Performance assessment of meta-heuristics for composite layup optimisation

Shahin Jalili[a,*] , Reza Khani[b], Alireza Maheri[a], Yousef Hosseinzadeh[b]

[a]*School of Engineering, University of Aberdeen, King's College, Aberdeen, AB24 3UE, United Kingdom*
[b]*Faculty of Civil Engineering, University of Tabriz, Tabriz, Iran*

Abstract:
This paper investigates the performance of several meta-heuristic algorithms, including particle swarm optimisation (PSO), different variants of differential evolution (DE), biogeography-based optimisation (BBO), cultural algorithm (CA), optics-inspired optimisation (OIO), and league championship algorithm (LCA), for optimum layup of laminated composite plates. The study provides detailed Pseudo codes for different algorithms. The buckling capacity maximisation of a 64-layer laminated composite plate under various load scenarios has been considered as the benchmark problem, in which the design variables are the stacking sequences of layers. A deep statistical comparison (DSC) method is employed to rank the performance of different algorithms. The DSC uses a non-parametric two-sample Kolmogorov–Smirnov (KS) test to conduct the performance comparisons between the algorithms. The overall performance rankings obtained from the DSC suggest that the LCA, OIO, and PSO algorithms perform remarkably better in comparison to other algorithms. The comparisons provide some interesting conclusions on the performance of different algorithms.

Keywords: Composite structure, Meta-heuristics, Deep statistical comparison, Kolmogorov–Smirnov test

1. Introduction

Composite materials refer to multi-phase materials with enhanced properties that are fabricated by an artificial combination of two or more distinct materials [1]. These materials have found important structural applications to date. One particular structural application of composite materials is in the laminated composite plates. Recent advances revealed that laminated composite plates are an increasingly popular structural type for a wide range of applications in different industries, such as aerospace, automotive, marine, building, and renewable energy industries. The high strength-to-weight ratios and flexibility in design are two major advantages of such structures. The different design parameters of laminated composite structures provide a great opportunity for designers to achieve the desired cost-efficient optimum designs for a given application.

* **Corresponding author:** Shahin Jalili, email: s.jalili@abdn.ac.uk, School of Engineering, University of Aberdeen, King's College, Aberdeen, AB24 3UE, United Kingdom

A recent literature review provided by Nikbakt et al. [2] reveals that the optimum design of laminated composite plates to enhance their mechanical behaviour and minimise costs have been attracted much attention in recent years. Researchers have been implemented different optimisation algorithms to attain optimum designs for different kinds of design variables, objective functions, and constraints. Among the different objective functions investigated in the literature, buckling load maximisation is one of the prominent objective functions that have been widely taken into consideration by researchers for the optimum design of laminated composites [3-5].

Meta-heuristic optimisation algorithms, such as genetic algorithms (GAs) [6], simulated annealing (SA) [7], differential evolution (DE) [8], ant colony optimisation (ACO) [9], particle swarm optimisation (PSO) [10], cultural algorithms (CAs) [11,12], biogeography-based optimisation (BBO) [13], and harmony search (HS) [14], are attractive techniques to solve complex optimisation problems [15-20]. Meta-heuristics are nature-inspired search techniques that take the advantage of solution perturbation and stochasticity to find acceptable solutions for real-world problems in a reasonable time [21]. Like other engineering disciplines, these algorithms have been successfully applied to optimise the laminated composite structures [22-29]. For example, Karakaya and Soykasap [30] employed GA and SA to enhance the buckling capacity and attain optimum stacking sequence for hybrid laminated composite plates with carbon/epoxy and glass/epoxy materials. Kaveh et al. [31] investigated the application of the BBO algorithm in stacking sequence optimisation of laminated composite plates for various load cases and aspect ratios to maximise the buckling load factor. HS was utilised by Almeida et al. [32] to optimise the buckling load factor of the balanced laminated composite plate under compressive in-plane loads. Akcair et al. [33] applied DE and SA to optimise the buckling load factor of the laminated composite plate under different increments in fibre orientations. Kaveh et al. [34] proposed a novel improved rank-based version of quantum-inspired evolutionary algorithm (QEA) for optimum stacking sequence of hybrid laminated composite plates under uncertain buckling loads. The literature review also reveals that researchers have been adopted/proposed novel meta-heuristic algorithms for buckling maximisation of laminated composite plates. For example, Rao et al. [35] employed a scatter search algorithm (SSA), Kaveh et al. [36,37] applied the charged system search (CSS) algorithm, Jaya algorithm (JA), and colliding bodies optimisation (CBO).

