Final Report

Project Title: Surrogate assisted meta-heuristics for engineering optimisation

By Asst.Prof.Dr. Nantiwat Pholdee

April 2018

Contract No. MRG5980238

Final Report

Project Title: Surrogate assisted meta-heuristics for engineering optimisation

Researcher	Institute
1. Asst.Prof.Dr. Nantiwat Pholdee	Khon Kaen University
2. Prof.Dr.Sujin Bureerat	Khon Kaen University

This project granted by the Thailand Research Fund

บทคัดย่อ

รหัสโครงการ: MRG5980238

ชื่อโครงการ: การเพิ่มประสิทธิภาพในการหาคำตอบของวิธิเมตาฮิวริสติกโดยใช้แบบจำลองเซอโรเกทช่วยสำหรับการหาค่าเหมาะที่สุดทางวิศวกรรม

ชื่อนักวิจัย และสถาบัน ผศ.ดร.ณัญธิวัฒน์ พลดี มหาวิทยาลัยขอนแก่น

อีเมล์: nantiwat@kku.ac.th

ระยะเวลาโครงการ: 2 ปี

บทคัดย่อ:

งานวิจัยนี้นำเสนอการเพิ่มประสิทธิภาพให้กับวิธีหาค่าเหมาะที่สุดแบบเมต้าฮิวริสติกโดยใช้แบบจำลองเซอโรเกทช่วย ใช้แนวคิดของตัวแปรปรับตัวได้และใช้แนวคิดแบบผสม สำหรับปัญหาทางวิสกรรม งานวิจัยนี้จะเริ่มจากนำเสนอการเพิ่มประสิทธิภาพให้กับเมต้าฮิวริสติกที่มีชื่อว่า teaching-learning based optimizer (TLBO) โดยใช้ opposition-based approach, binary crossover และ probability of operating the learning phase สำหรับปัญหาการหาค่าเหมาะสมสุดของกระบวนการม้วนเก็บแผ่นเหล็ก เมื่อทำการหาค่าเหมาะที่สุดสำหรับปัญหาการออกแบบที่กำหนด พบว่าวิธีการใหม่ที่นำเสนอนี้มีประสิทธิภาพสูงกว่าวิธีเมต้าฮิวริวติกที่มีใช้อยู่ในปัจจุบัน ต่อจากนั้นงานวิจัยนี้นำเสนอการใช้ตัวแปรแบบปรับตัวได้ร่วมกับการทำ mutation ของวิธี differential evolution (DE) เพื่อเพิ่มประสิทธิภาพในการหาคำตอบให้กับวิธี sine cosine algorithm โดยวิธีที่นำเสนอจะถูกนำมาใช้เพื่อหาคำตอบสำหรับปัญหาในการหาตำแหน่งการเสียหายของโครงถัก ผลที่ได้พบว่าเมต้าฮิวริสติกที่ได้นำเสนอขึ้นมามีประสิทธิภาพสูงกว่าเมต้าฮิวริสติกหลายๆตัวที่มีใช้อยู่ในปัจจุบัน หลังจากนั้นงานวิจัยนี้ได้นำเสนอวิธีเมต้าฮิวริสติกแบบไบนารี่รูปแบบใหม่ เรียกว่า วิธี estimation of distribution algorithm using correlation between binary elements (EDACE) ซึ่งทดสอบประสิทธิภาพโดยใช้ฟังก์ชันทดสอบ CEC2015 ผลการทดสอบพบว่าเมต้าฮิวริสติกที่นำเสนอใหม่นี้มีประสิทธิภาพสูงที่สุดเมื่อเทียบกับวิธีอื่นๆที่มีใช้อยู่ในปัจจุบัน นอกเหนือจากนี้ในงานวิจัยนี้ยังได้นำเสนอการใช้แบบจำลองเซอโรเกทร่วมกับเมต้าฮิวริสติกสำหรับปัญหาที่เป็น Inverse problem ของการหาตำแหน่งการเสียหายของโครงสร้าง ซึ่งแบบจำลองเซอโรเกทจะถูกนำมาใช้ในการประมาณค่าตัวแปรออกแบบ(คำตอบ)แทนการประมาณค่าฟังก์ชันก์เป้าหมายตามที่ใช้กันปกติ ผลการทดสอบพบว่าจากการเปรียบเทียบประสิทธิภาพในการหาคำตอบกับเมต้าฮิวริสติกหลายวิธี วิธีที่นำเสนอมาใหม่นี้มีประสิทธิภาพสูงที่สุด

คำหลัก : เมตาฮิวริสติก แบบจำลองเซอโรเกท การหาค่าเหมาะที่สุดในงานวิศวกรรม เมตาฮิวริสติกแบบตัวแปรปรับตัวได้

Abstract

Project Code : MRG5980238

Project Title : Surrogate assisted meta-heuristics for engineering optimisation

Investigator : Assist. Dr. Nantiwat Pholdee

E-mail Address : nantiwat@kku.ac.th

Project Period : 2 Years

Abstract:

In this work, development of MHs for real world engineering optimisation is conducted based on using surrogated assisted MHs, using parameter adaption and using hybridization concepts. Firstly, performance enhancement of a teaching-learning based optimizer (TLBO) using an opposition-based approach, binary crossover, and the probability of operating the learning phase is proposed for strip flatness optimization during a coiling process. The results reveal that the proposed method gives a better optimum solution compared to the present state-of-the-art methods. Next, a self-adaptive sine cosine algorithm is proposed. The proposed algorithm is used to tackle the test problems for structural damage detection. The results reveal that the new algorithm outperforms a number of established meta-heuristics. In addition, new meta-heuristic called estimation of distribution algorithm using correlation between binary elements (EDACE) is proposed. The performance assessment is conducted by comparing the new algorithm with existing binary-code MHs. The comparative results show that the new algorithm is competitive with other established binary-code meta-heuristics. Finally, the integration of an inverse problem process using surrogate model into meta-heuristics (MHs) for performance enhancement in solving structural health monitoring optimisation problems is proposed. The surrogate model is integrated into the MH algorithm for generating an approximate solution rather than approximating the function value as with traditional surrogate-assisted optimisation. The results obtained from using various MHs and the proposed algorithms indicate that the new algorithm is the best for all test problems.

Keywords : Meta-heuristic algorithm, Surrogate model, Engineering Optimisation, Self adaptive meta-heuristic

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

LIST OF TABLES

LIST OF TABLES (Cont.)

LIST OF FIGURES

NOMENCLATURE

$[\mathbf{K}]$ = structural stiffness matrix

$[\mathbf{M}]$ = structural mass matrix

λ_j = j^{th} mode eigenvalue

ϕ_j = j^{th} mode eigenvector or mode shape.

n_{dof} = size of the mass and stiffness matrices.

$[\mathbf{m}_e]$= element mass matrices

$[\mathbf{k}_e]$ = element stiffness matrices.

n_e = number of elements

p_i = percentage of damage in the i^{th} element.

n_{mode} = number of lowest vibration modes

F = scaling factor

F_{min} = maximum scaling factor

F_{max}= minimum scaling factor

$\mathbf{x}_{r,i}$ = i^{th} randomly selected individual

$\mathbf{x}_{old}$ = current solution (parent)

$\mathbf{x}_{new}$ = new candidate solution

rand = uniform random number ranged from 0 to 1

rand(0,1)= random number, either 0 or 1

CR = crossover rate

D= number of design variables

c_k = interpolation coefficients

φ = RBF kernel function

$\boldsymbol{\omega}_{damage}$ = natural frequencies of the damaged structure (Target vector)

$\mathbf{x}_{damage}$ = solution vector containing n_e element damage percentages

Chapter I
Executive Summary

1.1 Rationale of the study

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 is traditionally called mathematical programming or gradient-based optimisers whereas the latter has various subcategories. One of them is a meta-heuristic (MH). The term meta-heuristics can cover nature-inspired optimisers, swarm intelligent algorithms, and evolutionary algorithms. 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 on how to improve the search performance of MHs, particularly for real engineering design. In this work, development of MHs for real world engineering optimsiation is conducted based on using surrogated assisted MHs, using parameter adaption and using hybridization concept.

1.2 Literature review

1.2.1 Meta-Heuristics

Meta-Heuristics (MHs), also known as evolutionary algorithms are optimisation methods which are mostly developed according to inspiration of physical law or natural phenomena such as genetic evolution, food finding of animal or insect, etc. A genetic algorithm (GA) [1] is probably the best known MH while other popular methods are differential evolution (DE) [2] and particle swarm optimisation (PSO) [3]. 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 [4, 5] and evolution strategies [6] were proposed. Then, DE and PSO were introduced. Up to recently, the have been probably over a hundred new real-code MHs in the literature. Some recent algorithms include, for example, a sine-cosine algorithm [7], a grey wolf optimiser [8], teaching-learning based optimisation [9] , a Jaya algorithm [10] etc. Meanwhile, powerful existing algorithms such as PSO and DE have been upgraded by integrating into them some types of self-adaptive schemes e.g. adaptive differential evolution with optional external archive (JADE) [11], Success-History Based Parameter Adaptation for Differential Evolution (SHADE) [12], SHADE Using Linear Population Size Reduction (LSHADE) [13] and adaptive PSO [14-16]. 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 non-dominated sorting genetic algorithm (NSGA-I, II, III) [17-19], multiobjective particle swarm optimisation [20], strength Pareto evolutionary algorithm [21], multiobjective grey wolf optimisation [22] , multi-objective teaching-learning-based optimization [23], multiobjective evolutionary algorithm based on decomposition [24], multiobjective ant colony optimisation [25], multiobjective differential evolution [26] etc. 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 [27], an improved two-archive algorithm [28], preference-inspired co-evolutionary algorithms [29] etc.

1.2.2 Surrogate assisted MHs

Surrogate models (also known as metamodels, or response surface models) are widely used in many kinds of applications in engineering design optimisation. The surrogate model is the approximation of an objective function by using a function with much less time-consuming compared to the actual function evaluation. By using such a model, only a few actual function evaluations are required for construction of the meta-model. The optimization process can be

carried out by using the approximate model which is adequately accurate not time-consuming. The commonly used surrogate model are such as polynomial response surface (PRS) [30], radial basic function (RBF) [31], Kriging (KG) [32], neural network (NN) [33], and support vector regression (SVR) [34], etc.

Recently, a surrogate model based on optimum tuning parameters has been proposed as an improved version of the traditionally used surrogate models. The idea of this proposed is to use some metaheuristics to find optimum tuning parameters of the surrogate model to improve their accuracy. The most successful investigations are reported in references [30, 35-47].

1.3 Objectives

1.2.1 To improve MH search performance based on improvement of a reproduction process for an application of practical engineering optimisation.

1.2.3 To proposed a novel and efficient MH for an application of practical engineering optimisation.

1.2.3 To improve MH search performance based on using a surrogate model for an application of practical engineering optimisation.

1.4 Scope of research

1.4.1 MHs will be coded by the MATLAB program.

1.4.2 Self-adaptive and/or hybridization concepts are used to enhance the search performance of MHs.

1.4.3 A surrogate model employed in this study is a radial-basis function.

1.4.4 An optimum Latin Hypercube sampling technique is used for generating sampling points

1.4.5 Both real code and binary code MHs are used in this study

1.5 Chapter outline

Chapter 2, performance enhancement of a teaching-learning based optimizer (TLBO) for strip flatness optimization during a coiling process is proposed. The method is termed improved teaching-learning based optimization (ITLBO). The new algorithm is achieved by modifying the

teaching phase of the original TLBO. The design problem is set to find spool geometry and coiling tension in order to minimize flatness defects during the coiling process. Having implemented the new optimizer with flatness optimization for strip coiling, the results reveal that the proposed method gives a better optimum solution compared to the present state-of-the-art methods.

Chapter 3, a sine cosine algorithm is extended to be self-adaptive and its main reproduction operators are integrated with the mutation operator of differential evolution. The new algorithm is called adaptive sine cosine algorithm integrated with differential evolution (ASCA-DE) and used to tackle the test problems for structural damage detection. The results reveal that the new algorithm outperforms a number of established meta-heuristics.

Chapter 4, a new meta-heuristic 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 meta-heuristics 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 the 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 meta-heuristics.

Chapter 5 proposes the integration of an inverse problem process using radial basis functions (RBFs) into meta-heuristics (MHs) for performance enhancement in solving structural health monitoring optimisation problems. A differential evolution (DE) algorithm is chosen as the MH for this study. In this chapter, RBF is integrated into the DE algorithm for generating an approximate solution rather than approximating the function value as with traditional surrogate-assisted optimisation. Four structural damage detection test problems of three trusses are used to examine the search performance of the proposed algorithms. The results obtained from using various MHs and the proposed algorithms indicate that the new algorithm is the best for all test problems. DE search performance for structural damage detection can be considerably improved by integrating RBF into its procedure.

Chapter II
An Improved Teaching-Learning Based optimization for Optimization of Flatness of a Strip during a Coiling Process

2.1 Introduction

In this chapter, optimization of flatness of the strips has been enhanced by an improved teaching-learning based algorithm (ITLBO). This method is compared to several well established EAs, such as simulated annealing (SA) [48], differential evolution (DE) [2], artificial bee colony optimization (ABC) [49], real code ant colony optimization (ACOR) [50], original teaching-learning based optimization (TLBO) [9], league championship algorithm (LCA) [51], charged system search (ChSS) [52], Opposition-based Differential Evolution Algorithm (OPDE) [53] and Enhanced teaching-learning based optimization with differential evolution (ETLBO-DE) [54] to determine the spool geometry and coiling tension where the objective is to minimize the axial inhomogeneity of the stress to improve the flatness of the strip. For function evaluations, the analytical elastic model proposed by Park et al. [55] similar to the one suggested by Yanagi et al. [56] was employed.

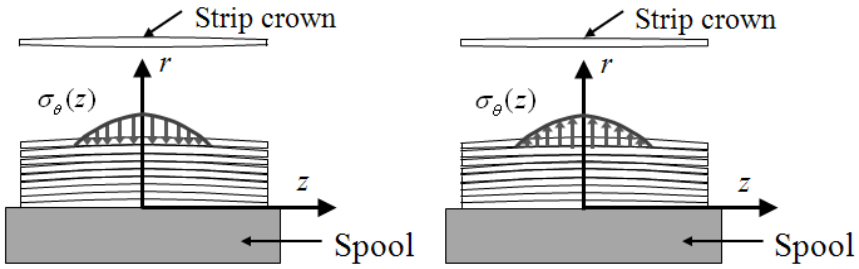
2.2 Formulation of the Optimization Design Problem

It is known that wavy edges occur during the strip coiling process, when the circumferential stress at the middle zone of the strip is highly compressed, while two edges are under tension or slight compression. Also, if the middle strip zone is under high tension while the two edges are compressed or slightly stretched, center buckle can happen. Figures 2.1(a) and (b) display the circumferential stress (σ_θ) distribution along the z direction within the thin strip, which respectively caused the wavy edge and center buckle.

Generally, it is impossible to obtain a flat strip after finishing a rolling process. The strip always has a crown shape. When the strips are being coiled, tension loads need to be applied, the middle zone (z = 0) of the strip at the inner coil will be considerably compressed in comparison with the two edges because of the coiling tension and the strip crown. In such a situation, the center buckle defect at the inner coil will not appear but the wavy edge defect can possibly occur. As such, the wavy edge defect at the inner coil is the major problem during the coiling process.

Figure 2.2 depicts the circumferential stress (σ_θ) distribution in the *z* direction at the radius (*r*) of the coil (computed by the Love's elastic solution proposed by Park et al. [9]) contributing to wavy edge defect formation during the strip coiling process. It is possible to reduce the wavy edge defect by decreasing the axial inhomogeneity of the stress distribution and the maximum compressive stress at the compressive zone.

In this paper, optimization using the ITLBO and other well-known and newly developed EAs will be used to find the optimum solution for the processing parameters including coiling tension (σ_T) and spool geometry, as illustrated in Fig. 2.3.



(a) the wavy edge (b) center buckle

Figure 2.1 Circumferential stress distributions for (a) the wavy edge and (b) center buckle, respectively

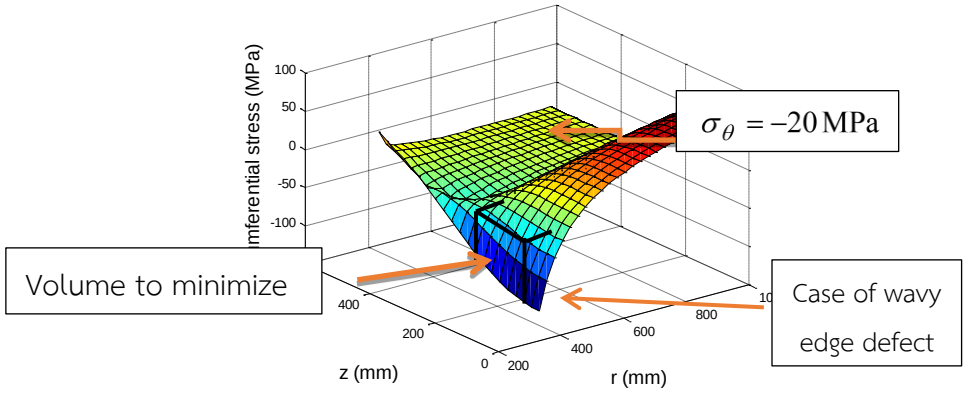


Figure 2.2 Circumferential stress distribution (σ_θ) in the coil determined by Love's elastic solution

To decrease the axial inhomogeneity of the stress distribution and the maximum compressive stress, minimization of the volume of the circumferential stress and maximum

compressive stress (shown in Fig. 2.2) is defined as an objective function. In Fig. 2.2, the volume can only be computed for the coil, where compressive stresses were higher than 20 MPa, in order to minimize the zone that is likely to have the wavy edge defect. The objective function of the optimization problem can then be written as:

Minimize $$f(\alpha_b, \eta_b, \sigma_{T,i}) = \frac{V}{V_0} + \frac{\max(\sigma_{\theta c})}{\max(\sigma_{\theta c0})} \quad (2.1)$$

Subject to

$$0 \le \alpha_b \le 4,$$
$$0 \le \eta_b \le 4,$$
$$25 \le \sigma_{T,i} \le 50\text{MPa}; \quad i = 1, \ldots, n_{\max},$$
$$\left|\sigma_{T,i} - \sigma_{T,i-1}\right| \le 2\text{MPa},$$

where $\sigma_{\theta c}$ and V are respectively the compressive circumferential stress higher than 20 MPa (refer to Fig. 2.2) and the approximate volume of the circumferential stress. $\sigma_{\theta c0}$ and V_0 are the respective values for the original design of the process. The $\sigma_{T,i}$ is the coiling tension at coil number *i*. The coiling tension is normally set to be constant for all coils. The variable $n_{\max}$ is the maximum number of coils, which has been assigned to be 220 in this paper. η_b and α_b in Eq. (2.2) are spool crown exponent and the spool crown height, which were used for defining the spool geometry, as described in Fig. 2.3:

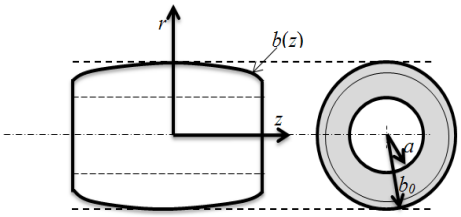


Figure 2.3 Spool Geometry used in the present investigation

$$b(z) = b_0 - \alpha_b \left(\frac{|z|}{z_{\max}} \right)^{\eta_b} \quad (2.2)$$

where b_0 (z = 330 mm) and $b(z)$ are the initial value of the outer radius of the spool and the outer radius of the spool along the z direction, respectively. z_{max} = 525 mm is the width of the spool. The inner radius of the spool (a) in Fig. 2.3 has been assigned to be 300 mm. The total number of design variables, therefore, is 222 (220 for coiling tensions and 2 for the spool geometry).

2.3 Improved teaching-learning based optimization

From the previous section, the optimization problem can be considered being large-scale. It has been found [53, 54], that TLBO is suitable for this type of design problem. The teaching-learning based optimization (TLBO) algorithm is an evolutionary algorithm, or an optimizer without using function derivatives, proposed by Rao et al. [9]. The concept of TLBO searching mechanism is based on mimicking a teacher on the output of learners in a classroom. Basically, the learners can improve their intellectual and knowledge by two stages i.e. learning directly from the teacher and learning among themselves. During the teacher stage, a teacher may teach the learners, however, only some learners can acquire all things presented by the teacher. Those who can accept what the teacher taught will improve their knowledge. For the second stage, which is called the learning phase, the learners can improve their knowledge during discussion with other learners. Based on the different levels of the learners' knowledge, the better learners may transfer knowledge to the inferior learners.

From the view point of optimization, the algorithm starts with a randomly created initial population, which is a group of design solutions. Learners are identical to design solutions whereas the best one is considered a teacher. The objective function is analogous to the knowledge which needs to be improved towards the optimum solution. Having identified a teacher and other learners for the current iteration, the population will be updated by two stages including "Teacher Phase" and "Learner Phase". In the "Teacher Phase", an individual ($\mathbf{x}_i$) will be updated based on the best individual ($\mathbf{x}_{teacher}$) and the mean values of all populations ($\mathbf{x}_{mean}$) as follows:

$$\mathbf{x}_{\text{new},i} = \mathbf{x}_{\text{old},i} + r\{\mathbf{x}_{\text{teacher}} - (T_F \cdot \mathbf{x}_{\text{mean}})\} \tag{3}$$

Where T_F is a teaching factor, which can be either 1 or 2 and $r \in [0,1]$ is a uniform random number.

For the "Learner Phase", the members in the current population will be modified by exchanging information between themselves. Two individuals $\mathbf{x}_i$ and $\mathbf{x}_j$ will be chosen at random, where $i \neq j$. The update of the solutions can then be calculated as:

$$\mathbf{x}_{\text{new},i} = \begin{cases} \mathbf{x}_{\text{old},i} + r\left(\mathbf{x}_i - \mathbf{x}_j\right) & \text{if } f\left(\mathbf{x}_i\right) < f\left(\mathbf{x}_j\right) \\ \mathbf{x}_{\text{old},i} + r\left(\mathbf{x}_j - \mathbf{x}_i\right) & \text{if } f\left(\mathbf{x}_j\right) < f\left(\mathbf{x}_i\right) \end{cases} \tag{4}$$

At both teacher and learner phases, the new solution ($\mathbf{x}_{\text{new}}$) will replace its parent if it has better knowledge or produces better objective function value, otherwise, it will be rejected. The two phases are sequentially operated until the termination criterion is fulfilled.

For the improved teaching-learning based optimization (ITLBO), an opposition-based approach, binary crossover, and the probability of operating the learning phase are added to the original TLBO to improve the balance of search exploration and exploitation. Four random numbers including, $rand_1$, $rand_2$, $rand_3$, and $rand_4$, have been used for performing opposition-based approach, binary crossover, and the learning phase. The main search procedure starts by generating an initial population, updating the population at the teaching phase and learning phase similarly to the original TLBO. However, at the teaching phase, the updating can be done by the following equation;

$$\mathbf{x}_{\text{new},i} = \mathbf{x}_{\text{old},i} + (-1)^{rand_1} r\left\{\mathbf{x}_{\text{teacher}} - \left(T_F \cdot \mathbf{x}_{\text{mean}}\right)\right\} \tag{5}$$

where $rand_1$ is a random value with either 0 or 1. Then, the binary crossover is applied if a uniform random number having an interval of 0 and 1 ($rand_2$) is lower than the crossover probability (P_r). For a new individual $\mathbf{x}_{\text{new}}^T = [x_{\text{new},1}, \ldots, x_{\text{new},D}]$ and an old individual $\mathbf{x}_{\text{old}}^T = [x_{\text{old},1}, \ldots, x_{\text{old},D}]$, the binary crossover step can be expressed as follow;

$$x_{new,j} \begin{cases} x_{old,j} & \text{if } rand_3 < CR_1 \quad j = 1,\ldots,D \\ x_{teacher,j} & \text{if } CR_1 \leq rand_3 < CR_2 \quad j = 1,\ldots,D \end{cases} \tag{6}$$

where the $rand_3$ is a uniform random number generated from 0 to 1. The CR_1 and CR_2 are the predefined crossover rates, while D is the number of design variables, respectively. Thereafter, the learning phase is conducted if a uniform random number generated from 0 to 1 ($rand_4$) is lower than the probability value (L_p), otherwise, the learning phase will be skipped. The search process will be repeated until the termination criterion is satisfied. The computational steps of the proposed algorithm are shown in Algorithm 2.1.

Algorithm 2.1 An improved TLBO

Input: Maximum iteration number ($maxiter$), population size (n_P), Crossover probability Crossover rate (CR_1 and CR_2), learning phase probability (L_p).
Output: $\mathbf{x}_{best}$, f_{best}
Initialization
1. Generate an initial population randomly.
2. Evaluate objective function values
Main algorithm
3. For i =1 to $maxiter$
  3.1 Identify the best solution ($\mathbf{x}_{teacher}$)
   (Teacher Phase)
     For j=1 to n_p
     3.2 Update the population using equation (5)
         If $rand_2$ < P_r
             3.2.1 Applied binary crossover using equation (6)
         End
         3.2.1 Evaluate the objective function value $f(\mathbf{x}_{new,j})$
         3.2.2 If $f(\mathbf{x}_{new,j}) < f(\mathbf{x}_{old,j})$
                 Replace $\mathbf{x}_{old,j}$ by $\mathbf{x}_{new,j}$
         End
     End

     If $rand_4$ < L_p
     (Learner Phase)
     For j=1 to n_p
     3.3 Update the population using equation (4)
         3.3.1 Evaluate the objective function value
   $f(\mathbf{x}_{new,j})$
         3.2.2 If $f(\mathbf{x}_{new,j}) < f(\mathbf{x}_{old,j})$
                 Replace $\mathbf{x}_{old,j}$ by $\mathbf{x}_{new,j}$
         End
     End
End

2.4 Numerical Experiments

In order to examine the search performance of the proposed ITLBO, several EAs have been used to solve the optimum design problem of the strip flatness as described in the previous section. The EAs used in this study are as follows [57] :

DE: The DE/best/2/bin strategy was used. DE scaling factor was random from 0.25 to 0.7 in each calculation and crossover probability was 0.7.

SA: An annealing temperature was reduced exponentially by 10 times from the value of 10 to 0.001 in the optimization searching process. On each loop 2n children were created by means of mutation to be compared with their parent. Here, n is the number of design variables.

ABC: The number of food sources was set to be $3n_{\mathrm{p}}$. A trial counter to discard a food source was 100.

ACOR: The parameters used for computing the weighting factor and the standard deviation in the algorithm were set to be ξ =1.0 and q =0.2, respectively.

TLBO: Parameter settings are not required.

LCA: The default parameter settings provided by the authors [51] were used.

ChSS: The number of solutions in the charge memory was $0.2n_{\mathrm{p}}$. Here, n_{p} is the population size. The charged moving considering rate and the parameter PAR were set to be 0.75 and 0.5, respectively.

OPDE: The DE/best/2/bin strategy was used .DE scaling factor was random from 0.25 to 0.5 in each calculation and crossover probability used was 0.7.

ETLBO-DE: Used the DE parameter setting and Latin hypercube sampling (LHS) technique to generate an initial population.

ITLBO (Algorithm 2.2): The P_{r}, CR_1, CR_2 and L_{p} were set to be 0.5, 0.33, 0.66 and 0.75, respectively.

Each optimizer was employed to solve the problem for 5 optimization runs. Both the maximum number of iterations and population size were set to be 100. For the optimizers using different population sizes, such as simulated annealing, their search processes were stopped with the total number of function evaluations as 100×100. The optimal results of the various optimizers from using this limited number of function evaluations were compared. The best optimizer was used to find the optimal processing parameters of the strip coiling process.

2.5 Results and Discussion

After applying each optimization algorithm to solve the problem for 5 runs, the results are given in Table 2.1. The mean values (Mean) are used to measure the convergence rate while the standard deviation (STD) determines search consistency. The lower the mean objective function value the better, and the lower the standard deviation the more consistent. In the table, max and min stand for the maximum and minimum values of the objective function, respectively.
For the measure of convergence speed based on the mean objective value, the best method is ITLBO while the second best and the third best performers are ETLBO-DE and OPDE, respectively. The worst results came from ABC. For the measure of search consistency based on STD, the best was also ITLBO while the worst was ABC, which was similar to the measure of the search convergence. The second best and the third best for consistency were ETLBO-DE and ACOR, respectively. The minimum objective function value was obtained by the ITLBO.

Based on the results obtained, it was clearly indicated that the proposed ITLBO by adding opposition based method, binary crossover, and learning phase probability can improve the search performance of the original TLBO for solving the optimization design problem of the strip coiling process.

The optimal spool crown exponent and height obtained are 1.0822 and 2.3645, respectively. The optimal distribution of coiling tensions as a function of coil numbers is shown in Fig. 2.4. The results reveal that the coiling tensions start with the highest value initially and then decrease when the number of coils increases. After a few series of coiling, the tension levels become almost constant, converging to the lower bound at the end of the process. Fig. 2.5 shows the plot of the circumferential stress distributions along the z and r directions of the original and optimum design solutions in that order. The comparison of the maximum compressive stresses and the standard deviation of stresses at the inner strip between the original and optimal designs is given in Table 2.2. The results show that the optimal processing parameters obtained by the proposed ITLBO algorithm can reduce the maximum compressive stress and the axial inhomogeneity of the stress distribution at the inner strip, which might cause undesirable wavy edge defects during the strip coiling process.

Table 2.1. Objective function values calculated

Evolutionary Algorithms	Mean	STD	Max.	Min.
DE	0.9700	0.0275	1.0096	0.9354
ABC	1.7637	0.0787	1.8800	1.6751
ACOR	1.0621	0.0070	1.0705	1.0546
ChSS	1.4026	0.0289	1.4448	1.3678
LCA	1.7116	0.0408	1.7580	1.6473
SA	1.5451	0.0645	1.6323	1.4841
TLBO	0.9915	0.0132	1.0066	0.9766
OPDE	0.9539	0.0179	0.9715	0.9297
ETLBO-DE	0.8850	0.0047	0.8897	0.8784
ITLBO	0.8740	0.0025	0.8783	0.8720

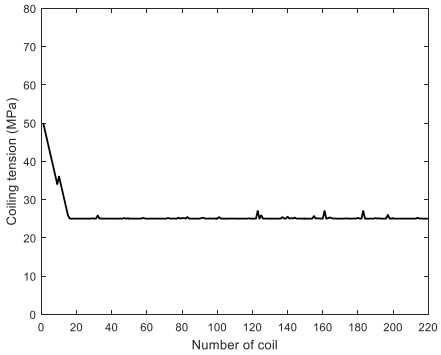
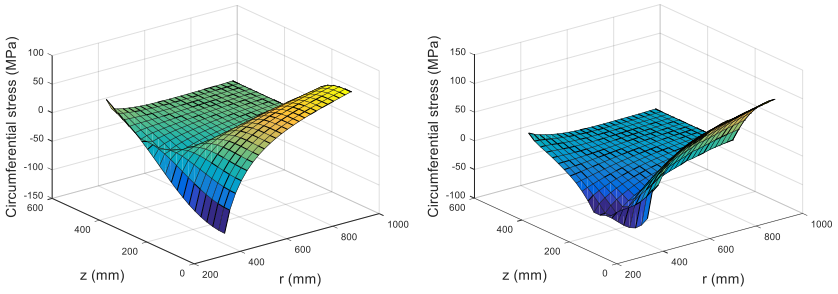


Figure 2.4 Coiling tension levels as a function of number of coils



(a) Original design (b) Optimal design

Figure 2.5 Comparison of circumferential stresses along the z and r directions for the original design and optimal design, respectively

Table 2.2 Maximum compressive stress and the standard deviation of stresses at the inner coil

	Original design	**Optimal design**
Maximum compressive stress (MPa)	111.546	68.0270
Standard deviation of stresses	48.375	29.3703

2.6 Conclusions

The new population-based optimization algorithm obtained by improving the original TLBO for solving the flatness optimization of the strip coiling process has been proposed. The search performance of the method was compared to various established evolutionary algorithms. The numerical results show that the new optimizer ITLBO is the best performer for both convergence rate and consistency. With this, the new parameters including the spool geometry and the coiling tension distribution have been obtained and can be used in the real strip coiling process. Further studies will be made to enhance the mathematical model of the strip coiling process. A self-adaptive version of ITLBO will be investigated for search performance enhancement.

Chapter III

Adaptive Sine Cosine Algorithm Integrated with Differential Evolution for Structural Damage Detection

3.1 Introduction

This chapter presents and extension of the sine cosine algorithm. An adaptive strategy is embedded into the new version while the mutation operator of differential evolution is integrated into the algorithm in order to further improve its performance. The new optimiser is then termed an adaptive sine cosine algorithm integrated with differential evolution (ASCA-DE). The optimiser is then implemented on several test problems for structural damage detection. Numerical results show that the proposed MH is superior to a number of established MHs found in the literature.

3.2 Formulation of a Damage Detection Optimization problem.

In this work, vibration based damage detection based on using natural frequencies is used for damage localization of truss structures. The main concept of using structural natural frequencies for damage detection of a truss structure is based on using a finite element model and the measured natural frequencies. When the natural frequencies and mode shapes are measured (usually the lowest n_{mode} natural frequencies), the finite element model is updated until the computed natural frequencies fit well with the measured ones. For the undamaged structure, natural frequencies can be calculated from a simple linear undamped free vibration finite element model which can be expressed as;

$$[\mathbf{K}]\{\phi_j\} - \omega_j^2[\mathbf{M}]\{\phi_j\} = 0 \tag{3.1}$$

where [**K**] is a structural stiffness matrix which can be expressed as the summation of element stiffness matrices [$\mathbf{k}_e$],

$$[\mathbf{K}] = \sum_{i=1}^{n_e}[\mathbf{k}_e] \tag{3.2}$$

where i is the i^{th} element of the structure while n_e is the total number of elements. The matrix [**M**] is a structural mass matrix computed in similar fashion to the stiffness matrix. The variables ϕ_j and ω_j are the j^{th} mode shape and its corresponding natural frequency, respectively. For the damaged structure, the stiffness matrix of the damaged element is assumed to be modified. The stiffness matrix of the damaged structure [$\mathbf{K}_d$] can be written as a percentage of damage in the elements as follows:

$$[\mathbf{K}_d] = \sum_{i=1}^{n_e} \frac{100 - p_i}{100} [\mathbf{k}_e] \tag{3.3}$$

where p_i is the percentage of damage on the i^{th} element. The natural frequency of the damaged structure can be computed by solving eq. (3.1) by replacing [**K**] with [$\mathbf{K}_d$].

The percentage of damage in the structural element (p_i) can be found by solving an optimisation problem to minimise the root mean square error (RMSE) between natural frequencies measured from the damaged structure and natural frequencies computed by using the finite element model. The problem can be expressed as follow:

$$\text{Min}: f(\mathbf{x}) = \sqrt{\frac{\sum_{j=1}^{n_{mode}} \left(\omega_{j,damage} - \omega_{j,computed}\right)^2}{n_{mode}}} \tag{3.4}$$

where $\omega_{j,damage}$ and $\omega_{j,computed}$ are the structural natural frequency of mode j obtained from a damaged structure and that from solving (3.1) – (3.3). The design variables are those damage percentages of structural elements ($\mathbf{x} = \{p_1, \ldots, p_{nele}\}^T$) respectively. In this work, six vibration modes are used for calculation.

3.3 Test problems with trusses

Four truss damage detection optimisation problems from two truss structures are used in this study. These are the test problems used in our previous studies [58]. Detail of the test problems are shown as follow:

3.3.1 Twenty-five-bar truss

The structure is shown in Fig. 3.1. The cross sections of all bar elements are set to be 6.4165 mm^2. Table 3.1 shown the material properties and simulated case study for this example. The data of natural frequencies of the undamaged and damaged 25-bar truss structures are shown in Table 3.2.

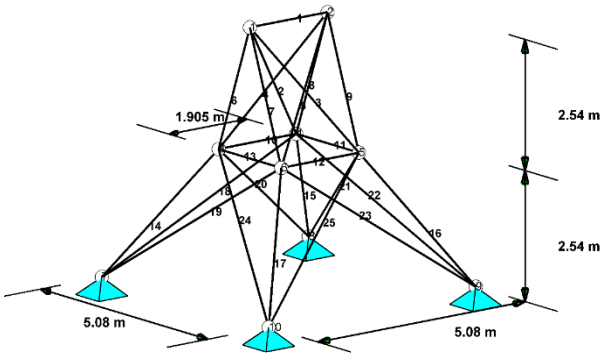


Figure 3.1 Twenty-five bar truss

Table 3.1 Material properties and simulated case study for 25-bar truss.

Material density	7,850 kg/m^3
Modulus of elasticity	200 GPa
Simulated case study	Case I: 35 %damage at element number 7 Case II: 35 %damage at element number 7 and 40 %damage at element number 9.

Table 3.2 Natural frequencies (Hz) of damaged and undamaged of 25 bar structure.

Mode	Undamaged	35 %damage at element number 7	35 %damage at element number 7 and 40 %damage at element number 9
1	69.7818	69.1393	68.5203
2	72.8217	72.2006	71.3167
3	95.8756	95.3372	94.5625
4	120.1437	119.8852	119.6514
5	121.5017	121.4774	121.4253

6	125.0132	125.0130	125.0129

3.3.2 Seventy-two-bar truss

The structure is shown in Fig. 3.2. Four non-structural masses of 2270 kg are attached to the top nodes. The cross sections of all bar elements are set to be 0.0025 m^2. Table 3.3 shown the material properties and simulated case study for this example. The data of natural frequencies of the undamaged and damaged 72-bar truss structure are shown in Table 3.3.

