [57]. In this work, only the lateral/directional motion control is considered. A state equation representing the dynamic motion of an aircraft is expressed as follows [57–60]:

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}, \quad (5)$$

TABLE 1: Summary of CEC2015 learning-based functions.

	Number	Functions	$f_{\min}$
Unimodal functions	1	Rotated high conditioned elliptic function	100
	2	Rotated Cigar function	200
Simple multimodal functions	3	Shifted and rotated Ackley's function	300
	4	Shifted and rotated Rastrigin's function	400
	5	Shifted and rotated Schwefel's function	500
Hybrid functions	6	Hybrid function 1 ($N = 3$)	600
	7	Hybrid function 2 ($N = 4$)	700
	8	Hybrid function 3 ($N = 5$)	800
Composition functions	9	Composition function 1 ($N = 3$)	900
	10	Composition function 2 ($N = 3$)	1000
	11	Composition function 3 ($N = 5$)	1100
	12	Composition function 4 ($N = 5$)	1200
	13	Composition function 5 ($N = 5$)	1300
	14	Composition function 6 ($N = 7$)	1400
	15	Composition function 7 ($N = 10$)	1500

where $\mathbf{x} = \{\beta, r, p, \phi\}^T$, β is the sideslip, a velocity in y direction, r is the yaw rate, rate of change of rotation about the x -axis, p is the roll rate, rate of change of rotation about the z -axis, ϕ is the bank angle, rotation about the x -axis, $\mathbf{A}$ is the kinetic energy matrix, $\mathbf{B}$ is Coriolis matrix, $\mathbf{u} = \{\delta_a, \delta_r\}$ is the control vector, δ_a is the aileron deflection, and δ_r is the rudder deflection.

The control vector $\mathbf{u}$ can be expressed as

$$\mathbf{u} = \mathbf{C}\mathbf{u}_p + \mathbf{K}\mathbf{x}, \quad (6)$$

where $\mathbf{u}_p$ is a pilot's control input vector while $\mathbf{C}$ and $\mathbf{K}$ are the gain matrices expressed as follows [59]:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \\ k_5 & 1 \end{bmatrix}, \quad (7)$$

$$\mathbf{K} = \begin{bmatrix} k_6 & k_1 & k_2 & 0 \\ k_7 & k_3 & k_4 & 0 \end{bmatrix},$$

where parameters k_1 – k_7 are control gain coefficients which need to be found.

From (5)–(6), the state equation for lateral/directional motion of an aircraft can be expressed as

$$\dot{\mathbf{x}} = (\mathbf{A} + \mathbf{BK})\mathbf{x} + \mathbf{B}\mathbf{C}\mathbf{u}_p. \quad (8)$$

Design optimisation of the control system of an aircraft is found to have many objectives as there are several criteria that need to be satisfied such as control stability, accuracy, sensitivity, and control effort, while the control gains coefficients are set to be design variables for an optimisation problem. In this work, the optimal flight control of an aircraft focuses on only the stability aspect. The objective function is posed

FIGURE 1: Stability axes of an aircraft.

to minimise spiral root subjected to stability performance constraints. The optimisation problem can then be written as

$$\begin{aligned} \min: \quad & f(\mathbf{x}) = \lambda_s \\ \text{Subjected to:} \quad & \lambda_s \leq -0.01 \\ & \lambda_R \leq -3.75 \\ & \xi_D \geq 0.5 \\ & \omega_d \geq 1, \end{aligned} \quad (9)$$

where λ_s , λ_R , ξ_D , and ω_d are spiral root, roll damping, damping ratio of Dutch-roll complex pair, and Dutch-roll frequency, respectively. These parameters can be calculated based on the eigenvalues associated with the matrix $\mathbf{A} + \mathbf{BK}$. The design variables are control gain coefficients in the matrix

$\mathbf{K}$ ($\mathbf{x} = \{k_1, k_2, k_3, k_4, k_6, k_7\}^T$). The kinetic energy matrix ($\mathbf{A}$) and the Coriolis matrix ($\mathbf{B}$) are defined as

$$\mathbf{A} = \begin{bmatrix} -0.2842 & -0.9879 & 0.1547 & 0.0204 \\ 10.8574 & -0.5504 & -0.2896 & 0 \\ -199.8942 & -0.4840 & -1.6025 & 0 \\ 0 & 0.1566 & 1 & 0 \end{bmatrix}, \quad (10)$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0.0524 \\ 0.4198 & -12.7393 \\ 50.5756 & 21.6753 \\ 0 & 0 \end{bmatrix}.$$

More details about this aircraft dynamic model can be found in [58–60]. To handle the constraints, the penalty function which was presented in [61] is used.