In recent years, some novel meta-heuristic algorithms have been developed by researchers to solve a wide range of engineering problems. League championship algorithm (LCA) [38] inspired

by the sporting competitions between the teams in sports leagues is one of such novel algorithms, which has been able to exhibit efficient performance for different kinds of problems [39-43]. Optics inspired optimisation (OIO) [44] algorithm is another recently developed meta-heuristic inspired by the optical characteristics of spherical mirrors in physics, which has been remarkably efficient for engineering applications [45-48]. The main objective of this paper is to assess the performance of different meta-heuristic algorithms, including PSO, different variants of DE, BBO, CA, OIO, and LCA, for optimum layup of laminated composite plates. The buckling capacity maximisation of a 64-layer laminated composite plate under various load cases is considered as the benchmark problem. A deep statistical comparison (DSC) is conducted to asses the performance of different algorithms. The DSC uses the non-parametric two-sample Kolmogorov–Smirnov (KS) test to pair-wise performance comparison between each pair of algorithms. Then, the algorithms are ranked based on the results obtained from the two-sample KS test. The study provides some interesting conclusions about the performance of investigated algorithms.

The rest of the paper is organised as follows. In Section 2, the algorithmic details of investigated meta-heuristics, as well as their Pseudo codes, will be presented. Section 3 will present the problem formulation for buckling load maximisation of laminated composite plates. The numerical test and comparisons will be presented in Section 4. Finally, the conclusions will be provided in Section 5.

2. Investigated optimisation algorithms

In this section, the algorithmic details of PSO, DE, CA, LCA, and OIO will be presented. Although the description of all details related to the mentioned algorithms in this study is not possible, this section will try to explain the main elements of algorithms without discussing the unnecessary details. However, the detailed Pseudo codes of algorithms would provide a clear picture to readers about how the algorithms work. All algorithms will be described for the minimisation problem with the objective function of $f(\boldsymbol{X})$ and vector of variables $\boldsymbol{X} = [x_1, x_2, \dots, x_n]$, in which variables should satisfy the search space constraints represented by vectors $\boldsymbol{X}^{\min} = [x_1^{\min}, x_2^{\min}, \dots, x_n^{\min}]$ and $\boldsymbol{X}^{\max} = [x_1^{\max}, x_2^{\max}, \dots, x_n^{\max}]$ as follows: $x_k^{\min} \le x_k \le x_k^{\max}, k = 1, 2, \dots, n$.

2.1. Particle Swarm Optimisation (PSO)

Cooperative behaviour observed between the individuals in a group of species enhances their capabilities to deal with environmental challenges. These cooperation capabilities make them able

to achieve difficult goals, which are almost impossible to gain by every single individual in the group. Swarm Intelligence (SI) techniques try to inspire the collective intelligence exhibited by a group or swarm of species in nature to perform the optimisation process more efficiently [49]. One of the prominent SI techniques is the Particle Swarm Optimisation (PSO) algorithm inspired by the social behaviour of natural swarms. The PSO was originally developed by Kennedy and Eberhart [10] in the 1990s. The algorithm assumes a given set of particles in the search space, representing potential solutions for the problem. Each particle has its position and velocity, which are defined as n-dimensional vectors for the n-dimensional problem. The algorithm tries to benefit from the experience acquired by every individual particle and the whole swarm to update the positions and velocities of the particles.