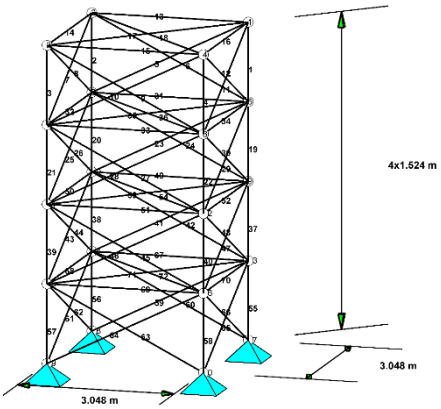


Figure 3.2 Seventy-two bar truss

Table 3.3 Material properties and simulated case study for 72-bar truss.

Material density	2,770 kg/m^3
Modulus of elasticity	6.98×10^{10} Pa
Simulated case study	Case I: 15 %damage at element number 55 (15% damage in element number 56, 57, or 58 results in the same set of natural frequencies) Case II: 10 %damage at element number 4 and 15 %damage at element number 58 (90, 180, and 270 degrees rotation along the *z* axis lead to the same set of natural frequencies).

Table 3.4 Natural frequencies (Hz) of damaged and undamaged of 72 bar structure.

Mode	Undamaged	15 %damage at element number 55	15 %damage at element number 58 and 10 % damage at element number 4
1	6.0455	5.9553	5.9530
2	6.0455	6.0455	6.0455
3	10.4764	10.4764	10.4764
4	18.2297	18.1448	18.0921
5	25.4939	25.4903	25.2437
6	25.4939	25.4939	25.4939

3.4 Adaptive Sine Cosine algorithm hybridized with differential evolution (ASCA-DE)

The Sine Cosine Algorithm (SCA) is a population based optimisation method proposed by Mirjalili, 2016 [7]. The algorithm is simple and efficient for various optimisation test problems as reported in [7]. The search procedure of SCA is similar to other MH which contains three main steps; population initialisation, population updating and population selection. For the SCA, updating population can be done based on a sine and cosine function. Given a current population having *NP* members $\mathbf{X}=\{\mathbf{x}_1, \mathbf{x}_2,..., \mathbf{x}_{NP}\}^T$, an element of a solution vector for the next generation can be calculated as follows:

$$x_{new,k} = \begin{cases} x_{old,k} + r_1 \sin(r_2)\left|r_3 x_{best,k} - x_{old,k}\right|, & if \quad r_4 < 0.5, \\ x_{old,k} + r_1 \cos(r_2)\left|r_3 x_{best,k} - x_{old,k}\right|, & otherwise \end{cases} \tag{3.5}$$

where $x_{best,k}$ is the k^{th} matrix element of the current best solution. The variables r_2, r_3, and r_4 are random parameters in the ranges of [0, 2π], [0, 2] and [0, 1], respectively. The variable r_1 is an iterative adaption parameter,

$$r_1 = a - T\frac{a}{T_{\max}} \tag{3.6}$$

where a is a constant parameter while T is an iteration number. T_{max} is maximum number of iterations.

The search process of SCA start with generating an initial population at random, and then calculating their objective function values where the best solution is found. Then, the new population for the next generation is generated using eq. (3.5) and the objective function values of its members are calculated. The current best will be compared with the best solution of the newly generated population and the better one is saved to the next generation. The process is repeated until a termination criterion is met. The computational steps of SCA are shown in Algorithm 3.1

Algorithm 3.1 Sine Cosine Algorithm

Input: population size (N_p), number of generations (T_{max}), number of design variable (D)

Output: $\mathbf{x}_{best}$, f_{best}

Main algorithm

1: Initialise a population and set as the current population.
2: Find the best solution ($\mathbf{x}_{best}$)
3: For T=1 to T_{max}
4: Calculate parameter r_1 using eq.(3.6)
5: For l=1 to N_p
6: For k = 1 to D
7: Generate the parameter r_2, r_3 and r_4
8: Update the k^{th} element of the l^{th} population ($\mathbf{x}_l$) using eq.(3.5)
9: End For
10: End For
11: Calculate objective function values of the newly generated population and find the best ones ($\mathbf{x}_{best,new}$)
12: Replace $\mathbf{x}_{best}$ by $\mathbf{x}_{best,new}$ if $f(\mathbf{x}_{best,new}) < f(\mathbf{x}_{best})$
13: End

For the proposed adaptive sine cosine algorithm with integration of DE mutation, the DE mutation operator as proposed in Bureerat and Pholdee (2015) [59] is integrated into the updating operation. The mutation equation is detailed as follow;

$$\mathbf{x}_{new} = \mathbf{x}_{best} + rand(-1,+1)F\left(\mathbf{x}_{r,1} + \mathbf{x}_{r,2} - \mathbf{x}_{r,3} - \mathbf{x}_{r,4}\right) \tag{3.7}$$

where *rand*(-1, 1) gives either -1 or 1 with equal probability. F is a scaling factor while $\mathbf{x}_{r,1}$–$\mathbf{x}_{r,4}$ are four solutions randomly selected from the population.

At ASCA-DE updating operation, if a generated uniform random number in the interval [0,1] is lower than a probability value ($rand < P_{DE}$), the population will be updated using the SCA updating operation based on Eq. (3.5), otherwise, the population will be updated by DE mutation as detailed in Eq. (3.7).

The term of self-adaption of the proposed algorithm is accomplished in such way that the parameter r_2, r_3 and F are regenerated for each calculation based on the information from the previous iteration. For each calculation, the r_2 and r_3 are generated based on normally distributed random numbers with mean values, r_{2m} and r_{3m} respectively and standard deviation values, STD = 0.1 for both r_2 and r_3. The values of r_{2m} and r_{3m} are iteratively adapted based on the following equations:

$$r_{2m}(T+1) = 0.9r_{2m}(T) + 0.1mean(good_{r2m}), \tag{3.8}$$

and,

$$r_{3m}(T+1) = 0.9r_{3m}(T) + 0.1mean(good_{r3m}), \tag{3.9}$$

where $mean(good_{r2m})$ and $mean(good_{r3m})$ are the mean values of all values of r_2 and r_3 used in current iteration that lead to successful updates. The successful update means the created offspring is better than its parent from the previous iteration. In addition, for each calculation, the scaling factor F is generated by Cauchy distribution randomisation with the mean value F_m and STD value of 0.1 [12]. The F_m is iteratively adapted using the Lehmer mean [12] defined as follows:

$$F_m(T+1) = 0.9F_m(T) + 0.1\frac{sum(good_F^2)}{sum(good_F)} \tag{3.10}$$

where $good_F$ is a tray of all F used in the current iteration with successful updates.

The parameter P_{DE} is also regenerated in the similar fashion to r_2 and r_3 before updating a population. For an individual solution, the P_{DE} is generated by normal distribution randomising with the mean value of P_{DEm} and standard deviation of 0.1. P_{DEm} is iteratively adapted based on the following equation:

$$P_{DE\mathrm{m}}(T+1) = 0.9\ P_{DE\mathrm{m}}\ (T)+0.1mean(good_{\mathrm{PDE}}), \tag{3.11}$$

where $good_{PDE}$ means all P_{DE} values used in the current iteration with successful updates.

The search process of ASCA-DE start with initilaising a population, r_{2m}, r_{3m}, F_m and P_{DEm}. The $good_{r2m}$, $good_{r3m}$, $good_F$ and $good_{PDE}$ trays are empty initially. After having calculated objective function values, the current best solution will be obtained. To update a population, firstly, P_{DE} and a random number in [0,1] are generated. If the generated random number is lower than P_{DE}, a scaling factor (F) is generated based on F_m and a new solution is created using eq. (3.7), otherwise, a new solution is generated based on eq. (3.5). For each calculation of eq. (3.5), r_2 and r_3 are generated based on r_{2m} and r_{3m}. If a newly generated solution is better than its parent, the new solution will be selected for the next generation while saving all used parameters P_{DE}, r_2, r_3 and F into the $good_{PDE}$, $good_{r2m}$, $good_{r3m}$, and $good_F$ trays, respectively. Then, update the r_{2m}, r_{3m}, F_m and P_{DEm} using eq. (3.8) - (3.11). The search process is repeated until a termination criterion is reached. The computational steps of ASCA-DE are shown in Algorithm 3.2

Algorithm 3.2 ASCA-DE

Input: population size (N_p), number of generations (T_{max}), number of design variable (D)

Output: $\mathbf{x}_{best}$, f_{best}

Main algorithm

1: Initialise a population, r_{2m}, r_{3m}, F_m and P_{DEm}.

2: Find the best solution ($\mathbf{x}_{best}$)

3: *For* T=1 to T_{max}

4: Calculate parameter r_1 using eq.(3.6)

5: Empty $good_{r2m}$, $good_{r3m}$, $good_{F}$ and $good_{PDE}$

5: *For* l=1 to N_p

6: Generate P_{DE} by normal distribution random with mean values P_{DEm} and STD =0.1

7: *IF rand*< P_{DE}

8: Generate F by Cauchy distribution random with mean value F_m and STD = 0.1

9: Updated a population using eq. (3.7)

10: *Else*

11: *For* k = 1 to D

12: Generate the parameter r_2 and r_3 by normal distribution random with mean values r_{2m}, r_{3m}, and STD =0.1

13: Random generate r_4 in rank [0, 1]

14: Update the k^{th} element of the l^{th} population ($\mathbf{x}_l$) using eq.(3.5)

14: *End For*

16: *End IF*

17: Calculate objective function values of the newly generated population

18: *IF* $f(\mathbf{x}_{l,new}) < f(\mathbf{x}_{l,old})$

19: Replace $\mathbf{x}_{l,old}$ by $\mathbf{x}_{l,new}$

20: Add all generated r_2, r_3, F, and P_{DE}, into the $good_{r2m}$, $good_{r3m}$, $good_{F}$ and $good_{PDE}$ tray, respectively.

21: *End IF*

22: *End For*

23: Find the best solution ($\mathbf{x}_{best}$)

24: Update r_{2m}, r_{3m}, F_m, and P_{DEm} using eq. (3.8) - (3.11)

24: End

3.5 Numerical Experiment

The performance investigation of the proposed ASCA-DE for structural damage detection is carried out by employing the algorithm to solve the test problems in the previous section. ASCA-DE along with a number of MHs in the literature implemented to solve the test problems

include (Note that the details of variables can be found in the original sources of each method) [58]:

Differential evolution (DE): a DE/best/2/bin strategy was used. A scaling factor, and probability of choosing elements of mutant vectors (*CR*) are 0.5 and 0.8 respectively.

Artificial bee colony algorithm (ABC): The number of food sources for employed bees is set to be $n_P/2$. A trial counter to discard a food source is 100.

Real-code ant colony optimisation (ACOR): The parameter settings are q = 0.2, and ξ = 1.

Charged system search (ChSS): The number of solutions in the charge memory is $0.2 \times n_P$. The charged moving considering rate and the parameter *PAR* are set to be 0.75 and 0.5 respectively.

League championship algorithm (LCA): The probability of success P_c and the decreasing rate to decrease P_c are set to be 0.9999 and 0.9995, respectively.

Simulated annealing (SA): Starting and ending temperatures are 10 and 0.001 respectively. For each loop, n_{mode} candidates are created by mutating on the current best solution while other n_{mode} candidates are created from mutating the current parent. The best of those $2n_{mode}$ solutions are set as an offspring to be compared with the parent.

Particle swarm optimisation (PSO): The starting inertia weight, ending inertia weight, cognitive learning factor, and social learning factor are assigned as 0.5, 0.01, 0.5 and 0.5 respectively.

Evolution strategies (ES): The algorithm uses a binary tournament selection operator and a simple mutation without the effect of rotation angles.

Teaching-learning-based optimisation (TLBO): Parameter settings are not required.

Adaptive differential evolution (JADE): The parameters are self-adapted during an optimisation process.

Evolution strategy with covariance matrix adaptation (CMAES): The parameters are self-adapted during an optimisation process.

Sine Cosine Algorithm (SCA) (Algorithm 3.1): The constant a parameter is set to be 2.

Adaptive Sine Cosine algorithm with integrating DE mutation (ASCDE) (Algorithm 3.2): The parameter a is set to be 2 while initial r_{2m}, r_{3m}, F_m and P_{DEm} are set to be 0.5.

Each optimiser is used to tackle each truss damage detection test problem for 30 optimisation runs. The number of iterations (generations) is 300 for all case studies while the population size is set to be 30 and 50 for 25-bar and 72-bar trusses respectively. All methods will be terminated with two criteria: the maximum numbers of functions evaluatio as 30×300 and 50×300 for the 25-bar and 72-bar trusses respectively, and the objective function value being less than or equal to 1×10^{-3}. The six lowest natural frequencies (n_{mode} = 6) are used to compute the objective function value. This number of selected frequencies is reasonable since it is practically easier to measure fewer lowest natural frequencies with sufficient accuracy.

3.6 Results and discussions

After performing 30 optimisation runs of all MHs on solving the four truss damage detection optimisation problems, the results obtained from the various MHs are given in Tables 5-8. The mean of the objective function is used to indicate the search convergence of the algorithms in cases that the objective function threshold (1×10^{-3}) is not met during searching. Otherwise, the mean number of FEs is used as an indicator. The number of successful runs out of 30 runs is used to measure the search consistency. The algorithm that is terminated by the objective function threshold is obviously superior and any run being stopped with this criterion is considered a successful run.

3.6.1 Twenty-five-bar truss

Table 3.5 shown the results of the 25-bar truss with 35% damage at element 7. The best performer based on the mean objective function values is ASCA-DE while the second best and the third best are DE and JADE respectively. When considering the number of successful runs, seven optimisers including DE, TLBO, JADE, SCA and ASCA-DE can detect the damage of the structure. The most efficient optimiser is ASCA-DE that can detect the damages of the structure for 29 times out of 30 runs with the average of 2835 function evaluations.

Table 3.5 Results for 25 bar truss with 35 %damage at element number 7

Optimisers	Mean objective function values	No. of successful runs from 30 runs	Mean of FEs
DE	0.0017	19	6019
ABC	0.0135	0	9000
ACOR	0.0089	0	9000
ChSS	0.1385	0	9000
LCA	0.9036	0	9000
SA	0.0089	0	9000
TLBO	0.0077	6	7772
CMAES	0.0033	0	9000
ES	0.0308	0	9000
PSO	8.3830	0	9000
JADE	0.0026	2	8953
SCA	0.0270	24	3262
ASCA-DE	0.0009	29	2835

Results of the 25 bar truss with 35% damage at element 7 and 40% damage at the element number 9 are reported in Table 3.6. The best performer based on mean objective function values is ASCA-DE while the second best and the third best are JADE and DE respectably. When examining the number of successful runs, only two optimisers, DE and ASCA-DE can consistently detect the damage of the structure for 27 and 26 runs respectively while the average number of function evaluations to obtain the results are 5220 and 5511 respectively.

Table 3.6 Results for 25 bar truss with 35 %damage at element number 7 and 40 %damage at element number 9

Optimisers	Mean objective function values	No. of successful runs from 30 runs	Mean of FEs
DE	0.0096	27	5220
ABC	0.0326	0	9000

ACOR	0.0125	0	9000
ChSS	0.1590	0	9000
LCA	0.8080	0	9000
SA	0.0269	0	9000
TLBO	0.0405	1	8917
CMAES	0.0115	0	9000
ES	0.0356	0	9000
PSO	8.6012	0	9000
JADE	0.0042	6	8875
SCA	0.0930	0	9000
ASCA-DE	0.0032	26	5511

3.6.2 Seventy-two-bar truss

Table 3.7 shows comparison results of the 72-bar truss with 15% damage at element 5. The best performer based on mean objective function values is ASCA-DE while the second best and the third best are ES and ACOR. When examining the number of successful runs, the most efficient method is ASCA-DE which can detect the damage of the structure for 30 times while the average numbers of function evaluations for the convergence results is only 1715.

Table 3.7 Results for 72 bar truss with 15 %damage at element number 55

Optimisers	Mean objective function Values	No. of successful runs from 30 runs	Mean of FEs
DE	0.0087	14	12887
ABC	0.2184	0	15000
ACOR	0.0014	6	14831
ChSS	0.1727	0	15000
LCA	1.1499	0	15000
SA	0.0097	0	15000
TLBO	0.0035	27	5781
CMAES	0.0053	0	15000

ES	0.0010	29	9335
PSO	1.9146	0	15000
JADE	0.0019	1	15000
SCA	0.0070	23	4793
ASCA-DE	0.0008	30	1715

Results of the72 bar truss with 15% damage at element number 58 and 10% damage at element number 4 are given in Table 3.8. The best performer based on the mean of objective function values is ES while the second best and the third best are JADE and ASCA-DE respectively. The minimum objective function value is obtained by SCA. When considering the number of successful runs, only ASCA-DE can consistently detect the damage of the structure for 22 times from totally 30 optimisation runs while the average number of function evaluations for the convergence results is 9235. Although ES and JADE given better mean objective function values, they fail to search for the damage location. ASCA-DE is said to be the most efficient optimizer for this case.

Table 3.8 Results for 72 bar truss with 15 %damage at element number 58 and 10 %damage at element number 4

Optimisers	Mean objective function values	No. of successful runs from 30 runs	Mean of FEs
DE	0.0127	7	13963
ABC	0.1591	0	15000
ACOR	0.0058	0	15000
ChSS	0.1348	0	15000
LCA	1.1049	0	15000
SA	0.0129	0	15000
TLBO	0.0045	7	13503
CMAES	0.0050	0	15000
ES	0.0023	2	14940
PSO	1.7726	0	15000

JADE	0.0031	0	15000
SCA	0.0260	2	14502
ASCA-DE	0.0035	21	9235

Overall, it was found that integrating DE mutation into and applying adaptive parameters to SCA lead to performance enhancement of the original SCA. The proposed ASCA-DE is the best performer on solving truss damage detection optimisation problem. It is considered the most reliable method for this study.

3.7 Conclusions

Performance enhancement of a meta-heuristics called a sine cosine algorithm is proposed by integrating into it a mutation strategy of DE. Self-adaptive optimisation parameters are employed to improve the search performance of the new algorithm. The proposed optimiser is implemented on solving a number of truss damage detection inverse problems. The results reveal that the new meta-heuristic is the best and most reliable method. Our future work is to investigate the new MH for solving other practical engineering design problems.

Chapter IV

Estimation of distribution algorithm using correlation between binary elements – a new binary-code meta-heuristic

4.1 Introduction

This chapter presents a development of a binary-code meta-heuristic. 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.

4.2 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:

$$\text{Min } f(\mathbf{x}); \ \mathbf{x}_L \le \mathbf{x} \le \mathbf{x}_U \tag{4.1}$$

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 [60]. 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 4.1 where $\mathbf{b}$ is a binary design solution, $\mathbf{b}_{REF}$ is the reference binary solution, n_P is a population size and $rand \in [0, 1]$ 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, $\mathbf{b}_{REF}$. 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.

Algorithm 4.1 Generation of a binary population from a CBE matrix

Input:$\mathbf{b}_{REF}$, **P**

Output: **B** = {$\mathbf{b}^i$} for i =1, .., n_P

Main procedure

1: Set **B** = {}.

2: For i = 1 to nP

3: Set **a** = {} a vector used to contain elements of a generated binary string.

4: Randomly select a position (*r*-th row) of **P**.

5: Set $a_r = b_{REF, r.}$% Set the *r*-th element of a as the *r*-th element of $\mathbf{b}_{REF}$.

6: For j = {1, 2, ..., m} – {r}

7: If $rand < P_{rj}$

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 **B** = **B** ∪ **a**.
14:End

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 ($\mathbf{b}_{best}$). The learning rate (L_R) with 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 $m(m\text{-}1)/2$ updates since P_{ii} is always set to be 1. The updated P_{ij} denoted by P'_{ij} can be calculated from

$$P'_{ij} = (1 - L_R)P_{ij} + L_R(1 - \left|b_{best,i} - b_{best,j}\right|) \tag{4.2}$$

where L_R is the learning rate randomly generated in the interval $[L_{R,L}, L_{R,U}]$. $b_{best,i}$ and $b_{best,j}$ are the i-th and j-th elements of $\mathbf{b}_{best}$ 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

$$0 \leq P_L \leq P_{ij} \leq P_U \leq 1. \tag{4.3}$$

where P_L and P_U are the predefined lower and upper limits of P_{ij}. Equation (4.3) 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 4.2. The decision on

selecting $\mathbf{b}_{REF}$ for generating a binary solution and $\mathbf{b}_{best}$ 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 $\mathbf{b}_{REF}$ and $\mathbf{b}_{best}$ are found. Then, new binary solutions are generated using Algorithm 4.1 while the greedy selection (steps 6-8) is activated with $\mathbf{b}_{REF}$ and $\mathbf{b}_{best}$ being determined. The CBE matrix is updated by using $\mathbf{b}_{best}$ as detailed in Equations (4.2) – (4.3). 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 [61] and binary quantum-inspired gravitational search algorithm (BQIGSA) [62] 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 the reference binary solution $\mathbf{b}_{REF}$. Apart from that, the update of CBE tend to be similar to the concept employed in PBIL with a learning rate and this is totally different from binary PSO and BQIGSA.

Algorithm 4.2 Procedure for EDACE

Input: number of generation (n_{iter}), population size (n_P), binary length (m)

Output: $\mathbf{b}_{best}$, 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 $f^i = fun(\mathbf{x}^i)$ where fun is an objective function evaluation.

0.4: Find f_{best}, $\mathbf{b}_{REF}$, $\mathbf{b}_{best}$

Main iterations

1: For $iter$ =1 to n_{iter}

2: Update $\mathbf{P}$ using Equation (4.2)

3: Generate $\mathbf{b}^i_{new}$ from $\mathbf{P}$ using Algorithm 1, and decode them to be $\mathbf{x}^i_{new}$.

4: For i =1 to n_P

5: Calculate objective function values $f^i_{new} = fun(\mathbf{x}^i_{new})$.

6: If $f^i_{new} < f_i$

7: $f_i = f^i_{new}$, $\mathbf{b}^i = \mathbf{b}^i_{new}$, $\mathbf{x}^i = \mathbf{x}^i_{new}$

8: End
9: End
10: Update f_{best}, $\mathbf{b}_{REF}$, $\mathbf{b}_{best}$
11: End

In selecting $\mathbf{b}_{REF}$ and $\mathbf{b}_{best}$, if both solutions are the same which is $\mathbf{b}_{best}$, 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 $\mathbf{b}_{REF}$ to maintain intensification. For updating the CBE matrix, we use the new updating scheme as

$$P'_{ij} = (1 - L_R)P_{ij} + L_R\left(1 - \left|b_{best1,i} - b_{best2,j}\right|\right) \tag{4.4}$$

The solutions $\mathbf{b}_{best1}$ and $\mathbf{b}_{best2}$ are two types of best solutions. Firstly, n_P best solutions are selected from $\{\mathbf{b}^i\} \cup \{\mathbf{b}^i_{new}\}$ (see Algorithm 4.2 for both solution sets), sorted according to their functions, and then saved to a set *Best_sol*. Four $m\times1$ 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. $\mathbf{b}_{best1}$ is randomly chosen from the aforementioned solutions ($\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$, and $\mathbf{b}_4$) with equal probability while $\mathbf{b}_{best2}$ is randomly chosen from the members of *Best_sol*. With this idea, the balance between exploration and exploitation is maintained throughout the search process. Algorithm 4.3 shows the new CBE updating strategy.

Algorithm 4.3 Updating scheme for CBE

Input: $L_{R,L}$, $L_{R,U}$, $\mathbf{P}$, $\mathbf{b}^i$, $\mathbf{b}_{REF}$, *Best_sol*, n_{best}
Output: $\mathbf{P}'$
Main procedure
Create $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$, $\mathbf{b}_4$

For i = 1 to m

1: Assign P_R = *rand*.

2: If $P_R \in$ [0, 0.25), set $b_{best1,i} = b_{1,i}$

3: If $P_R \in$ [0.25, 0.5), set $b_{best1,i} = b_{2,i}$

4: If $P_R \in$ [0.5, 0.75), set $b_{best1,i} = b_{3,i}$

5: Otherwise, set $b_{best1,i} = b_{4,i}$

6: Random selected a vector $\mathbf{b}_{best2}$ from *Best_sol*.

For $j = i + 1$ to m

7: Generate L_R.

8: Update P_{ij} using Equation (4.4).

9. Limit P_{ij} to the interval $[P_L, P_U]$.

End

End

4.3 Experimental set up

To investigate the search performance of the proposed algorithm, fifteen learning-based test problems from CEC 2015 and one flight dynamic control optimisation problem are used. The former is used for testing the performance of EDACE for general types of box-constrained optimisation while the latter is the real-world application.

4.3.1 CEC 2015 learning-based test problems

The CEC2015 learning-based test problems are box-constrained single objective benchmark functions proposed in [63]. 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 4.1. It should be noted that the details and the codes for the test problems can be downloaded from the website of CEC 2015 competition.

Table 4.1 Summary of CEC2015 learning-based functions

	No.	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

4.3.2 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 4.1. The motion of the aircraft is described by the 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 [64]. In this work, only the lateral/directional motion control is considered. A state equation representing the dynamic motion of an aircraft is expressed as [64-67]:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{4.5}$$

where $\mathbf{x}=\{\beta, r, p, \phi\}^T$

β = Sideslip, a velocity in *y* direction

r= yaw rate, rate of change of rotation about the *x* axis

p= roll rate, rate of change of rotation about the *z* axis

ϕ = bank angle, rotation about the *x* axis

A = kinetic energy matrix

B = Coriolis matrix

$\mathbf{u} = \begin{Bmatrix} \delta_a \\ \delta_r \end{Bmatrix}$ = control vector

δ_a = aileron deflection

δ_r = rudder deflection.

The control vector **u** can be expressed as:

$$\mathbf{u} = \mathbf{C}\mathbf{u}_p + \mathbf{K}\mathbf{x} \tag{4.6}$$

where $\mathbf{u}_p$ is a pilot's control input vector while **C** and **K** are the gain matrices expressed as [67]

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \\ k_5 & 1 \end{bmatrix},$$

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

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

From Equations (4.5) - (4.6), the state equation for lateral/directional motion of an aircraft can be expressed as:

$$\dot{\mathbf{x}} = (\mathbf{A} + \mathbf{B}\mathbf{K})\mathbf{x} + \mathbf{B}\mathbf{C}\mathbf{u}_p \tag{4.7}$$

Design optimisation of the control system of an aircraft is found to have many objectives as there are several criteria need to be satisfied such as control stability, accuracy, sensitivity,

control effort, etc, 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 to minimise spiral root subjected to stability performance constraints. The optimisation problem can then be written as:

$$\text{Min: } f(\mathbf{x}) = \lambda_s \tag{4.8}$$

Subjected to:

$$\lambda_s \le -0.01$$
$$\lambda_R \le -3.75$$
$$\xi_D \ge 0.5$$
$$\omega_d \ge 1$$

where $\lambda_s, \lambda_R, \xi_D$ 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 **A**+**BK**. The design variables are control gain coefficients in the matrix **K** (**x** = $\{k_1, k_2, k_3, k_4, k_6, k_7\}^T$). The kinetic energy matrix (**A**) and the Coriolis matrix (**B**) are defined as;

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

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

More details about this aircraft dynamic model can be found in the references [64-67]. To handle the constraints, the penalty function which was presented in [59] is used.

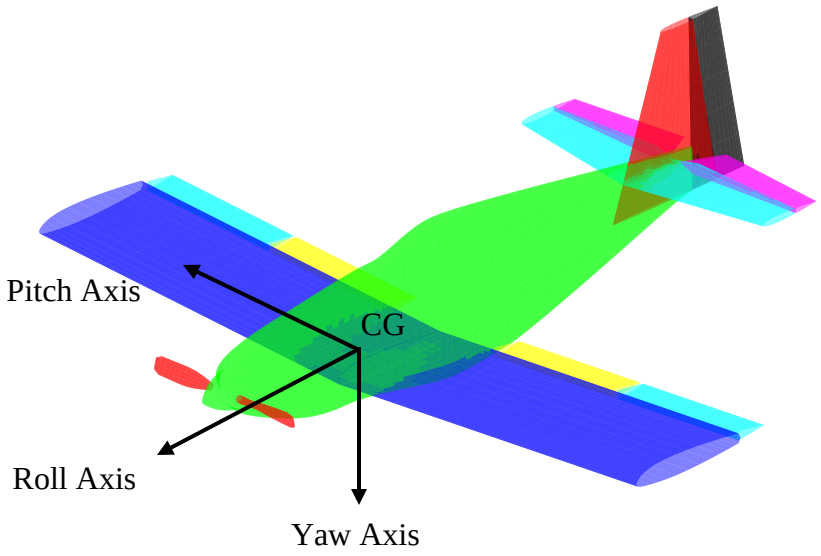


Figure 4.1 Stability axes of an aircraft

The proposed EDACE and several well established binary-code meta-heuristics are used to solve the fifteen CEC2015 learning-based test problems and the flight dynamic control test problem. The meta-heuristic optimisers are [68]:

Genetic Algorithm (GA): used binary codes with crossover and mutation rates are 1 and 0.1 respectively.

Binary Simulated Annealing (BSA): 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): used binary codes with the learning rate, mutation shift, and mutation rate as 0.5, 0.7, and 0.2 respectively.

Binary based Particle Swarm Optimisation (BPSO): used binary codes with V-shaped transfer function while the transfer function used is the V-shaped version 4 (V4). It is noted that this version is said to be the most efficient version based on the results obtained in [61].

Univariate Marginal Distribution Algorithm (UMDA); used binary codes. The first 20 best binary solutions is used to update the probability matrix.

Estimation of Distribution Algorithm with Correlation of binary Elements (EDACE) (Algorithm 4.2): used binary codes with P_L = 0.1, P_U = 0.9, $L_{R,L}$ = 0.4, $L_{R,U}$ = 0.6, and n_{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.

4.4 Optimum Results

4.4.1 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 4.2-4.4. 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 [69] are also included in the comparison. From Table 4.2, 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, the 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, f1 and f2, is EDACE while the second best is BPSO. For the simple multimodal functions, the best performer for f4 and f5 is SabDE while the best performer for the f3 is BPSO. The second best for the f3, f4 and f5 are SabDE, BEDACE and UMDE, respectively. For the hybrid functions, the best performer for the function f6, f7, and f8, are SabDE, EDACE, and BPSO, respectively, while the second best for f6, f7 is BPSO and the second best for f8 is EDACE. For the final group of CEC2015 test problems, composition functions, the best performer for the f11, f12 and f14 is SabDE while the best performer for the f10 and f15

are BPSO and EDACE, respectively. For f9, the best performers are UMDA, BPSO, GA, PBIL, and EDACE, which obtain the same mean values while, for f13, 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 method for unimodal functions, f1 and f2, are BPSO and EDACE while the second most consistent methods are EDACE and BPSO, respectively. For the simple multimodal functions, the best for f3 and f5 is SabDE while the best for f4 is the proposed EDACE. EDACE is the best for the hybrid function of f7 while BPSO is the best for the hybrid functions f6 and f8. For the composition functions, EDACE is the best for the problems f9 and f12 while BPSO is the best for f10. For the composition functions, f11, f14 and f15, the best is SabDE while the best for f13 is BSA.

The value Min in Table 4.2 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 the 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 4.2 Objective values obtained

CEC2015		MHs	UMDA	BPSO	GA	PBIL	BSA	EDACE	*GBABC	*BQIGSA	*SabDE
Unimodal Functions	f1	**Mean**	7.415E+06	1.807E+06	5.508E+06	1.586E+07	4.365E+07	**1.692E+06**	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	2.267E+07	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			
	f2	**Mean**	1.728E+08	1.278E+08	2.415E+08	1.443E+08	1.018E+09	**7.802E+07**	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**	2.374E+09	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	f3	**Mean**	3.203E+02	**3.197E+02**	3.203E+02	3.202E+02	3.202E+02	3.201E+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	2.641E+02	**2.044E-02**
		Min	3.201E+02	**3.107E+02**	3.201E+02	3.201E+02	3.201E+02	3.200E+02			
	f4	**Mean**	4.213E+02	4.220E+02	4.286E+02	4.278E+02	4.226E+02	4.182E+02	4.358E+02	4.389E+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	3.621E+02	7.606E+00
		Min	**4.105E+02**	4.124E+02	4.138E+02	4.123E+02	**4.105E+02**	**4.105E+02**			
	f5	**Mean**	1.010E+03	1.066E+03	1.339E+03	1.353E+03	1.275E+03	1.014E+03	1.108E+03	1.542E+03	**9.330E+02**
		STD	1.300E+02	1.352E+02	1.981E+02	2.279E+02	1.975E+02	1.500E+02	9.736E+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**			
Hybrid Functions	f6	**Mean**	1.951E+05	7.345E+04	2.288E+05	4.894E+05	6.403E+06	1.133E+05	7.442E+06	5.582E+05	**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	6.055E+05	4.076E+04
		Min	3.661E+04	3.661E+04	3.702E+04	8.124E+04	1.320E+05	**3.659E+04**			
	f7	**Mean**	7.047E+02	7.032E+02	7.044E+02	7.046E+02	7.118E+02	**7.030E+02**	7.589E+02	7.392E+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	4.324E+02	4.155E+03
		Min	7.027E+02	7.024E+02	7.027E+02	7.024E+02	7.025E+02	**7.021E+02**			
	f8	**Mean**	9.309E+04	**1.511E+04**	5.503E+04	4.808E+05	2.305E+06	2.727E+04	3.949E+07	2.948E+06	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	2.469E+06	5.432E+07
		Min	1.497E+04	**1.287E+04**	**1.287E+04**	1.312E+04	1.589E+04	**1.287E+04**			
Composition Functions	f9	**Mean**	**1.001E+03**	**1.001E+03**	**1.001E+03**	**1.001E+03**	1.003E+03	**1.001E+03**	1.017E+03	1.048E+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	8.643E+02	4.102E+01
		Min	1.000E+03	1.000E+03	1.000E+03	1.001E+03	1.000E+03	1.000E+03			
	f10	**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	2.416E+07
		STD	8.308E+03	**2.140E+03**	1.291E+04	3.728E+04	6.444E+05	4.897E+03	3.659E+06	4.477E+04	8.862E+07
		Min	3.199E+03	1.733E+03	1.738E+03	2.275E+03	1.805E+03	**1.731E+03**			

	f11	**Mean**	1.510E+03	1.232E+03	1.360E+03	1.396E+03	1.427E+03	1.240E+03	1.159E+03	1.172E+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	9.669E+02	**1.131E+01**
		Min	1.401E+03	**1.109E+03**	1.118E+03	1.132E+03	1.402E+03	**1.109E+03**			
	f12	**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	**1.224E+03**
		STD	9.674E-01	1.125E+00	1.259E+00	2.595E+00	4.486E+00	**1.115E+00**	1.044E+03	1.035E+03	1.302E+00
		Min	**1.303E+03**	**1.303E+03**	1.304E+03	1.304E+03	**1.303E+03**	**1.303E+03**			
	f13	**Mean**	**1.300E+03**	**1.300E+03**	**1.300E+03**	**1.300E+03**	**1.300E+03**	**1.300E+03**	1.446E+03	1.452E+03	2.815E+09
		STD	8.140E-04	1.039E-03	1.248E-03	9.095E-04	**2.313E-13**	7.944E-04	1.193E+03	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			
	f14	**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	**1.727E+03**
		STD	2.120E+03	2.144E+03	1.856E+03	2.732E+03	3.603E+03	2.199E+03	2.091E+03	2.869E+03	**4.411E+02**
		Min	4.817E+03	2.816E+03	4.425E+03	2.818E+03	4.453E+03	**2.401E+03**			
	f15	**Mean**	1.623E+03	1.616E+03	1.621E+03	1.618E+03	1.639E+03	**1.614E+03**	2.012E+03	1.530E+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	1.262E+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 [69] with 1,000,000 function evaluations and 50 binary length for each design variable

Table 4.3 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.

Table 4.3 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
Sum of ranking	72	**40**	80	85	93	**40**	96	92	76

In order to further investigate the performance comparison of the binary-code MHs, the statistical t-test is employed. Table 4.4 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 is set to be 1, otherwise, it is set to be 0. When 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 4.5 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.