The proposed EDACE and several well-established binary-code metaheuristics are used to solve the fifteen CEC2015 learning-based test problems and the flight dynamic control test problem. The metaheuristic optimisers are as follows:

Genetic algorithm (GA) [21] used binary codes with crossover and mutation rates are 1 and 0.1, respectively.

Binary simulated annealing (BSA) [50] used binary codes with exponentially decreasing temperature. The starting and ending temperature are set to be 10 and 0.001, respectively. The cooling step is set as 10.

Population-based incremental learning (PBIL) [24] used binary codes with the learning rate, mutation shift, and mutation rate as 0.5, 0.7, and 0.2, respectively.

Binary particle swarm optimisation (BPSO) [49] used binary codes with V-shaped transfer function while the transfer function used is the V-shaped version 4 (V4) as reported in [49]. It is noted that this version is said to be the most efficient version based on the results obtained in [49].

Univariate marginal distribution algorithm (UMDA) [48] used binary codes. The first 20 best binary solutions are used to update the probability matrix.

Estimation of distribution algorithm with correlation of binary elements (EDACE) (Algorithm 2) used binary codes with $P_L = 0.1$, $P_U = 0.9$, $L_{R,L} = 0.4$, $L_{R,U} = 0.6$, and $n_{\text{best}} = 10$.

Each algorithm is used to solve the problems for 30 optimisation runs. The population sizes are set to be 100 and 20 while number of generation is set to be 100 and 500 for the CEC2015 learning-based test problems and the flight dynamic control test problem, respectively. For an algorithm using different population size and number of generations such as BSA, it will be terminated at the same number function evaluations, which is 10,000 for all test problems. The binary length is set to be 5 for each design variable for all optimisers.

4. Optimum Results

4.1. CEC2015. After applying the proposed EDACE and several well-established binary MHs for solving the CEC2015 learning-based benchmark functions, the results are shown in Tables 2–4. Note that, apart from the algorithms used in this study, the results of solving CEC2015 test suit obtained from efficient binary artificial bee colony algorithm based on genetic operator (GBABC), binary quantum-inspired gravitational search algorithm (BQIGSA), and self-adaptive binary variant of a differential evolution algorithm (SabDE) as reported in [53] are also included in the comparison. From Table 2, the mean (Mean) and standard deviation (STD) values of the objective functions are used to measure the search convergence and consistency of the algorithms. The lower Mean is the better convergence while the lower STD is the better consistency. The value of Mean is more important; thus, for method A with lower Mean but higher STD than method B, method A is considered to be superior.

For the measure of search convergence based on the mean objective function values, the best performer for the unimodal test functions, f_1 and f_2 , is EDACE while the second best is BPSO. For the simple multimodal functions, the best performer for f_4 and f_5 is SabDE while the best performer for f_3 is BPSO. The second best performers for f_3 , f_4 , and f_5 are SabDE, BEDACE, and UMDE, respectively. For the hybrid functions, the best performers for the functions f_6 , f_7 , and f_8 , are SabDE, EDACE, and BPSO, respectively, while the second best performer for f_6 and f_7 is BPSO and the second best for f_8 is EDACE. For the final group of CEC2015 test problems, composition functions, the best performer for the f_{11} , f_{12} , and f_{14} is SabDE while the best performers for the f_{10} and f_{15} are BPSO and EDACE, respectively. For f_9 , the best performers are UMDA, BPSO, GA, PBIL, and EDACE, which obtain the same mean values while, for f_{13} , the best performers are UMDA, BPSO, GA, PBIL, BSA, and EDACE, which obtained the same mean values. It should be noted that the results from [53] were obtained from using the total number of function evaluations as 1,000,000 with the binary length of 50 for each design variable whereas this work uses 10,000 function evaluations with the binary length of 5 for each design variable. This indirect comparison with GBABC, BQIGSA, and SabDE can only be used to show that the proposed EDACE also has good performance and cannot be used to claim which method is superior.