To explain how the algorithm works, let us assume $\boldsymbol{X}_i^t = [x_{i,1}^t, x_{i,2}^t, \dots, x_{i,n}^t]$ and $\boldsymbol{V}_i^t = [v_{i,1}^t, v_{i,2}^t, \dots, v_{i,n}^t]$ are the position and velocity of the ith particle in the solution space at iteration t, in which $i = 1, 2, \dots, N_p$ and N_p is the swarm size. The PSO updates the velocities and positions of particles using the following formulas [10]:

$$\boldsymbol{V}_i^{t+1} = \omega^t \boldsymbol{V}_i^t + c_1 rand(\boldsymbol{P}_i^t - \boldsymbol{X}_i^t) + c_2 rand(\boldsymbol{G}^t - \boldsymbol{X}_i^t) \quad (1)$$

$$\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t + \boldsymbol{V}_i^{t+1} \quad (2)$$

In equations (1) and (2), $\boldsymbol{X}_i^{t+1}$ and $\boldsymbol{V}_i^{t+1}$ represent the position and velocity of the ith particle at iteration $t+1$, ω^t is the weighting parameter at iteration t which controls the influence of the previous velocity of the particle on its new velocity, $rand$ represents a uniformly generated random number between 0 and 1, c_1 and c_2 are the acceleration parameters, $\boldsymbol{P}_i^t$ indicates the position with the best objective function value experienced by the ith individual until iteration t, and $\boldsymbol{G}^t$ is the best position experienced by the whole swarm until iteration t. This study assumes that the value of the weighting parameter is gradually reduced in each iteration as $\omega^t = \omega_{\text{damp}}\omega^{t-1}$, in which ω_{damp} is a damping factor. In this study, the initial value for the weighting parameter is set to 1 (i.e., $\omega^0 = 1$). Algorithm 1 shows the Pseudo code of the PSO algorithm for a minimisation problem.

2.2. Differential Evolution (DE)

The Differential Evolution (DE) algorithm originally developed by Storn and Price [50] is a population-based evolutionary algorithm (EA). The overall idea of the algorithm is to use the

weighted differences of different individuals for generating new individuals in the search space. In DE, the solution-finding process is performed by three main operators, including mutation, cross-over, and selection operators.

Algorithm 1: Pseudo code of PSO algorithm

Initialise parameters n, $\boldsymbol{X}^{\min}$, $\boldsymbol{X}^{\max}$, N_p, c_1, c_2, ω_{damp};
Initialise the velocity vectors $\boldsymbol{V}_i^0$ as n-dimensional zero vectors;
Generate N_p particles randomly within the search space:
$\boldsymbol{X}_i^0 \sim \boldsymbol{U}(\boldsymbol{X}^{\min}, \boldsymbol{X}^{\max}), i = 1, 2, \dots, N_p$;
Initialise the personal experience of each particle as $\boldsymbol{P_i^0} = \boldsymbol{X_i^0}$;
Evaluate the objective function values of particles $f(\boldsymbol{X}_i^0)$;
Initialise the global experience vector $\boldsymbol{G}^0$ as follows: $\boldsymbol{G}^0 = arg\min\{f(\boldsymbol{X}_i^0)\}$;
Set $t = 0$;
Set $\omega^0 = 1$;
while *termination criteria are not met* **do**
    Set $\omega^{t+1} = \omega_{\text{damp}}\omega^t$;
    for $i \leftarrow 1$ **to** N_p **do**
        Update the velocity:
        $\boldsymbol{V}_i^{t+1} = \omega^{t+1}\boldsymbol{V}_i^t + c_1 rand(\boldsymbol{P}_i^t - \boldsymbol{X}_i^t) + c_2 rand(\boldsymbol{G}^t - \boldsymbol{X}_i^t)$;
        Update the position: $\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t + \boldsymbol{V}_i^{t+1}$;
        Evaluate the objective function value: $f(\boldsymbol{X}_i^{t+1})$;
        Update the personal experience:
        if $f(\boldsymbol{X}_i^{t+1}) < f(\boldsymbol{P}_i^t)$ **then**
            $\boldsymbol{P}_i^t = \boldsymbol{X}_i^{t+1}$;
            Update the global experience:
            if $f(\boldsymbol{X}_i^{t+1}) < f(\boldsymbol{G}^t)$ **then**
                $\boldsymbol{G}^t = \boldsymbol{X}_i^{t+1}$;
            end
        end
    end
    Set $\boldsymbol{G}^{t+1} = \boldsymbol{G}^t$;
    Set $t = t + 1$;
end