Figure 4.2-4.5 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 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 4.4 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

Table 4.5 Ranking of the 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

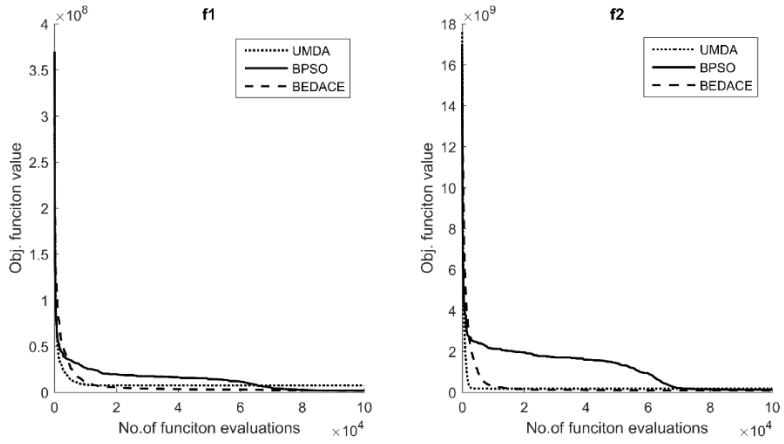


Figure 4.2 Search history of the top three best optimisers based on the t-test for the unimodal function

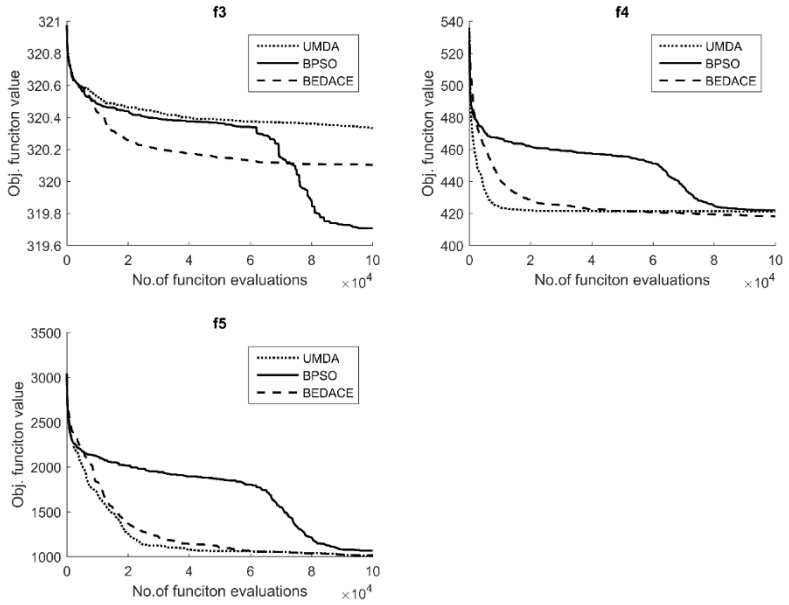


Figure 4.3 Search history of the top three best optimisers based on the t-test for the simple multimodal functions

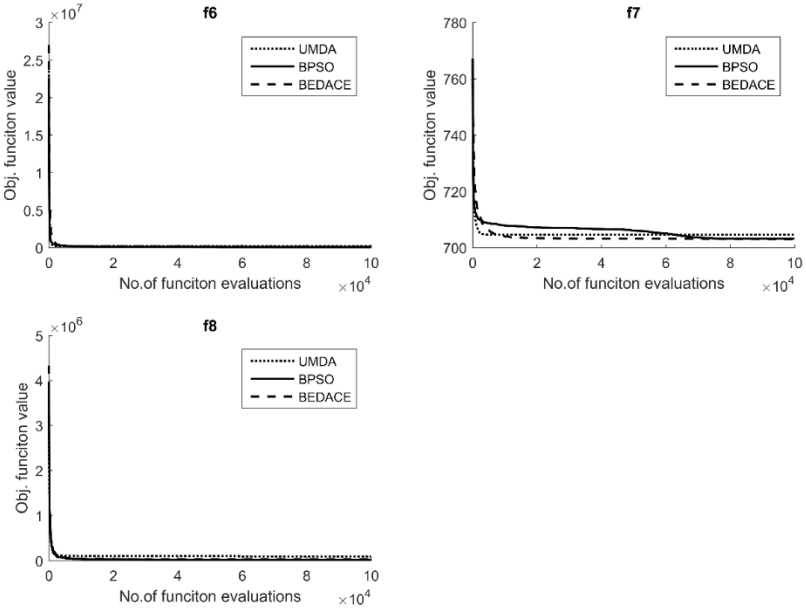


Figure 4.4 Search history of the top three best optimisers based on the t-test for the hybrid functions

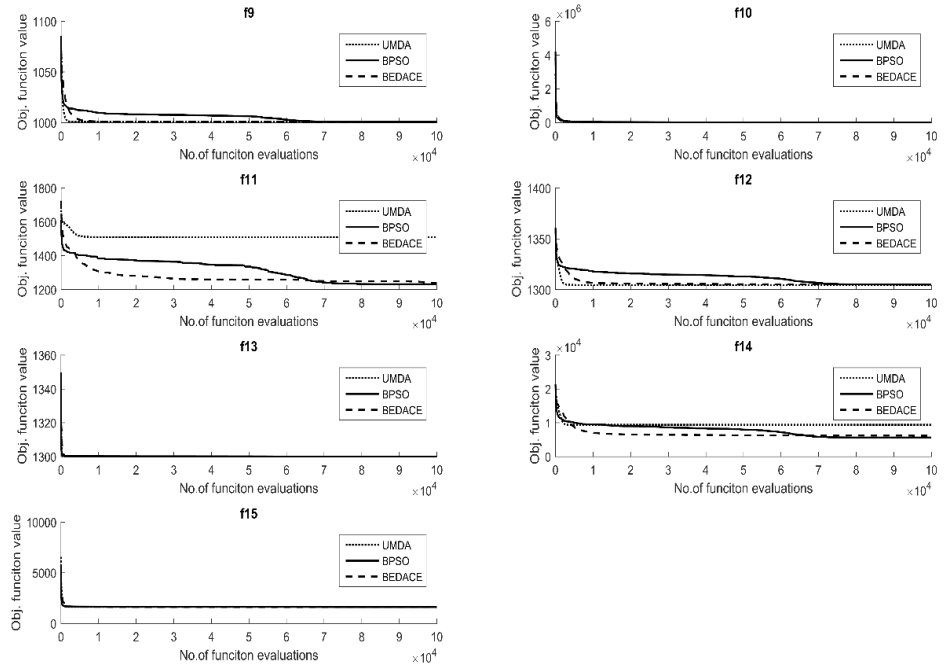


Figure 4.5 Search history of the top three best optimisers based on the t-test for the composition functions

Table 4.6 shown 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 bit increases, the computational time increase 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.

Table 4.6 shown performance of EDACE for various number of binary bits

No. of binary bits	5	10	25	50
Mean function values	2.314E+6	1.101 E+6	1.079 E+6	1.143 E+6
Average computational time (Sec.)	9.371	10.748	18.634	52.773

4.4.2 *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 4.6). 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 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 having the smallest gap between the maximum and minimum for all of optimisation runs is UMDA. However, the worst function value found be 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 4.7 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.

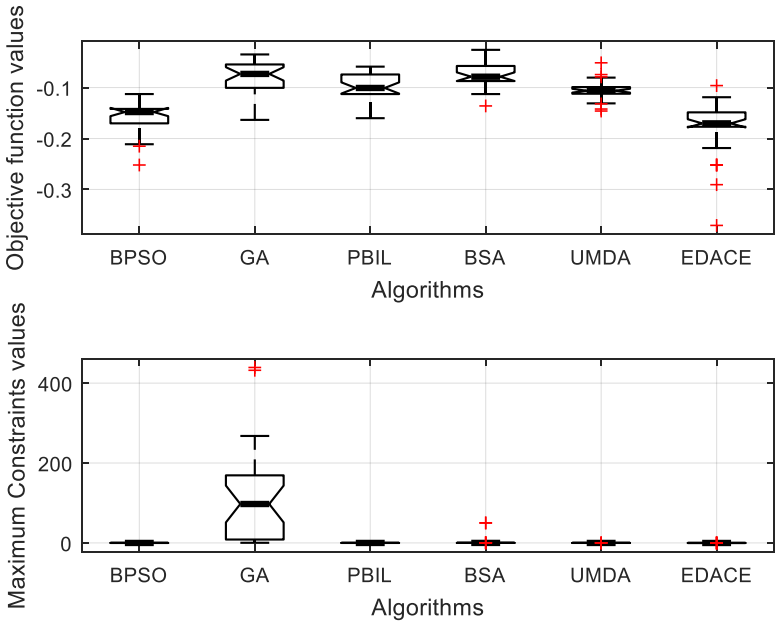


Figure 4.6 Box-plot of objective function values from 30 optimisation runs

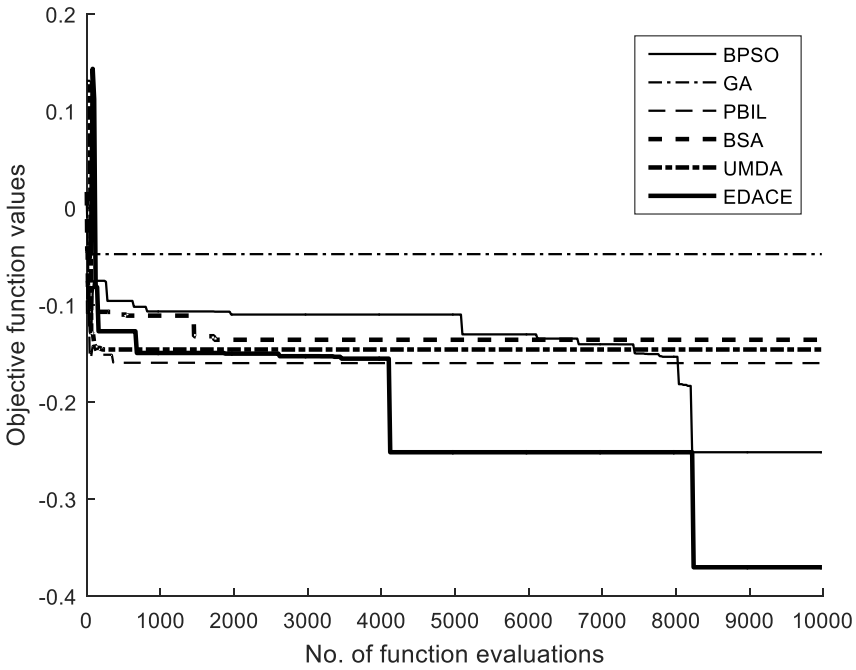


Figure 4.7 Search history of the best run of all optimisers

4.5 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 investigated the search performance of the proposed algorithm. Several well-establish 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 $\mathbf{b}_{REF}$ needs further studies. The use of EDACE for hyper-heuristic development is also possible. The extension to multiobjective optimisation and many-objective optimisation is also under investigation. Appling EDACE for the more complex problems such as large scale problems, mixed-variable problems, and reliability optimisation is for future work. The fight 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.

Chapter V

Inverse problem based differential evolution for efficient structural health monitoring of trusses

5.1 Introduction

This chapter presents a new efficient MH for structural damage detection as a hybridisation of a radial basis function (RBF) interpolation and differential evolution (DE). In this work, the RBF is integrated into the main procedures of DE for approximating design solutions rather than objective functions as with traditional surrogate-assisted optimisation. Four structural damage detection and localisation test problems from two truss structures are used for performance assessment of a number of MHs and the proposed algorithm. The results obtained from the various algorithms will be statistically compared in terms of both convergence rate and consistency.

5.2 Natural-frequency-based damage detection and localisation

In this study, structural damage detection using changes in structural natural frequencies is considered. The detection strategy can be used for damage detection of truss elements due to corrosion, crack and yielding of members due to fatigue. This approach is based on implementing modal testing incorporated with a finite element model. Initially, the natural frequencies (usually the lowest n_{mode} natural frequencies) of the structure in a normal condition will be used as the baseline. In practice, the natural frequencies and mode shapes will be measured and the finite element model will be updated so that both measured and computed modal parameters are equivalent. The finite element model used herein is a simple linear undamped free vibration which can be expressed as:

$$[\mathbf{K}]\{\phi_j\} - \lambda_j [\mathbf{M}]\{\phi_j\} = 0 \tag{5.1}$$

The structural natural frequencies can be computed as

$$\omega_j = \sqrt{\lambda_j}\ ,\ j=1,2,3,\ldots, n_{dof} \tag{5.2}$$

The mass and stiffness matrices can be obtained from assembling all element mass and stiffness matrices, which can be expressed as:

$$[\mathbf{M}] = \sum_{i=1}^{n_e} [\mathbf{m}_e]$$

and

$$[\mathbf{K}] = \sum_{i=1}^{n_e} [\mathbf{k}_e]. \tag{5.3}$$

In cases that damage in the structural element occurs, the structural natural frequencies of the structure will be different from those of the baseline structure. To localise the damage, it is assumed that the values of the structural stiffness matrix are altered, which can be written in terms of element structural damage percentage. As a result, the altered structural stiffness matrix of the damaged structure is of the form

$$[\mathbf{K}_d] = \sum_{i=1}^{n_e} \frac{100 - p_i}{100} [\mathbf{k}_e]. \tag{5.4}$$

The optimisation problem is then formulated by assigning all the values of element damage percentages as a design solution $\mathbf{x} = \{p_1, ..., p_{ne}\}^T$. The objective function is to minimise the root mean square error:

$$\text{Min} : f(\mathbf{x}) = \sqrt{\frac{\sum_{j=1}^{n_{mode}} \left(\omega_{j,damage} - \omega_{j,computed}\right)^2}{n_{mode}}} \tag{5.5}$$

where $\omega_{j,damage}$ is the structural natural frequency of mode *j* obtained from measuring a damaged structure. n_{mode} is the number of lowest vibration modes used for the damage detection. $\omega_{j,computed}$ is the structural natural frequency of mode *j* obtained from solving (5.1) using $[\mathbf{K}_d]$ instead of [**K**]. The optimum solution having the objective function value close to zero gives accurate damage localisation. The values of the element damage percentage indicate where the damage takes place.

5.3 Test problems with trusses

To study performance assessment of a number of MHs on tackling damage detection optimisation, two truss structures are employed in this work. For the sake of simple investigation, truss damage is simulated whereas the natural frequencies of structures are computed from finite element analysis rather than measuring real structure modal data. Only truss element damages are taken into consideration. It should be noted that free vibration is simulated for all cases without considering gravity loads. The trusses are detailed as follows.

5.3.1 Twenty-five-bar truss

The structure having 25 bars is depicted in Fig. 5.1 [70]. All bar element cross-sectional areas are set to be 6.4165 mm^2. Material density and Young modulus are given as 7,850 kg/m^3 and 200 GPa, respectively. Two damage case studies are assumed as Case I: 35% damage on element 7 (Note that 35% damage on elements 6, 8 or 9 will result in the same set of natural frequencies), and Case II: 35% and 40% damage at elements 7 and 9 (Note that 35% damage in element 6 and 40% damage in element 8 will result in the same set of natural frequencies for this case). The pin supports are applied to node numbers 7, 8, 9 and 10. The data of natural frequencies of the damaged and undamaged 25-bar truss are given in Table 5.1.

Table 5.1 Natural frequencies (Hz) of damaged and undamaged 25 bar structure.

Mode	Undamaged reported in [6]*	Undamaged calculated by commercial software (Ansys academic version)*	Undamaged calculated in this study*	35% damage at element number 7	35% damage at element number 7 and 40% damage at element number 9
1	70.9924	69.782	69.7818	69.1393	68.5203
2	74.0851	72.822	72.8217	72.2006	71.3167
3	97.5390	95.876	95.8756	95.3372	94.5625

4	122.2281	120.14	120.1437	119.8852	119.6514
5	121.9300	121.50	121.5017	121.4774	121.4253
6	-	125.01	125.0132	125.0130	125.0129

* The natural frequencies are slightly different which could be due to the numerical algorithm used and truncation errors.

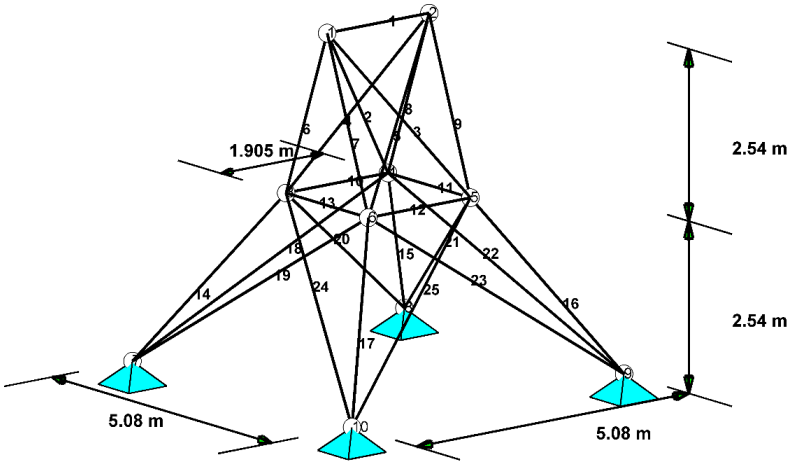


Figure 5.1 Twenty-five bar truss

5.3.2 Seventy-two-bar truss

The 72-bar truss structure is displayed in Fig. 5.2 [71] where four non-structural masses of 2270 kg are attached to the top nodes. The values of all bar element cross-sectional areas are set to be 0.0025 m^2. Material density and modulus of elasticity are 2,770 kg/m^3 and 6.98×10^{10} Pa, respectively. Two cases of damage are generated as Case I: 15% damage at element number 55 (Note that 15% damage in elements 56, 57, or 58 will result in the same set of natural frequencies as that of element 55), and Case II: 10% damage at element number 4 and 15% damage at element number 58 (90, 180, and 270 degrees rotation along the *z* axis will lead to the same set of natural frequencies). The pin supports are applied to nodes number 17, 18, 19 and 20. The values of natural frequencies of the damaged and undamaged 72-bar truss are given in Table 5.2.

Table 5.2 Natural frequencies (Hz) of damaged and undamaged 72 bar structure.

Mode	Undamaged reported in [11]*	Undamaged calculated by commercial software (Ansys academic version)*	Undamaged calculated in this study*	15% damage at element number 55	15% damage at element number 58 and 10% damage at element number 4
1	6.0434	5.4977	6.0455	5.9553	5.9530
2	6.0441	5.4977	6.0455	6.0455	6.0455
3	10.4627	9.5181	10.4764	10.4764	10.4764
4	18.2275	16.594	18.2297	18.1448	18.0921
5	25.4466	23.213	25.4939	25.4903	25.2437
6	25.4510	23.213	25.4939	25.4939	25.4939

* The natural frequencies are slightly different which could be due to the numerical algorithm used and truncation errors.

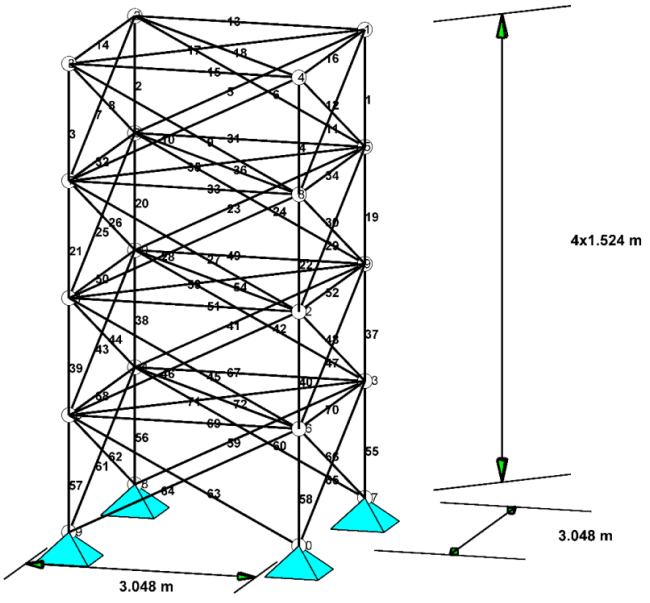


Figure 5.2 Seventy-two bar truss

5.4 Hybrid radial basis function and differential evolution for truss damage detection

The purpose of using MHs for truss damage detection is to solve the optimisation problem with the objective function (5.5) subject to bound constraints of **x**. The advantages of using MHs are their simplicity in use, capability of global search, derivative-free feature, and robustness. Using meta-heuristics implies that a user has less worry about mode switching during an optimisation run while this phenomenon may occur in cases of using a gradient-based optimiser. The detection approach can be used for real-time monitoring provided that an employed MH is adequately powerful.

5.4.1 Differential evolution

Differential evolution is a population based method which was first proposed by Storn and Price in 1997 [2]. The method contains two main steps for searching an optimum, including mutation and crossover where the acronym DE/*x*/*y*/*z* is used to specify different mutation and crossover strategies. The variable *x* is used to specify a vector for mutation which can be *best* (the best individual) or *rand* (random individual) while *y* and *z* specify the number of vector pairs used in mutation and the choice of a crossover scheme, respectively. For example, as used in this work, DE/best/2/bin means that the best individual and two different vector pairs are used in the mutation step while the binomial crossover is employed. The mutation operation can be expressed as follows:

$$\mathbf{u}_i = \mathbf{x}_{\text{best}} + (-1)^{rand(-1,0)} F\left(\mathbf{x}_{r,1} + \mathbf{x}_{r,2} - \mathbf{x}_{r,3} - \mathbf{x}_{r,4}\right). \tag{5.6}$$

In this work, F is a uniform random number in the range of $[F_{\text{min}}, F_{\text{max}}]$. For the i-th mutant individual $\mathbf{u}_i^T = [x_{\text{new},1}, \ldots, x_{\text{new},D}]$ and its corresponding parent $\mathbf{x}_{\text{old}}^T = [x_{\text{old},1}, \ldots, x_{\text{old},D}]$, the binary crossover can be operated leading to a new candidate solution $\mathbf{x}_{new}$ as

$$x_{new,j} = \begin{cases} u_j; & rand < CR \\ x_{old,j}; & otherwise \end{cases} \qquad j=1,2,3,\ldots,D. \tag{5.7}$$

The selection operator is carried out by comparing $\mathbf{x}_{new}$ and its parent $\mathbf{x}_{old}$ where the better will survive to the next generation.

The DE computational steps are shown in Algorithm 5.1. Initially, a set of the population is generated by means of randomisation and their objective function values are evaluated. After obtaining the best individual, the offspring are generated by mutation (eq. 5.6) and then crossover (eq. 5.7). Then, the next generation is selected and the search process will be repeated until a termination criterion is reached.

Algorithm 5.1 DE search procedure

Input: population size, number of generations, algorithm parameters.

Output: $\mathbf{x}_{best}, f_{best}$

Main algorithm

1::Initialise a population, calculate their objective function values and set as the current population.

2: Find the best individual

3: Generate a new population from the current population using DE mutation (eq.6) and DE crossover (eq.7).

4: Evaluate objective function values of the members of the new population.

5: Select the next generation from the newly generated and current populations.

6: Set the selected population from step 5 as the next generation.

7: If a termination condition is not met, go to step 2. Otherwise, stop the algorithm.

5.4.2 Inverse problem-based differential evolution

This subsection details the proposed differential evolution based on using an inverse problem concept. In optimisation, the radial basis function is traditionally used for approximating an objective function value for problems with expensive function evaluation [47, 72]. Nevertheless, in this work RBF is conversely implemented. It will be used to approximate a design solution **x** that is expected corresponding to the target damage conditions. Given that the vector of target natural frequencies ($\boldsymbol{\omega}_{damage}$) contains n_{mode} lowest natural frequencies of the damaged structure, the idea is to find a solution vector $\mathbf{x}_{damage}$ containing n_e element damage

percentages by means of interpolation. During MH search, if we have a set of N design solutions $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ which corresponds to a set of N vectors of natural frequencies $\{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_N\}$, these data will be used for RBF training. In contrast to surrogate-assisted optimisation, the natural frequency vector will be set as independent variables whereas the design vector $\mathbf{x}$ will be set as dependent variables. The i^{th} element of $\mathbf{x}_{damage}$ that is expected to give the target vector of natural frequencies of the damaged truss is expressed as:

$$x_{damage,i} = \sum_{k=1}^{N} c_k \varphi\left(\left\|\boldsymbol{\omega}_k - \boldsymbol{\omega}_{damage}\right\|\right) \tag{5.8}$$

where c_k is the interpolation coefficients to be determined, and $\boldsymbol{\varphi}$ is a RBF kernel function. $\left\|\boldsymbol{\omega}_k - \boldsymbol{\omega}_{damage}\right\|$ is the distance between $\boldsymbol{\omega}_k$ and $\boldsymbol{\omega}_{damage}$. For x_i, interpolation coefficients c_k can be found from solving the system of linear equations

$$\sum_{k=1}^{N} c_k \varphi\left(\left\|\boldsymbol{\omega}_k - \boldsymbol{\omega}_l\right\|\right) = x_i\left(\boldsymbol{\omega}_l\right) \;;\; \text{for } i = 1, \ldots, n_e, \text{ and } l = 1, \ldots, N \tag{5.9}$$

where $x_i(\boldsymbol{\omega}_l)$ is the i^{th} element of the l^{th} solution vector in the training set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. Equation (5.9) can be written in a matrix form as

$$\mathbf{A}\mathbf{c} = \mathbf{b} \tag{5.10}$$

where $A_{k,l} = \varphi(||\boldsymbol{\omega}_k - \boldsymbol{\omega}_l||)$. It is required to compute n_e sets of the interpolation coefficients according to n_e elements of $\mathbf{x}$. In practice, the matrix $\mathbf{A}$ is generated and inverted once, and will be used to calculate n_e sets of the coefficients.

Having determined the sets of interpolation coefficients c_k for all n_e elements of $\mathbf{x}$ by using (5.9), the elements of $\mathbf{x}_{damage}$ can be found from using Equation (5.8). The search procedure for hybridised RBF and DE which will be termed inverse problem-based differential evolution (IPB-DE) according to its computation nature can be carried out in such a way that, after the reproduction step 3 in Algorithm 5.1, the next generation is selected in step 5. The worst solution in the next generation is then replaced by $\mathbf{x}_{damage}$. The procedure of the hybrid algorithm IPB-DE

is detailed in Algorithm 5.2 while the flowchart for the IPB-DE algorithm is shown in Fig. 5.3. The process starts by creating an initial population by using the Latin hypercube sampling (LHS) technique instead of the Monte Carlo technique. Those solutions in the initial population are then saved to the RBF database for training RBF. Offspring are then created by means of reproduction of DE. The candidate solution $\mathbf{x}_{damage}$ is created using Equations (5.8-5.9). Having performed a selection operation, the worst solution in the next generation is replaced by $\mathbf{x}_{damage}$. The best solution from the offspring and $\mathbf{x}_{damage}$ are then added to the RBF database which will be used as training points during the optimisation search. As the process continues, the RBF database is improved and expected to give more accurate results. The procedure is repeated until fulfilling the termination criteria.

Algorithm 5.2 IPB-DE

Input: population size (n_p), number of generations (n_{iter}), algorithm parameters, the natural frequencies measured from the damaged structure ($\boldsymbol{\omega}_{damage}$)

Output: $\mathbf{x}_{best}, f_{best}$

Main algorithm

1: Generate an initial set of design variables $\mathbf{x}$ using LHS, calculate the natural frequencies ($\boldsymbol{\omega}$) and objective function values ($\mathbf{f}$), set $\mathbf{x}$ and $\mathbf{f}$ as the current population and save $\mathbf{x}$ and $\boldsymbol{\omega}$ in the RBF database.

2: Find the best solution.

3: Generate offspring from the current population using the DE mutation and binomial crossover operators (reproduction) and then perform function evaluations.

4: Select design solutions from the offspring and the current population.

5: Generate $\mathbf{x}_{damage}$ using the training points from the RBF database using Equations (9) and then (8).

6: Calculate the natural frequencies ($\boldsymbol{\omega}$) and objective function value (f) of $\mathbf{x}_{damage}$.

7: Update the RBF database by adding to it the data of the best solution from the offspring and $\mathbf{x}_{damage}$.

8: Replace the worst solution in the next generation with $\mathbf{x}_{damage}$.

9: If a termination condition is not met, go to step 2. Otherwise, stop the algorithm.

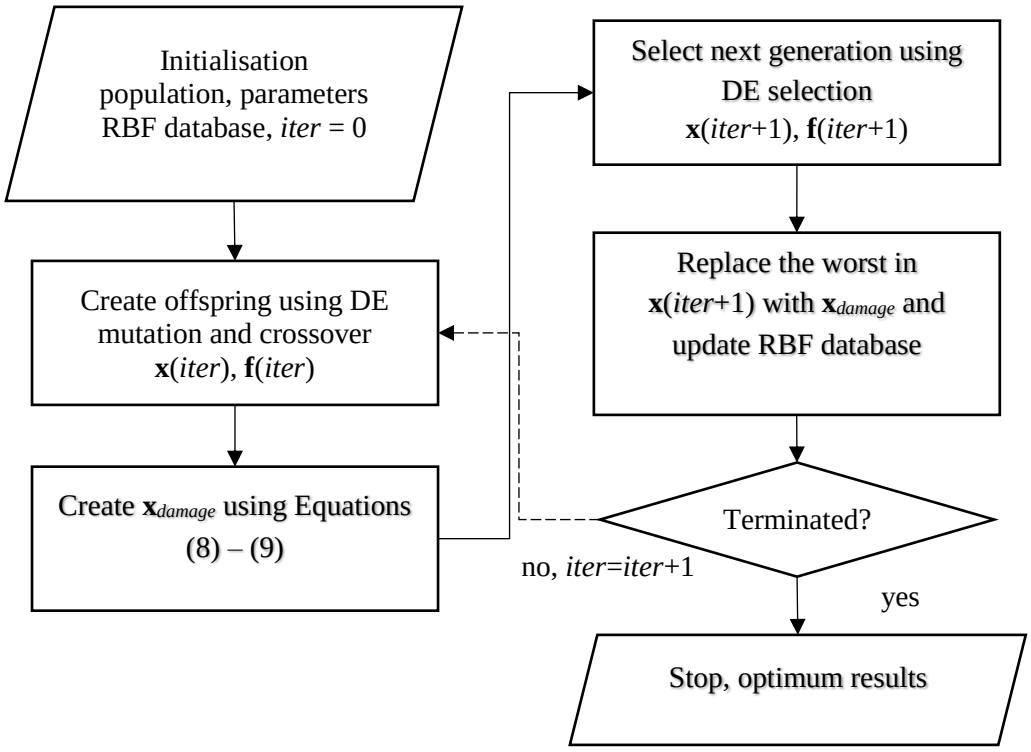


Figure 5.3 Flow chart of IPB-DE

5.5 Numerical Experiment

To verify the search performance of the proposed IPB-DE, several MHs are compared based on solving the aforementioned truss damage detection problems. The employed methods are said to be established while some of them are regarded as the currently best optimisers of this type. Given that n_P is a population size, MHs and their optimisation parameter settings used in this work are detailed in table 5.3 (it should be noted that details of notations can be found in the corresponding references for each method) [58, 73]:

Table 5.3 MH Parameters settings

MH	Parameter settings
Whale optimization algorithm (WOA) [74]	- The parameter b = 1 - Other parameters are iteratively adapted.
Sine Cosine algorithm (SCA) [7]	- The constant parameter a = 2.

Moth-flame optimisation algorithm (MFO) [75]	- The constant parameter b = 1 - Other parameters are iteratively adapted.
Differential evolution (DE) [2]	- Using DE/best/2/bin strategy - Scaling factor (F) =0.8, - probability of choosing elements of mutant vectors (CR) = 0.5
Artificial bee colony algorithm (ABC) [49]	- The number of food sources for employed bees = $n_P/2$. - A trial counter to discard a food source =100.
Real-code ant colony optimisation (ACOR) [50]	- The parameter, q = 0.2 - The parameter, ξ = 1
Charged system search (ChSS) [52]	- The number of solutions in the charge memory = $0.2 \times n_P$ - The charged moving considering rate = 0.75 - the parameter PAR = 0.5
League championship algorithm (LCA) [51]	-The probability of success P_c = 0.9999 -The decreasing rate to decrease P_c = 0.9995
Simulated annealing (SA) [48]	- Starting temperature = 10 - Ending temperature = 0.001 For each loop, n_{mode} candidates are created by mutating on the current best solution while other n_{mode} candidates are created from mutating the current parent. The best of those $2n_{mode}$ solutions are set as an offspring to be compared with the parent.
Particle swarm optimisation (PSO) [3]	- The starting inertia weight = 0.5 - The ending inertia weight = 0.01 - The cognitive learning factor = 0.5 - The social learning factor = 0.5

Evolution strategies (ES) [4]	The algorithm uses a binary tournament selection operator and a simple mutation without the effect of rotation angles.
Teaching-learning-based optimisation (TLBO) [9]	Parameter settings are not required.
Adaptive differential evolution (JADE) [11]	The parameters are self-adapted during an optimisation process.
Evolution strategy with covariance matrix adaptation (CMAES) [76]	The parameters are self-adapted during an optimisation process.
IPB-DE	Use the DE parameter setting.

Each optimisation algorithm is employed to solve each test problem for 30 independent runs. The number of iterations (generations) is 300 for all case studies while the population size is set to be 30 and 50 for 25-bar and 72-bar trusses respectively. For the optimisers using different population sizes from the aforementioned values, their search processes are terminated with the total number of functions evaluations (FEs) equal to 30×300 and 50×300 for 25-bar and 72-bar trusses respectively. Another termination criterion is when one of the design solutions in the current population has an objective function value less than or equal to 1×10^{-3}. It should be noted that the numbers of FEs used in this study can be considered insufficient for some MH optimisers. However, these values are used to find out really powerful algorithms. For all test problems, six lowest natural frequencies (n_{mode} = 6) are used to compute the objective function values. This number of selected frequencies is reasonable since, in practice, it is easier to accurately measure fewer lowest natural frequencies.

5.6 Results and discussion

Initially, the effect of RBF kernels on the performance of the proposed algorithm was investigated. The last test problem, 72 bar truss with 15% damage at element number 58 and 10% damage at element number 4 which is said to be the most complicated problem, was used. Table 5.4 shows the results obtained from using a variety of RBF kernel functions. The mean

values of the objective function are used to indicate the search convergence of the algorithms in cases that the objective function threshold (1×10^{-3}) is not met during an optimisation run. Otherwise, the mean number of FEs is used as an indicator. The algorithm that is terminated by the objective function threshold is clearly the superior method and any optimisation run being stopped with this criterion is considered a successful run. The number of successful runs from 30 optimisation runs denoted as "No. of successful runs from 30" is the total number that the algorithm can meet the target objective function value (1×10^{-3}). It is used to measure the algorithm reliability. From Table 5.4, the best performer is the Gaussian kernel, while the second best and the third best are the Polynomial kernel and the Inverse quadratic kernel, respectively. Thus, the Gaussian kernel is used in this study.

Table 5.4 Comparison of various RBF kernels for solving 72 bar truss Case II

DE with RBF kernel	Mean objective function Values	No. of successful runs from 30 runs	Mean of FEs
Gaussian	0.0011	25	6856
Multiquadric	0.0104	5	13993
Inverse quadratic	0.0032	14	12221
Linear	0.0117	8	13819
Polynomial order 2	0.0039	15	10807

Comparison of various ranges [F_{min}, F_{max}] of a scaling factor and *CR* values using DE with the best RBF kernel for solving the 72 bar truss with 15% damage at element number 58 and 10% damage at element number 4 is shown in Table 5.5. It is found that for all implemented intervals of [F_{min}, F_{max}], the performance increases when the value of *CR* increases. The highest DE performance is obtained when the range [F_{min}, F_{max}] and *CR* are set to be [0.2, 0.8] and 0.8, respectively.

Table 5.5 Comparison of various ranges of *F* and *CR* values for solving 72 bar truss Case II

DE with Gaussian RBF kernel		Mean objective function value	No. of successful runs from 30 runs	Mean of FEs
$[F_{min}, F_{max}]$	*CR*			
[-1.5,1.5]	0.3	0.0027	1	15000
[-1.5,1.5]	0.5	0.0013	16	12983
[-1.5,1.5]	0.8	0.0011	24	7648
[0.2,0.8]	0.3	0.0025	0	15000
[0.2,0.8]	0.5	0.0011	21	12344
[0.2,0.8]	0.8	0.0011	25	6856
[-2,-2]	0.3	0.0042	0	15000
[-2,-2]	0.5	0.0014	9	14496
[-2,-2]	0.8	0.0014	21	9940

The results obtained from the various MHs from solving the six test problems are given in Tables 5.6-5.9.

5.6.1 Twenty-five-bar truss

For the 25-bar truss with 35% damage at element 7, the results are given in Table 5.6. The best performer based on the mean objective function values is IPB-DE while the second and third best are DE and JADE respectively. When considering the number of successful runs, seven optimisers including WOA, MFO, SCA, DE, TLBO, JADE and IPB-DE can detect the damage in the structures. The most efficient optimisers are SCA and IPB-DE that can detect the damages of the structure for 24 and 25 times out of 30 runs within the average of 3262 and 4486 function evaluations respectively.

For the 25 bar truss with 35% damage at element 7 and 40% damage at the element number 9, the results are reported in Table 5.7. The best performer based on mean values is IPB-DE while the second and third best are JADE and DE respectively. When examining the number of successful runs, only IPB-DE can detect the damage in the structure for all 30 runs. For this case, IPB-DE is said to be the most efficient optimiser, which obtained the minimum

objective function mean value and successfully detected the damage in the structure for all optimisation runs with the average number of function evaluations being 3735.

Table 5.6 Results for 25 bar truss Case I

Optimiser	Mean objective function value	No. of successful runs from 30 runs	Mean of FEs
WOA	0.0357	8	6993
MFO	0.0279	3	8686
SCA	0.0270	24	3262
DE	0.0017	19	6019
ABC	0.0135	0	9000
ACOR	0.0089	0	9000
ChSS	0.1385	0	9000
LCA	0.9036	0	9000
SA	0.0089	0	9000
TLBO	0.0077	6	7772
CMAES	0.0033	0	9000
ES	0.0308	0	9000
PSO	8.3830	0	9000
JADE	0.0026	2	8953
IPB-DE	0.0012	25	4486

Table 5.7 Results for 25 bar truss Case II

Optimiser	Mean objective function value	No. of successful runs from 30 runs	Mean of FEs
WOA	0.1301	0	9000
MFO	0.0336	1	8876
SCA	0.0930	0	9000
DE	0.0096	27	5220
ABC	0.0326	0	9000

ACOR	0.0125	0	9000
ChSS	0.1590	0	9000
LCA	0.8080	0	9000
SA	0.0269	0	9000
TLBO	0.0405	1	8917
CMAES	0.0115	0	9000
ES	0.0356	0	9000
PSO	8.6012	0	9000
JADE	0.0042	6	8875
IPB-DE	0.0010	30	3757

5.6.2 Seventy-two-bar truss

For the 72-bar truss with 15% damage at element 5, the results are reported in Table 5.8. The best performer based on the mean objective function values is IPB-DE, while the second and the third best are ES and ACOR. When looking at the number of successful runs (*f* reaching 1×10^{-3} or lower), the most efficient method is IPB-DE which can detect the damage of the structure 30 times from implementing it in 30 optimisation runs, while the average number of function evaluations for convergent results is only 3155.