For the measure of search consistency based on the STD values, the most consistent methods for unimodal functions, f_1 and f_2 , are BPSO and EDACE while the second most consistent methods are EDACE and BPSO, respectively. For the simple multimodal functions, the best for f_3 and f_5 is SabDE while the best for f_4 is the proposed EDACE. EDACE is the best for the hybrid function of f_7 while BPSO is the best for the hybrid functions f_6 and f_8 . For the composition functions, EDACE is the best for the problems f_9 and f_{12} while BPSO is the best for f_{10} . For the composition functions, f_{11} , f_{14} , and f_{15} , the best is SabDE while the best for f_{13} is BSA.

TABLE 2: Objective values obtained.

CEC2015	MHs	UMDA	BPSO	GA	PBIL	BSA	EDACE	*GBABC	*BQIGSA	*SabDE
Unimodal functions	f_1	Mean	7.415E+06	1.807E+06	5.508E+06	1.586E+07	4.365E+07	2.729E+07	8.419E+07	3.093E+08
		STD	5.648E+06	1.224E+06	3.510E+06	1.226E+07	4.391E+07	2.297E+06	7.354E+07	1.168E+08
		Min.	5.203E+05	1.914E+05	1.016E+06	2.688E+05	1.325E+06	2.454E+05		
	f_2	Mean	1.728E+08	1.278E+08	2.415E+08	1.443E+08	1.018E+09	2.864E+09	7.834E+09	2.541E+09
		STD	1.287E+08	1.236E+08	1.880E+08	1.371E+08	1.680E+09	3.046E+07	6.527E+09	5.008E+09
		Min	4.359E+07	3.525E+07	6.713E+07	4.834E+07	1.133E+08	3.277E+07		
Simple multimodal functions	f_3	Mean	3.203E+02	3.197E+02	3.203E+02	3.202E+02	3.202E+02	3.202E+02	3.202E+02	3.200E+02
		STD	8.505E-02	1.900E+00	9.050E-02	7.945E-02	6.006E-02	3.300E-02	2.641E+02	2.044E-02
		Min	3.201E+02	3.107E+02	3.201E+02	3.201E+02	3.201E+02	3.200E+02		
	f_4	Mean	4.213E+02	4.220E+02	4.286E+02	4.278E+02	4.226E+02	4.182E+02	4.358E+02	4.116E+02
		STD	4.647E+00	4.915E+00	8.553E+00	9.507E+00	7.150E+00	4.173E+00	3.599E+02	7.606E+00
		Min	4.105E+02	4.124E+02	4.138E+02	4.123E+02	4.105E+02	4.105E+02		
Hybrid functions	f_5	Mean	1.010E+03	1.066E+03	1.339E+03	1.353E+03	1.275E+03	1.014E+03	1.108E+03	9.330E+02
		STD	1.300E+02	1.352E+02	1.981E+02	2.279E+02	1.975E+02	1.500E+02	1.275E+03	9.464E+01
		Min	7.791E+02	8.526E+02	1.049E+03	8.120E+02	9.628E+02	6.907E+02		
	f_6	Mean	1.951E+05	7.345E+04	2.288E+05	4.894E+05	6.403E+06	1.133E+05	7.442E+06	4.625E+04
		STD	1.120E+05	3.958E+04	1.813E+05	3.224E+05	8.635E+06	8.936E+04	1.321E+07	4.076E+04
		Min	3.661E+04	3.661E+04	3.702E+04	8.124E+04	1.320E+05	3.659E+04		
Composition functions	f_7	Mean	7.047E+02	7.032E+02	7.044E+02	7.046E+02	7.118E+02	7.030E+02	7.589E+02	7.752E+02
		STD	1.054E+00	6.183E-01	1.036E+00	1.113E+00	8.660E+00	5.927E-01	4.668E+02	4.155E+03
		Min	7.027E+02	7.024E+02	7.027E+02	7.024E+02	7.025E+02	7.021E+02		
	f_8	Mean	9.309E+04	1.511E+04	5.503E+04	4.808E+05	2.305E+06	2.727E+04	3.949E+07	2.395E+07
		STD	1.120E+05	6.918E+03	5.630E+04	5.295E+05	2.400E+06	2.635E+04	2.442E+08	5.432E+07
		Min	1.497E+04	1.287E+04	1.287E+04	1.312E+04	1.589E+04	1.287E+04		
	f_9	Mean	1.001E+03	1.001E+03	1.001E+03	1.001E+03	1.003E+03	1.001E+03	1.017E+03	1.177E+03
		STD	2.090E-01	2.231E-01	9.437E-01	4.017E-01	4.284E+00	1.700E-01	8.397E+02	4.102E+01
		Min	1.000E+03	1.000E+03	1.000E+03	1.001E+03	1.000E+03	1.000E+03		
	f_{10}	Mean	1.285E+04	3.930E+03	1.367E+04	4.235E+04	5.317E+05	7.819E+03	9.909E+05	4.426E+04
		STD	8.308E+03	2.140E+03	1.291E+04	3.728E+04	6.444E+05	4.897E+03	3.659E+06	2.416E+07
		Min	3.199E+03	1.733E+03	1.738E+03	2.275E+03	1.805E+03	1.731E+03	4.477E+04	8.862E+07
	f_{11}	Mean	1.510E+03	1.232E+03	1.360E+03	1.396E+03	1.427E+03	1.240E+03	1.159E+03	1.114E+03
		STD	8.727E+01	1.410E+02	1.129E+02	5.002E+01	4.384E+01	1.436E+02	9.557E+02	1.131E+01
		Min	1.401E+03	1.109E+03	1.118E+03	1.132E+03	1.402E+03	1.109E+03		
	f_{12}	Mean	1.304E+03	1.305E+03	1.306E+03	1.308E+03	1.308E+03	1.305E+03	1.264E+03	1.255E+03
		STD	9.674E-01	1.125E+00	1.259E+00	2.595E+00	4.486E+00	1.115E+00	1.044E+03	1.302E+00
		Min	1.303E+03	1.303E+03	1.304E+03	1.304E+03	1.303E+03	1.303E+03		
	f_{13}	Mean	1.300E+03	1.300E+03	1.300E+03	1.300E+03	1.300E+03	1.300E+03	1.446E+03	2.815E+09
		STD	8.140E-04	1.039E-03	1.248E-03	9.095E-04	2.313E-13	7.944E-04	1.197E+03	4.158E+09
		Min	1.300E+03	1.300E+03	1.300E+03	1.300E+03	1.300E+03	1.300E+03		
	f_{14}	Mean	9.364E+03	5.623E+03	7.023E+03	7.355E+03	8.736E+03	6.167E+03	2.162E+03	3.356E+03
		STD	2.120E+03	2.144E+03	1.856E+03	2.732E+03	3.603E+03	2.199E+03	2.091E+03	1.727E+03
		Min	4.817E+03	2.816E+03	4.425E+03	2.818E+03	4.453E+03	2.401E+03	2.869E+03	4.411E+02
	f_{15}	Mean	1.623E+03	1.616E+03	1.621E+03	1.618E+03	1.639E+03	1.614E+03	2.012E+03	1.700E+03
		STD	3.445E+00	3.132E+00	5.108E+00	4.295E+00	3.614E+01	3.945E+00	1.659E+03	2.177E-05
		Min	1.618E+03	1.609E+03	1.612E+03	1.610E+03	1.610E+03	1.607E+03	—	—