The DE kick-starts the optimisation process with N_{I} randomly generated individuals within the search space. For each solution $\boldsymbol{X}_i^t$, the algorithm applies the mutation operator to generate a mutant vector $\boldsymbol{V}_i^{t+1} = [v_{i,1}^{t+1}, v_{i,2}^{t+1}, \dots, v_{i,n}^{t+1}]$ in each iteration. These mutant vectors will be used to form a new generation of individuals. According to the literature, a variety of mutation operators

have been developed for the DE algorithm [51], in which the mutant vectors $\boldsymbol{V}_i^{t+1}$ are generated by weighted difference combinations of different individuals. The most popular mutation operators of DE are as follows [51,52]:

- "DE\best\1":

$$\boldsymbol{V}_i^{t+1} = \boldsymbol{X}_{\text{Best}}^t + F(\boldsymbol{X}_{r_1}^t - \boldsymbol{X}_{r_2}^t) \tag{3}$$

- "DE\rand-to-best\ 1":

$$\boldsymbol{V}_i^{t+1} = \boldsymbol{X}_{r_1}^t + F\left(\boldsymbol{X}_{\text{Best}}^t - \boldsymbol{X}_{r_2}^t\right) + F\left(\boldsymbol{X}_{r_3}^t - \boldsymbol{X}_{r_4}^t\right) \tag{4}$$

- "DE\current-to-rand\1"

$$\boldsymbol{V}_i^{t+1} = \boldsymbol{X}_{r_1}^t + F\left(\boldsymbol{X}_{r_2}^t - \boldsymbol{X}_i^t\right) + F\left(\boldsymbol{X}_{r_3}^t - \boldsymbol{X}_{r_4}^t\right) \tag{5}$$

- "DE\current-to-best\1":

$$\boldsymbol{V}_i^{t+1} = \boldsymbol{X}_{r_1}^t + F(\boldsymbol{X}_{\text{Best}}^t - \boldsymbol{X}_i^t) + F\left(\boldsymbol{X}_{r_2}^t - \boldsymbol{X}_{r_3}^t\right) \tag{6}$$

In equations (3-6), $\boldsymbol{V}_i^{t+1}$ represents the mutant vector constructed for the ith individual at iteration $t+1$, indexes $r_1, r_2, r_3, r_4 \in [1, 2, \dots, N_{\text{I}}]$ indicate the randomly generated numbers which must satisfy $r_1 \neq r_2 \neq r_3 \neq r_4 \neq i$, $\boldsymbol{X}_{\text{Best}}^t$ represents the individual with the best objective function value at iteration t, and F is the scaling factor which is usually assumed as a constant value.

The obtained mutant vectors $\boldsymbol{V}_i^{t+1}$ are different than their parent solutions $\boldsymbol{X}_i^t$ in all dimensions. However, changing current solutions in all dimensions may not provide good outcomes at all times. The main reason for this statement stems from the fact that the small changes in the solution vector can result in significant changes in objective function values. Hence, the DE tries to randomly keep the original values of some variables in the offspring solutions. To this end, the algorithm employs the cross-over operator to generate a new trial vector $\boldsymbol{U}_i^{t+1} = [u_{i,1}^{t+1}, u_{i,2}^{t+1}, \dots, u_{i,n}^{t+1}]$ based on the mutant vectors $\boldsymbol{V}_i^{t+1}$ obtained through one of the above-mentioned mutation operators as follows:

$$u_{i,k}^{t+1} = \begin{cases} v_{i,k}^{t+1} & \text{if } rand < CR \text{ or } k = randi(1,n) \\ x_{i,k}^t & \text{otherwise} \end{cases} \tag{7}$$

where $u_{i,k}^{t+1}$ denotes the kth variable of trial vector $\boldsymbol{U}_i^{t+1}$ constructed for the ith individual at iteration $t+1$, $v_{i,k}^{t+1}$ is the kth variable of mutant vector $\boldsymbol{V}_i^{t+1}$ obtained for the ith individual at iteration $t+1$, $rand$ is the random number between 0 and 1, CR is the cross-over rate control

which is usually considered as a constant value, and $randi(a,b)$ represents a random integer number between a and b.