For the 72 bar truss with 15% damage at element number 58 and 10% damage at element number 4, the results are given in Table 5.9. The best performer based on the mean of objective function values is IPB-DE, while the second and third best are ES and JADE respectively. When considering the number of successful runs, the most efficient is IPB-DE, which can detect the damage of the structure 25 times from a total of 30 optimisation runs, while the average number of function evaluations for the convergence results is 6856.

Table 5.8 Results for 72 bar truss Case I

Optimiser	Mean objective function value	No. of successful runs from 30 runs	Mean of FEs
WOA	0.0082	22	4832
MFO	0.0270	2	14783

SCA	0.0070	23	4793
DE	0.0087	14	12887
ABC	0.2184	0	15000
ACOR	0.0014	6	14831
ChSS	0.1727	0	15000
LCA	1.1499	0	15000
SA	0.0097	0	15000
TLBO	0.0035	27	5781
CMAES	0.0053	0	15000
ES	0.0010	29	9335
PSO	1.9146	0	15000
JADE	0.0019	1	15000
IPB-DE	0.0009	30	3155

Table 5.9 Results for 72 bar truss Case II

Optimiser	Mean objective function value	No. of successful runs from 30 runs	Mean of FEs
WOA	0.0189	0	15000
MFO	0.0137	1	14935
SCA	0.0260	2	14502
DE	0.0127	7	13963
ABC	0.1591	0	15000
ACOR	0.0058	0	15000
ChSS	0.1348	0	15000
LCA	1.1049	0	15000
SA	0.0129	0	15000
TLBO	0.0045	7	13503
CMAES	0.0050	0	15000
ES	0.0023	2	14940
PSO	1.7726	0	15000

JADE	0.0031	0	15000
IPB-DE	0.0011	25	6856

Overall, it is clearly indicated from the results that integrating RBF into the DE can improve the search performance of the optimiser in solving structural damage detection of truss structures in terms of both search convergence and consistency. Based on the most crucial indicators, the average number of successful runs and the average number of function evaluations, IPB-DE is unanimously the most powerful method.

Figure 5.4-5.7 shows the search history of the top five best algorithms (sorted based on number of successful runs from 30 runs). For the 25 bar truss with 35% damage at element number 7, the proposed IPB-DE and WOA show a similar convergence curve while WOA is slightly faster than IPB-DE after 200 function evaluations. Similarly, for the case of the 72 bar truss with 15% damage at element number 55, the proposed IPB-DE and WOA show the best convergence curves at the beginning while WOA is faster than IPB-DE. The WOA can converge to the goal before 500 function evaluations for this case. For the 25 bar truss with 35% damage at element number 7 and 40% damage at element number 9, and the 72 bar truss with 15% damage at element number 58 and 10% damage at element number 4, the IPB-DE gives the best convergence curves since the beginning.

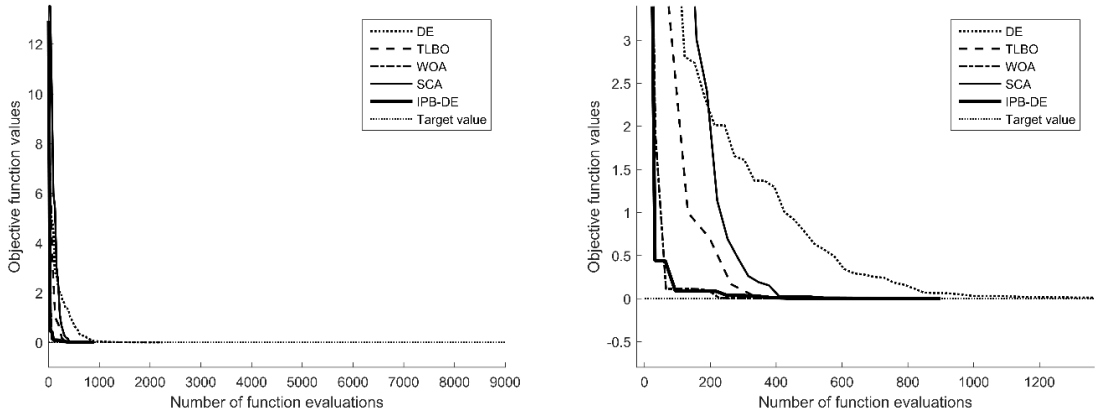
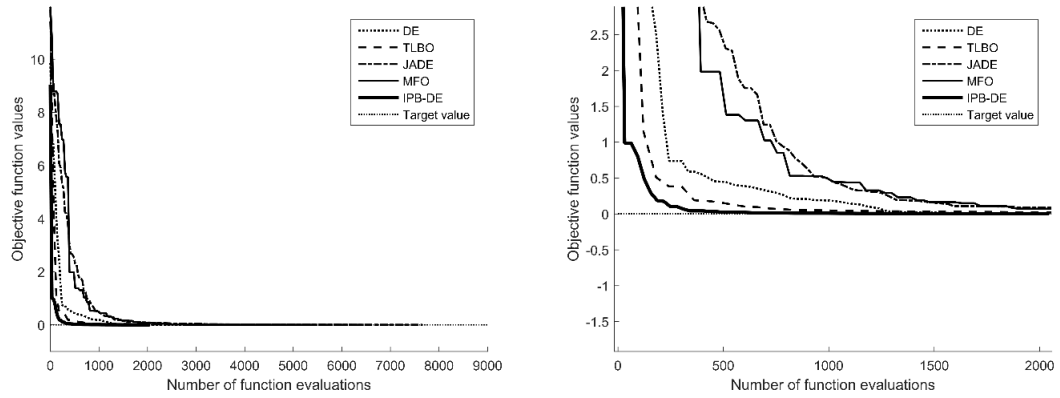
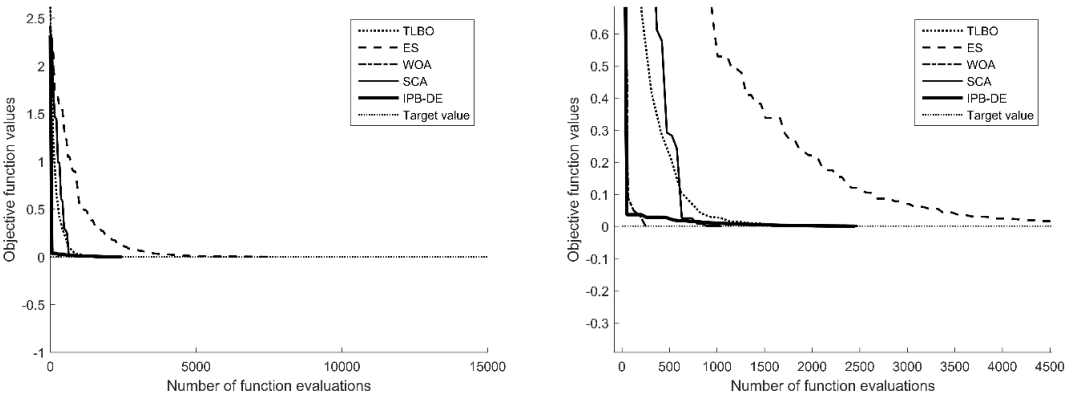


Figure 5.4 Search history for the case 25 bar Case I, (a) original, (b) zoom in



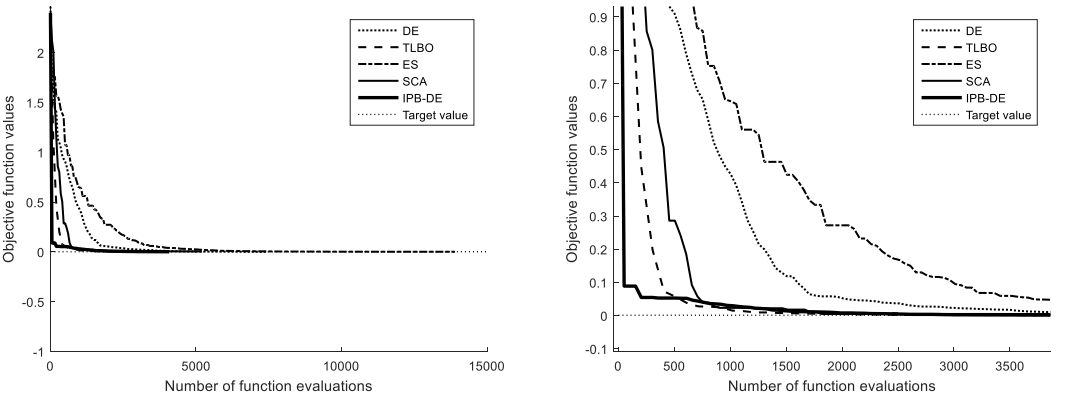
(a) (b)

Figure 55. Search history for the case 25 bar truss Case II, (a) original, (b) zoom in



(a) (b)

Figure 5.6 Search history for the case, 72 bar truss Case I, (a) original, (b) zoom in



(a) (b)

Figure 5.7 Search history for the case, 72 bar truss Case II, (a) original, (b) zoom in

Tables 5.10-5.11 show a comparison of the damage locations of the simulated problems and the results obtained from the best run of IPB-DE. It was found that IPB-DE can correctly detect the damage locations for Case I of the twenty-five bar truss while for Case II of the twenty-five bar truss, the structure is simulated to have 35% and 40% damage at element 7 and element 9 respectively, while the result obtained from IPB-DE gives 34.39 and 39.83% damage at element 6 and element 8. For this case, it can be said that the results are accurate, as both groups can obtain the same set of natural frequencies as mentioned in Section 5.3. Similarly, for Case I of the seventy-two bar truss, the structure is simulated to have 15% damage at element number 55 while 15% damage at element 56, 57, or 58 gives the same values of $\boldsymbol{\omega}_{damage}$. Therefore, it can be concluded that the results are accurate for this case. For Case II of the seventy-two bar truss, IPB-DE found damage in many elements, while the resulting natural frequencies are similar to the values of $\boldsymbol{\omega}_{damage}$. This implies that using only natural frequencies as an objective function can possibly fail to identify the damage locations for the cases of symmetric structures. The proposed algorithm is obviously effective and efficient but more reliable objective functions for damage localisation such as the use of both natural frequencies and mode shapes should be invented.

Table 5.10 Comparison of the simulated solution and the best results obtained by IPB-DE for 25 bar truss

% damage at element no.	Case I		Case II	
	Simulated damage (%)	Damage found by IPB-DE (%)	Simulated damage (%)	Damage found by IPB-DE (%)
1	0.00	0.06	0.00	0.00
2	0.00	0.83	0.00	0.74
3	0.00	0.00	0.00	0.02
4	0.00	0.00	0.00	0.35
5	0.00	0.05	0.00	0.02
6	0.00	0.44	0.00	***34.39**
7	**35.00**	**34.16**	**35.00**	0.58
8	0.00	0.45	0.00	***39.83**

9	0.00	0.00	**40.00**	0.00
10	0.00	0.00	0.00	0.00
11	0.00	0.02	0.00	0.00
12	0.00	0.10	0.00	0.00
13	0.00	0.00	0.00	0.00
14	0.00	0.00	0.00	0.00
15	0.00	0.01	0.00	0.00
16	0.00	0.00	0.00	0.00
17	0.00	0.00	0.00	0.00
18	0.00	0.00	0.00	0.00
19	0.00	0.00	0.00	0.00
20	0.00	0.01	0.00	0.00
21	0.00	0.00	0.00	0.00
22	0.00	0.00	0.00	0.00
23	0.00	0.00	0.00	0.00
24	0.00	0.00	0.00	0.00
25	0.00	0.00	0.00	0.00
ω_1	69.1393	69.139	68.5203	68.52002
ω_2	72.2006	72.200	71.3167	71.31654
ω_3	95.3372	95.337	94.5625	94.56267
ω_4	119.8852	119.886	119.6514	119.6496
ω_5	121.4774	121.477	121.4253	121.4256
ω_6	125.0130	125.011	125.0129	125.0121

* 35% damage in elements 6 and 40% damage in elements 8 will result in the same set of natural frequencies for the Case II as mentioned in Section 5.3

Table 5.11 Comparison of the simulated solution and the best results obtained by IPB-DE for 72 bar truss

% damage at element no.	Case I						Case II					
	Simulated damage (%)			Damage found by IPB-DE (%)			Simulated damage (%)			Damage found by IPB-DE (%)		
1, 26, 51	0,	0,	0	0,	0,	0.39	0,	0,	0	0,	0,	0,
2, 27, 52	0,	0,	0	0,	0,	0.36	0,	0,	0	**9.88,	0,	0.76,
3, 28, 53	0,	0,	0	0,	0,	0.03	0,	0,	0	0,	0.04,	0.51,
4, 29, 54	0,	0,	0	0,	0.01,	0.90	10.00,	0,	0	0.01,	0.05,	0.06,
5, 30, 55	0,	0,	15.00	0,	0.01,	0.02	0,	0,	0	0.01,	0,	0,
6, 31, 56	0,	0,	0	0.06,	0,	0.36	0,	0,	0	0.04,	0.16,	*9.08,
7, 32, 57	0,	0,	0	0.03,	0.06,	0	0,	0,	15.00	0.02,	0.45,	0,
8, 33, 58	0,	0,	0	0,	0,	*14.51	0,	0,	0	0,	0.73,	*6.58,
9, 34, 59	0,	0,	0	0.01,	0.01,	0.04	0,	0,	0	0.03,	0.02,	0.02,
10, 35, 60	0,	0,	0	0,	1.91,	0.01	0,	0,	0	0.01,	0,	0,
11, 36, 61	0,	0,	0	0.03,	0.87,	0	0,	0,	0	0,	0.69,	0,
12, 37, 62	0,	0,	0	0.01,	0.01,	0.07	0,	0,	0	0.04,	0,	0,
13, 38, 63	0,	0,	0	0.01,	0.00,	0.01	0,	0,	0	0.03,	0,	0,
14, 39, 64	0,	0,	0	0,	0.02,	0	0,	0,	0	0.13,	0,	0,
15, 40, 65	0,	0,	0	0.02,	0.03,	0.05	0,	0,	0	0,	0.06,	0.02,
16, 41, 66	0,	0,	0	0.14,	0.01,	0.00	0,	0,	0	0.21,	0,	0,
17, 42, 67	0,	0,	0	0.73,	0.00,	0.17	0,	0,	0	3.04,	0.02,	0.09,
18, 43, 68	0,	0,	0	0.27,	0,	0	0,	0,	0	0.55,	0.05,	1.79,
19, 44, 69	0,	0,	0	0,	0.01,	0	0,	0,	0	0,	0.06,	0.12,
20, 45, 70	0,	0,	0	0,	0,	0.62	0,	0,	0	0,	0,	0,
21, 46, 71	0,	0,	0	0,	0.01,	0.04	0,	0,	0	0,	0,	0.46,
22, 47, 72	0,	0,	0	0,	0,	0.01	0,	0,	0	0.01,	0.02,	0,
23, 48	0,	0		0.03,	0.01		0,	0,		0.03,	0.01,	

24, 49 25, 50	0, 0 0, 0	0.06, 0.54 0, 0.38	0, 0, 0, 0,	0.05, 0.07, 0.05, 0.15,
ω_1	5.9553	5.9562	5.9530	5.9534
ω_2	6.0455	6.0451	6.0455	6.0451
ω_3	10.4764	10.4757	10.4764	10.4755
ω_4	18.1448	18.1443	18.0921	18.0904
ω_5	25.4903	25.4892	25.2437	25.2436
ω_6	25.4939	25.4929	25.4939	25.4927

* 15% damage in elements 55, 56, 57 or 58 will result in the same set of natural frequencies.

** 10% damage in elements 1, 2, 3 or 4 will result in the same set of natural frequencies.

5.7 Conclusions

Hybridisation of RBF into DE leading to IPB-DE is presented for truss structural damage detection problems. Four structural damage detection test problems from three different truss structures are used to examine the search performance of the proposed approach. Several well established MHs and the proposed algorithms are then employed to solve the test problems. Numerical results reveal that the proposed hybrid algorithms of DE with RBF are the top performers for all test problems. Integrating RBF into the DE obviously improves DE performance. The proposed idea has the potential to be further applied to other inverse problems such as robot inverse kinematic analysis. Further improvement for meta-heuristic based structural health monitoring should be the purpose of a more reliable objective function rather than solely using the set of lowest natural frequencies. Detection of joint damage is another issue that will be focused on in future work.

Chapter VI
Conclusions and Future work

In this work, development of MHs for practical engineering optimisation is successfully conducted based on using surrogated assisted MHs, using parameter adaption and using a hybridization concept. Firstly, performance enhancement of a teaching-learning based optimizer (TLBO) for strip flatness optimization during a coiling process is proposed. The method is termed improved teaching-learning based optimization (ITLBO). The new algorithm is achieved by modifying the teaching phase of the original TLBO. The design problem is set to find a spool geometry and coiling tension in order to minimize flatness defects during the coiling process. Having implemented the new optimizer with flatness optimization for strip coiling, the results reveal that the proposed method gives a better optimum solution compared to the present state-of-the-art methods. Next, a sine cosine algorithm is extended to be self-adaptive and its main reproduction operators are integrated with the mutation operator of differential evolution. The new algorithm is called adaptive sine cosine algorithm integrated with differential evolution (ASCA-DE) and used to tackle the test problems for structural damage detection. The results reveal that the new algorithm outperforms a number of established meta-heuristics. In addition, a new meta-heuristic 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 meta-heuristics 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 the 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 meta-heuristics. Finally, this work proposes the integration of an inverse problem process using radial basis functions (RBFs) into meta-heuristics (MHs) for performance enhancement in solving structural health monitoring optimisation problems. A differential evolution (DE) algorithm is chosen as the MH for this study.

In this chapter, RBF is integrated into the DE algorithm for generating an approximate solution rather than approximating a function value as with traditional surrogate-assisted optimisation. Four structural damage detection test problems of three trusses are used to examine the search performance of the proposed algorithms. The results obtained from using various MHs and the proposed algorithms indicate that the new algorithm is the best for all test problems. DE search performance for structural damage detection can be considerably improved by integrating RBF into its procedure.

Base on this study, performance of MHs can be improve for various engineering applications based on using surrogated assisted MHs, using parameter adaption and using a hybridisation concept. The MH proposed in this work can be extended to other engineering optimisation problems such as robot inverse kinematic problem, robot and aircraft trajectory planning, flight dynamic and control etc., while the performance can be still more improve.

Reference

[1] D. Goldberg, K. Deb, and B. Korb, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex systems,* pp. 493-530, 1989.

[2] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization,* vol. 11, pp. 341-359, 1997/12/01 1997.

[3] G. Venter and J. Sobieszczanski-Sobieski, "Particle swarm optimization," *AIAA Journal,* vol. 41, pp. 1583-1589, 2003.

[4] T. Back, *Evolutionary Algorithms in Theory and Practice*. Oxford: Oxford University Press, 1996.

[5] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial intelligence through simulated evolution*. New York: John Wiley, 1966.

[6] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – A comprehensive introduction," *Natural Computing,* vol. 1, pp. 3-52, March 01 2002.

[7] S. Mirjalili, "SCA: A Sine Cosine Algorithm for solving optimization problems," *Knowledge-Based Systems,* vol. 96, pp. 120-133, 2016.

[8] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software,* vol. 69, pp. 46-61, 2014.

[9] 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, pp. 303-315, 2011.

[10] R. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *International Journal of Industrial Engineering Computations,* vol. 7, pp. 19-34, 2016.

[11] J. Zhang and A. C. Sanderson, "JADE: Adaptive Differential Evolution With Optional External Archive," *Evolutionary Computation, IEEE Transactions on,* vol. 13, pp. 945-958, 2009.

[12] R. Tanabe and A. Fukunaga, "Evaluating the performance of SHADE on CEC 2013 benchmark problems," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2013, pp. 1952-1959.

[13] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, 2014, pp. 1658-1665.

[14] 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,* vol. PP, pp. 1-1, 2017.

[15] 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,* vol. 28, pp. 138-149, 2015/03/01/ 2015.

[16] X. Liang, W. Li, Y. Zhang, and M. Zhou, "An adaptive particle swarm optimization method based on clustering," *Soft Computing,* vol. 19, pp. 431-448, February 01 2015.

[17] 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, pp. 182-197, 2002.

[18] N. Srinivas and K. Deb, "Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation,* vol. 2, pp. 221-248, 1994.

[19] M. 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 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014, pp. 1263-1270.

[20] C. A. C. Coello and M. S. Lechuga, "MOPSO: a proposal for multiple objective particle swarm optimization," in *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, 2002, pp. 1051-1056.

[21] 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,, Barcelona Spain, 2002.

[22] S. Mirjalili, S. Saremi, S. M. Mirjalili, and L. d. S. Coelho, "Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization," *Expert Systems with Applications,* vol. 47, pp. 106-119, 2016/04/01/ 2016.

[23] 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, pp. 1291-1300, 2013/04/01/ 2013.

[24] Z. Qingfu and L. Hui, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," *Evolutionary Computation, IEEE Transactions on,* vol. 11, pp. 712-731, 2007.

[25] D. Angus and C. Woodward, "Multiple objective ant colony optimisation," *Swarm Intelligence,* vol. 3, pp. 69-85, 2008.

[26] T. Robič and B. Filipič, "DEMO: Differential Evolution for Multiobjective Optimization," in *Evolutionary Multi-Criterion Optimization*. vol. 3410, C. Coello Coello, A. Hernández Aguirre, and E. Zitzler, Eds., ed: Springer Berlin Heidelberg, 2005, pp. 520-533.

[27] X. Zhang, Y. Tian, and Y. Jin, "A Knee Point-Driven Evolutionary Algorithm for Many-Objective Optimization," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION,* vol. 19, pp. 761-776, 2015.

[28] 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, pp. 524-541, 2015.

[29] R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-Inspired Coevolutionary Algorithms for Many-Objective Optimization," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION,* vol. 17, pp. 474-494, 2013.

[30] N. Pholdee, H. M. Baek, S. Bureerat, and Y.-T. Im, "Process optimization of a non-circular drawing sequence based on multi-surrogate assisted meta-heuristic algorithms," *Journal of Mechanical Science and Technology,* vol. 29, pp. 3427-3436, 2015.

[31] H.-M. Gutmann, "A radial basis function method for global optimization," *Journal of Global Optimization,* vol. 19, pp. 201-227, 2001.

[32] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard, "DACE a MATLAB Kriging Toolbox," Technical University of Denmark, Kongens Lyngby, 2002.

[33] X. Xiang, Y. Fan, A. Fan, and W. Liu, "Cooling performance optimization of liquid alloys GaIny in microchannel heat sinks based on back-propagation artificial neural network," *Applied Thermal Engineering,* vol. 127, pp. 1143-1151, 2017/12/25/ 2017.

[34] A. Rosales-Pérez, J. A. Gonzalez, C. A. Coello Coello, H. J. Escalante, and C. A. Reyes-Garcia, "Surrogate-assisted multi-objective model selection for support vector machines," *Neurocomputing,* vol. 150, Part A, pp. 163-172, 2015.

[35] N. Pholdee, S. Bureerat, H. M. Baek, and Y.-T. Im, "Surrogate Assisted Teaching Learning Based Optimisation for Process Design of a Non-circular Drawing Sequence," in *International Conference on Manufacture Engineering, Quality and Production System*, London UK, 2015.

[36] F. Lambiase, "Optimization of shape rolling sequences by integrated artificial intelligent techniques," *The International Journal of Advanced Manufacturing Technology,* vol. 68, pp. 443-452, 2013/09/01 2013.

[37] M. Costas, J. Díaz, L. Romera, and S. Hernández, "A multi-objective surrogate-based optimization of the crashworthiness of a hybrid impact absorber," *International Journal of Mechanical Sciences,* vol. 88, pp. 46-54, 2014.

[38] J. Wang, W. Shen, Z. Wang, M. Yao, and X. Zeng, "Multi-objective optimization of drive gears for power split device using surrogate models," *Journal of Mechanical Science and Technology,* vol. 28, pp. 2205-2214, 2014/06/01 2014.

[39] T. Braconnier, M. Ferrier, J. C. Jouhaud, M. Montagnac, and P. Sagaut, "Towards an adaptive POD/SVD surrogate model for aeronautic design," *Computers & Fluids,* vol. 40, pp. 195-209, 2011.

[40] C. Luo, S.-L. Zhang, C. Wang, and Z. Jiang, "A metamodel-assisted evolutionary algorithm for expensive optimization," *Journal of Computational and Applied Mathematics,* vol. 236, pp. 759-764, 2011.

[41] T. Massé, L. Fourment, P. Montmitonnet, C. Bobadilla, and S. Foissey, "The optimal die semi-angle concept in wire drawing, examined using automatic optimization techniques," *International Journal of Material Forming,* vol. 6, pp. 377-389, 2013/09/01 2013.

[42] L. Bo, Z. Qingfu, and G. G. E. Gielen, "A Gaussian Process Surrogate Model Assisted Evolutionary Algorithm for Medium Scale Expensive Optimization Problems," *Evolutionary Computation, IEEE Transactions on,* vol. 18, pp. 180-192, 2014.

[43] S. Chakraborty and A. Sen, "Adaptive response surface based efficient Finite Element Model Updating," *Finite Elements in Analysis and Design,* vol. 80, pp. 33-40, 2014.

[44] S. M. Elsayed, T. Ray, and R. A. Sarker, "A surrogate-assisted differential evolution algorithm with dynamic parameters selection for solving expensive optimization problems," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, 2014, pp. 1062-1068.

[45] L. Tong, S. Chaoli, Z. Jianchao, X. Songdong, and J. Yaochu, "Similarity- and reliability-assisted fitness estimation for particle swarm optimization of expensive problems," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, 2014, pp. 640-646.

[46] S. Bureerat, K. Wansasueb, and N. Pholdee, "Optimum radii and heights of U-shaped baffles in a square duct heat exchanger using surrogate-assisted optimization," *Engineering and Applied Science Research,* vol. 44, pp. 84-89, 2017.

[47] N. Pholdee, S. Bureerat, H. M. Baek, and Y.-T. Im, "Two-stage surrogate assisted differential evolution for optimization of a non-circular drawing sequence," *International Journal of Precision Engineering and Manufacturing,* vol. 18, pp. 567-573, 2017.

[48] S. Bureerat and J. Limtragool, "Structural topology optimisation using simulated annealing with multiresolution design variables," *Finite Elements in Analysis and Design,* vol. 44, pp. 738-747, 2008.

[49] 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. 459-471, 2007/11/01 2007.

[50] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research,* vol. 185, pp. 1155-1173, 2008.

[51] A. Husseinzadeh Kashan, "An efficient algorithm for constrained global optimization and application to mechanical engineering design: League championship algorithm (LCA)," *Computer-Aided Design,* vol. 43, pp. 1769-1792, 2011.

[52] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: charged system search," *Acta Mechanica,* vol. 213, pp. 267-289, 2010.

[53] N. Pholdee, S. Bureerat, W.-W. Park, D.-K. Kim, Y.-T. Im, H.-C. Kwon, and M.-S. Chun, "Optimization of flatness of strip during coiling process based on evolutionary algorithms," *International Journal of Precision Engineering and Manufacturing,* vol. 16, pp. 1493-1499, 2015.

[54] N. Pholdee, W.-W. Park, D.-K. Kim, Y.-T. Im, S. Bureerat, H.-C. Kwon, and M.-S. Chun, "Efficient hybrid evolutionary algorithm for optimization of a strip coiling process," *Engineering Optimization,* vol. 47, pp. 521-532, 2015.

[55] W.-W. Park, D.-K. Kim, Y.-T. Im, H.-C. Kwon, and M.-S. Chun, "Effects of processing parameters on elastic deformation of the coil during the thin-strip coiling process," *Metals and Materials International,* vol. 20, pp. 719-726, 2014.

[56] S. Yanagi, S. Hattori, and Y. Maeda, "Analysis model for deformation of coil of thin strip under coiling process," *Journal- Japan Society for Technology of Plasticity,* vol. 39, pp. 51-55, 1998.

[57] S. Bureerat, N. Pholdee, W.-W. Park, and D.-K. Kim, "An Improved Teaching-Learning Based Optimization for Optimization of Flatness of a Strip During a Coiling Process," in *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, 2016, pp. 12-23.

[58] N. Pholdee and S. Bureerat, "Structural health monitoring through meta-heuristics – comparative performance study," *Advances in Computational Design, An International Journal,* vol. 1, pp. 315-327, 2016.

[59] S. Bureerat and N. Pholdee, "Optimal Truss Sizing Using an Adaptive Differential Evolution Algorithm," *Journal of Computing in Civil Engineering,* p. 04015019, 2015.

[60] G. Lindfield and J. Penny, *Numerical methods: using MATLAB*: Academic Press, 2012.

[61] 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.

[62] H. Nezamabadi-pour, "A quantum-inspired gravitational search algorithm for binary encoded optimization problems," *Engineering Applications of Artificial Intelligence,* vol. 40, pp. 62-75, 2015/04/01/ 2015.

[63] 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," *Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore,* 2014.

[64] D. Tabak, A. Schy, D. Giesy, and K. Johnson, "Application of multiobjective optimization in aircraft control systems design," *Automatica,* vol. 15, pp. 595-600, 1979.

[65] 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 *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 2005, pp. 1-8.

[66] 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,* pp. 14853-7501, 2011.

[67] S. Rostami and F. Neri, "Covariance matrix adaptation pareto archived evolution strategy with hypervolume-sorted adaptive grid algorithm," *Integrated Computer-Aided Engineering,* vol. 23, pp. 313-329, 2016.

[68] N. Pholdee and S. Bureerat, "Estimation of Distribution Algorithm Using Correlation between Binary Elements: A New Binary-Code Metaheuristic," *Mathematical Problems in Engineering,* vol. 2017, 2017.

[69] 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, pp. 487-511, 2016/11/01/ 2016.

[70] A. Majumdar, D. K. Maiti, and D. Maity, "Damage assessment of truss structures from changes in natural frequencies using ant colony optimization," *Applied Mathematics and Computation,* vol. 218, pp. 9759-9772, 2012.

[71] A. Kaveh and A. Zolghadr, "An improved CSS for damage detection of truss structures using changes in natural frequencies and mode shapes," *Advances in Engineering Software,* vol. 80, pp. 93-100, 2015.

[72] K. Wansaseub, N. Pholdee, and S. Bureerat, "Optimal U-shaped baffle square-duct heat exchanger through surrogate-assisted self-adaptive differential evolution with

neighbourhood search and weighted exploitation-exploration," *Applied Thermal Engineering,* vol. 118, pp. 455-463, 2017/05/25/ 2017.

[73] S. Bureerat and N. Pholdee, "Inverse problem based differential evolution for efficient structural health monitoring of trusses," *Applied Soft Computing,* vol. 66, pp. 462-472, 2018/05/01/ 2018.

[74] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," *Advances in Engineering Software,* vol. 95, pp. 51-67, 2016.

[75] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-Based Systems,* vol. 89, pp. 228-249, 2015.

[76] N. Hansen, S. D. Muller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evol. Comput.,* vol. 11, pp. 1-18, 2003.

APPENDIX A

LIST OF PUBLICATIONS

LIST OF PUBLICATIONS

[1] Bureerat, S., & Pholdee, N. (2018). Inverse problem based differential evolution for efficient structural health monitoring of trusses. Applied Soft Computing, 66, 462-472 (IF =3.541, Q1)

[2] Pholdee, N. and Bureerat, S., (2017) Estimation of Distribution Algorithm Using Correlation between Binary Elements: A New Binary-Code Metaheuristic, Mathematical Problems in Engineering, 2017, 6043109, 15 pages. doi:10.1155/2017/6043109, (IF = 0.802, Q3)

[3] Bureerat, S. and Pholdee, N., (2017) Adaptive Sine Cosine Algorithm Integrated with Differential Evolution for Structural Damage Detection, Lecture Notes in Computer Science, 10404, 71-86. (SNIP=0.552)

[4] Pholdee, N. and Bureerat, S., (2016) An Improved Teaching-Learning Based optimization for Optimization of Flatness of a Strip during Coiling Process, Lecture Notes in Computer Science, 10053, 12-23. (SNIP=0.552)

Applied Soft Computing 66 (2018) 462–472

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Inverse problem based differential evolution for efficient structural health monitoring of trusses

Sujin Bureerat, Nantiwat Pholdee*

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

ARTICLE INFO

Article history:
Received 29 December 2016
Received in revised form
10 November 2017
Accepted 23 February 2018
Available online 27 February 2018

Keywords:
Structural health monitoring
Meta-heuristics
Inverse problem
Differential evolution
Damage detection

ABSTRACT

This paper proposes the integration of an inverse problem process using radial basis functions (RBFs) into meta-heuristics (MHs) for performance enhancement in solving structural health monitoring optimisation problems. A differential evolution (DE) algorithm is chosen as the MH for this study. In this work, RBF is integrated into the DE algorithm for generating an approximate solution rather than approximating the function value as with traditional surrogate-assisted optimisation. Four structural damage detection test problems of two trusses are used to examine the search performance of the proposed algorithms. The results obtained from using various MHs and the proposed algorithms indicate that the new algorithm is the best for all test problems. DE search performance for structural damage detection can be considerably improved by integrating RBF into its procedure.

1. Introduction

Structural damage detection is a technique used to identify the presence of structural damage, localising it, and assessing the severity [1]. Structural damage takes place due to several reasons such as defects in structures, cracks and corrosion in structural elements, and incomplete construction of the structures. Such mistakes can cause the structures to have a shortened service life and other undesirable accidents. As a result, engineers have had to develop techniques to predict and prevent it. Visual inspection of damage is one straightforward technique usually employed, however, its main disadvantage is the inability to detect internal defects and cracks. Moreover, it is difficult to check throughout a large structure and find damage locations. Therefore, a more sophisticated means should be used to detect damage locations using only one measurement.

One of the most popular damage detection techniques is the use of changes in structural modal data. The idea is that the modal data of a healthy structure is measured and used as the baseline. Once it has been found that the modal data alters from its normal values, it means structural damage may have taken place. Over several decades, researchers have investigated vibration-based damage detection of mechanical systems and structures [2–7]. The use of fuzzy logic systems [8], neural networks [4,7], and other types of soft computing has been proposed. Recently, meta-heuristics have been implemented for perform structural health monitoring based on vibration measurement. The problem of damage detection is treated as an optimisation inverse problem [6,9–12]. The advantage of this strategy is that it is easy to use, can be used to check throughout a large structure, and can locate damage positions within one measurement of modal testing. Although many researchers have demonstrated using a number of MHs for solving the optimisation problems [6,10,11,13–15], it has been found that they failed to assess the performance of MHs properly. The algorithm search convergence and usability was reported but the search consistency has never been examined. For practicality, an algorithm without the guarantee of search consistency will be always questioned, whether it can be used in reality or not. In this regards, developing MHs for optimising an inverse problem of damage detection to improve search convergence simultaneously with search consistency is an interesting topic.

Over the last few decades, development of MHs with an emphasis on improving the convergence rate and consistency can be accomplished in several ways, such as introducing new search concept MHs [16–18], using a hybridisation concept [19], using parameter adaption [20,21], or using surrogate assisted MHs [22]. The implementation of a surrogate assisted MH is usually required when the optimisation problem has computationally expensive

* Corresponding author.
E-mail address: nantiwat@kku.ac.th (N. Pholdee).

https://doi.org/10.1016/j.asoc.2018.02.046
1568-4946/

Nomenclature

$[\mathbf{K}]$	Structural stiffness matrix
$[\mathbf{M}]$	Structural mass matrix
λ_j	jth mode eigenvalue
ϕ_j	jth mode eigenvector or mode shape.
n_{dof}	Size of the mass and stiffness matrices.
$[\mathbf{m}_e]$	Element mass matrices
$[\mathbf{k}_e]$	Element stiffness matrices.
n_e	Number of elements
p_i	Percentage of damage in the ith element.
n_{mode}	Number of lowest vibration modes
F	Scaling factor
$F_{\min}$	Maximum scaling factor
$F_{\max}$	Minimum scaling factor
$\mathbf{x}_{r,i}$	ith randomly selected individual
$\mathbf{x}_{old}$	Current solution (parent)
$\mathbf{x}_{new}$	New candidate solution
rand	Uniform random number ranged from 0 to 1
rand(0,1)	Random number, either 0 or 1
CR	Crossover rate
D	Number of design variables
c_k	Interpolation coefficients
φ	RBF kernel function
$\boldsymbol{\omega}_{damage}$	Natural frequencies of the damaged structure (Target vector)
$\mathbf{x}_{damage}$	Solution vector containing n_e element damage percentages

function evaluations. The simple strategy of surrogate assisted optimisation is carried out in such a way that the design of the experiment uses a technique such as Latin hypercube sampling to generate a set of training points. With those training points, actual function evaluations are performed. A surrogate model, a form of function that requires significantly less computation time, is then constructed based on the training points and their function values. Thereafter, optimisation can be performed based on using the surrogate model instead of actual function evaluations. This can greatly reduce optimisation running time. Although a surrogate model can be used to improve MHs search convergence (by reducing the number of real expensive function evaluations) and also search consistency, it is yet to find that such a model is applied to an inverse problem for structural damage detection.

Therefore, this paper presents a new, efficient MH for structural damage detection as a hybridisation of a radial basis function (RBF) interpolation and differential evolution (DE). In this work, the RBF is integrated into the main procedures of DE for approximating design solutions rather than objective functions as with traditional surrogate-assisted optimisation. Four structural damage detection and localisation test problems from two truss structures are used for performance assessment of a number of MHs and the proposed algorithm. The results obtained from the various algorithms will be statistically compared in terms of both convergence rate and consistency.