*Results reported in [53] with 1,000,000 function evaluations and 50 binary lengths for each design variable.

TABLE 3: Ranking of all optimisers based on the Mean values.

	UMDA	BPSO	GA	PBIL	BSA	EDACE	GBABC	BQIGSA	SabDE
$f1$	4	2	3	5	7	1	6	8	9
$f2$	4	2	5	3	6	1	8	9	7
$f3$	9	1	8	7	4	3	5	5	2
$f4$	3	4	7	6	5	2	8	9	1
$f5$	2	4	7	8	6	3	5	9	1
$f6$	4	2	5	6	8	3	9	7	1
$f7$	5	2	3	4	6	1	8	7	9
$f8$	4	1	3	5	6	2	9	7	8
$f9$	3	2	5	4	6	1	7	8	9
$f10$	3	1	4	5	7	2	8	6	9
$f11$	9	4	6	7	8	5	2	3	1
$f12$	4	6	7	9	8	5	3	2	1
$f13$	3	2	6	5	1	4	7	8	9
$f14$	9	4	6	7	8	5	2	3	1
$f15$	6	3	5	4	7	2	9	1	8
<i>Sum of ranking</i>	72	40	80	85	93	40	96	92	76

TABLE 4: Comparison based on the statistical t -test of the test problem.