The DE algorithm uses the selection operator to decide whether the trial vector $\boldsymbol{U}_i^{t+1}$ generated by the cross-over operator could be a member of the next generation or not. The selection operator can be expressed as follows [52]:

$$\boldsymbol{X}_i^{t+1} = \begin{cases} \boldsymbol{U}_i^{t+1} & \text{if } f(\boldsymbol{U}_i^{t+1}) < f(\boldsymbol{X}_i^t) \\ \boldsymbol{X}_i^t & \text{otherwise} \end{cases} \tag{8}$$

The Pseudo code of the DE algorithm for a minimisation problem is presented in Algorithm 2. In this study, we categorise the DE algorithm based on the applied mutation operators into different versions, including DE\best\1, DE\rand-to-best\1, DE\current-to-rand\1, and DE\current-to-best\1.

Algorithm 2: Pseudo code of DE algorithm

Initialise parameters n, $\boldsymbol{X}^{\min}$, $\boldsymbol{X}^{\max}$, N_{I}, CR, F;
Generate N_{I} individuals randomly within the search space:
$\boldsymbol{X}_i^0 \sim \boldsymbol{U}(\boldsymbol{X}^{\min}, \boldsymbol{X}^{\max}), i = 1, 2, \dots, N_{\mathrm{I}}$;
Evaluate the objective function values of individuals $f(\boldsymbol{X}_i^0)$;
Find the individual with the best objective function value $\boldsymbol{X}_{\mathrm{Best}}^0$ as follows:
$\boldsymbol{X}_{\mathrm{Best}}^0 = arg\min\{f(\boldsymbol{X}_i^0)\}$;
Set $t = 0$;
while *termination criteria are not met* **do**
  for $i \leftarrow 1$ **to** N_{I} **do**
    Select the random numbers as $r_1 \neq r_2 \neq r_3 \neq r_4 \neq i$;
    Generate the mutant vector $\boldsymbol{V}_i^{t+1}$ using one of equations (3-6)
    Find a random integer between 1 and n as follows: $j_{\mathrm{rand}} = randi(1, n)$;
    for $k \leftarrow 1$ **to** n **do**
      if $rand > CR$ or $j == j_{\mathrm{rand}}$ **then**
        $u_{i,k}^{t+1} = v_{i,k}^{t+1}$;
      else
        $u_{i,k}^{t+1} = x_{i,k}^t$;
      end
    end
    if $f(\boldsymbol{U}_i^{t+1}) < f(\boldsymbol{X}_i^t)$ **then**
      $\boldsymbol{X}_i^{t+1} = \boldsymbol{U}_i^{t+1}$;
    else
      $\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t$;
    end
  end
  $\boldsymbol{X}_{\mathrm{Best}}^{t+1} = arg\min\{f(\boldsymbol{X}_i^{t+1})\}$;
  Set $t = t + 1$;
end

2.3. Cultural Algorithms (CAs)

Cultural Algorithms (CAs) developed by Reynolds [11] in the 1990s is an EA inspired by the principles of human social evolution. In real human societies, culture can be viewed as a source of information exchanged between individuals, which can affect the behaviours of the individuals [53]. According to the bio-cultural evolution theory [54], the overall human evolutionary process is a combination of genetic and cultural evolutionary processes. CA was developed based on the bio-cultural evolutionary mechanism [11], which is computationally different from the conventional EAs. The algorithm has found interesting applications in different research areas, ranging from computer science to different branches of engineering [12,55-59].

In contrast to the conventional EAs which are based on a single population space, CA works with two population and belief spaces. These two spaces affect each other through some sorts of communication protocols. Like other EAs, the population space consists of a set of individuals who are potential solutions for the problem. The belief space records the cultural information gained by the individuals during the evolutionary process, which includes different types of knowledge components. The general Pseudo-code of the CA is illustrated in Algorithm 3. The algorithm consists of a population space $\wp^t$ and a belief space $\boldsymbol{\mathcal{B}}^t$, which interact each other using *Accept()*, *Update()*, and *Influence()* functions.