2. Natural-frequency-based damage detection and localisation

In this study, structural damage detection using changes in structural natural frequencies is considered. The detection strategy can be used for damage detection of truss elements due to corrosion, crack and yielding of members due to fatigue. This approach is based on implementing modal testing incorporated with a finite element model. Initially, the natural frequencies (usually the lowest n_{mode} natural frequencies) of the structure in a normal condition will be used as the baseline. In practice, the natural frequencies and mode shapes will be measured and the finite element model will be updated so that both measured and computed modal parameters are equivalent. The finite element model used herein is a simple linear undamped free vibration which can be expressed as:

$$[\boldsymbol{K}]\left\{\phi_j\right\} - \lambda_j[\boldsymbol{M}]\left\{\phi_j\right\} = 0 \tag{1}$$

The structural natural frequencies can be computed as

$$\omega_j = \sqrt{\lambda_j} \quad , j = 1, 2, 3, \ldots, n_{dof} \tag{2}$$

The mass and stiffness matrices can be obtained from assembling all element mass and stiffness matrices, which can be expressed as:

$$[\mathbf{M}] = \sum_{i=1}^{n_e} [\mathbf{m}_e]$$

and

$$[\boldsymbol{K}] = \sum_{i=1}^{n_e} [\boldsymbol{k}_e]\,. \tag{3}$$

In cases that damage in the structural element occurs, the structural natural frequencies of the structure will be different from those of the baseline structure. To localise the damage, it is assumed that the values of the structural stiffness matrix are altered, which can be written in terms of element structural damage percentage. As a result, the altered structural stiffness matrix of the damaged structure is of the form

$$[\boldsymbol{K}_d] = \sum_{i=1}^{n_e} \frac{100 - p_i}{100} [\boldsymbol{k}_e]\,. \tag{4}$$

The optimisation problem is then formulated by assigning all the values of element damage percentages as a design solution $\mathbf{x} = \{p_1, \ldots, p_{ne}\}^T$. The objective function is to minimise the root mean square error:

$$\text{Min} : f(\mathbf{x}) = \sqrt{\frac{\sum_{j=1}^{n_{mode}} \left(\omega_{j,damage} - \omega_{j,computed}\right)^2}{n_{mode}}} \tag{5}$$

where $\omega_{j,damage}$ is the structural natural frequency of mode j obtained from measuring a damaged structure. n_{mode} is the number of lowest vibration modes used for the damage detection. $\omega_{j,computed}$ is the structural natural frequency of mode j obtained from solving (1) using $[\mathbf{K}_d]$ instead of $[\mathbf{K}]$. The optimum solution having the objective function value close to zero gives accurate damage localisation. The values of the element damage percentage indicate where the damage takes place.

3. Test problems with trusses

To study performance assessment of a number of MHs on tackling damage detection optimisation, two truss structures are employed in this work. For the sake of simple investigation, truss damage is simulated whereas the natural frequencies of structures are computed from finite element analysis rather than measuring real structure modal data. Only truss element damages are taken into consideration. It should be noted that free vibration is simulated for all cases without considering gravity loads. The trusses are detailed as follows.

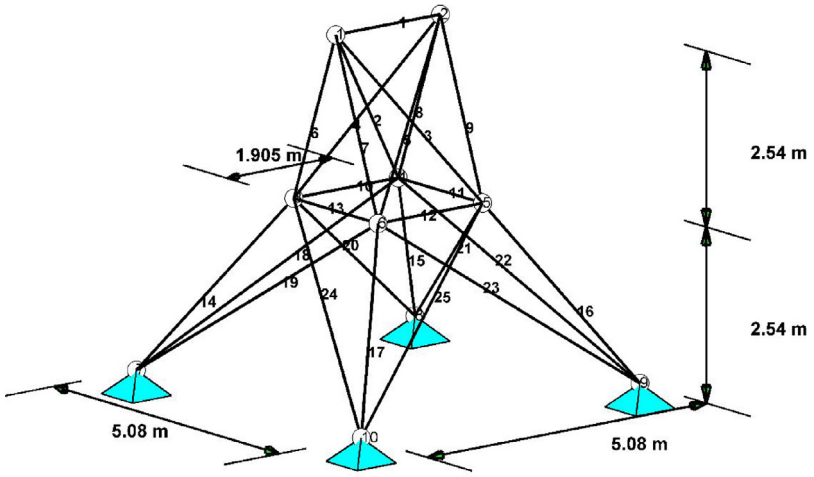


Fig. 1. Twenty-five bar truss.

Table 1
Natural frequencies (Hz) of damaged and undamaged 25 bar structure.

Mode	Undamaged reported in [6][a]	Undamaged calculated by commercial software (Ansys academic version)[a]	Undamaged calculated in this study[a]	35% damage at element number 7	35% damage at element number 7 and 40% damage at element number 9
1	70.9924	69.782	69.7818	69.1393	68.5203
2	74.0851	72.822	72.8217	72.2006	71.3167
3	97.5390	95.876	95.8756	95.3372	94.5625
4	122.2281	120.14	120.1437	119.8852	119.6514
5	121.9300	121.50	121.5017	121.4774	121.4253
6	–	125.01	125.0132	125.0130	125.0129

[a] The natural frequencies are slightly different which could be due to the numerical algorithm used and truncation errors.

Table 2
Natural frequencies (Hz) of damaged and undamaged 72 bar structure.

Mode	Undamaged reported in [11][a]	Undamaged calculated by commercial software (Ansys academic version)[a]	Undamaged calculated in this study[a]	15% damage at element number 55	15% damage at element number 58 and 10% damage at element number 4
1	6.0434	5.4977	6.0455	5.9553	5.9530
2	6.0441	5.4977	6.0455	6.0455	6.0455
3	10.4627	9.5181	10.4764	10.4764	10.4764
4	18.2275	16.594	18.2297	18.1448	18.0921
5	25.4466	23.213	25.4939	25.4903	25.2437
6	25.4510	23.213	25.4939	25.4939	25.4939

[a] The natural frequencies are slightly different which could be due to the numerical algorithm used and truncation errors.

3.1. Twenty-five-bar truss

The structure having 25 bar is depicted in Fig. 1 [6]. All bar element cross-sectional areas are set to be 6.4165 mm^2. Material density and Young modulus are given as 7850 kg/m^3 and 200 GPa, respectively. Two damage case studies are assumed as Case I: 35% damage on element 7 (Note that 35% damage on elements 6, 8 or 9 will result in the same set of natural frequencies), and Case II: 35% and 40% damage at elements 7 and 9 (Note that 35% damage in element 6 and 40% damage in element 8 will result in the same set of natural frequencies for this case). The pin supports are applied to node numbers 7, 8, 9 and 10. The data of natural frequencies of the damaged and undamaged 25-bar truss are given in Table 1.

3.2. Seventy-two-bar truss

The 72-bar truss structure is displayed in Fig. 2 [11] where four non-structural masses of 2270 kg are attached to the top nodes. The values of all bar element cross-sectional areas are set to be 0.0025 m^2. Material density and modulus of elasticity are 2770 kg/m^3 and 6.98×10^{10} Pa, respectively. Two cases of damage are generated as Case I: 15% damage at element number 55 (Note that 15% damage in elements 56, 57, or 58 will result in the same set of natural frequencies as that of element 55), and Case II: 10% damage at element number 4 and 15% damage at element number 58 (90, 180, and 270° rotation along the z axis will lead to the same set of natural frequencies). The pin supports are applied to nodes number 17, 18, 19 and 20. The values of natural frequencies of the damaged and undamaged 72-bar truss are given in Table 2.

4. Hybrid radial basis function and differential evolution for truss damage detection

The purpose of using MHs for truss damage detection is to solve the optimisation problem with the objective function (5) subject to bound constraints of **x**. The advantages of using MHs are their simplicity in use, capability of global search, derivative-free feature, and robustness. Using *meta*-heuristics implies that a user has less worry about mode switching during an optimisation run while this phenomenon may occur in cases of using a gradient-based opti-

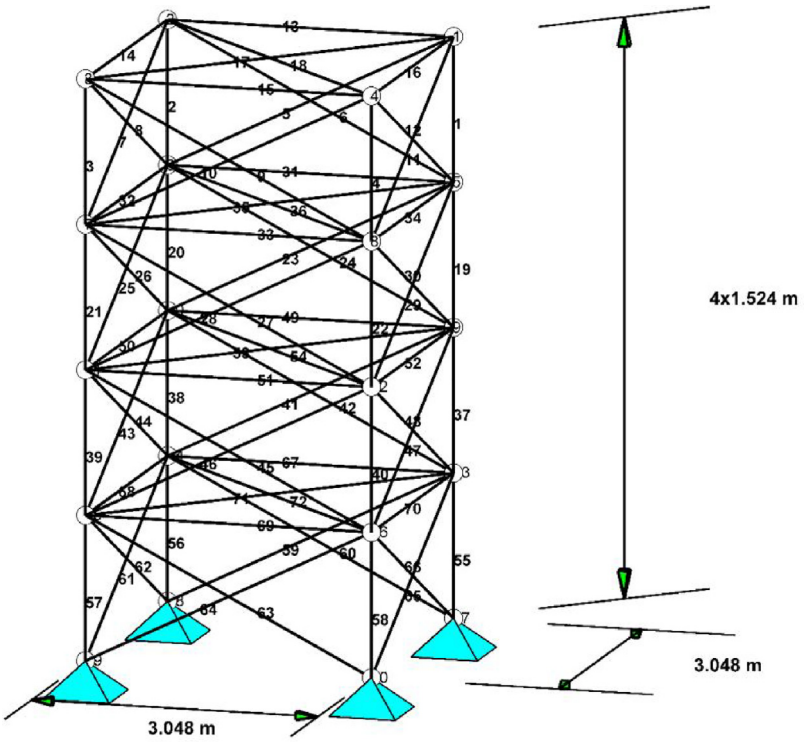


Fig. 2. Seventy-two bar truss.

miser. The detection approach can be used for real-time monitoring provided that an employed MH is adequately powerful.

4.1. Differential evolution

Differential evolution is a population based method which was first proposed by Storn and Price in 1997 [23]. The method contains two main steps for searching an optimum, including mutation and crossover where the acronym DE/*x*/*y*/*z* is used to specify different mutation and crossover strategies. The variable *x* is used to specify a vector for mutation which can be *best* (the best individual) or *rand* (random individual) while *y* and *z* specify the number of vector pairs used in mutation and the choice of a crossover scheme, respectively. For example, as used in this work, DE/best/2/bin means that the best individual and two different vector pairs are used in the mutation step while the binomial crossover is employed. The mutation operation can be expressed as follows:

$$\boldsymbol{u}_i = \boldsymbol{x}_{\text{best}} + (-1)^{rand(-1,0)} F\left(\boldsymbol{x}_{r,1} + \boldsymbol{x}_{r,2} - \boldsymbol{x}_{r,3} - \boldsymbol{x}_{r,4}\right). \quad (6)$$

In this work, F is a uniform random number in the range of $[F_{\min}, F_{\max}]$. For the i-th mutant individual $\boldsymbol{u}_i^T = [x_{new,1}, \ldots, x_{new,D}]$ and its corresponding parent $\boldsymbol{x}_{\text{old}}^T = [x_{old,1}, \ldots, x_{old,D}]$, the binary crossover can be operated leading to a new candidate solution $\mathbf{x}_{new}$ as

$$x_{new,j} = \begin{cases} u_j\,; & rand < CR \\ x_{old,j}\,; & otherwise \end{cases} \quad j = 1, 2, 3, \ldots, D. \quad (7)$$

The selection operator is carried out by comparing $\mathbf{x}_{new}$ and its parent $\mathbf{x}_{old}$ where the better will survive to the next generation.

The DE computational steps are shown in Algorithm 1. Initially, a set of the population is generated by means of randomisation and their objective function values are evaluated. After obtaining the best individual, the offspring are generated by mutation (eq.6) and then crossover (eq.7). Then, the next generation is selected and the search process will be repeated until a termination criterion is reached.

Algorithm 1 DE search procedure.

Input: population size, number of generations, algorithm parameters.
Output: $\mathbf{x}_{best}$, f_{best}
Main algorithm
1.:Initialise a population, calculate their objective function values and set as the current population.
2: Find the best individual
3: Generate a new population from the current population using DE mutation (eq.6) and DE crossover (eq.7).
4: Evaluate objective function values of the members of the new population.
5: Select the next generation from the newly generated and current populations.
6: Set the selected population from step 5 as the next generation.
7: If a termination condition is not met, go to step 2. Otherwise, stop the algorithm.

4.2. Inverse problem-based differential evolution

This subsection details the proposed differential evolution based on using an inverse problem concept. In optimisation, the radial basis function is traditionally used for approximating an objective function value for problems with expensive function evaluation [22,24]. Nevertheless, in this work RBF is conversely implemented. It will be used to approximate a design solution **x** that is expected

Table 3
MH Parameters settings.

MH	Parameter settings
Whale optimization algorithm (WOA) [16]	– The parameter $b = 1$ – Other parameters are iteratively adapted.
Sine Cosine algorithm (SCA) [17]	– The constant parameter $a = 2$.
Moth-flame optimisation algorithm (MFO) [18]	– The constant parameter $b = 1$ – Other parameters are iteratively adapted.
Differential evolution (DE) [23]	– Using DE/best/2/bin strategy – Scaling factor (F) = 0.8, – probability of choosing elements of mutant vectors (CR) = 0.5
Artificial bee colony algorithm (ABC) [25]	– The number of food sources for employed bees = $n_P/2$. – A trial counter to discard a food source = 100.
Real-code ant colony optimisation (ACOR) [26]	– The parameter, $q = 0.2$ – The parameter, $\xi = 1$
Charged system search (ChSS) [27]	– The number of solutions in the charge memory = $0.2 \times n_P$ – The charged moving considering rate = 0.75 − the parameter $PAR = 0.5$
League championship algorithm (LCA) [28]	– The probability of success P_c = 0.9999 – The decreasing rate to decrease P_c = 0.9995
Simulated annealing (SA) [29]	– Starting temperature = 10 – Ending temperature = 0.001
Particle swarm optimisation (PSO) [30]	For each loop, n_{mode} candidates are created by mutating on the current best solution while other n_{mode} candidates are created from mutating the current parent. The best of those $2n_{mode}$ solutions are set as an offspring to be compared with the parent. – The starting inertia weight = 0.5 – The ending inertia weight = 0.01 – The cognitive learning factor = 0.5 – The social learning factor = 0.5
Evolution strategies (ES) [31]	The algorithm uses a binary tournament selection operator and a simple mutation without the effect of rotation angles.
Teaching-learning-based optimisation (TLBO) [32]	Parameter settings are not required.
Adaptive differential evolution (JADE) [20]	The parameters are self-adapted during an optimisation process.
Evolution strategy with covariance matrix adaptation (CMAES) [21]	The parameters are self-adapted during an optimisation process.
IPB-DE	Use the DE parameter setting.

corresponding to the target damage conditions. Given that the vector of target natural frequencies ($\boldsymbol{\omega}_{damage}$) contains n_{mode} lowest natural frequencies of the damaged structure, the idea is to find a solution vector $\mathbf{x}_{damage}$ containing n_e element damage percentages by means of interpolation. During MH search, if we have a set of N design solutions $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ which corresponds to a set of N vectors of natural frequencies $\{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_N\}$, these data will be used for RBF training. In contrast to surrogate-assisted optimisation, the natural frequency vector will be set as independent variables whereas the design vector $\mathbf{x}$ will be set as dependent variables. The i^{th} element of $\mathbf{x}_{damage}$ that is expected to give the target vector of natural frequencies of the damaged truss is expressed as:

$$x_{damage,i} = \sum_{k=1}^{N} c_k \varphi\left(\|\boldsymbol{\omega}_k - \boldsymbol{\omega}_{damage}\|\right) \tag{8}$$

where c_k is the interpolation coefficients to be determined, and φ is a RBF kernel function. $\|\boldsymbol{\omega}_k - \boldsymbol{\omega}_{damage}\|$is the distance between $\boldsymbol{\omega}_k$ and $\boldsymbol{\omega}_{damage}$. For x_i, interpolation coefficients c_k can be found from solving the system of linear equations

$$\sum_{k=1}^{N} c_k \varphi\left(\|\boldsymbol{\omega}_k - \boldsymbol{\omega}_l\|\right) = x_i\left(\boldsymbol{\omega}_l\right); \quad \text{for } i = 1, \ldots, n_e \text{ and } l = 1, \ldots, N \tag{9}$$

where $x_i(\boldsymbol{\omega}_l)$ is the i^{th} element of the l^{th} solution vector in the training set $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$. Eq. (9) can be written in a matrix form as

$$\mathbf{Ac} = \mathbf{b} \tag{10}$$

where $A_{k,l} = \varphi(||\boldsymbol{\omega}_k - \boldsymbol{\omega}_l||)$. It is required to compute n_e sets of the interpolation coefficients according to n_e elements of $\mathbf{x}$. In practice, the matrix $\mathbf{A}$ is generated and inverted once, and will be used to calculate n_e sets of the coefficients.

Having determined the sets of interpolation coefficients c_k for all n_e elements of $\mathbf{x}$ by using (9), the elements of $\mathbf{x}_{damage}$ can be found from using Eq. (8). The search procedure for hybridised RBF and DE which will be termed inverse problem-based differential evolution (IPB-DE) according to its computation nature can be carried out in such a way that, after the reproduction step 3 in Algorithm 1, the next generation is selected in step 5. The worst solution in the next generation is then replaced by $\mathbf{x}_{damage}$. The procedure of the hybrid algorithm IPB-DE is detailed in Algorithm 2 while the flowchart for the IPB-DE algorithm is shown in Fig. 3. The process starts by creating an initial population by using the Latin hypercube sampling (LHS) technique instead of the Monte Carlo technique. Those solutions in the initial population are then saved to the RBF database for training RBF. Offspring are then created by means of reproduction of DE. The candidate solution $\mathbf{x}_{damage}$ is created using Equations (8–9). Having performed a selection operation, the worst solution in the next generation is replaced by $\mathbf{x}_{damage}$. The best solution from the

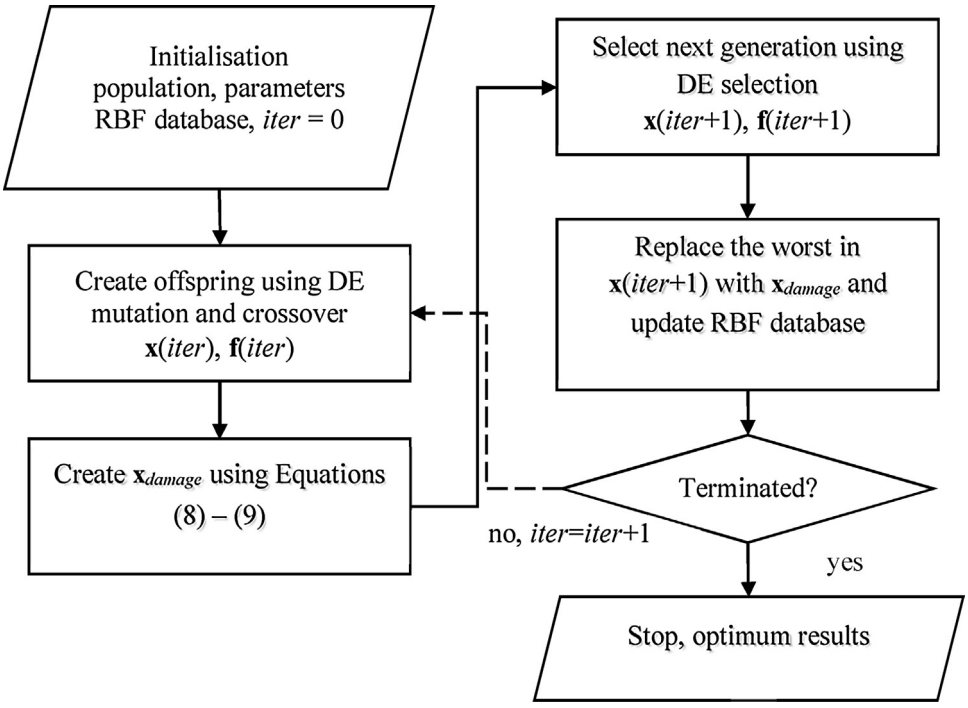


Fig. 3. Flow chart of IPB-DE.

offspring and $\mathbf{x}_{damage}$ are then added to the RBF database which will be used as training points during the optimisation search. As the process continues, the RBF database is improved and expected to give more accurate results. The procedure is repeated until fulfilling the termination criteria.

Algorithm 2 IPB-DE

Input: population size (n_p), number of generations (n_{iter}), algorithm parameters, the natural frequencies measured from the damaged structure ($\boldsymbol{\omega}_{damage}$)

Output: $\mathbf{x}_{best}$, f_{best}

Main algorithm

1: Generate an initial set of design variables $\mathbf{x}$ using LHS, calculate the natural frequencies ($\boldsymbol{\omega}$) and objective function values ($\mathbf{f}$), set $\mathbf{x}$ and $\mathbf{f}$ as the current population and save $\mathbf{x}$ and $\boldsymbol{\omega}$ in the RBF database.
2: Find the best solution.
3: Generate offspring from the current population using the DE mutation and binomial crossover operators (reproduction) and then perform function evaluations.
4: Select design solutions from the offspring and the current population.
5: Generate $\mathbf{x}_{damage}$ using the training points from the RBF database using Equations (9) and then (8).
6: Calculate the natural frequencies ($\boldsymbol{\omega}$) and objective function value (f) of $\mathbf{x}_{damage}$.
7: Update the RBF database by adding to it the data of the best solution from the offspring and $\mathbf{x}_{damage}$.
8: Replace the worst solution in the next generation with $\mathbf{x}_{damage}$.
9: If a termination condition is not met, go to step 2. Otherwise, stop the algorithm.

5. Numerical experiment

To verify the search performance of the proposed IPB-DE, several MHs are compared based on solving the aforementioned truss damage detection problems. The employed methods are said to be established while some of them are regarded as the currently best optimisers of this type. Given that n_P is a population size, MHs and their optimisation parameter settings used in this work are detailed in Table 3 (it should be noted that details of notations can be found in the corresponding references for each method) [9]:

Each optimisation algorithm is employed to solve each test problem for 30 independent runs. The number of iterations (generations) is 300 for all case studies while the population size is set to be 30 and 50 for 25-bar and 72-bar trusses respectively. For the optimisers using different population sizes from the aforementioned values, their search processes are terminated with the total number of functions evaluations (FEs) equal to 30×300 and 50×300 for 25-bar and 72-bar trusses respectively. Another termination criterion is when one of the design solutions in the current population has an objective function value less than or equal to 1×10^{-3}. It should be noted that the numbers of FEs used in this study can be considered insufficient for some MH optimisers. However, these values are used to find out really powerful algorithms. For all test problems, six lowest natural frequencies ($n_{mode} = 6$) are used to compute the objective function values. This number of selected frequencies is reasonable since, in practice, it is easier to accurately measure fewer lowest natural frequencies.

Table 4
Comparison of various RBF kernels for solving 72 bar truss Case II.

DE with RBF kernel	Mean objective function Values	No. of successful runs from 30 runs	Mean of FEs
Gaussian	0.0011	25	6856
Multiquadric	0.0104	5	13993
Inverse quadratic	0.0032	14	12221
Linear	0.0117	8	13819
Polynomial order 2	0.0039	15	10807

6. Results and discussion

Initially, the effect of RBF kernels on the performance of the proposed algorithm was investigated. The last test problem, 72 bar truss with 15% damage at element number 58 and 10% damage at element number 4 which is said to be the most complicated problem, was used. Table 4 shows the results obtained from using a variety of RBF kernel functions. The mean values of the objective function are used to indicate the search convergence of the algorithms in cases that the objective function threshold (1×10^{-3}) is not met during an optimisation run. Otherwise, the mean number of FEs is used as an indicator. The algorithm that is terminated by the objective function threshold is clearly the superior method and any optimisation run being stopped with this criterion is considered a successful run. The number of successful runs from 30 optimisation runs denoted as "No. of successful runs from 30" is the total number that the algorithm can meet the target objective function value (1×10^{-3}). It is used to measure the algorithm reliability. From Table 4, the best performer is the Gaussian kernel, while the second best and the third best are the Polynomial kernel and the Inverse quadratic kernel, respectively. Thus, the Gaussian kernel is used in this study.

Comparison of various ranges $[F_{\min}, F_{\max}]$ of a scaling factor and *CR* values using DE with the best RBF kernel for solving the 72 bar truss with 15% damage at element number 58 and 10% damage at element number 4 is shown in Table 5. It is found that for all implemented intervals of $[F_{\min}, F_{\max}]$, the performance increases when the value of *CR* increases. The highest DE performance is obtained when the range $[F_{\min}, F_{\max}]$ and *CR* are set to be [0.2, 0.8] and 0.8, respectively.

The results obtained from the various MHs from solving the six test problems are given in Tables 6–9.

6.1. Twenty-five-bar truss

For the 25-bar truss with 35% damage at element 7, the results are given in Table 6. The best performer based on the mean objective function values is IPB-DE while the second and third best are DE and JADE respectively. When considering the number of successful runs, seven optimisers including WOA, MFO, SCA, DE, TLBO, JADE and IPB-DE can detect the damage in the structures. The most efficient optimisers are SCA and IPB-DE that can detect the dam-

Table 6
Results for 25 bar truss Case I.

Optimiser	Mean objective function value	No. of successful runs from 30 runs	Mean of FEs
WOA	0.0357	8	6993
MFO	0.0279	3	8686
SCA	0.0270	24	3262
DE	0.0017	19	6019
ABC	0.0135	0	9000
ACOR	0.0089	0	9000
ChSS	0.1385	0	9000
LCA	0.9036	0	9000
SA	0.0089	0	9000
TLBO	0.0077	6	7772
CMAES	0.0033	0	9000
ES	0.0308	0	9000
PSO	8.3830	0	9000
JADE	0.0026	2	8953
IPB-DE	0.0012	25	4486

Table 7
Results for 25 bar truss Case II.

Optimiser	Mean objective function value	No. of successful runs from 30 runs	Mean of FEs
WOA	0.1301	0	9000
MFO	0.0336	1	8876
SCA	0.0930	0	9000
DE	0.0096	27	5220
ABC	0.0326	0	9000
ACOR	0.0125	0	9000
ChSS	0.1590	0	9000
LCA	0.8080	0	9000
SA	0.0269	0	9000
TLBO	0.0405	1	8917
CMAES	0.0115	0	9000
ES	0.0356	0	9000
PSO	8.6012	0	9000
JADE	0.0042	6	8875
IPB-DE	0.0010	30	3757

Table 8
Results for 72 bar truss Case I.

Optimiser	Mean objective function value	No. of successful runs from 30 runs	Mean of FEs
WOA	0.0082	22	4832
MFO	0.0270	2	14783
SCA	0.0070	23	4793
DE	0.0087	14	12887
ABC	0.2184	0	15000
ACOR	0.0014	6	14831
ChSS	0.1727	0	15000
LCA	1.1499	0	15000
SA	0.0097	0	15000
TLBO	0.0035	27	5781
CMAES	0.0053	0	15000
ES	0.0010	29	9335
PSO	1.9146	0	15000
JADE	0.0019	1	15000
IPB-DE	0.0009	30	3155

Table 5
Comparison of various ranges of *F* and *CR* values for solving 72 bar truss Case II.

DE with Gaussian RBF kernel		Mean objective function value	No. of successful runs from 30 runs	Mean of FEs
$[F_{\min}, F_{\max}]$	*CR*			
[−1.5,1.5]	0.3	0.0027	1	15000
[−1.5,1.5]	0.5	0.0013	16	12983
[−1.5,1.5]	0.8	0.0011	24	7648
[0.2,0.8]	0.3	0.0025	0	15000
[0.2,0.8]	0.5	0.0011	21	12344
[0.2,0.8]	0.8	0.0011	25	6856
[−2,−2]	0.3	0.0042	0	15000
[−2,−2]	0.5	0.0014	9	14496
[−2,−2]	0.8	0.0014	21	9940

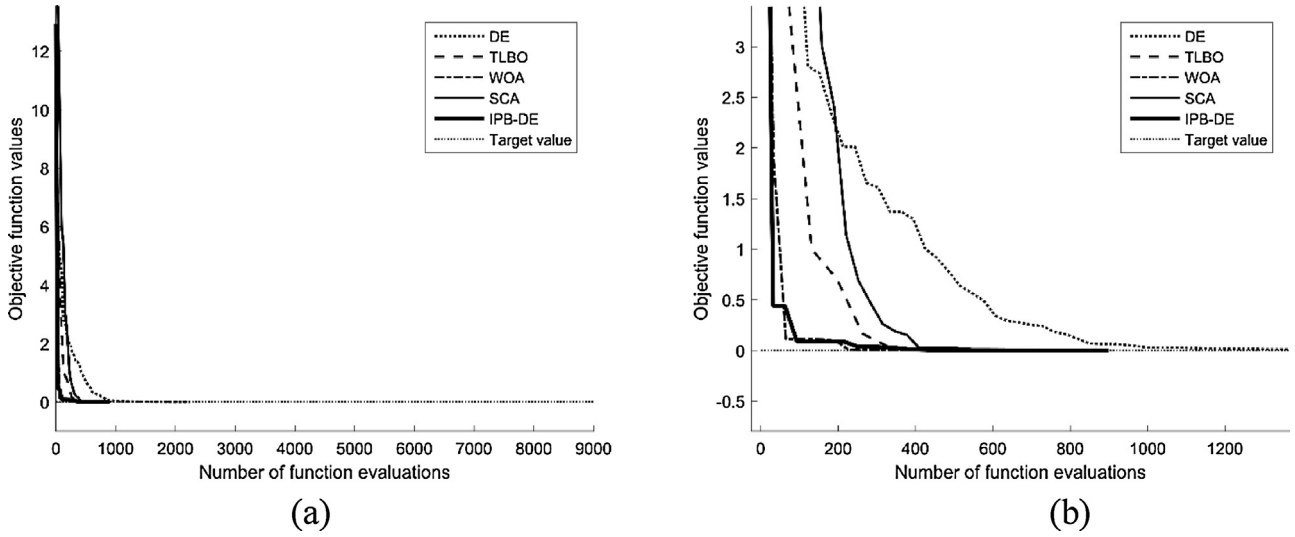


Fig. 4. Search history for the case 25 bar Case I, (a) original, (b) zoom in.

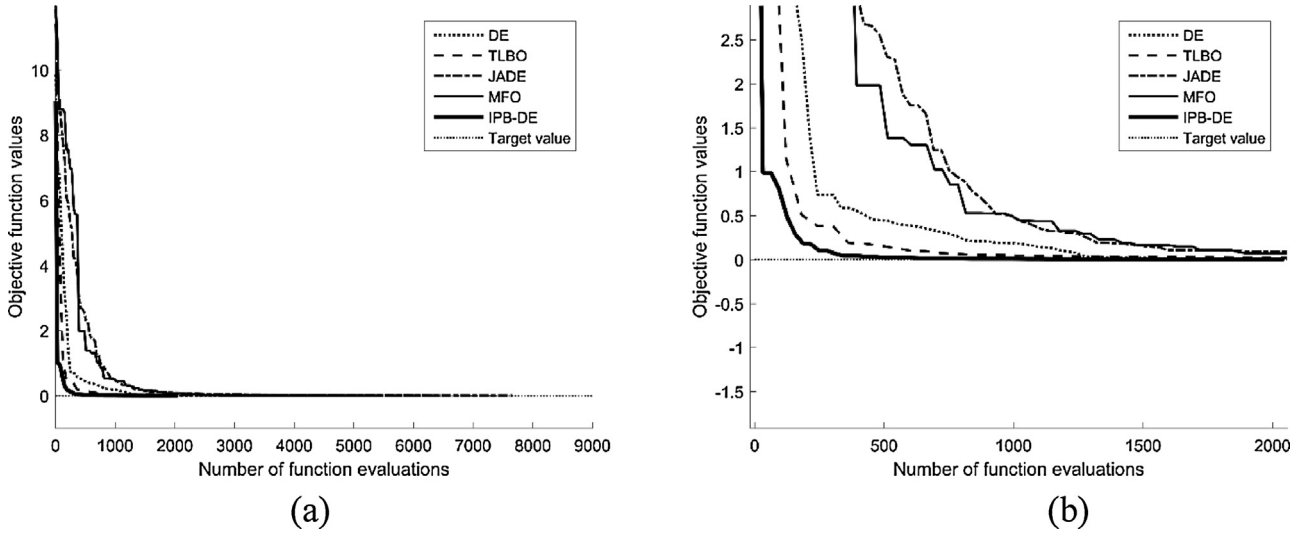


Fig. 5. Search history for the case 25 bar truss Case II, (a) original, (b) zoom in.

Table 9
Results for 72 bar truss Case II.

Optimiser	Mean objective function value	No. of successful runs from 30 runs	Mean of FEs
WOA	0.0189	0	15000
MFO	0.0137	1	14935
SCA	0.0260	2	14502
DE	0.0127	7	13963
ABC	0.1591	0	15000
ACOR	0.0058	0	15000
ChSS	0.1348	0	15000
LCA	1.1049	0	15000
SA	0.0129	0	15000
TLBO	0.0045	7	13503
CMAES	0.0050	0	15000
ES	0.0023	2	14940
PSO	1.7726	0	15000
JADE	0.0031	0	15000
IPB-DE	0.0011	25	6856

ages of the structure for 24 and 25 times out of 30 runs within the average of 3262 and 4486 function evaluations respectively.

For the 25 bar truss with 35% damage at element 7 and 40% damage at the element number 9, the results are reported in Table 7. The best performer based on mean values is IPB-DE while the second and third best are JADE and DE respectively. When examining the number of successful runs, only IPB-DE can detect the damage in the structure for all 30 runs. For this case, IPB-DE is said to be the most efficient optimiser, which obtained the minimum objective function mean value and successfully detected the damage in the structure for all optimisation runs with the average number of function evaluations being 3735.

6.2. Seventy-two-bar truss

For the 72-bar truss with 15% damage at element 5, the results are reported in Table 8. The best performer based on the mean objective function values is IPB-DE, while the second and the third best are ES and ACOR. When looking at the number of successful runs (f reaching 1×10^{-3} or lower), the most efficient method is IPB-DE which can detect the damage of the structure 30 times from implementing it in 30 optimisation runs, while the average number of function evaluations for convergent results is only 3155.

For the 72 bar truss with 15% damage at element number 58 and 10% damage at element number 4, the results are given in Table 9. The best performer based on the mean of objective function values

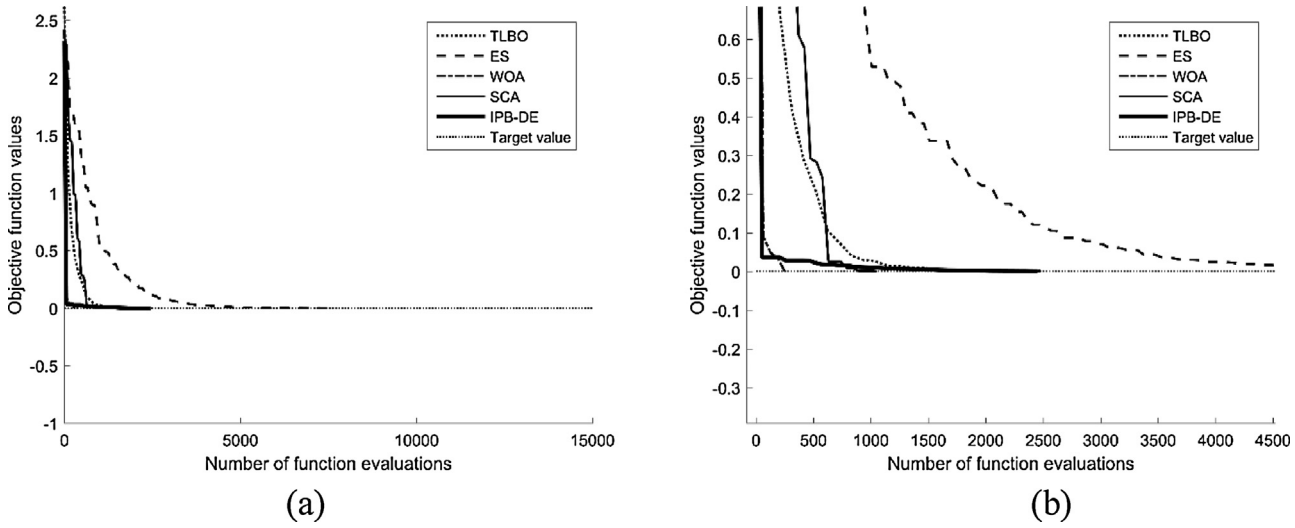


Fig. 6. Search history for the case, 72 bar truss Case I, (a) original, (b) zoom in.

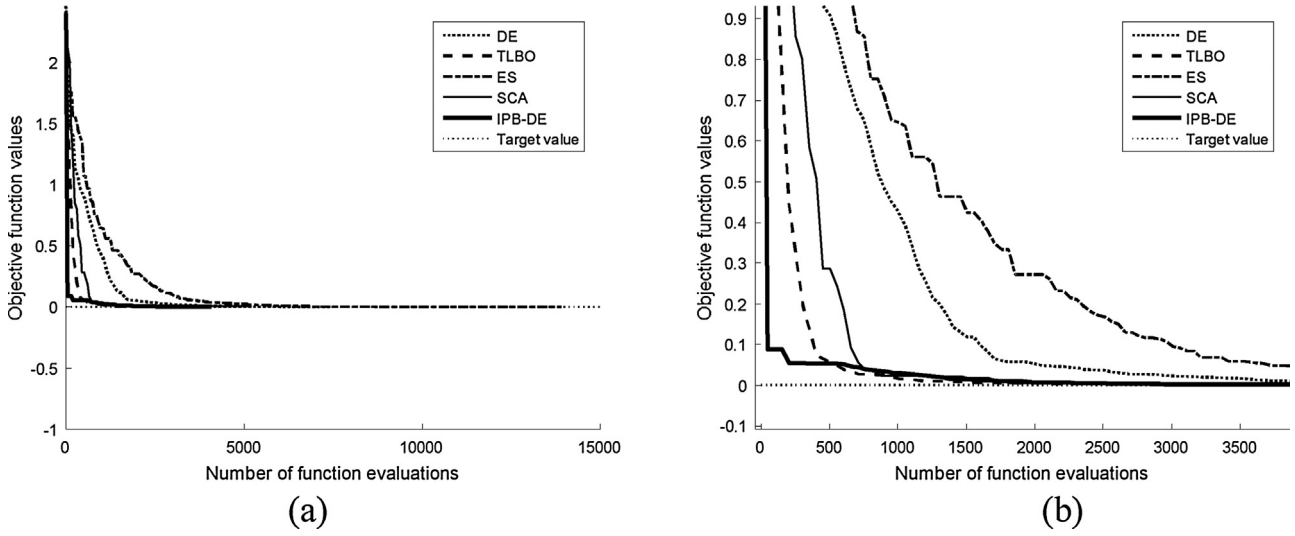


Fig. 7. Search history for the case, 72 bar truss Case II, (a) original, (b) zoom in.

is IPB-DE, while the second and third best are ES and JADE respectively. When considering the number of successful runs, the most efficient is IPB-DE, which can detect the damage of the structure 25 times from a total of 30 optimisation runs, while the average number of function evaluations for the convergence results is 6856.