	UMDA	BPSO	GA	PBIL	BSA	EDACE	GBABC	BQIGSA	SabDE
UMDA	0	1	1	0	0	1	0	0	0
BPSO	0	0	0	0	0	1	0	0	0
GA	0	1	0	0	0	1	0	0	0
PBIL	1	1	1	0	0	1	0	0	0
BSA	1	1	1	1	0	1	1	0	0
EDACE	0	0	0	0	0	0	0	0	0
GBABC	1	1	1	1	0	1	0	0	0
BQIGSA	1	1	1	1	1	1	1	0	0
SabDE	1	1	1	1	1	1	1	1	0
Sum	5	7	6	4	2	8	3	1	0
Ranking	4	2	3	5	7	1	6	8	9

The value Min in Table 2 is the objective function value of the best run from a particular method. Note that only the UMDA, BPSO, GA, PBIL, BSA, and EDACE were compared. For the unimodal function, the minimum objective function values of $f1$ and $f2$ were obtained by BPSO and EDACE, respectively. For the simple multimodal functions, the minimum objective function values for $f3$ and $f5$ are obtained from BPSO and EDACE, respectively, while for $f4$, the minimum is obtained from UMDA, BSA, and EDACE. The EDACE obtained minimum objective function values for all test functions in the hybrid function group. However, for the hybrid function $f8$, three algorithms including BPSO, GA, and EDACE obtained the minimum values. For the composition functions, EDACE obtained the minimum function values for all test functions. However, for the functions $f9$ and $f13$, all algorithms obtained the same minimum values while

for the $f11$, BPSO and EDACE obtained the same minimum function values. Similarly, for $f12$, UMDA, BPSO, BSA, and EDACE obtained the same minimum values.

Table 3 shows the summary of ranking based on the mean objective function values from 30 optimisation runs. It was found that the proposed EDACE is mostly ranked in top three best from solving fifteen CEC2015 learning-based test problems. After summing up the ranking score, it is found that EDACE and BPSO are equal best performer while the third best is UMDA.

In order to further investigate the performance comparison of the binary-code MHs, the statistical t -test is employed. Table 4 shows a 9×9 comparison matrix of the 9 optimisers. If method i is significantly better than method j based on the t -test at 5% significant level, the column i and row j of the matrix are set to be 1; otherwise, they are set to be 0. When

FIGURE 2: Search history of the top three best optimisers based on the t -test for the unimodal function.TABLE 5: Ranking of all optimisers for all CEC2015 learning-based test problem based on statistical t -test.

	UMDA	BPSO	GA	PBIL	BSA	EDACE	GBABC	BQIGSA	SabDE
$f1$	4	2	3	5	7	1	6	8	9
$f2$	4	2	5	3	6	1	8	9	7
$f3$	5	2	5	5	4	2	5	5	1
$f4$	3	4	6	6	4	2	8	8	1
$f5$	2	4	7	8	6	2	5	9	1
$f6$	4	2	5	6	8	3	9	7	1
$f7$	3	1	3	3	6	1	8	7	8
$f8$	4	1	3	5	6	2	9	7	8
$f9$	3	1	4	4	6	1	7	8	9
$f10$	3	1	4	5	7	2	8	6	9
$f11$	9	4	6	7	8	5	2	2	1
$f12$	4	5	7	8	8	5	2	2	1
$f13$	1	1	1	1	1	1	7	7	9
$f14$	9	4	6	7	8	5	2	3	1
$f15$	6	3	5	4	7	2	9	1	8
Sum	64	37	70	77	92	35	95	89	74

summing up along the columns, the highest score indicates the best optimiser based on this type of comparison. In the table, it means EDACE is the best. Table 5 shows the ranking of the 9 optimisers when solving all CEC2015 learning-based test problems based on the t -test. After summing up the ranking numbers of all test problems, it is found that EDACE is the overall best optimiser while BPSO and UMDA are the second and the third best, respectively.

Figures 2–5 show the search history of the top three optimisers EDACE, BPSO, and UMDA on solving all CEC2015 learning-based test problems where the vertical axis is the average objective function from 30 runs of each method. For all test functions, it was found that EDACE and UMDA converged to the optimal values at higher speed while BPSO seems to converge slowly and consistently. However, for all functions, BPSO finally moves to the minimum or near

FIGURE 3: Search history of the top three best optimisers based on the t -test for the simple multimodal functions.

TABLE 6: The table shows performance of EDACE for various number of binary bits.