Algorithm 3: The general Pseudo code of CA [12]

Set $t = 0$;
Initialise $\wp^t$ and $\boldsymbol{\mathcal{B}}^t$
while *termination criteria are not met* **do**
    Set $t = t + 1$;
    Evaluate the fitness of individuals in $\wp^t$ using $Obj()$;
    Accept some elite individuals from $\wp^t$ using $Accept()$;
    Update $\boldsymbol{\mathcal{B}}^t$ using $Update()$
    Produce new generation $\wp^{t+1}$ using $influence()$
end

Let us assume $\boldsymbol{\mathcal{P}}^t$ be the population space, which is consisted of N_S individual solutions represented by $\boldsymbol{X}_i^t = [x_{i,1}^t, x_{i,2}^t, \dots, x_{i,n}^t]$, where $i = 1, 2, \dots, N_S$. As it was mentioned earlier, the belief space includes different knowledge components acquired by individuals. There are different kinds of knowledge components in the literature that could be used in CAs depending on the type of problem to be solved, including situational, normative, historical, topographical, and domain

knowledge components. In this study, it is assumed that the belief space consists of situational and normative knowledge components as follows:

$$\boldsymbol{\mathcal{B}}^t = \{\boldsymbol{\mathcal{S}}^t, \boldsymbol{\mathcal{N}}^t\} \tag{9}$$

where $\boldsymbol{\mathcal{B}}^t$, $\boldsymbol{\mathcal{S}}^t$, and $\boldsymbol{\mathcal{N}}^t$ are the belief space, situational knowledge, and normative knowledge, respectively. The situational knowledge component, which can be represented by $\boldsymbol{\mathcal{S}}^t = [s_1^t, s_2^t, \dots, s_n^t]$, contains the best solution obtained so far. The normative knowledge component $\boldsymbol{\mathcal{N}}^t$ is consisted of a set of information for each variable of the problem, which can be mathematically expressed as follows:

$$\boldsymbol{\mathcal{N}}^t = \begin{bmatrix} \boldsymbol{I}_1^t & \boldsymbol{I}_2^t & \dots & \boldsymbol{I}_n^t \\ L_1^t & L_2^t & \dots & L_n^t \\ U_1^t & U_2^t & \dots & U_n^t \end{bmatrix} \tag{10}$$

where $\boldsymbol{I}_k^t = \left[x_{\min,k}^t, x_{\max,k}^t\right]$ indicates the belief interval of the kth dimension of the problem, $x_{\min,k}^t$ and $x_{\max,k}^t$ are the lower and upper normative bounds for the kth dimension of the problem, respectively, L_k^t and U_k^t represent the objective function values corresponding to the lower and upper normative bounds, respectively.

The *Accept()* function of CA selects some elite individuals from the population space to update the belief space in each iteration. Usually, a given percentage of the individuals with better objective functions are selected to update the belief space. In this study, it is assumed that N_{Accept} of the population would be accepted to update the belief space.

The *Update()* function updates the different knowledge components of the belief space using the accepted individuals as follows:

$$\boldsymbol{\mathcal{S}}^{t+1} = \begin{cases} \boldsymbol{X}_l^t & \text{if} \quad f(\boldsymbol{X}_l^t) < f(\boldsymbol{\mathcal{S}}^t) \\ \boldsymbol{\mathcal{S}}^t & otherwise \end{cases} \tag{11}$$

$$x_{\min,k}^{t+1} = \begin{cases} x_{l,k}^t & \text{if } x_{l,k}^t \le x_{\min,k}^t \text{ or } f(\boldsymbol{X}_l^t) < L_k^t \\ x_{\min,k}^t & otherwise \end{cases} \tag{12}$$

$$x_{\max,k}^{t+1} = \begin{cases} x_{l,k}^t & \text{if } x_{l,k}^t \ge x_{\max,k}^t \text{ or } f(\boldsymbol{X}_l^t) < U_k^t \\ x_{max,k}^t & otherwise \end{cases} \tag{13}$$