Overall, it is clearly indicated from the results that integrating RBF into the DE can improve the search performance of the optimiser in solving structural damage detection of truss structures in terms of both search convergence and consistency. Based on the most crucial indicators, the average number of successful runs and the average number of function evaluations, IPB-DE is unanimously the most powerful method.

Figs. 4–7 shows the search history of the top five best algorithms (sorted based on number of successful runs from 30 runs). For the 25 bar truss with 35% damage at element number 7, the proposed IPB-DE and WOA show a similar convergence curve while WOA is slightly faster than IPB-DE after 200 function evaluations. Similarly, for the case of the 72 bar truss with 15% damage at element number 55, the proposed IPB-DE and WOA show the best convergence curves at the beginning while WOA is faster than IPB-DE. The WOA can converge to the goal before 500 function evaluations for this case. For the 25 bar truss with 35% damage at element number 7 and 40% damage at element number 9, and the 72 bar truss with 15% damage at element number 58 and 10% damage at element number 4, the IPB-DE gives the best convergence curves since the beginning.

Tables 10–11 show a comparison of the damage locations of the simulated problems and the results obtained from the best run of IPB-DE. It was found that IPB-DE can correctly detect the damage locations for Case I of the twenty-five bar truss while for Case II of the twenty-five bar truss, the structure is simulated to have 35% and 40% damage at element 7 and element 9 respectively, while the result obtained from IPB-DE gives 34.39 and 39.83% damage at element 6 and element 8. For this case, it can be said that the results are accurate, as both groups can obtain the same set of natural frequencies as mentioned in Section 3. Similarly, for Case I of the seventy-two bar truss, the structure is simulated to have 15% damage at element number 55 while 15% damage at element 56, 57, or 58 gives the same values of $\boldsymbol{\omega}_{damage}$. Therefore, it can be concluded that the results are accurate for this case. For Case II of the seventy-two bar truss, IPB-DE found damage in many elements, while the resulting natural frequencies are similar to the values of $\boldsymbol{\omega}_{damage}$. This implies that using only natural frequencies as an objective function can possibly fail to identify the damage

Table 10
Comparison of the simulated solution and the best results obtained by IPB-DE for 25 bar truss.

% damage at element no.	Case I		Case II	
	Simulated damage (%)	Damage found by IPB-DE (%)	Simulated damage (%)	Damage found by IPB-DE (%)
1	0.00	0.06	0.00	0.00
2	0.00	0.83	0.00	0.74
3	0.00	0.00	0.00	0.02
4	0.00	0.00	0.00	0.35
5	0.00	0.05	0.00	0.02
6	0.00	0.44	0.00	[a]34.39
7	35.00	34.16	35.00	0.58
8	0.00	0.45	0.00	[a]39.83
9	0.00	0.00	40.00	0.00
10	0.00	0.00	0.00	0.00
11	0.00	0.02	0.00	0.00
12	0.00	0.10	0.00	0.00
13	0.00	0.00	0.00	0.00
14	0.00	0.00	0.00	0.00
15	0.00	0.01	0.00	0.00
16	0.00	0.00	0.00	0.00
17	0.00	0.00	0.00	0.00
18	0.00	0.00	0.00	0.00
19	0.00	0.00	0.00	0.00
20	0.00	0.01	0.00	0.00
21	0.00	0.00	0.00	0.00
22	0.00	0.00	0.00	0.00
23	0.00	0.00	0.00	0.00
24	0.00	0.00	0.00	0.00
25	0.00	0.00	0.00	0.00
ω_1	69.1393	69.139	68.5203	68.52002
ω_2	72.2006	72.200	71.3167	71.31654
ω_3	95.3372	95.337	94.5625	94.56267
ω_4	119.8852	119.886	119.6514	119.6496
ω_5	121.4774	121.477	121.4253	121.4256
ω_6	125.0130	125.011	125.0129	125.0121

[a] 35% damage in elements 6 and 40% damage in elements 8 will result in the same set of natural frequencies for the Case II as mentioned in Section 3.

Table 11
Comparison of the simulated solution and the best results obtained by IPB-DE for 72 bar truss.

% damage at element no.	Case I						Case II					
	Simulated damage (%)			Damage found by IPB-DE (%)			Simulated damage (%)			Damage found by IPB-DE (%)		
1, 26, 51	0,	0,	0	0,	0,	0.39	0,	0,	0	0,	0,	0,
2, 27, 52	0,	0,	0	0,	0,	0.36	0,	0,	0	[b]9.88,	0,	0.76,
3, 28, 53	0,	0,	0	0,	0,	0.03	0,	0,	0	0,	0.04,	0.51,
4, 29, 54	0,	0,	0	0,	0.01,	0.90	10.00,	0,	0	0.01,	0.05,	0.06,
5, 30, 55	0,	0,	15.00	0,	0.01,	0.02	0,	0,	0	0.01,	0,	0,
6, 31, 56	0,	0,	0	0.06,	0,	0.36	0,	0,	0	0.04,	0.16,	[a]9.08,
7, 32, 57	0,	0,	0	0.03,	0.06,	0	0,	0,	15.00	0.02,	0.45,	0,
8, 33, 58	0,	0,	0	0,	0,	[a]14.51	0,	0,	0	0,	0.73,	[a]6.58,
9, 34, 59	0,	0,	0	0.01,	0.01,	0.04	0,	0,	0	0.03,	0.02,	0.02,
10, 35, 60	0,	0,	0	0,	1.91,	0.01	0,	0,	0	0.01,	0,	0,
11, 36, 61	0,	0,	0	0.03,	0.87,	0	0,	0,	0	0,	0.69,	0,
12, 37, 62	0,	0,	0	0.01,	0.01,	0.07	0,	0,	0	0.04,	0,	0,
13, 38, 63	0,	0,	0	0.01,	0.00,	0.01	0,	0,	0	0.03,	0,	0,
14, 39, 64	0,	0,	0	0,	0.02,	0	0,	0,	0	0.13,	0,	0,
15, 40, 65	0,	0,	0	0.02,	0.03,	0.05	0,	0,	0	0,	0.06,	0.02,
16, 41, 66	0,	0,	0	0.14,	0.01,	0.00	0,	0,	0	0.21,	0,	0,
17, 42, 67	0,	0,	0	0.73,	0.00,	0.17	0,	0,	0	3.04,	0.02,	0.09,
18, 43, 68	0,	0,	0	0.27,	0,	0	0,	0,	0	0.55,	0.05,	1.79,
19, 44, 69	0,	0,	0	0,	0.01,	0	0,	0,	0	0,	0.06,	0.12,
20, 45, 70	0,	0,	0	0,	0,	0.62	0,	0,	0	0,	0,	0,
21, 46, 71	0,	0,	0	0,	0.01,	0.04	0,	0,	0	0,	0,	0.46,
22, 47, 72	0,	0,	0	0,	0,	0.01	0,	0,	0	0.01,	0.02,	0,
23, 48	0,	0		0.03,	0.01		0,	0,		0.03,	0.01,	
24, 49	0,	0		0.06,	0.54		0,	0,		0.05,	0.07,	
25, 50	0,	0		0,	0.38		0,	0,		0.05,	0.15,	
ω_1	5.9553			5.9562			5.9530			5.9534		
ω_2	6.0455			6.0451			6.0455			6.0451		
ω_3	10.4764			10.4757			10.4764			10.4755		
ω_4	18.1448			18.1443			18.0921			18.0904		
ω_5	25.4903			25.4892			25.2437			25.2436		
ω_6	25.4939			25.4929			25.4939			25.4927		

[a] 15% damage in elements 55, 56, 57 or 58 will result in the same set of natural frequencies.
[b] 10% damage in elements 1, 2, 3 or 4 will result in the same set of natural frequencies.

locations for the cases of symmetric structures. The proposed algorithm is obviously effective and efficient but more reliable objective functions for damage localisation such as the use of both natural frequencies and mode shapes should be invented.

7. Conclusions

Hybridisation of RBF into DE leading to IPB-DE is presented for truss structural damage detection problems. Four structural damage detection test problems from two different truss structures are used to examine the search performance of the proposed approach. Several well established MHs and the proposed algorithms are then employed to solve the test problems. Numerical results reveal that the proposed hybrid algorithms of DE with RBF are the top performers for all test problems. Integrating RBF into the DE obviously improves DE performance. The proposed idea has the potential to be further applied to other inverse problems such as robot inverse kinematic analysis. Further improvement for meta-heuristic based structural health monitoring should be the purpose of a more reliable objective function rather than solely using the set of lowest natural frequencies. Detection of joint damage is another issue that will be focused on in future work.

Acknowledgement

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

References

[1] J.-J. Sinou, A review of damage detection and health monitoring of mechanical systems from changes in the measurement of linear and non-linear vibrations, in: C.S. Robert (Ed.), Mechanical Vibrations: Measurement, Effects and Control, Nova Science Publishers, Inc., 2009, 2018, pp. 643–702.
[2] A. Ghods, H.-H. Lee, Probabilistic frequency-domain discrete wavelet transform for better detection of bearing faults in induction motors, Neurocomputing 188 (2016) 206–216.
[3] Z.D. Zheng, Z.R. Lu, W.H. Chen, J.K. Liu, Structural damage identification based on power spectral density sensitivity analysis of dynamic responses, Comput. Struct. 146 (2015) 176–184.
[4] M. Rajendra, K. Shankar, Improved complex-valued radial basis function (ICRBF) neural networks on multiple crack identification, Appl. Soft Comput. 28 (2015) 285–300.
[5] A. Labib, D. Kennedy, C.A. Featherston, Crack localisation in frames using natural frequency degradations, Comput. Struct. 157 (2015) 51–59.
[6] A. Majumdar, D.K. Maiti, D. Maity, Damage assessment of truss structures from changes in natural frequencies using ant colony optimization, Appl. Math. Comput. 218 (2012) 9759–9772.
[7] Ł. Jedliński, J. Jonak, Early fault detection in gearboxes based on support vector machines and multilayer perceptron with a continuous wavelet transform, Appl. Soft Comput. 30 (2015) 636–641.
[8] M.A. de Oliveira, D.J. Inman, Performance analysis of simplified Fuzzy ARTMAP and Probabilistic Neural Networks for identifying structural damage growth, Appl. Soft Comput. 52 (2017) 53–63.
[9] N. Pholdee, S. Bureerat, Structural health monitoring through meta-heuristics –comparative performance study, Adv. Comput. Des.: Int. J. 1 (2016) 315–327.
[10] Z.H. Ding, M. Huang, Z.R. Lu, Structural damage detection using artificial bee colony algorithm with hybrid search strategy, Swarm Evol. Comput. 28 (2016) 1–13.
[11] A. Kaveh, A. Zolghadr, An improved CSS for damage detection of truss structures using changes in natural frequencies and mode shapes, Adv. Eng. Software 80 (2015) 93–100.
[12] M. Mehrjoo, N. Khaji, M. Ghafory-Ashtiany, Application of genetic algorithm in crack detection of beam-like structures using a new cracked Euler–Bernoulli beam element, Appl. Soft Comput. 13 (2013) 867–880.
[13] A. Mortazavi, V. Toğan, Sizing and layout design of truss structures under dynamic and static constraints with an integrated particle swarm optimization algorithm, Appl. Soft Comput. 51 (2017) 239–252.
[14] H. Zhao, M. Zhao, C. Zhu, Reliability-based optimization of geotechnical engineering using the artificial bee colony algorithm, KSCE J. Civ. Eng. 20 (2016) 1728–1736.
[15] A. Kaveh, B. Mirzaei, A. Jafarvand, An improved magnetic charged system search for optimization of truss structures with continuous and discrete variables, Appl. Soft Comput. 28 (2015) 400–410.
[16] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[17] S. Mirjalili, SCA.;1;: a sine cosine algorithm for solving optimization problems, Knowledge-Based Syst. 96 (2016) 120–133.
[18] S. Mirjalili, Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm, Knowledge-Based Syst. 89 (2015) 228–249.
[19] F. Sardari, M. Ebrahimi Moghaddam, A hybrid occlusion free object tracking method using particle filter and modified galaxy based search meta-heuristic algorithm, Appl. Soft Comput. 50 (2017) 280–299.
[20] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, evolutionary computation, IEEE Trans. 13 (2009) 945–958.
[21] N. Hansen, S.D. Muller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Evol. Comput. 11 (2003) 1–18.
[22] N. Pholdee, S. Bureerat, H.M. Baek, Y.-T. Im, Two-stage surrogate assisted differential evolution for optimization of a non-circular drawing sequence, Int. J. Precis. Eng. Manuf. 18 (2017) 567–573.
[23] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[24] K. Wansaseub, N. Pholdee, S. Bureerat, Optimal U-shaped baffle square-duct heat exchanger through surrogate-assisted self-adaptive differential evolution with neighbourhood search and weighted exploitation-exploration, Appl. Therm. Eng. 118 (2017) 455–463.
[25] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (2007) 459–471.
[26] K. Socha, M. Dorigo, Ant colony optimization for continuous domains, Eur. J. Oper. Res. 185 (2008) 1155–1173.
[27] A. Kaveh, S. Talatahari, A novel heuristic optimization method: charged system search, Acta Mech. 213 (2010) 267–289.
[28] A. Husseinzadeh Kashan, An efficient algorithm for constrained global optimization and application to mechanical engineering design: league championship algorithm (LCA), Comput.-Aided Des. 43 (2011) 1769–1792.
[29] S. Bureerat, J. Limtragool, Structural topology optimisation using simulated annealing with multiresolution design variables, Finite Elem. Anal. Des. 44 (2008) 738–747.
[30] G. Venter, J. Sobieszczanski-Sobieski, Particle swarm optimization, AIAA J. 41 (2003) 1583–1589.
[31] T. Back, Evolutionary Algorithms in Theory and Practice, Oxford University Press, Oxford, 1996.
[32] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput.-Aided Des. 43 (2011) 303–315.

Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 6043109, 15 pages
https://doi.org/10.1155/2017/6043109

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

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:

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

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 rth row). The rth element of a generated binary solution is set to be the rth 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 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 rth element of a generated binary solution is set equal to the rth 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 rth 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) with 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 $i = 1, \ldots, n_P$
Main procedure
(1) Set $\mathbf{B} = \{\}$.
(2) For $i = 1$ to nP
(3) Set $\mathbf{a} = \{\}$ a vector used to contain elements of a generated binary string.
(4) Randomly select a position (rth row) of $\mathbf{P}$.
(5) Set $a_r = b_{\text{REF},r}$. % Set the rth element of $\mathbf{a}$ as the rth element of $\mathbf{b}_{\text{REF}}$.
(6) For $j = \{1, 2, \ldots, m\} - \{r\}$
(7) If 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

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

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 ith and jth elements of $\mathbf{b}_{\text{best}}$, respectively. From the updating equation, if the ith and jth 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

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

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 rth 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

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

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 $\{\mathbf{b}^i\} \cup \{\mathbf{b}^i_{\text{new}}\}$ (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 $\mathbf{b}^i_{\text{new}}$ from $\mathbf{P}$ using Algorithm 1, and decode them to be $\mathbf{x}^i_{\text{new}}$.
(4) For $i = 1$ to n_P
(5) Calculate objective function values $f^i_{\text{new}} = \text{fun}(\mathbf{x}^i_{\text{new}})$.
(6) If $f^i_{\text{new}} < f_i$
(7) $f_i = f^i_{\text{new}}$, $\mathbf{b}^i = \mathbf{b}^i_{\text{new}}$, $\mathbf{x}^i = \mathbf{x}^i_{\text{new}}$
(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: $\mathbf{P}'$
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]:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{5}$$

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, $\mathbf{u} = \begin{Bmatrix} \delta_a \\ \delta_r \end{Bmatrix}$ 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

$$\mathbf{u} = \mathbf{C}\mathbf{u}_p + \mathbf{K}\mathbf{x}, \tag{6}$$

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]:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \\ k_5 & 1 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} k_6 & k_1 & k_2 & 0 \\ k_7 & k_3 & k_4 & 0 \end{bmatrix}, \tag{7}$$

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

$$\dot{\mathbf{x}} = (\mathbf{A} + \mathbf{B}\mathbf{K})\,\mathbf{x} + \mathbf{B}\mathbf{C}\mathbf{u}_p. \tag{8}$$

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

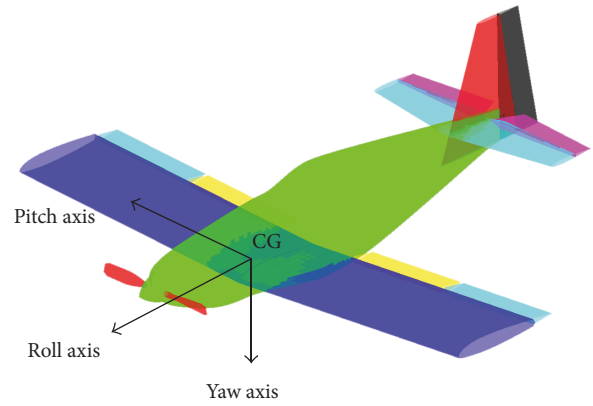


Figure 1: Stability axes of an aircraft.

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

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

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 $\mathbf{A} + \mathbf{B}\mathbf{K}$. 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

$$\mathbf{A} = \begin{bmatrix} -0.2842 & -0.9879 & 0.1547 & 0.0204 \\ 10.8574 & -0.5504 & -0.2896 & 0 \\ -199.8942 & -0.4840 & -1.6025 & 0 \\ 0 & 0.1566 & 1 & 0 \end{bmatrix},$$
$$\mathbf{B} = \begin{bmatrix} 0 & 0.0524 \\ 0.4198 & -12.7393 \\ 50.5756 & 21.6753 \\ 0 & 0 \end{bmatrix}. \quad (10)$$

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, $f1$ and $f2$, is EDACE while the second best is BPSO. For the simple multimodal functions, the best performer for $f4$ and $f5$ is SabDE while the best performer for $f3$ is BPSO. The second best performers for $f3$, $f4$, and $f5$ are SabDE, BEDACE, and UMDE, respectively. For the hybrid functions, the best performers for the functions $f6$, $f7$, and $f8$, are SabDE, EDACE, and BPSO, respectively, while the second best performer for $f6$ and $f7$ is BPSO and the second best for $f8$ is EDACE. For the final group of CEC2015 test problems, composition functions, the best performer for the $f11$, $f12$, and $f14$ is SabDE while the best performers for the $f10$ and $f15$ are BPSO and EDACE, respectively. For $f9$, the best performers are UMDA, BPSO, GA, PBIL, and EDACE, which obtain the same mean values while, for $f13$, 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, $f1$ and $f2$, are BPSO and EDACE while the second most consistent methods are EDACE and BPSO, respectively. For the simple multimodal functions, the best for $f3$ and $f5$ is SabDE while the best for $f4$ is the proposed EDACE. EDACE is the best for the hybrid function of $f7$ while BPSO is the best for the hybrid functions $f6$ and $f8$. For the composition functions, EDACE is the best for the problems $f9$ and $f12$ while BPSO is the best for $f10$. For the composition functions, $f11$, $f14$, and $f15$, the best is SabDE while the best for $f13$ is BSA.

Table 2: Objective values obtained.

CEC2015		MHs	UMDA	BPSO	GA	PBIL	BSA	EDACE	*GBABC	*BQIGSA	*SabDE
Unimodal functions	$f1$	Mean	$7.415E+06$	$1.807E+06$	$5.508E+06$	$1.586E+07$	$4.365E+07$	**$1.692E+06$**	$2.729E+07$	$8.419E+07$	$3.093E+08$
		STD	$5.648E+06$	**$1.224E+06$**	$3.510E+06$	$1.226E+07$	$4.391E+07$	$2.297E+06$	$2.267E+07$	$7.354E+07$	$1.168E+08$
		Min.	$5.203E+05$	**$1.914E+05$**	$1.016E+06$	$2.688E+05$	$1.325E+06$	$2.454E+05$			
	$f2$	Mean	$1.728E+08$	$1.278E+08$	$2.415E+08$	$1.443E+08$	$1.018E+09$	**$7.802E+07$**	$2.864E+09$	$7.834E+09$	$2.541E+09$
		STD	$1.287E+08$	$1.236E+08$	$1.880E+08$	$1.371E+08$	$1.680E+09$	**$3.046E+07$**	$2.374E+09$	$6.527E+09$	$5.008E+09$
		Min	$4.359E+07$	$3.525E+07$	$6.713E+07$	$4.834E+07$	$1.133E+08$	**$3.277E+07$**			
Simple multimodal functions	$f3$	Mean	$3.203E+02$	**$3.197E+02$**	$3.203E+02$	$3.202E+02$	$3.202E+02$	$3.201E+02$	$3.202E+02$	$3.202E+02$	$3.200E+02$
		STD	$8.505E-02$	$1.900E+00$	$9.050E-02$	$7.945E-02$	$6.006E-02$	$3.300E-02$	$2.641E+02$	$2.641E+02$	**$2.044E-02$**
		Min	$3.201E+02$	**$3.107E+02$**	$3.201E+02$	$3.201E+02$	$3.201E+02$	$3.200E+02$			
	$f4$	Mean	$4.213E+02$	$4.220E+02$	$4.286E+02$	$4.278E+02$	$4.226E+02$	$4.182E+02$	$4.358E+02$	$4.389E+02$	**$4.116E+02$**
		STD	$4.647E+00$	$4.915E+00$	$8.553E+00$	$9.507E+00$	$7.150E+00$	**$4.173E+00$**	$3.599E+02$	$3.621E+02$	$7.606E+00$
		Min	**$4.105E+02$**	$4.124E+02$	$4.138E+02$	$4.123E+02$	**$4.105E+02$**	**$4.105E+02$**			
	$f5$	Mean	$1.010E+03$	$1.066E+03$	$1.339E+03$	$1.353E+03$	$1.275E+03$	$1.014E+03$	$1.108E+03$	$1.542E+03$	**$9.330E+02$**
		STD	$1.300E+02$	$1.352E+02$	$1.981E+02$	$2.279E+02$	$1.975E+02$	$1.500E+02$	$9.736E+02$	$1.275E+03$	**$9.464E+01$**
		Min	$7.791E+02$	$8.526E+02$	$1.049E+03$	$8.120E+02$	$9.628E+02$	**$6.907E+02$**			
Hybrid functions	$f6$	Mean	$1.951E+05$	$7.345E+04$	$2.288E+05$	$4.894E+05$	$6.403E+06$	$1.133E+05$	$7.442E+06$	$5.582E+05$	**$4.625E+04$**
		STD	$1.120E+05$	**$3.958E+04$**	$1.813E+05$	$3.224E+05$	$8.635E+06$	$8.936E+04$	$1.321E+07$	$6.055E+05$	$4.076E+04$
		Min	$3.661E+04$	$3.661E+04$	$3.702E+04$	$8.124E+04$	$1.320E+05$	**$3.659E+04$**			
	$f7$	Mean	$7.047E+02$	$7.032E+02$	$7.044E+02$	$7.046E+02$	$7.118E+02$	**$7.030E+02$**	$7.589E+02$	$7.392E+02$	$7.752E+02$
		STD	$1.054E+00$	$6.183E-01$	$1.036E+00$	$1.113E+00$	$8.660E+00$	**$5.927E-01$**	$4.668E+02$	$4.324E+02$	$4.155E+03$
		Min	$7.027E+02$	$7.024E+02$	$7.027E+02$	$7.024E+02$	$7.025E+02$	**$7.021E+02$**			
	$f8$	Mean	$9.309E+04$	**$1.511E+04$**	$5.503E+04$	$4.808E+05$	$2.305E+06$	$2.727E+04$	$3.949E+07$	$2.948E+06$	$2.395E+07$
		STD	$1.120E+05$	**$6.918E+03$**	$5.630E+04$	$5.295E+05$	$2.400E+06$	$2.635E+04$	$2.442E+08$	$2.469E+06$	$5.432E+07$
		Min	$1.497E+04$	**$1.287E+04$**	**$1.287E+04$**	$1.312E+04$	$1.589E+04$	**$1.287E+04$**			
Composition functions	$f9$	Mean	**$1.001E+03$**	**$1.001E+03$**	**$1.001E+03$**	**$1.001E+03$**	$1.003E+03$	**$1.001E+03$**	$1.017E+03$	$1.048E+03$	$1.177E+03$
		STD	$2.090E-01$	$2.231E-01$	$9.437E-01$	$4.017E-01$	$4.284E+00$	**$1.700E-01$**	$8.397E+02$	$8.643E+02$	$4.102E+01$
		Min	$1.000E+03$	$1.000E+03$	$1.000E+03$	$1.001E+03$	$1.000E+03$	$1.000E+03$			
	$f10$	Mean	$1.285E+04$	**$3.930E+03$**	$1.367E+04$	$4.235E+04$	$5.317E+05$	$7.819E+03$	$9.909E+05$	$4.426E+04$	$2.416E+07$
		STD	$8.308E+03$	**$2.140E+03$**	$1.291E+04$	$3.728E+04$	$6.444E+05$	$4.897E+03$	$3.659E+06$	$4.477E+04$	$8.862E+07$
		Min	$3.199E+03$	$1.733E+03$	$1.738E+03$	$2.275E+03$	$1.805E+03$	**$1.731E+03$**			
	$f11$	Mean	$1.510E+03$	$1.232E+03$	$1.360E+03$	$1.396E+03$	$1.427E+03$	$1.240E+03$	$1.159E+03$	$1.172E+03$	**$1.114E+03$**
		STD	$8.727E+01$	$1.410E+02$	$1.129E+02$	$5.002E+01$	$4.384E+01$	$1.436E+02$	$9.557E+02$	$9.669E+02$	**$1.131E+01$**
		Min	$1.401E+03$	**$1.109E+03$**	$1.118E+03$	$1.132E+03$	$1.402E+03$	**$1.109E+03$**			
	$f12$	Mean	$1.304E+03$	$1.305E+03$	$1.306E+03$	$1.308E+03$	$1.308E+03$	$1.305E+03$	$1.264E+03$	$1.255E+03$	**$1.224E+03$**
		STD	$9.674E-01$	$1.125E+00$	$1.259E+00$	$2.595E+00$	$4.486E+00$	**$1.115E+00$**	$1.044E+03$	$1.035E+03$	$1.302E+00$
		Min	**$1.303E+03$**	**$1.303E+03$**	$1.304E+03$	$1.304E+03$	**$1.303E+03$**	**$1.303E+03$**			
	$f13$	Mean	**$1.300E+03$**	**$1.300E+03$**	**$1.300E+03$**	**$1.300E+03$**	**$1.300E+03$**	**$1.300E+03$**	$1.446E+03$	$1.452E+03$	$2.815E+09$
		STD	$8.140E-04$	$1.039E-03$	$1.248E-03$	$9.095E-04$	**$2.313E-13$**	$7.944E-04$	$1.193E+03$	$1.197E+03$	$4.158E+09$
		Min	$1.300E+03$	$1.300E+03$	$1.300E+03$	$1.300E+03$	$1.300E+03$	$1.300E+03$			
	$f14$	Mean	$9.364E+03$	$5.623E+03$	$7.023E+03$	$7.355E+03$	$8.736E+03$	$6.167E+03$	$2.162E+03$	$3.356E+03$	**$1.727E+03$**
		STD	$2.120E+03$	$2.144E+03$	$1.856E+03$	$2.732E+03$	$3.603E+03$	$2.199E+03$	$2.091E+03$	$2.869E+03$	**$4.411E+02$**
		Min	$4.817E+03$	$2.816E+03$	$4.425E+03$	$2.818E+03$	$4.453E+03$	**$2.401E+03$**			
	$f15$	Mean	$1.623E+03$	$1.616E+03$	$1.621E+03$	$1.618E+03$	$1.639E+03$	**$1.614E+03$**	$2.012E+03$	$1.530E+03$	$1.700E+03$
		STD	$3.445E+00$	$3.132E+00$	$5.108E+00$	$4.295E+00$	$3.614E+01$	$3.945E+00$	$1.659E+03$	$1.262E+03$	**$2.177E-05$**
		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
Sum of ranking	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

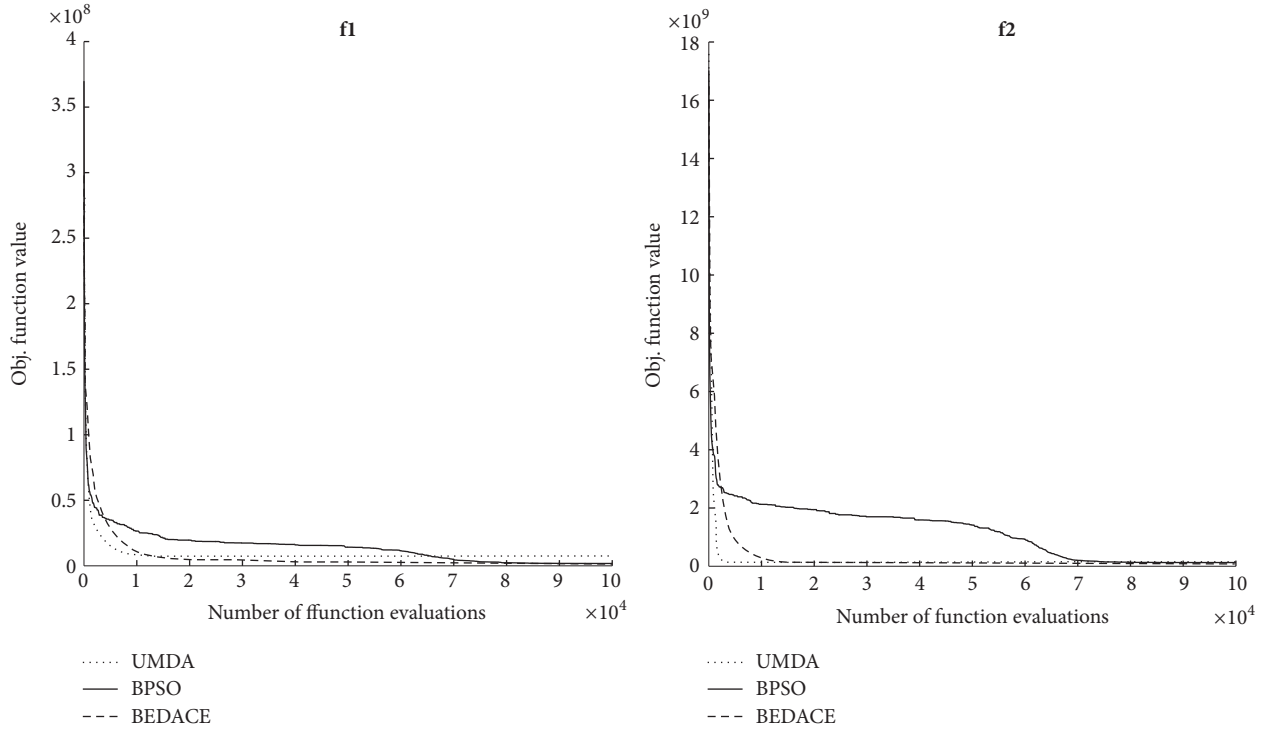


Figure 2: Search history of the top three best optimisers based on the t-test for the unimodal function.

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

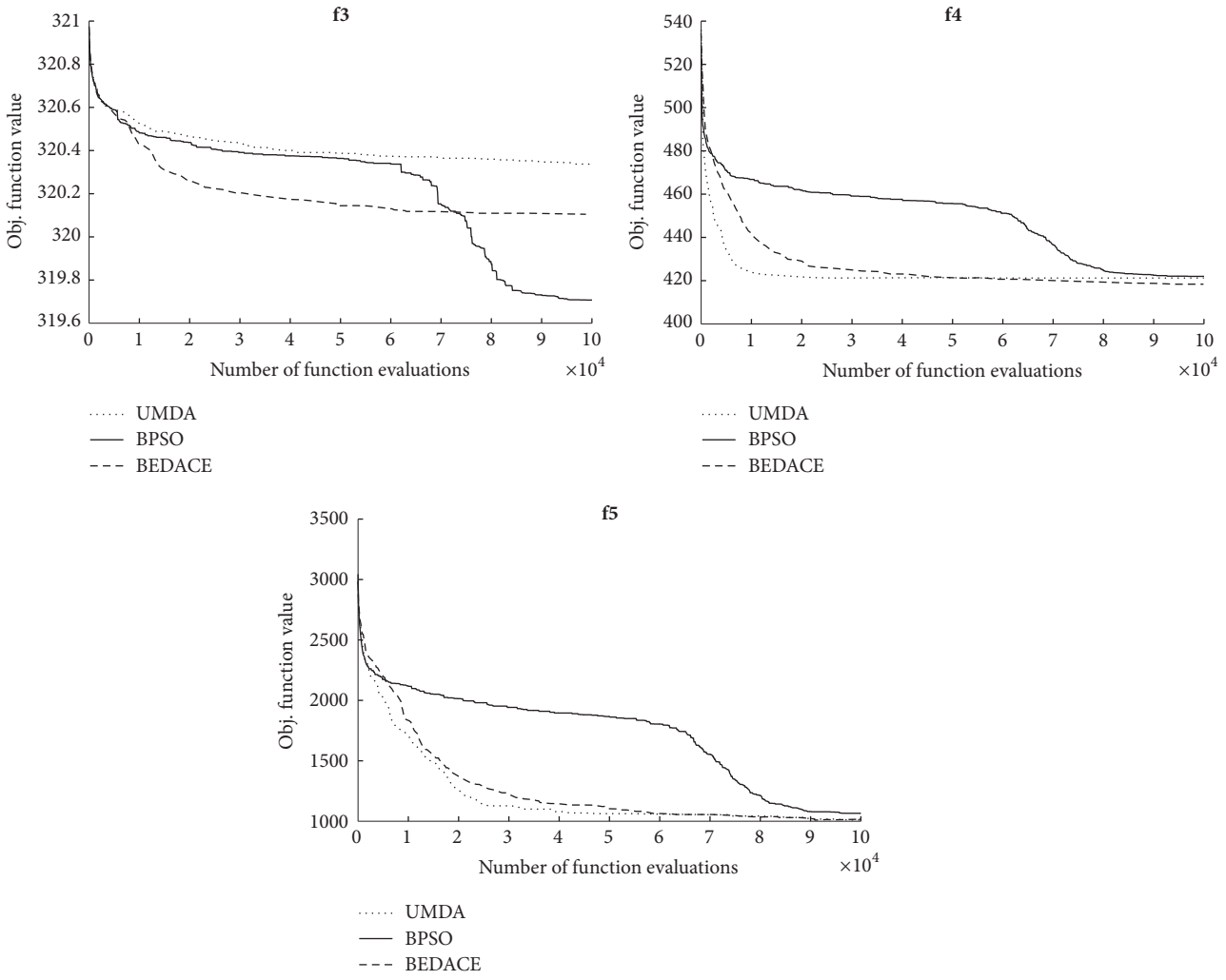


Figure 3: Search history of the top three best optimisers based on the t-test for the simple multimodal functions.