Number of binary bits	5	10	25	50
Mean function values	$2.314E + 6$	$1.101E + 6$	$1.079E + 6$	$1.143E + 6$
Average computational time (Sec.)	9.371	10.748	18.634	52.773

minimum function values at the end of the runs. EDACE shows fast convergence from the beginning and obtained the minimum or near minimum values for all test functions except for $f3$. This indicates the ability of search exploitation and search exploration of the proposed EDACE since the CEC2015 test functions were assigned to test both aspects of MHs.

Table 6 shows performance of EDACE on solving unimodal function, $f1$, when the binary lengths for each design variable are 5, 10, 25, and 50 for 10 optimisation runs. It was found that when the number of binary bits increases, the computational time increases and the resulting mean objective function values decrease for the binary lengths less

than 25. However, for the binary length of 50, the mean objective function value increases meaning EDACE performance deteriorates. Without considering computational time, the best number of binary length is 25.

4.2. Flight Dynamic Control System Design. After applying the six binary-code MHs to solve the real engineering application of flight dynamic and control system for 30 optimisation runs, the comparison results are shown as box-plots of the objective and constraint violation values (Figure 6). The upper and lower horizontal lines of each box represent the maximum and minimum of objective function values, respectively, while the internal line shows the median

FIGURE 4: Search history of the top three best optimisers based on the t -test for the hybrid functions.

of objective function values. From this figure, based on median values of objective function, it is found that the best performer is EDACE while the second best and the third best are BPSO and UMDA, respectively. The most consistent method having the smallest gap between the maximum and minimum for all of optimisation runs is UMDA. However, the worst function value found by EDACE is almost as good as the best found by UMDA. Thus, the proposed EDACE is superior. Based on the figure, it was found that GA failed to solve the problem as it cannot obtain a feasible optimum point. The minimum objective function value is obtained from using the proposed EDACE.

Figure 7 shows the best run search history of all optimisers (selection based on the minimum objective function values of feasible solutions). From the figure, UMDA and PBIL seem to be the fastest convergent methods initially. However, after the process goes on for about 4,000 function evaluations, the proposed EDACE converged to the minimum objective function value with a faster rate than the others. It has better

exploration rate as the best function value is still decreased at the late iteration numbers. BPSO, on the other hand, seems to be slower than UMDA, PBIL, and BSA in the beginning. It however can converge to the better results after around 8,000 function evaluations.

5. Conclusions and Discussion

In this work, a new concept of a binary-code optimiser is proposed. Fifteen CEC2015 learning-based test problems and a real engineering design problem of flight dynamic and control system are used to investigate the search performance of the proposed algorithm. Several well-established binary-code MHs are used in comparison. The results obtained show that the proposed EDACE is the best performer on solving the 15 CEC2015 learning-based test problems and real engineering design problem of flight dynamic and control. Further improvement of EDACE by means of self-adaptation will be investigated in the future. The choice for $\mathbf{b}_{\text{REF}}$ needs further

FIGURE 5: Search history of the top three best optimisers based on the t -test for the composition functions.

studies. The use of EDACE for hyperheuristic development is also possible. The extension to multiobjective optimisation and many-objective optimisation is also under investigation. Applying EDACE for the more complex problems such as large scale problems, mixed-variable problems, and reliability optimisation is for future work. The flight control optimisation problem, one of our recent research focuses, has more than three objective functions to be optimised; thus, it should be formulated as many-objective optimisation. This along

with aircraft path planning dynamic optimisation still needs considerably more investigation while EDACE will be one of optimisers to be used for solving such design problems.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

FIGURE 6: Box-plot of objective function values from 30 optimisation runs.

FIGURE 7: Search history of the best run of all optimisers.

Acknowledgments

The authors are grateful for the support from the Thailand Research Fund (TRF), Grant no. MRG5980238.