$$L_k^{t+1} = \begin{cases} f(x_l^t) & \text{if } x_{l,k}^t \le x_{\min,k}^t \text{ or } f(\boldsymbol{X}_l^t) < L_k^t \\ L_k^t & otherwise \end{cases} \tag{14}$$

$$U_k^{t+1} = \begin{cases} f(x_l^t) & \text{if } x_{l,k}^t \ge x_{\max,k}^t \text{ or } f(\boldsymbol{X}_l^t) < U_k^t \\ U_k^t & otherwise \end{cases} \tag{15}$$

where $l = 1, 2, \dots, \%N_{\text{Accept}} \times N_{\text{S}}$, $\boldsymbol{X}_l^t$ is the lth accepted individual at iteration t and $x_{l,k}^t$ represents its kth variable.

Finally, the *Influence()* function generates a new generation of individuals based on cultural information. There are different influence functions developed for CA in literature. Based on the previous experience of authors, the following influence function is considered in this study:

$$x_{i,k}^{t+1} = \begin{cases} x_{i,k}^{t} + \left|size(\boldsymbol{I}_k^t) N_{i,k}(0,1)\right| & if\ x_{i,k}^{t} < x_{\min,k}^{t} \\ x_{i,k}^{t} - \left|size(\boldsymbol{I}_k^t) N_{i,k}(0,1)\right| & if\ x_{i,k}^{t} > x_{\max,k}^{t} \\ x_{i,k}^{t} + \beta size(\boldsymbol{I}_k^t) N_{i,k}(0,1) & \text{otherwise} \end{cases} \quad (16)$$

where $x_{i,k}^{t+1}$ is the kth variable of the ith individual at iteration $t+1$, $size(\boldsymbol{I}_k^t) = x_{\max,k}^{t} - x_{\min,k}^{t}$ is the size of the normative interval for the kth variable, $N_{i,k}(0,1)$ represents the random number generated by a normal distribution with the mean value of 0 and standard deviation of 1, and $\beta > 0$ is a constant parameter.

Algorithm 4 shows the detailed Pseudo code of CA. Interested readers are referred to Ref. [12] for more details on CA and its variants.

Algorithm 4: The detailed Pseudo code of CA

Initialise parameters $N_S, n, \boldsymbol{X}^{\min}, \boldsymbol{X}^{\max}, \beta, N_{\text{Accept}}$;
Initialise the normative knowledge $\boldsymbol{\mathcal{N}}^0$ as follows: $\boldsymbol{\mathcal{N}}^0 = \begin{bmatrix} \infty & \infty & \dots & \infty \\ \infty & \infty & \dots & \infty \\ \infty & \infty & \dots & \infty \end{bmatrix}$;
Initialise the empty situational knowledge vector $\boldsymbol{\mathcal{S}}^0$ and set $f(\boldsymbol{\mathcal{S}}^0) = \infty$;
Initialise the belief space as follows: $\boldsymbol{\mathcal{B}}^0 = \{\boldsymbol{\mathcal{S}}^0, \boldsymbol{\mathcal{N}}^0\}$;
Generate N_S individuals randomly within the search space as follows:
$\boldsymbol{X}_i^0 \sim \boldsymbol{U}(\boldsymbol{X}^{\min}, \boldsymbol{X}^{\max}), i = 1, 2, \dots, N_S$;
Evaluate the objective function values of individuals $f(\boldsymbol{X}_i^0)$;
Select $\%N_{\text{Accept}} \times N_S$ of individuals with better objective function values;
Update the belief space using equations (11-15)
Set $t = 0$;
while *termination criteria are not met* **do**
  for $i \leftarrow 1$ **to** N_S **do**
    for $k \leftarrow 1$ **to** n **do**
      if $x_{i,k}^{t} < x_{\min,k}^{t}$ **then**
      $x_{i,k}^{t+1} = x_{i,k}^{t} + \left|size(\boldsymbol{I}_k^t) N_{i,k}(0,1)\right|$;
      else if $x_{i,k}^{t} > x_{\max,k}^{t}$ **then**
      $x_{i,k}^{t+1} = x_{i,k}^{t} - \left|size(\boldsymbol{I}_k^t) N_{i,k}(0,1)\right|$;
      else
      $x_{i,k}^{t+1} = x_{i,k}^{t} + \beta size(\boldsymbol{I}_k^t) N_{i,k}(0,1)$
      end
    end
    Evaluate the objective function value $f(\boldsymbol{X}_i^{t+1})$;
  end
  Select $\%N_{\text{Accept}} \times N_S$ of individuals with better objective function values;
  Update the belief space using equations (11-15)
  Set $t = t + 1$;
end