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

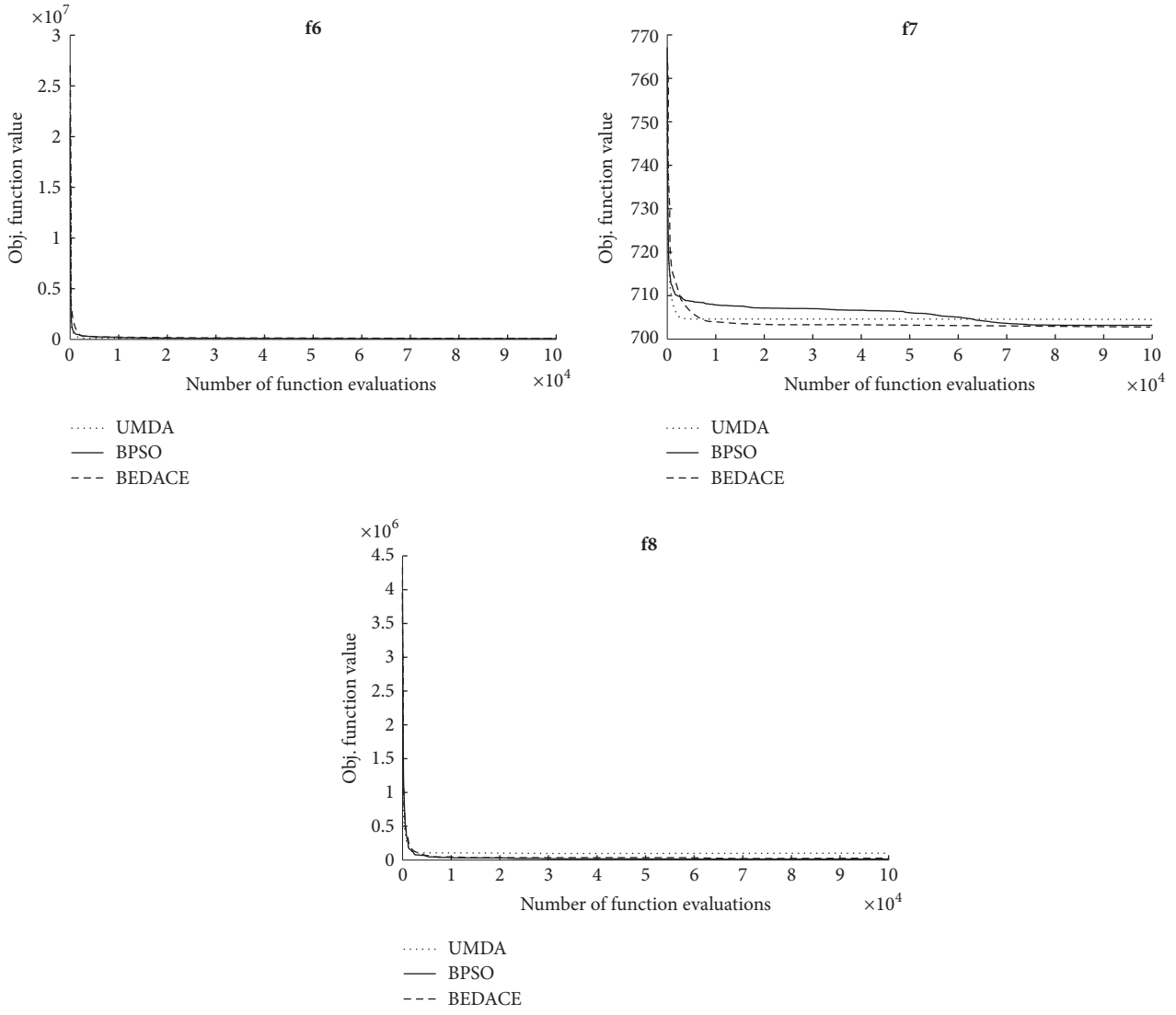


Figure 4: Search history of the top three best optimisers based on the t-test for the hybrid functions.

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

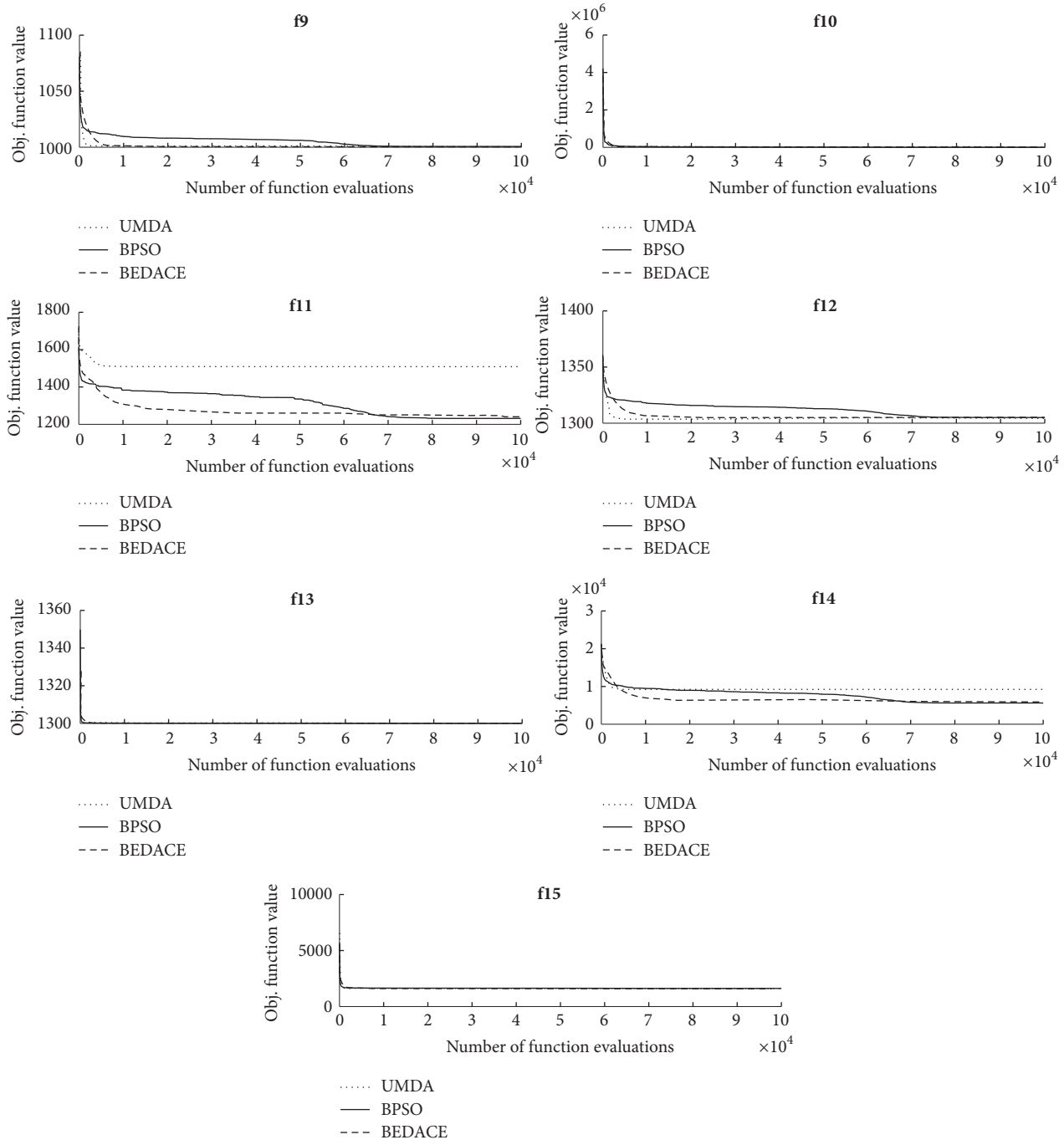


Figure 5: Search history of the top three best optimisers based on the t-test for the composition functions.

studies. The use of EDACE for hyperheuristic development is also possible. The extension to multiobjective optimisation and many-objective optimisation is also under investigation. Appling EDACE for the more complex problems such as large scale problems, mixed-variable problems, and reliability optimisation is for future work. The fight 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.

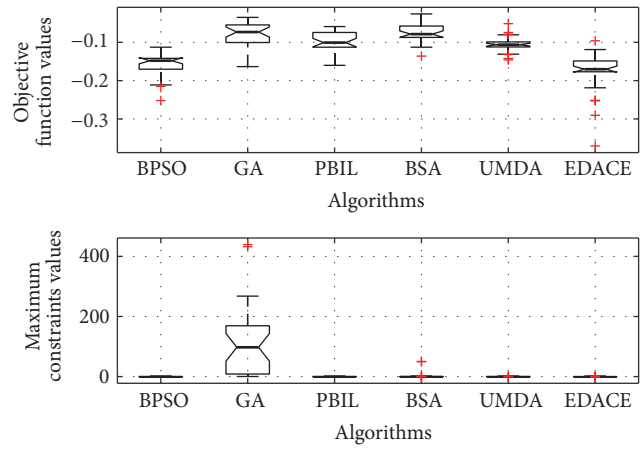


Figure 6: Box-plot of objective function values from 30 optimisation runs.

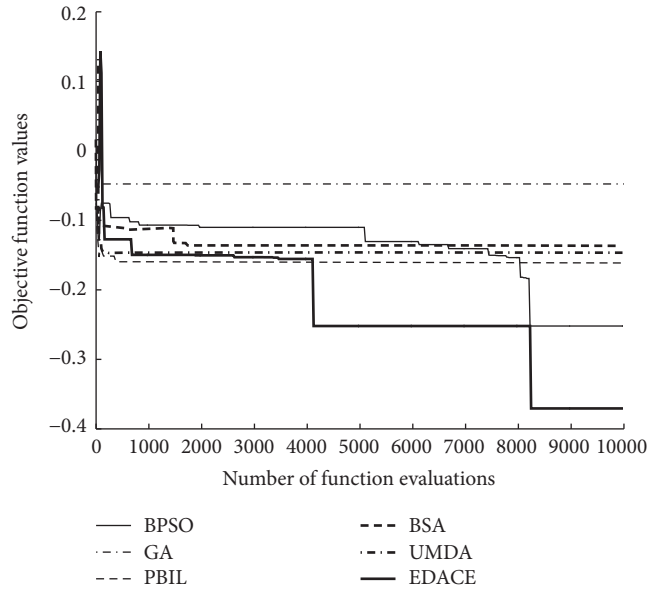


Figure 7: Search history of the best run of all optimisers.

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*, Carnagie 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. Nuaekaew, 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.

An Improved Teaching-Learning Based Optimization for Optimization of Flatness of a Strip During a Coiling Process

Sujin Bureerat[1], Nantiwat Pholdee[1(✉)], Won-Woong Park[2], and Dong-Kyu Kim[3]

[1] Department of Mechanical Engineering, Faculty of Engineering, Sustainable and Infrastructure Research and Development Centre, Khon Kaen University, 123 Moo 16, Mittraphap Road, Tambon Muang, Khon Kaen, Thailand
nantiwat@kku.ac.th
[2] National Research Laboratory for Computer Aided Materials Processing, Department of Mechanical Engineering, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon, South Korea
[3] Neutron Science Division, Korea Atomic Energy Research Institute, 111 Daeduk-daero 989 beon-gil, Yuseong-gu, Daejeon, South Korea

Abstract. Performance enhancement of a teaching-learning basedz optimizer (TLBO) for strip flatness optimization during a coiling process is proposed. The method is termed improved teaching-learning based optimization (ITLBO). The new algorithm is achieved by modifying the teaching phase of the original TLBO. The design problem is set to find spool geometry and coiling tension in order to minimize flatness defects during the coiling process. Having implemented the new optimizer with flatness optimization for strip coiling, the results reveal that the proposed method gives a better optimum solution compared to the present state-of-the-art methods.

Keywords: Evolutionary algorithm · Flatness defect · Optimization · Strip coiling · Teaching-learning based optimization

1 Introduction

There are several processing stages during the manufacturing of a coil strip, e.g. roughing, rolling, cooling, and coiling. Based on the previous investigation by Jung and Im [1, 2], the final strip shape had non-uniform thickness profiles consisting of ∩, ∪, M, and W shapes. Generally, it is difficult to predict the final shape of the strip due to various related processing parameters in production facilities. The strip crown, while being coiled, may include imperfections that were initiated during the rolling process resulting in flatness imperfection taking place on the coil strip [3, 4].

As a result, the strip is normally welded, cut, and recoiled in the recoiling line so as to satisfy customer strip flatness requirements. However, although adding the recoiling line to the process, flatness problems sometimes cannot be avoided especially for the high-strength coil strip. In order to understand the flatness defect formation mechanism

C. Sombattheera et al. (Eds.): MIWAI 2016, LNAI 10053, pp. 12–23, 2016.
DOI: 10.1007/978-3-319-49397-8_2

during the coiling process, Sims and Place [5] proposed a stress model of the coil assuming that the coil was an axial-symmetry hollow cylinder. Miller and Thornton [6] and Sarban [7] introduced a finite element method and a semi-analytical model to calculate the three-dimensional stress distribution within the coil. Nevertheless, in those models, they did not consider the physical clearance between each coiled wrap due to the strip crown as a cause of the axial inhomogeneity. Yanagi et al. [8] proposed an analytical model by wrapping the thick cylinder (the coil) with the thin-walled cylinders (the new coiling strips) to deal with inhomogeneous deformation of the cold-rolled thin-strip in the axial direction caused by the clearance and the strip crown. Moreover, Park et al. [9] studied the effect of processing parameters including a strip crown, a spool geometry, and coiling tension on the stress distribution on the strip during the coiling process where the analytical elastic model was used. In this study, it was found that enhancement of strip flatness of the cold-rolled thin-strip could be accomplished by suppressing the strip crown and lowering the coiling tension intensity compared to the measured circumferential strain distribution.

To alleviate the undesirable formation of flatness defects, manufacturing the strip coil without the strip crown is suggested as the best solution for fulfilling the strip flatness requirement. Nevertheless, suppressing the strip crown during the rolling process, as illustrated in Fig. 1, is somewhat difficult or even impossible to carry out due to many processing parameters involved. Therefore, use of optimization to find the optimum solution for a spool geometry and coiling tension was conducted [10, 11] in order to improve the strip flatness during the strip coiling process.

Optimization is a special kind of mathematical problem assigned to search for a design solution optimizing a predefined objective or merit indicator within a given feasible region. A numerical optimizer is usually employed to find such a solution. It can be categorized as an optimization method either with and without using function derivatives. The former is based on hard computing while the latter is based on a stochastic process and soft computing. The most popular non-gradient optimizer is an evolutionary algorithm (EAs) or later known as a meta-heuristic (MH). It has been implemented on a wide range of engineering applications and has shown several advantages [12–21]. For metal strip manufacturing, optimization by means of meta-heuristics has been used most commonly in the rolling process so as to control the flatness problem, whereas their use in the strip coiling process has been rarely reported [22–27].

In this study, optimization of flatness of the strips has been enhanced by an improved teaching-learning based algorithm (ITLBO). This method is compared to several well established EAs, such as simulated annealing (SA) [16], differential evolution (DE) [28], artificial bee colony optimization (ABC) [29], real code ant colony optimization (ACOR) [30], original teaching-learning based optimization (TLBO) [31], league championship algorithm (LCA) [32], charged system search (ChSS) [33], Opposition-based Differential Evolution Algorithm (OPDE) [10] and Enhanced teaching-learning based optimization with differential evolution (ETLBO-DE) [11] to determine the spool geometry and coiling tension where the objective is to minimize

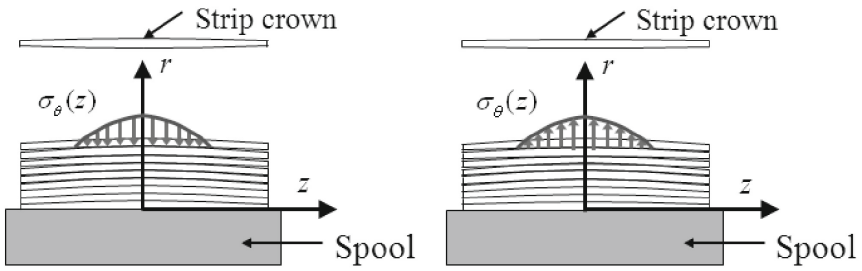
the axial inhomogeneity of the stress to improve the flatness of the strip. For function evaluations, the analytical elastic model proposed by Park et al. [9] similar to the one suggested by Yanagi et al. [8] was employed.

2 Formulation of the Optimization Design Problem

It is known that wavy edges occur during the strip coiling process, when the circumferential stress at the middle zone of the strip is highly compressed, while two edges are under tension or slight compression. Also, if the middle strip zone is under high tension while the two edges are compressed or slightly stretched, center buckle can happen [8, 9]. Figures 1(a) and (b) display the circumferential stress (σ_θ) distribution along the z direction within the thin strip, which respectively caused the wavy edge and center buckle.

Generally, it is impossible to obtain a flat strip after finishing a rolling process. The strip always has a crown shape. When the strips are being coiled, tension loads need to be applied, the middle zone (z = 0) of the strip at the inner coil will be considerably compressed in comparison with the two edges because of the coiling tension and the strip crown. In such a situation, the center buckle defect at the inner coil will not appear but the wavy edge defect can possibly occur. As such, the wavy edge defect at the inner coil is the major problem during the coiling process. Figure 2 depicts the circumferential stress (σ_θ) distribution in the z direction at the radius (r) of the coil (computed by the Love's elastic solution proposed by Park et al. [9]) contributing to wavy edge defect formation during the strip coiling process. It is possible to reduce the wavy edge defect by decreasing the axial inhomogeneity of the stress distribution and the maximum compressive stress at the compressive zone [10].

In this paper, optimization using the ITLBO and other well-known and newly developed EAs will be used to find the optimum solution for the processing parameters including coiling tension (σ_T) and spool geometry, as illustrated in Fig. 3.



(a) the wavy edge (b) center buckle

Fig. 1. Circumferential stress distributions for (a) the wavy edge and (b) center buckle, respectively [8, 9]

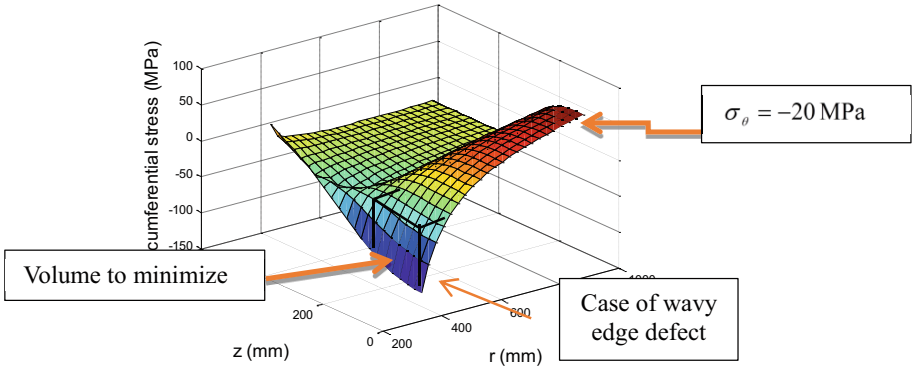


Fig. 2. Circumferential stress distribution (σ_θ) in the coil determined by Love's elastic solution [9]

To decrease the axial inhomogeneity of the stress distribution and the maximum compressive stress, minimization of the volume of the circumferential stress and maximum compressive stress (shown in Fig. 2) is defined as an objective function. In Fig. 2, the volume can only be computed for the coil, where compressive stresses were higher than 20 MPa, in order to minimize the zone that is likely to have the wavy edge defect. The objective function of the optimization problem can then be written as:

$$\text{Minimize} \qquad f\left(\alpha_b, \eta_b, \sigma_{T,i}\right) = \frac{V}{V_0} + \frac{\max(\sigma_{\theta c})}{\max(\sigma_{\theta c0})} \tag{1}$$

minimize

$$\begin{aligned} &0 \le \alpha_b \le 4, \\ &0 \le \eta_b \le 4, \\ &25 \le \sigma_{T,i} \le 50 \text{ MPa}; \qquad i = 1, \ldots n_{\max} \\ &\left|\sigma_{T,i} - \sigma_{T,i-1}\right| \le 2 \text{ MPa}, \end{aligned}$$

where $\sigma_{\theta c}$ and V are respectively the compressive circumferential stress higher than 20 MPa (refer to Fig. 2) and the approximate volume of the circumferential stress. $\sigma_{\theta c0}$ and V_0 are the respective values for the original design of the process. The $\sigma_{T,i}$ is the coiling tension at coil number i. The coiling tension is normally set to be constant for all coils [34]. The variable $n_{\max}$ is the maximum number of coils, which has been assigned to be 220 in this paper. η_b and α_b in Eq. (2) are spool crown exponent and the spool crown height, which were used for defining the spool geometry, as described in Fig. 3:

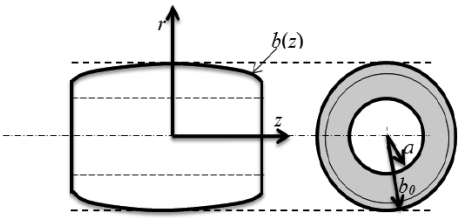


Fig. 3. Spool Geometry used in the present investigation

$$b(z) = b_0 - \alpha_b \left(\frac{|z|}{z_{\max}} \right)^{\eta_b} \tag{2}$$

where b_0 (z = 330 mm) and $b(z)$ are the initial value of the outer radius of the spool and the outer radius of the spool along the z direction, respectively. $z_{\max}$ = 525 mm is the width of the spool. The inner radius of the spool (a) in Fig. 3 has been assigned to be 300 mm. The total number of design variables, therefore, is 222 (220 for coiling tensions and 2 for the spool geometry).

3 Improved Teaching-Learning Based Optimization

From the previous section, the optimization problem can be considered being large-scale. It has been found [10, 11], that TLBO is suitable for this type of design problem. The teaching-learning based optimization (TLBO) algorithm is an evolutionary algorithm, or an optimizer without using function derivatives, proposed by Rao et al. [31]. The concept of TLBO searching mechanism is based on mimicking a teacher on the output of learners in a classroom. Basically, the learners can improve their intellectual and knowledge by two stages i.e. learning directly from the teacher and learning among themselves. During the teacher stage, a teacher may teach the learners, however, only some learners can acquire all things presented by the teacher. Those who can accept what the teacher taught will improve their knowledge. For the second stage, which is called the learning phase, the learners can improve their knowledge during discussion with other learners. Based on the different levels of the learners' knowledge, the better learners may transfer knowledge to the inferior learners.

From the view point of optimization, the algorithm starts with a randomly created initial population, which is a group of design solutions. Learners are identical to design solutions whereas the best one is considered a teacher. The objective function is analogous to the knowledge which needs to be improved towards the optimum solution. Having identified a teacher and other learners for the current iteration, the population will be updated by two stages including "Teacher Phase" and "Learner Phase". In the "Teacher Phase", an individual ($\mathbf{x}_i$) will be updated based on the best individual ($\mathbf{x}_{\text{teacher}}$) and the mean values of all populations ($\mathbf{x}_{\text{mean}}$) as follows:

$$\mathbf{x}_{\text{new},i} = \mathbf{x}_{\text{old},i} + r\{\mathbf{x}_{\text{teacher}} - (T_F \cdot \mathbf{x}_{\text{mean}})\} \tag{3}$$

Where T_F is a teaching factor, which can be either 1 or 2 and $r \in [0,1]$ is a uniform random number.

For the "Learner Phase", the members in the current population will be modified by exchanging information between themselves. Two individuals $\mathbf{x}_i$ and $\mathbf{x}_j$ will be chosen at random, where $i \neq j$. The update of the solutions can then be calculated as:

$$\mathbf{x}_{\text{new},i} = \begin{cases} \mathbf{x}_{\text{old},i} + r\left(\mathbf{x}_i - \mathbf{x}_j\right) & \text{if } f(\mathbf{x}_i) < f\left(\mathbf{x}_j\right) \\ \mathbf{x}_{\text{old},i} + r\left(\mathbf{x}_j - \mathbf{x}_i\right) & \text{if } f\left(\mathbf{x}_j\right) < f(\mathbf{x}_i) \end{cases} \tag{4}$$

At both teacher and learner phases, the new solution ($\mathbf{x}_{\text{new}}$) will replace its parent if it has better knowledge or produces better objective function value, otherwise, it will be rejected. The two phases are sequentially operated until the termination criterion is fulfilled.

For the improved teaching-learning based optimization (ITLBO), an opposition-based approach, binary crossover, and the probability of operating the learning phase are added to the original TLBO to improve the balance of search exploration and exploitation. Four random numbers including, $rand_1$, $rand_2$, $rand_3$, and $rand_4$, have been used for performing opposition-based approach, binary crossover, and the learning phase. The main search procedure starts by generating an initial population, updating the population at the teaching phase and learning phase similarly to the original TLBO. However, at the teaching phase, the updating can be done by the following equation;

$$\mathbf{x}_{\text{new},i} = \mathbf{x}_{\text{old},i} + (-1)^{rand_1} r\{\mathbf{x}_{\text{teacher}} - (T_F \cdot \mathbf{x}_{\text{mean}})\} \tag{5}$$

where $rand_1$ is a random value with either 0 or 1. Then, the binary crossover is applied if a uniform random number having an interval of 0 and 1 ($rand_2$) is lower than the crossover probability (P_r). For a new individual $\mathbf{x}_{\text{new}}^T = [x_{\text{new},1}, \ldots, x_{\text{new},D}]$ and an old individual $\mathbf{x}_{\text{old}}^T = [x_{\text{old},1}, \ldots, x_{\text{old},D}]$, the binary crossover step can be expressed as follow;

$$x_{new,j} \begin{cases} x_{old,j} & \text{if } rand_3 < CR_1 \quad j = 1, \ldots, D \\ x_{teacher,j} & \text{if } CR_1 \leq rand_3 < CR_2 \quad j = 1, \ldots, D \end{cases} \tag{6}$$

where the $rand_3$ is a uniform random number generated from 0 to 1. The CR_1 and CR_2 are the predefined crossover rates, while D is the number of design variables, respectively. Thereafter, the learning phase is conducted if a uniform random number generated from 0 to 1 ($rand_4$) is lower than the probability value (L_p), otherwise, the learning phase will be skipped. The search process will be repeated until the termination criterion is satisfied. The computational steps of the proposed algorithm are shown in Algorithm 1.

Algorithm 1 An improved TLBO

Input: Maximum iteration number ($maxiter$), population size (n_P), Crossover probability Crossover rate (CR_1 and CR_2), learning phase probability (L_p).
Output: $\mathbf{x}_{best}$, f_{best}
Initialization
1. Generate an initial population randomly.
2. Evaluate objective function values
Main algorithm
3. For i =1 to $maxiter$
  3.1 Identify the best solution ($\mathbf{x}_{teacher}$)
  (Teacher Phase)
    For j=1 to n_p
    3.2 Update the population using equation (5)
      If $rand_2$ < P_r
        3.2.1 Applied binary crossover using equation (6)
      End
      3.2.1 Evaluate the objective function value $f(\mathbf{x}_{new,j})$
      3.2.2 If $f(\mathbf{x}_{new,j}) < f(\mathbf{x}_{old,j})$
          Replace $\mathbf{x}_{old,j}$ by $\mathbf{x}_{new,j}$
      End
    End

    If $rand_4$ < L_p
    (Learner Phase)
    For j=1 to n_p
    3.3 Update the population using equation (4)
      3.3.1 Evaluate the objective function value $f(\mathbf{x}_{new,j})$
      3.2.2 If $f(\mathbf{x}_{new,j}) < f(\mathbf{x}_{old,j})$
          Replace $\mathbf{x}_{old,j}$ by $\mathbf{x}_{new,j}$
      End
    End
End

4 Numerical Experiments

In order to examine the search performance of the proposed ITLBO, several EAs have been used to solve the optimum design problem of the strip flatness as described in the previous section. The EAs used in this study are as follows:

- DE [28]: The DE/best/2/bin strategy was used. DE scaling factor was random from 0.25 to 0.7 in each calculation and crossover probability was 0.7.
- SA [16]: An annealing temperature was reduced exponentially by 10 times from the value of 10 to 0.001 in the optimization searching process. On each loop 2n children were created by means of mutation to be compared with their parent. Here, n is the number of design variables.
- ABC [29]: The number of food sources was set to be $3n_p$. A trial counter to discard a food source was 100.
- ACOR [30]: The parameters used for computing the weighting factor and the standard deviation in the algorithm were set to be $\xi = 1.0$ and $q = 0.2$, respectively.
- TLBO [31]: Parameter settings are not required.
- LCA [32]: The default parameter settings provided by the authors were used.
- ChSS [33]: The number of solutions in the charge memory was $0.2n_p$. Here, n_p is the population size. The charged moving considering rate and the parameter PAR were set to be 0.75 and 0.5, respectively.
- OPDE [10]: The DE/best/2/bin strategy was used. DE scaling factor was random from 0.25 to 0.5 in each calculation and crossover probability used was 0.7.
- ETLBO-DE [11]: Used the DE parameter setting and Latin hypercube sampling (LHS) technique to generate an initial population.
- ITLBO (Algorithm 1): The P_r, CR_1, CR_2 and L_p were set to be 0.5, 0.33, 0.66 and 0.75, respectively.

Each optimizer was employed to solve the problem for 5 optimization runs. Both the maximum number of iterations and population size were set to be 100. For the optimizers using different population sizes, such as simulated annealing, their search processes were stopped with the total number of function evaluations as 100×100. The optimal results of the various optimizers from using this limited number of function evaluations were compared. The best optimizer was used to find the optimal processing parameters of the strip coiling process.

5 Results and Discussion

After applying each optimization algorithm to solve the problem for 5 runs, the results are given in Table 1. The mean values (Mean) are used to measure the convergence rate while the standard deviation (STD) determines search consistency. The lower the mean objective function value the better, and the lower the standard deviation the more consistent. In the table, max and min stand for the maximum and minimum values of the objective function, respectively.

For the measure of convergence speed based on the mean objective value, the best method is ITLBO while the second best and the third best performers are ETLBO-DE and OPDE, respectively. The worst results came from ABC. For the measure of search consistency based on STD, the best was also ITLBO while the worst was ABC, which was similar to the measure of the search convergence. The second best and the third best for consistency were ETLBO-DE and ACOR, respectively. The minimum objective function value was obtained by the ITLBO.

Based on the results obtained, it was clearly indicated that the proposed ITLBO by adding opposition based method, binary crossover, and learning phase probability can improve the search performance of the original TLBO for solving the optimization design problem of the strip coiling process.

The optimal spool crown exponent and height obtained are 1.0822 and 2.3645, respectively. The optimal distribution of coiling tensions as a function of coil numbers is shown in Fig. 4. The results reveal that the coiling tensions start with the highest value initially and then decrease when the number of coils increases. After a few series of coiling, the tension levels become almost constant, converging to the lower bound at the end of the process. Figure 5 shows the plot of the circumferential stress distributions along the z and r directions of the original and optimum design solutions in that order. The comparison of the maximum compressive stresses and the standard deviation of stresses at the inner strip between the original and optimal designs is given in Table 2. The results show that the optimal processing parameters obtained by the proposed ITLBO algorithm can reduce the maximum compressive stress and the axial inhomogeneity of the stress distribution at the inner strip, which might cause undesirable wavy edge defects during the strip coiling process.

Table 1. Objective function values calculated

Evolutionary algorithms	Mean	STD	Max.	Min.
DE	0.9700	0.0275	1.0096	0.9354
ABC	1.7637	0.0787	1.8800	1.6751
ACOR	1.0621	0.0070	1.0705	1.0546
ChSS	1.4026	0.0289	1.4448	1.3678
LCA	1.7116	0.0408	1.7580	1.6473
SA	1.5451	0.0645	1.6323	1.4841
TLBO	0.9915	0.0132	1.0066	0.9766
OPDE	0.9539	0.0179	0.9715	0.9297
ETLBO-DE	0.8850	0.0047	0.8897	0.8784
ITLBO	0.8740	0.0025	0.8783	0.8720

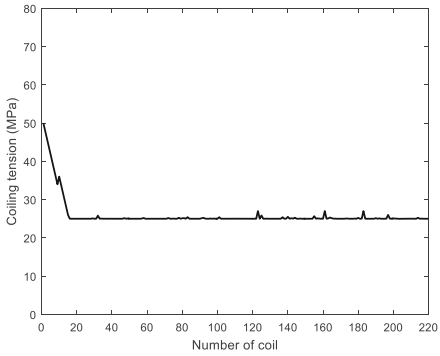
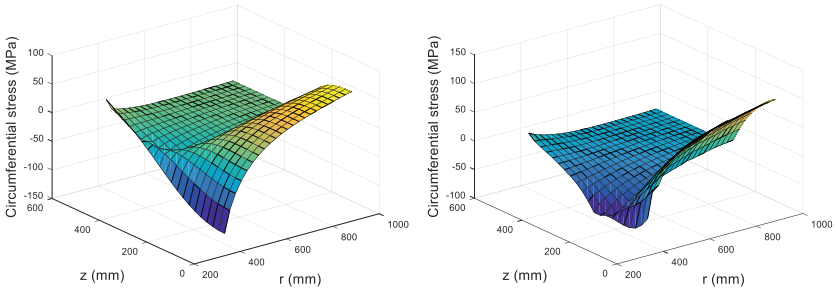


Fig. 4. Coiling tension levels as a function of number of coils



(a) Original design (b) Optimal design

Fig. 5. Comparison of circumferential stresses along the z and r directions for the original design and optimal design, respectively

Table 2. Maximum compressive stress and the standard deviation of stresses at the inner coil

	Original design	Optimal design
Maximum compressive stress [MPa]	111.546	68.0270
Standard deviation of stresses	48.375	29.3703

6 Conclusions

The new population-based optimization algorithm obtained by improving the original TLBO for solving the flatness optimization of the strip coiling process has been proposed. The search performance of the method was compared to various established evolutionary algorithms. The numerical results show that the new optimizer ITLBO is the best performer for both convergence rate and consistency. With this, the new parameters including the spool geometry and the coiling tension distribution have been obtained and can be used in the real strip coiling process. Further studies will be made to enhance the mathematical model of the strip coiling process. A self-adaptive version of ITLBO will be investigated for search performance enhancement.

Acknowledgements. The authors gratefully acknowledge the financial support from Thailand Research Fund (TRF). The research grant from the POSCO was also appreciated.

References

1. Jung, J.Y., Im, Y.T.: Simulation of fuzzy shape control for cold-rolled strip with randomly irregular strip shape. J. Mater. Process. Tech. **63**, 248–253 (1997)
2. Jung, J.Y., Im, Y.T.: Fuzzy control algorithm for prediction of tension variations in hot rolling. J. Mater. Process. Tech. **96**, 163–172 (1999)
3. Kawanami, T., Asamura, T., Matsumoto, H.: Development of high-precision shape and crown control technology for strip rolling. J. Mater. Process. Tech. **22**, 257–275 (1990)
4. Kwon, H.C., Han, I.S., Chun, M.S.: Examination of thermal behavior of hot rolled coil based on the finite element modeling and thermal measurement. In: 10th International Conference on Technology of Plasticity, pp. 37–40 (2011)
5. Sims, R.B., Place, J.A.: The stresses in the reels of cold reduction mills. Br. J. Appl. Phys. **4**, 213–216 (1953)
6. Miller, D.B., Thornton, M.: Prediction of changes in flatness during coiling. In: 5th International Rolling Conference, pp. 73–78 (1990)
7. Sarban, A.A.: An elasto-plastic analysis for the prediction of changes in flatness during coiling. In: 2nd International Conference on Modelling of Metal Rolling Processes, pp. 92–100 (1996)
8. Yanagi, S., Hattori, S., Maeda, Y.: Analysis model for deformation of coil of thin strip under coiling process. J. JSTP. **39**, 51–55 (1998)
9. Park, W.W., Kim, D.K., Kwon, H.C., Chun, M.S., Im, Y.T.: The effect of processing parameters on elastic deformation of the coil during the thin-strip coiling process. Metals Mater. Inter. **20**, 719–726 (2014)
10. Pholdee, N., Bureerat, S., Park, W.-W., Kim, D.-K., Im, Y.-T., Kwon, H.-C., Chun, M.-S.: Optimization of flatness of strip during coiling process based on evolutionary algorithms. Int. J. Precis. Eng. Manuf. **16**(7), 1493–1499 (2015)
11. Pholdee, N., Park, W.-W., Kim, D.-K., Im, Y.-T., Bureerat, S., Kwon, H.-C., Chun, M.-S.: Efficient hybrid evolutionary algorithm for optimization of a strip coiling process. Eng. Opt. **47**(4), 521–532 (2015)
12. Pholdee, N., Bureerat, S.: Passive vibration control of an automotive component using evolutionary optimization. J. Res. Appl. Mech. Eng. **1**, 19–22 (2010)
13. Pholdee, N., Bureerat, S.: Surrogate-assisted evolutionary optimizers for multiobjective design of a torque arm structure. Appl. Mech. Mater. **101–102**, 324–328 (2012)
14. Pholdee, N., Bureerat, S.: Performance enhancement of multiobjective evolutionary optimizers for truss design using an approximate gradient. Comput. Struct. **106–107**, 115–124 (2012)
15. Pholdee, N., Bureerat, S.: Hybridisation of real-code population-based incremental learning and differential evolution for multiobjective design of trusses. Inform. Sci. **223**, 136–152 (2013)
16. Bureerat, S., Limtragool, J.: Structural topology optimisation using simulated annealing with multiresolution design variables. Finite Elem. Anal. Des. **44**, 738–747 (2008)
17. Srisoporn, S., Bureerat, S.: Geometrical design of plate-fin heat sinks using hybridization of MOEA and RSM. IEEE Trans. Compon. Packag. Technol. **31**, 351–360 (2008)
18. Yildiz, A.R.: A novel particle swarm optimization approach for product design and manufacturing. Int. J. Adv. Manuf. Tech. **40**, 617–628 (2009)

19. Goldstein, M.: DEEPSAM: a hybrid evolutionary algorithm for the prediction of biomolecules structure. In: Blesa, M.J., Blum, C., Cangelosi, A., Cutello, V., Di Nuovo, A., Pavone, M., Talbi, E.-G. (eds.) HM 2016. LNCS, vol. 9668, pp. 218–221. Springer, Heidelberg (2016). doi:10.1007/978-3-319-39636-1_16
20. Kumar, S., Dubey, A.K., Pandey, A.K.: Computer-aided genetic algorithm based multi-objective optimization of laser trepan drilling. Int. J. Precis. Eng. Manuf. **14**, 1119–1125 (2013)
21. Oladimeji, M.O., Turkey, M., Dudley, S.: A heuristic crossover enhanced evolutionary algorithm for clustering wireless sensor network. In: Squillero, G., Burelli, P. (eds.) EvoApplications 2016. LNCS, vol. 9597, pp. 251–266. Springer, Heidelberg (2016). doi:10.1007/978-3-319-31204-0_17
22. Park, H.S., Nguyen, T.T.: Optimization of roll forming process with evolutionary algorithm for green product. Int. J. Precis. Eng. Manuf. **14**, 2127–2135 (2013)
23. Kandananond, K.: The optimization of a lathing process based on neural network and factorial design method. In: Fujita, H., Ali, M., Selamat, A., Sasaki, J., Kurematsu, M. (eds.) IEA/AIE 2016. LNCS (LNAI), vol. 9799, pp. 609–619. Springer, Heidelberg (2016). doi:10.1007/978-3-319-42007-3_53
24. Hetmaniok, E., Słota, D., Zielonka, A.: Solution of the inverse continuous casting problem with the aid of modified harmony search algorithm. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) PPAM 2013. LNCS, vol. 8384, pp. 402–411. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55224-3_38
25. Tiwari, A., Oduguwa, V., Roy, R.: Rolling system design using evolutionary sequential process optimization. IEEE Trans. Evol. Comput. **12**, 196–202 (2008)
26. Chakraborti, N., Siva Kumar, B., Satish Babu, V., Moitra, S., Mukhopadhyay, A.: Optimizing surface profiles during hot rolling: a genetic algorithms based multi-objective optimization. Comp. Mater. Sci. **37**, 159–165 (2006)
27. Zhang, J., Wang, Y.: Defection recognition of cold rolling strip steel based on ACO algorithm with quantum action. In: Pan, Z., Cheok, A.D., Müller, W., Chang, M., Zhang, M. (eds.). LNCS, vol. 7145, pp. 263–271Springer, Heidelberg (2012). doi:10.1007/978-3-642-29050-3_26
28. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimisation over continuous spaces. J. Global Optim. **11**, 341–359 (1997)
29. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimisation: artificial bee colony (ABC) algorithm. J. Global Optim. **39**, 459–471 (2007)
30. Socha, K., Dorigo, M.: Ant colony optimisation for continuous domains. Eur. J. Oper. Res. **185**, 1155–1173 (2008)
31. Rao, R.V., Savsani, V.J., Vakharia, D.P.: Teaching–learning-based optimisation: a novel method for constrained mechanical design optimisation problems. Comput. Aided Design. **43**, 303–315 (2011)
32. Kashan, A.H.: An efficient algorithm for constrained global optimisation and application to mechanical engineering design: league championship algorithm (LCA). Comput. Aided Design. **43**, 1769–1792 (2011)
33. Kaveh, A., Talatahari, S.: A novel heuristic optimisation method: charged system search. Acta Mech. **213**, 267–289 (2010)
34. Choi, Y.J., Lee, M.C.: A downcoiler simulator for high performance coiling in hot strip mill lines. Int. J. Precis. Eng. Manuf. **10**, 53–61 (2009)

http://www.springer.com/978-3-319-49396-1

Multi-disciplinary Trends in Artificial Intelligence
10th International Workshop, MIWAI 2016, Chiang Mai, Thailand, December 7-9, 2016, Proceedings
Sombattheera, C.; Stolzenburg, F.; Lin, F.; Nayak, A. (Eds.)
2016, XXIII, 314 p. 82 illus., Softcover
ISBN: 978-3-319-49396-1

Adaptive Sine Cosine Algorithm Integrated with Differential Evolution for Structural Damage Detection

Sujin Bureerat and Nantiwat Pholdee(✉)

Faculty of Engineering, Department of Mechanical Engineering,
Sustainable and Infrastructure Research and Development Center,
Khon Kaen University, Khon Kaen 40002, Thailand
nantiwat@kku.ac.th

Abstract. A sine cosine algorithm is one promising meta-heuristic recently proposed. In this work, the algorithm is extended to be self-adaptive and its main reproduction operators are integrated with the mutation operator of differential evolution. The new algorithm is called adaptive sine cosine algorithm integrated with differential evolution (ASCA-DE) and used to tackle the test problems for structural damage detection. The results reveal that the new algorithm outperforms a number of established meta-heuristics.