References

- [1] S. Mirjalili, "Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm," *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.
- [2] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems," *Computer-Aided Design*, vol. 43, no. 3, pp. 303–315, 2011.
- [3] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *Advances in Swarm Intelligence*, Y. Tan, Y. Shi, and K. Tan, Eds., vol. 6145, pp. 355–364, Springer, Berlin Heidelberg, Germany, 2010.
- [4] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: charged system search," *Acta Mechanica*, vol. 213, no. 3–4, pp. 267–289, 2010.
- [5] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.
- [6] M. Fesanghary, "Harmony search applications in mechanical, chemical and electrical engineering," in *Music-Inspired Harmony Search Algorithm*, Z. Geem, Ed., vol. 191, pp. 71–86, Springer Publishing Company, Berlin, Heidelberg, Germany, 2009.
- [7] X. Wu, Y. Zhou, and Y. Lu, "Elite opposition-based water wave optimization algorithm for global optimization," *Mathematical Problems in Engineering*, vol. 2017, Article ID 3498363, 25 pages, 2017.
- [8] C. Wang, Y. Wang, K. Wang, Y. Dong, and Y. Yang, "An improved hybrid algorithm based on biogeography/complex and Metropolis for many-objective optimization," *Mathematical Problems in Engineering*, vol. 2017, Article ID 2462891, 14 pages, 2017.

- [9] W. Lei, H. Manier, M.-A. Manier, and X. Wang, "A hybrid quantum evolutionary algorithm with improved decoding scheme for a robotic flow shop scheduling problem," *Mathematical Problems in Engineering*, vol. 2017, Article ID 3064724, 13 pages, 2017.
- [10] C.-R. Hwang, "Simulated annealing: theory and applications," *Acta Applicandae Mathematicae*, vol. 12, pp. 108–111, 1988.
- [11] C. Qu, S. Zhao, Y. Fu, and W. He, "Chicken swarm optimization based on elite opposition-based learning," *Mathematical Problems in Engineering*, vol. 2017, Article ID 2734362, 20 pages, 2017.
- [12] N. Dong, X. Fang, and A.-g. Wu, "A novel chaotic particle swarm optimization algorithm for parking space guidance," *Mathematical Problems in Engineering*, vol. 2016, Article ID 5126808, 14 pages, 2016.
- [13] X.-S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012.
- [14] X.-S. Yang and S. Deb, "Engineering optimisation by Cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.
- [15] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155–1173, 2008.
- [16] 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, no. 3, pp. 459–471, 2007.
- [17] G. Venter and J. Sobieszczanski-Sobieski, "Particle swarm optimization," *AIAA Journal*, vol. 41, no. 8, pp. 1583–1589, 2003.
- [18] V. Muthiah-Nakarajan and M. M. Noel, "Galactic swarm optimization: a new global optimization metaheuristic inspired by galactic motion," *Applied Soft Computing Journal*, vol. 38, pp. 771–787, 2016.
- [19] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.
- [20] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.
- [21] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine Learning*, vol. 3, no. 2-3, pp. 95–99, 1998.
- [22] 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.
- [23] T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, NY, USA, 1996.
- [24] S. Baluja, *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*, Carnegie Mellon University, Pittsburgh, PA, USA, 1994.
- [25] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, John Wiley, New York, NY, USA, 1966.
- [26] H. Beyer and H. Schwefel, "Evolution strategies—a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.
- [27] S. Mirjalili, "SCA: a sine cosine algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.
- [28] R. V. Rao, "Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 7, no. 1, pp. 19–34, 2016.
- [29] J. Q. 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.
- [30] R. Tanabe and A. Fukunaga, "Evaluating the performance of SHADE on CEC 2013 benchmark problems," in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC '13)*, pp. 1952–1959, June 2013.
- [31] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 1658–1665, IEEE, July 2014.
- [32] M. P. Wachowiak, M. C. Timson, and D. J. DuVal, "Adaptive particle swarm optimization with heterogeneous multicore parallelism and GPU acceleration," *IEEE Transactions on Parallel and Distributed Systems*, 2017.
- [33] L. Zhang, Y. Tang, C. Hua, and X. Guan, "A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques," *Applied Soft Computing Journal*, vol. 28, pp. 138–149, 2015.
- [34] X. Liang, W. Li, Y. Zhang, and M. C. Zhou, "An adaptive particle swarm optimization method based on clustering," *Soft Computing*, vol. 19, no. 2, pp. 431–448, 2015.
- [35] N. Srinivas and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithms," *Evolutionary Computation*, vol. 2, pp. 221–248, 1994.
- [36] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [37] W. Mkaouer, M. Kessentini, S. Bechikh, K. Deb, and M. Ó. Cinnéide, "High dimensional search-based software engineering: Finding tradeoffs among 15 objectives for automating software refactoring using NSGA-III," in *Proceedings of the 16th Genetic and Evolutionary Computation Conference (GECCO '14)*, pp. 1263–1270, July 2014.
- [38] C. A. Coello Coello and M. S. Lechuga, "MOPSO: a proposal for multiple objective particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, pp. 1051–1056, May 2002.
- [39] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization, presented at the Evolutionary Methods for Design, Optomozation and Control," *Evolutionary Methods for Design, Optomozation and Control, Barcelona Spain*, 2002.
- [40] K. Nuaekae, P. Artrit, N. Pholdee, and S. Bureerat, "Optimal reactive power dispatch problem using a two-archive multi-objective grey wolf optimizer," *Expert Systems with Applications*, vol. 87, pp. 79–89, 2017.
- [41] F. Zou, L. Wang, X. Hei, D. Chen, and B. Wang, "Multi-objective optimization using teaching-learning-based optimization algorithm," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, pp. 1291–1300, 2013.
- [42] Q. Zhang and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.
- [43] D. Angus and C. Woodward, "Multiple objective ant colony optimisation," *Swarm Intelligence*, vol. 3, no. 1, pp. 69–85, 2009.
- [44] B. V. Babu and M. M. L. Jehan, "Differential evolution for multi-objective optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '03)*, pp. 2696–2703, Canberra, Australia, December 2003.
- [45] X. Zhang, Y. Tian, and Y. Jin, "A knee point driven evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 6, pp. 761–776, 2015.

- [46] H. Wang, L. Jiao, and X. Yao, "Two_Arch2: an improved two-archive algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, pp. 524–541, 2015.
- [47] R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-inspired coevolutionary algorithms for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 4, pp. 474–494, 2013.
- [48] H. Mühlenbein, "The equation for response to selection and its use for prediction," *Evolutionary Computation*, vol. 5, no. 3, pp. 303–346, 1997.
- [49] S. Mirjalili and A. Lewis, "S-shaped versus V-shaped transfer functions for binary particle swarm optimization," *Swarm and Evolutionary Computation*, vol. 9, pp. 1–14, 2013.
- [50] S. Bureerat and J. Limtragool, "Structural topology optimisation using simulated annealing with multiresolution design variables," *Finite Elements in Analysis and Design*, vol. 44, no. 12-13, pp. 738–747, 2008.
- [51] C. Ozturk, E. Hancer, and D. Karaboga, "A novel binary artificial bee colony algorithm based on genetic operators," *Information Sciences*, vol. 297, pp. 154–170, 2015.
- [52] H. Nezamabadi-Pour, "A quantum-inspired gravitational search algorithm for binary encoded optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 40, pp. 62–75, 2015.
- [53] A. Banitalebi, M. I. A. Aziz, and Z. A. Aziz, "A self-adaptive binary differential evolution algorithm for large scale binary optimization problems," *Information Sciences*, vol. 367-368, pp. 487–511, 2016.
- [54] B. Crawford, R. Soto, G. Astorga, J. Garcia, C. Castro, and F. Paredes, "Putting continuous metaheuristics to work in binary search spaces," *Complexity*, vol. 2017, Article ID 8404231, 19 pages, 2017.
- [55] G. Lindfield and J. Penny, *Numerical Methods: Using MATLAB*, Academic Press, 2012.
- [56] J. Liang, B. Qu, P. Suganthan, and Q. Chen, "Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization," Tech. Rep., Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2014.
- [57] D. A. Caughey, *Introduction to Aircraft Stability and Control Course Notes for M&AE 5070*, Sibley School of Mechanical & Aerospace Engineering, Cornell University, Ithaca, New York, NY, USA, 2011.
- [58] S. Rostami and F. Neri, "Covariance matrix adaptation pareto archived evolution strategy with hypervolume-sorted adaptive grid algorithm," *Integrated Computer-Aided Engineering*, vol. 23, no. 4, pp. 313–329, 2016.
- [59] S. F. Adra, A. I. Hamody, I. Griffin, and P. J. Fleming, "A hybrid multi-objective evolutionary algorithm using an inverse Neural Network for aircraft control system design," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (IEEE CEC '05)*, pp. 1–8, September 2005.
- [60] D. Tabak, A. A. Schy, D. P. Giesy, and K. G. Johnson, "Application of multiobjective optimization in aircraft control systems design," *Automatica*, vol. 15, no. 5, pp. 595–600, 1979.
- [61] S. Bureerat and N. Pholdee, "Optimal truss sizing using an adaptive differential evolution algorithm," *Journal of Computing in Civil Engineering*, vol. 30, no. 2, p. 04015019, 2015.