Keywords: Sine cosine algorithm · Differential evolution · Structural damage detection · Optimization · Meta-heuristics

1 Introduction

Structural health monitoring is an essential topic in the structural engineering field due to the requirement of reliability and safety use of the structure thought its life time. The key point of structural health monitoring is to identify the presence of structural damage, localising the damage and predicting its severity without destroying the structure or using nondestructive testing [1–3]. One of the most popular nondestructive techniques to identify damage location is to use static and/or dynamic testing data such as changing on strain data, structural deflection [4, 5], or modal data such as natural frequencies and mode shapes [1, 4–8].

Numerous work on damage detection based on changing of modal data has been reported worldwide [1, 4–16]. The main idea of this technique is to measure the modal data of an undamaged structure and use it as the baseline. When the modal data has changed, structural damage is supposed to occur. Identifying damage location can be achieved by applying a soft computing technique such as fuzzy logic systems [17, 18] and neural networks systems [17, 19–21]. For the latter, a large number of modal data of the damage and undamaged structures must be provided and used for training the system. Although much work has claimed that the technique works well, the requirement of a large number of training data and analysis time is an inevitable obstacle.

O. Gervasi et al. (Eds.): ICCSA 2017, Part I, LNCS 10404, pp. 71–86, 2017.
DOI: 10.1007/978-3-319-62392-4_6

Recently, the damage detection is traded as an optimization inverse problem. The main idea is to update mechanical properties of a mathematical model such as a finite element model until the modal data such as natural frequency of the model agree well with the testing data while the optimum parameters of the mechanical properties can be obtained by means of optimization [7, 9–14]. Over the last few decades, much research has successfully applied meta-heuristics (MHs) for this kind of problem, for example, genetic algorithm (GA) [9], differential evolution algorithm (DE) [16], ant colony optimization (ACO) [10], charged system search (ChSS) [11], etc. [7, 9–14, 16]. Although successful results have been reported, it is found that most of them choose to present only the best runs of the algorithms. This is not a proper way to investigate the use of MHs for real world applications. The methods must be run to solve a particular design problem many times and use the mean values of objective function as a performance indicator. Over a decade, over a thousand of MHs have been introduced based on a wide variety of search concepts and mechanisms. The algorithms can be regarded as a search method or an optimizer. The methods are simple to understand/use/code due to it being a soft computing technique. They can be used as an alternative to classical gradient-based optimization methods particularly for optimization problems in which sufficiently accurate function derivatives are not affordable. Moreover, they can be used for multiobjective or many-objective optimization more effectively than using the gradient-based optimizers since they can explore a Pareto optimal front within one optimization run. A genetic algorithm is arguable the best known and most used MH. Then, there can be differential evolution, particle swarm optimization and so on. Recently, there have been numerous MHs being published each year some of the new algorithms are a grey wolf optimizer [22], moth-fame optimization [23], the whale optimization algorithm [24] and a sine cosine algorithm [25]. Those algorithms are remained to be investigated when applying to practical design problem.

Therefore, this paper presents and extension of the sine cosine algorithm. An adaptive strategy is embedded into the new version while the mutation operator of differential evolution is integrated into the algorithm in order to further improve its performance. The new optimizer is then termed an adaptive sine cosine algorithm integrated with differential evolution (ASCA-DE). The optimizer is then implemented on several test problems for structural damage detection. Numerical results show that the proposed MH is superior to a number of established MHs found in the literature.

2 Formulation of a Damage Detection Optimization Problem

In this work, vibration based damage detection based on using natural frequencies is used for damage localization of truss structures. The main concept of using structural natural frequencies for damage detection of a truss structure is based on using a finite element model and the measured natural frequencies. When the natural frequencies and mode shapes are measured (usually the lowest n_{mode} natural frequencies), the finite element model is updated until the computed natural frequencies fit well with the measured ones. For the undamaged structure, natural frequencies can be calculated from a simple linear undamped free vibration finite element model which can be expressed as;

$$[\mathbf{K}]\{\phi_j\} - \omega_j^2[\mathbf{M}]\{\phi_j\} = 0 \tag{1}$$

where [**K**] is a structural stiffness matrix which can be expressed as the summation of element stiffness matrices [$\mathbf{k}_e$],

$$[\mathbf{K}] = \sum_{i=1}^{n_e} [\mathbf{k}_e] \tag{2}$$

where i is the i^{th} element of the structure while n_e is the total number of elements. The matrix [**M**] is a structural mass matrix computed in similar fashion to the stiffness matrix. The variables ϕ_j and ω_j are the j^{th} mode shape and its corresponding natural frequency, respectively. For the damaged structure, the stiffness matrix of the damaged element is assumed to be modified. The stiffness matrix of the damaged structure [$\mathbf{K}_d$] can be written as a percentage of damage in the elements as follows:

$$[\mathbf{K}_d] = \sum_{i=1}^{n_e} \frac{100 - p_i}{100} [\mathbf{k}_e] \tag{3}$$

where p_i is the percentage of damage on the i^{th} element. The natural frequency of the damaged structure can be computed by solving Eq. (1) by replacing [**K**] with [$\mathbf{K}_d$].

The percentage of damage in the structural element (p_i) can be found by solving an optimization problem to minimise the root mean square error (RMSE) between natural frequencies measured from the damaged structure and natural frequencies computed by using the finite element model. The problem can be expressed as follow:

$$\text{Min: } f(\mathbf{x}) = \sqrt{\frac{\sum_{j=1}^{n_{mode}} \left(\omega_{j,damage} - \omega_{j,computed}\right)^2}{n_{mode}}} \tag{4}$$

where $\omega_{j,damage}$ and $\omega_{j,computed}$ are the structural natural frequency of mode j obtained from a damaged structure and that from solving (1) – (3). The design variables are those damage percentages of structural elements ($\mathbf{x} = \{p_1, \ldots, p_{\text{nele}}\}^T$) respectively. In this work, six vibration modes are used for calculation.

3 Test Problems with Trusses

Four truss damage detection optimization problems from two truss structures are used in this study. These are the test problems used in our previous studies [14]. Detail of the test problems are shown as follow:

3.1 Twenty-Five-Bar Truss

The structure is shown in Fig. 1 [10, 14]. The cross sections of all bar elements are set to be 6.4165 mm^2. Table 1 shown the material properties and simulated case study for this example. The data of natural frequencies of the undamaged and damaged 25-bar truss structures are shown in Table 2.

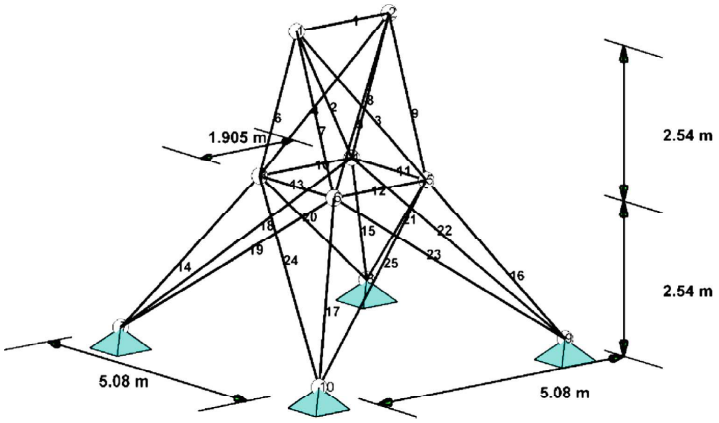


Fig. 1. Twenty-five bar truss

Table 1. Material properties and simulated case study for 25-bar truss.

Material density	7,850 kg/m^3
Modulus of elasticity	200 GPa
Simulated case study	Case I: 35% damage at element number 7
	Case II: 35% damage at element number 7 and 40% damage at element number 9

Table 2. Natural frequencies (Hz) of damaged and undamaged of 25 bar structure.

Mode	Undamaged	35% damage at element number 7	35% damage at element number 7 and 40% damage at element number 9
1	69.7818	69.1393	68.5203
2	72.8217	72.2006	71.3167
3	95.8756	95.3372	94.5625
4	120.1437	119.8852	119.6514
5	121.5017	121.4774	121.4253
6	125.0132	125.0130	125.0129

3.2 Seventy-Two-Bar Truss

The structure is shown in Fig. 2 [7, 14]. Four non-structural masses of 2270 kg are attached to the top nodes. The cross sections of all bar elements are set to be 0.0025 m^2. Table 3 shown the material properties and simulated case study for this example. The data of natural frequencies of the undamaged and damaged 72-bar truss structure are shown in Table 4.

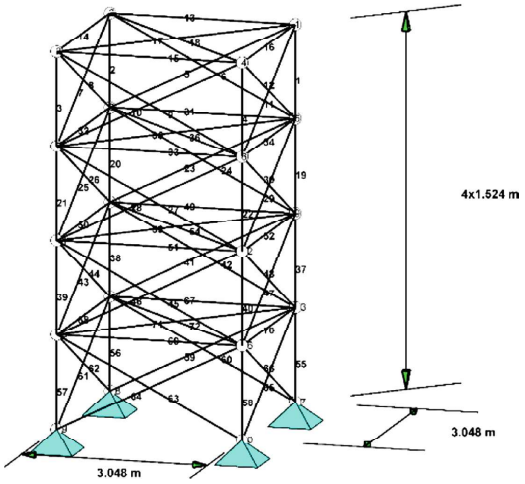


Fig. 2. Seventy-two bar truss

Table 3. Material properties and simulated case study for 72-bar truss.

Material density	2,770 kg/m^3
Modulus of elasticity	6.98×10^{10} Pa
Simulated case study	Case I: 15% damage at element number 55 (15% damage in element number 56, 57, or 58 results in the same set of natural frequencies)
	Case II: 10% damage at element number 4 and 15% damage at element number 58 (90, 180, and 270° rotation along the *z* axis lead to the same set of natural frequencies)

Table 4. Natural frequencies (Hz) of damaged and undamaged of 72 bar structure.

Mode	Undamaged	15% damage at element number 55	15% damage at element number 58 and 10% damage at element number 4
1	6.0455	5.9553	5.9530
2	6.0455	6.0455	6.0455
3	10.4764	10.4764	10.4764
4	18.2297	18.1448	18.0921
5	25.4939	25.4903	25.2437
6	25.4939	25.4939	25.4939

4 Adaptive Sine Cosine Algorithm Hybridized with Differential Evolution (ASCA-dE)

The Sine Cosine Algorithm (SCA) is a population based optimization method proposed by Mirjalili [25]. The algorithm is simple and efficient for various optimization test problems as reported in [25]. The search procedure of SCA is similar to other MH which contains three main steps; population initialisation, population updating and population selection. For the SCA, updating population can be done based on a sine and cosine function. Given a current population having *NP* members $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\mathrm{NP}}\}^{\mathrm{T}}$, an element of a solution vector for the next generation can be calculated as follows:

$$x_{new,k} = \begin{cases} x_{old,k} + r_1 \sin(r_2)\left|r_3 x_{best,k} - x_{old,k}\right|, & if \quad r_4 < 0.5, \\ x_{old,k} + r_1 \cos(r_2)\left|r_3 x_{best,k} - x_{old,k}\right|, & otherwise \end{cases} \tag{5}$$

where $x_{\mathrm{best,k}}$ is the k^{th} matrix element of the current best solution. The variables r_2, r_3, and r_4 are random parameters in the ranges of [0, 2π], [0, 2] and [0, 1], respectively. The variable r_1 is an iterative adaption parameter,

$$r_1 = a - T\frac{a}{T_{\max}} \tag{6}$$

where a is a constant parameter while T is an iteration number. $T_{\max}$ is maximum number of iterations.

The search process of SCA start with generating an initial population at random, and then calculating their objective function values where the best solution is found. Then, the new population for the next generation is generated using Eq. (5) and the objective function values of its members are calculated. The current best will be compared with the best solution of the newly generated population and the better one is saved to the next generation. The process is repeated until a termination criterion is met. The computational steps of SCA are shown in Algorithm 1.

Algorithm 1 Sine Cosine Algorithm
Input: population size (N_p), number of generations (T_{max}), number of design variable (D)
Output: $\mathbf{x}_{best}$, f_{best}
Main algorithm
1: Initialise a population and set as the current population.
2: Find the best solution ($\mathbf{x}_{best}$)
3: For T=1 to T_{max}
4: Calculate parameter r_1 using eq.(6)
5: For l=1 to N_p
6: For k = 1 to D
7: Generate the parameter r_2, r_3 and r_4
8: Update the k^{th} element of the l^{th} population ($\mathbf{x}_1$) using eq.(5)
9: End For
10: End For
11: Calculate objective function values of the newly generated population and find the best ones ($\mathbf{x}_{best,new}$)
12: Replace $\mathbf{x}_{best}$ by $\mathbf{x}_{best,new}$ if $f(\mathbf{x}_{best,new}) < f(\mathbf{x}_{best})$
13: End

For the proposed adaptive sine cosine algorithm with integration of DE mutation, the DE mutation operator as proposed in Pholdee and Bureerat [26] is integrated into the updating operation. The mutation equation is detailed as follow;

$$\mathbf{x}_{new} = \mathbf{x}_{best} + rand(-1, +1)F\left(\mathbf{x}_{r,1} + \mathbf{x}_{r,2} - \mathbf{x}_{r,3} - \mathbf{x}_{r,4}\right) \tag{7}$$

where $rand(-1, 1)$ gives either -1 or 1 with equal probability. F is a scaling factor while $\mathbf{x}_{r,1}$–$\mathbf{x}_{r,4}$ are four solutions randomly selected from the population.

At ASCA-DE updating operation, if a generated uniform random number in the interval [0,1] is lower than a probability value ($rand < P_{DE}$), the population will be updated using the SCA updating operation based on Eq. (5), otherwise, the population will be updated by DE mutation as detailed in Eq. (7).

The term of self-adaption of the proposed algorithm is accomplished in such way that the parameter r_2, r_3 and F are regenerated for each calculation based on the information from the previous iteration. For each calculation, the r_2 and r_3 are generated based on normally distributed random numbers with mean values, r_{2m} and r_{3m} respectively and standard deviation values, STD = 0.1 for both r_2 and r_3. The values of r_{2m} and r_{3m} are iteratively adapted based on the following equations:

$$r_{2m}(T+1) = 0.9r_{2m}(T) + 0.1mean(good_{r2m}), \tag{8}$$

and,

$$r_{3m}(T+1) = 0.9r_{3m}(T) + 0.1mean(good_{r3m}), \tag{9}$$

where $mean(good_{\mathrm{r2m}})$ and $mean(good_{\mathrm{r3m}})$ are the mean values of all values of r_2 and r_3 used in current iteration that lead to successful updates. The successful update means the created offspring is better than its parent from the previous iteration. In addition, for each calculation, the scaling factor F is generated by Cauchy distribution randomisation with the mean value F_{m} and STD value of 0.1 [27]. The F_{m} is iteratively adapted using the Lehmer mean [27] defined as follows:

$$F_m(T+1) = 0.9F_m(T) + 0.1\frac{sum(good_F^2)}{sum(good_F)} \tag{10}$$

where $good_F$ is a tray of all F used in the current iteration with successful updates.

The parameter P_{DE} is also regenerated in the similar fashion to r_2 and r_3 before updating a population. For an individual solution, the P_{DE} is generated by normal distribution randomising with the mean value of P_{DEm} and standard deviation of 0.1. P_{DEm} is iteratively adapted based on the following equation:

$$P_{DE\mathrm{m}}(T+1) = 0.9P_{DE\mathrm{m}}(T) + 0.1mean(good_{\mathrm{PDE}}), \tag{11}$$

where $good_{\mathrm{PDE}}$ means all P_{DE} values used in the current iteration with successful updates.

The search process of ASCA-DE start with initilaising a population, $r_{2\mathrm{m}}$, $r_{3\mathrm{m}}$, F_m and P_{DEm}. The $good_{\mathrm{r2m}}$, $good_{\mathrm{r3m}}$, $good_{\mathrm{F}}$ and $good_{\mathrm{PDE}}$ trays are empty initially. After having calculated objective function values, the current best solution will be obtained. To update a population, firstly, P_{DE} and a random number in [0, 1] are generated. If the generated random number is lower than P_{DE}, a scaling factor (F) is generated based on F_{m} and a new solution is created using Eq. (7), otherwise, a new solution is generated based on Eq. (5). For each calculation of Eq. (5), r_2 and r_3 are generated based on $r_{2\mathrm{m}}$ and $r_{3\mathrm{m}}$. If a newly generated solution is better than its parent, the new solution will be selected for the next generation while saving all used parameters P_{DE}, r_2, r_3 and F into the $good_{\mathrm{PDE}}$, $good_{\mathrm{r2m}}$, $good_{\mathrm{r3m}}$, and $good_{\mathrm{F}}$ trays, respectively. Then, update the $r_{2\mathrm{m}}$, $r_{3\mathrm{m}}$, F_m and P_{DEm} using Eqs. (8)–(11). The search process is repeated until a termination criterion is reached. The computational steps of ASCA-DE are shown in Algorithm 2.

Algorithm 2 ASCA-DE

Input: population size (N_p), number of generations (T_{max}), number of design variable (D)

Output: $\mathbf{x}_{best}$, f_{best}

Main algorithm

1: Initialise a population , r_{2m}, r_{3m}, F_m and P_{DEm}.
2: Find the best solution ($\mathbf{x}_{best}$)
3: *For* T=1 to T_{max}
4: Calculate parameter r_1 using eq.(6)
5: Empty $good_{r2m}$, $good_{r3m}$, $good_F$ and $good_{PDE}$
5: *For* l=1 to N_p
6: Generate P_{DE} by normal distribution random with mean values P_{DEm} and STD =0.1
7: *IF* $rand < P_{DE}$
8: Generate F by Cauchy distribution random with mean value F_m and STD = 0.1
9: Updated a population using eq. (7)
10: *Else*
11: *For* k = 1 to D
12: Generate the parameter r_2 and r_3 by normal distribution random with mean values r_{2m}, r_{3m}, and STD =0.1
13: Random generate r_4 in rank [0, 1]
14: Update the k^{th} element of the l^{th} population ($\mathbf{x}_1$) using eq.(5)
14: *End For*
16: *End IF*
17: Calculate objective function values of the newly generated population
18: *IF* $f(\mathbf{x}_{1,new}) < f(\mathbf{x}_{1,old})$
19: Replace $\mathbf{x}_{1,old}$ by $\mathbf{x}_{1,new}$
20: Add all generated r_2, r_3, F, and P_{DE}, into the $good_{r2m}$, $good_{r3m}$, $good_F$ and $good_{PDE}$ tray, respectively.
21: *End IF*
22: *End For*
23: Find the best solution ($\mathbf{x}_{best}$)
24: Update r_{2m}, r_{3m}, F_m, and P_{DEm} using eq.(8)-(11)
24: End

5 Numerical Experiment

The performance investigation of the proposed ASCA-DE for structural damage detection is carried out by employing the algorithm to solve the test problems in the previous section. ASCA-DE along with a number of MHs in the literature implemented to solve the test problems include (Note that the details of variables can be found in the original sources of each method) [14]:

- Differential evolution (DE) [28]: a DE/best/2/bin strategy was used. A scaling factor, and probability of choosing elements of mutant vectors (*CR*) are 0.5 and 0.8 respectively.
- Artificial bee colony algorithm (ABC) [29]: The number of food sources for employed bees is set to be $n_P/2$. A trial counter to discard a food source is 100.
- Real-code ant colony optimization (ACOR) [30]: The parameter settings are $q = 0.2$, and $\xi = 1$.
- Charged system search (ChSS) [31]: The number of solutions in the charge memory is $0.2 \times n_P$. The charged moving considering rate and the parameter *PAR* are set to be 0.75 and 0.5 respectively.
- League championship algorithm (LCA) [32]: The probability of success P_c and the decreasing rate to decrease P_c are set to be 0.9999 and 0.9995, respectively.
- Simulated annealing (SA) [33]: Starting and ending temperatures are 10 and 0.001 respectively. For each loop, n_{mode} candidates are created by mutating on the current best solution while other n_{mode} candidates are created from mutating the current parent. The best of those $2n_{mode}$ solutions are set as an offspring to be compared with the parent.
- Particle swarm optimization (PSO) [34]: The starting inertia weight, ending inertia weight, cognitive learning factor, and social learning factor are assigned as 0.5, 0.01, 0.5 and 0.5 respectively.
- Evolution strategies (ES) [35]: The algorithm uses a binary tournament selection operator and a simple mutation without the effect of rotation angles.
- Teaching-learning-based optimization (TLBO) [36]: Parameter settings are not required.
- Adaptive differential evolution (JADE) [27]: The parameters are self-adapted during an optimization process.
- Evolution strategy with covariance matrix adaptation (CMAES) [37]: The parameters are self-adapted during an optimization process.
- Sine Cosine Algorithm (SCA) (Algorithm 1) [25]: The constant a parameter is set to be 2.
- Adaptive Sine Cosine algorithm with integrating DE mutation (ASCDE) (Algorithm 2): The parameter a is set to be 2 while initial r_{2m}, r_{3m}, F_m and P_{DEm} are set to be 0.5.

Each optimizer is used to tackle each truss damage detection test problem for 30 optimization runs. The number of iterations (generations) is 300 for all case studies while the population size is set to be 30 and 50 for 25-bar and 72-bar trusses respectively. All methods will be terminated with two criteria: the maximum numbers

of functions evaluatio as 20 × 300, 30 × 300 and 50 × 300 for the 25-bar and 72-bar trusses respectively, and the objective function value being less than or equal to 1×10^{-3}. The six lowest natural frequencies ($n_{mode} = 6$) are used to compute the objective function value. This number of selected frequencies is reasonable since it is practically easier to measure fewer lowest natural frequencies with sufficient accuracy.

6 Results and Discussions

After performing 30 optimization runs of all MHs on solving the four truss damage detection optimization problems, the results obtained from the various MHs are given in Tables 5, 6, 7 and 8. The mean of the objective function are used to indicate the search convergence of the algorithms in cases that the objective function threshold (1×10^{-3}) is not met during searching. Otherwise, the mean number of FEs is used as an indicator. The number of successful runs out of 30 runs is used to measure the search consistency. The algorithm that is terminated by the objective function threshold is obviously superior and any run being stopped with this criterion is considered a successful run.

Table 5. Results for 25 bar truss with 35% damage at element number 7

Optimizers	Mean objective function values	No. of successful runs from 30 runs	Mean of FEs
DE	0.0017	19	6019
ABC	0.0135	0	9000
ACOR	0.0089	0	9000
ChSS	0.1385	0	9000
LCA	0.9036	0	9000
SA	0.0089	0	9000
TLBO	0.0077	6	7772
CMAES	0.0033	0	9000
ES	0.0308	0	9000
PSO	8.3830	0	9000
JADE	0.0026	2	8953
SCA	0.0270	24	3262
ASCA-DE	0.0009	29	2835

6.1 Twenty-Five-Bar Truss

Table 5 shown the results of the 25-bar truss with 35% damage at element 7. The best performer based on the mean objective function values is ASCA-DE while the second best and the third best are DE and JADE respectively. When considering the number of successful runs, seven optimizers including DE, TLBO, JADE, SCA and ASCA-DE can detect the damage of the structure. The most efficient optimizer is ASCA-DE that can detect the damages of the structure for 29 times out of 30 runs with the average of 2835 function evaluations.

Table 6. Results for 25 bar truss with 35% damage at element number 7 and 40% damage at element number 9

Optimizers	Mean objective function values	No. of successful runs from 30 runs	Mean of FEs
DE	0.0096	27	5220
ABC	0.0326	0	9000
ACOR	0.0125	0	9000
ChSS	0.1590	0	9000
LCA	0.8080	0	9000
SA	0.0269	0	9000
TLBO	0.0405	1	8917
CMAES	0.0115	0	9000
ES	0.0356	0	9000
PSO	8.6012	0	9000
JADE	0.0042	6	8875
SCA	0.0930	0	9000
ASCA-DE	0.0032	26	5511

Table 7. Results for 72 bar truss with 15% damage at element number 55

Optimizers	Mean objective function values	No. of successful runs from 30 runs	Mean of FEs
DE	0.0087	14	12887
ABC	0.2184	0	15000
ACOR	0.0014	6	14831
ChSS	0.1727	0	15000
LCA	1.1499	0	15000
SA	0.0097	0	15000
TLBO	0.0035	27	5781
CMAES	0.0053	0	15000
ES	0.0010	29	9335
PSO	1.9146	0	15000
JADE	0.0019	1	15000
SCA	0.0070	23	4793
ASCA-DE	0.0008	30	1715

Results of the 25 bar truss with 35% damage at element 7 and 40% damage at the element number 9 are reported in Table 6. The best performer based on mean objective function values is ASCA-DE while the second best and the third best are JADE and DE respectably. When examining the number of successful runs, only two optimizers, DE and ASCA-DE can consistently detect the damage of the structure for 27 and 26 runs respectively while the average number of function evaluations to obtain the results are 5220 and 5511 respectively.

Table 8. Results for 72 bar truss with 15% damage at element number 58 and 10% damage at element number 4

Optimizers	Mean objective function values	No. of successful runs from 30 runs	Mean of FEs
DE	0.0127	7	13963
ABC	0.1591	0	15000
ACOR	0.0058	0	15000
ChSS	0.1348	0	15000
LCA	1.1049	0	15000
SA	0.0129	0	15000
TLBO	0.0045	7	13503
CMAES	0.0050	0	15000
ES	0.0023	2	14940
PSO	1.7726	0	15000
JADE	0.0031	0	15000
SCA	0.0260	2	14502
ASCA-DE	0.0035	21	9235

6.2 Seventy-Two-Bar Truss

Table 7 shows comparison results of the 72-bar truss with 15% damage at element 5. The best performer based on mean objective function values is ASCA-DE while the second best and the third best are ES and ACOR. When examining the number of successful runs, the most efficient method is ASCA-DE which can detect the damage of the structure for 30 times while the average numbers of function evaluations for the convergence results is only 1715.

Results of the 72 bar truss with 15% damage at element number 58 and 10% damage at element number 4 are given in Table 8. The best performer based on the mean of objective function values is ES while the second best and the third best are JADE and ASCA-DE respectively. The minimum objective function value is obtained by SCA. When considering the number of successful runs, only ASCA-DE can consistently detect the damage of the structure for 22 times from totally 30 optimization runs while the average number of function evaluations for the convergence results is 9235. Although ES and JADE given better mean objective function values, they fail to search for the damage location. ASCA-DE is said to be the most efficient optimizer for this case.

Overall, it was found that integrating DE mutation into and applying adaptive parameters to SCA lead to performance enhancement of the original SCA. The proposed ASCA-DE is the best performer on solving truss damage detection optimization problem. It is considered the most reliable method for this study.

7 Conclusions

Performance enhancement of a meta-heuristics called a sine cosine algorithm is proposed by integrating into it a mutation strategy of DE. Self-adaptive optimization parameters are employed to improve the search performance of the new algorithm. The proposed optimizer is implemented on solving a number of truss damage detection inverse problems. The results reveal that the new meta-heuristic is the best and most reliable method. Our future work is to investigate the new MH for solving other practical engineering design problems.

Acknowledgments. The authors are grateful for the support from the Thailand Research Fund (TRF).

References

1. Sinou, J.J.: A review of damage detection and health monitoring of mechanical systems from changes in the measurement of linear and non-linear vibrations. In: Sapri, R.C. (ed.) Mechanical Vibrations: Measurement, Effects and Control, pp. 643–702. Nova Science Publishers, Inc., Hauppauge (2009)
2. Chen, H., Shi, X., He, Q., Mao, J.H., Liu, Y., Kang, H., Shen, J.: A multiresolution investigation on fatigue damage of aluminum alloys at micrometer level, Int. J. Damage. Mech. **26** (2017). doi:10.1177/1056789517693411
3. Shen, J., Mao, J., Boileau, J., Chow, C.L.: Material damage estimated via linking micro/macroscale defects to macroscopic mechanical properties. Int. J. Damage Mech **23**, 537–566 (2014)
4. Wang, X., Hu, N., Fukunaga, H., Yao, Z.: Structural damage identification using static test data and changes in frequencies. Eng. Struct. **23**, 610–621 (2001). doi:10.1016/S0141-0296(00)00086-9
5. Gerist, S., Maheri, M.R.: Multi-stage approach for structural damage detection problem using basis pursuit and particle swarm optimization. J. Sound Vib. **384**, 210–226 (2016). doi:10.1016/j.jsv.2016.08.024
6. Koh, B.H., Dyke, S.J.: Structural health monitoring for flexible bridge structures using correlation and sensitivity of modal data. Comput. Struct. **85**, 117–130 (2007). doi:10.1016/j.compstruc.2006.09.005
7. Kaveh, A., Zolghadr, A.: An improved CSS for damage detection of truss structures using changes in natural frequencies and mode shapes. Adv. Eng. Softw. **80**, 93–100 (2015). doi:10.1016/j.advengsoft.2014.09.010
8. Laier, J.E., Villalba, J.D.: Ensuring reliable damage detection based on the computation of the optimal quantity of required modal data. Comput. Struct. **147**, 117–125 (2015). doi:10.1016/j.compstruc.2014.09.020
9. Chou, J.H., Ghaboussi, J.: Genetic algorithm in structural damage detection. Comput. Struct. **79**, 1335–1353 (2001). doi:10.1016/S0045-7949(01)00027-X
10. Majumdar, A., Maiti, D.K., Maity, D.: Damage assessment of truss structures from changes in natural frequencies using ant colony optimization. Appl. Math. Comput. **218**, 9759–9772 (2012). doi:10.1016/j.amc.2012.03.031

11. Tabrizian, Z., Amiri G.G., Beigy, M.H.A.: Charged system search algorithm utilized for structural damage detection. Shock Vib. **2014**, Article ID 194753, 13 p. (2014). doi:10.1155/2014/194753
12. Xu, H., Ding, Z., Lu, Z., Liu, J.: Structural damage detection based on Chaotic Artificial Bee Colony algorithm. Struct. Eng. Mech. **55**(6), 1223–1239 (2015). doi:10.12989/sem.2015.55.6.1223
13. Ding, Z.H., Huang, M., Lu, Z.R.: Structural damage detection using artificial bee colony algorithm with hybrid search strategy. Swarm Evol. Comput. **28**, 1–13 (2016). doi:10.1016/j.swevo.2015.10.010
14. Pholdee, N., Bureerat, S.: Structural health monitoring through meta-heuristics – comparative performance study. Adv. Comput. Des. **1**, 315–327 (2016). doi:10.12989/acd.2016.1.4.315
15. Pal, J., Banerjee, S.: A combined modal strain energy and particle swarm optimization for health monitoring of structures. J. Civil Struct. Health Monit. **5**, 353–363 (2015). doi:10.1007/s13349-015-0106-y
16. Casciati, S.: Stiffness identification and damage localization via differential evolution algorithms. Struct. Control Health Monit. **15**, 436–449 (2008). doi:10.1002/stc.236
17. Agarwalla, D.K., Dash, A.K., Bhuyan, S.K., Nayak, P.S.K.: Damage detection of fixed-fixed beam: a fuzzy neuro hybrid system based approach. In: Panigrahi, B.K., Suganthan, P.N., Das, S. (eds.) SEMCCO 2014. LNCS, vol. 8947, pp. 363–372. Springer, Cham (2015). doi:10.1007/978-3-319-20294-5_32
18. Jiao, Y.B., Liu, H.B., Cheng, Y.C., Gong, Y.F.: Damage identification of bridge based on chebyshev polynomial fitting and fuzzy logic without considering baseline model parameters. Shock Vib. **2015**, Article ID 187956, 10 p. (2015). doi:10.1155/2015/187956
19. Pan, D.-G., Lei, S.-S., Wu, S.-C.: Two-stage damage detection method using the artificial neural networks and genetic algorithms. In: Zhu, R., Zhang, Y., Liu, B., Liu, C. (eds.) ICICA 2010. LNCS, vol. 6377, pp. 325–332. Springer, Heidelberg (2010). doi:10.1007/978-3-642-16167-4_42
20. Abdeljaber, O., Avci, O.: Nonparametric structural damage detection algorithm for ambient vibration response: utilizing artificial neural networks and self-organizing maps. J. Archit. Eng. **22**, 04016004 (2016). doi:10.1061/(ASCE)AE.1943-5568.0000205
21. Sidibe, Y., Druaux, F., Lefebvre, D., Maze, G., Léon, F.: Signal processing and Gaussian neural networks for the edge and damage detection in immersed metal plate-like structures. Artif. Intell. Rev. **46**, 289–305 (2016). doi:10.1007/s10462-016-9464-z
22. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014). doi:10.1016/j.advengsoft.2013.12.007
23. Mirjalili, S.: Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl.-Based Syst. **89**, 228–249 (2015). doi:10.1016/j.knosys.2015.07.006
24. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016). doi:10.1016/j.advengsoft.2016.01.008
25. Mirjalili, S.: SCA: a sine cosine algorithm for solving optimization problems. Knowl.-Based Syst. **96**, 120–133 (2016). doi:10.1016/j.knosys.2015.12.022
26. Bureerat, S., Pholdee, N.: Optimal truss sizing using an adaptive differential evolution algorithm. J. Comput. Civil Eng. **30**, 04015019 (2015). doi:10.1061/(ASCE)CP.1943-5487.0000487
27. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE T. Evolut. Comput. **13**, 945–958 (2009). doi:10.1109/TEVC.2009.2014613
28. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359 (1997). doi:10.1023/A:1008202821328

29. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. **39**, 459–471 (2007). doi:10.1007/s10898-007-9149-x
30. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. Eur. J. Oper. Res. **185**, 1155–1173 (2008). doi:10.1016/j.ejor.2006.06.046
31. Kaveh, A., Talatahari, S.: A novel heuristic optimization method: charged system search. Acta Mech. **213**, 267–289 (2010). doi:10.1007/s00707-009-0270-4
32. Husseinzadeh, K.A.: An efficient algorithm for constrained global optimization and application to mechanical engineering design: League championship algorithm (LCA). Comput. Aided Design. **43**, 1769–1792 (2011). doi:10.1016/j.cad.2011.07.003
33. Bureerat, S., Limtragool, J.: Structural topology optimization using simulated annealing with multiresolution design variables. Finite Elem. Anal. Des. **44**, 738–747 (2008). doi:10.1016/j.finel.2008.04.002
34. Venter, G., Sobieszczanski-Sobieski, J.: Particle swarm optimization. AIAA J. **41**, 1583–1589 (2003). doi:10.2514/2.2111
35. Back, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, Oxford (1996)
36. Rao, R.V., Savsani, V.J., Vakharia, D.P.: Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput. Aided Des. **43**, 303–315 (2011). doi:10.1016/j.cad.2010.12.015
37. Hansen, N., Muller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol. Comput. **11**, 1–18 (2003). doi:10.1162/106365603321828970