2.4. Biogeography-Based Optimisation (BBO)

Biogeography-Based Optimisation (BBO) is another EA inspired by the emigration and immigration behaviours of different species in nature. It was originally developed by Simon [13] in 2008 based on the mathematical migration models in biogeography science [60] and has been found significant applications in different fields [61-64]. In nature, the biological species migrate from one habitat to another one to access new resources. The migration process of species follows given mathematical patterns. BBO inspires this concept to perform the optimisation process. BBO works with a population of habitats, in which each habitat is a potential solution for the problem. The algorithm employs two main operators to form a new generation of solutions, including migration and mutation operators.

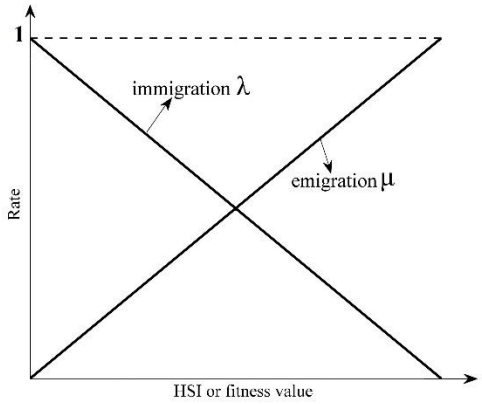

Fig. 1. Migration model in BBO [13]

Similar to other algorithms explained earlier, let us assume a set of N_{H} habitats represented by vectors $\boldsymbol{X}_i^t = [x_{i,1}^t, x_{i,2}^t, \dots, x_{i,n}^t]$, where $i = 1, 2, \dots, N_{\mathrm{H}}$. In BBO, the fitness function of each habitat is called as habitat suitability index (HSI), which means the habitats with high HSI values represent better solutions for the problem. In the migration operator of the BBO algorithm, two immigration and emigration rates are defined for each habitat based on the objective function values. Let λ_i and μ_i be the immigration and emigration rates for the ith habitat. The values of these rates for each habitat are obtained based on the migration models available from biogeography science. Fig. 1 shows a simple linear migration model which is usually used in literature to define the migration rates of BBO. As it can be seen from this figure, the sum of immigration and emigration rates for each habitat is equal to one (i.e., $\lambda_i + \mu_i = 1$). Based on Fig. 1, the habitats with higher (lower) fitness function values will have lower (larger) immigration rates λ_i and larger (lower) emigration rates μ_i. The BBO uses these rates to perform the migration

operator between the habitats in the search space. In BBO, the migration operator replaces the kth variable of ith habitat by the corresponding variable of the jth habitat as follows:

$$x_{i,k}^{t} \leftarrow x_{j,k}^{t} \tag{17}$$

where $x_{i,k}^{t}$ represents the kth variable of habitat i at iteration t and $x_{j,k}^{t}$ is the kth variable of habitat j at iteration t. It should be noted that the index j is selected based on the emigration rates μ_i and roulette wheel selection method. After performing the migration operator, the mutation operator randomly replaces the variables of habitats with a random value generated in the feasible search domain. The mutation probability for each habitat depends on the mutation rates m_i, which are defined based on the fitness values. The details for calculating mutation rates m_i are available in Ref. [13]. Algorithm 5 shows the detailed Pseudo code for the BBO algorithm.

Algorithm 5: Pseudo code of BBO algorithm

Initialise parameters n, $\boldsymbol{X}^{\min}$, $\boldsymbol{X}^{\max}$, N_{H};
Generate N_{H} habitats randomly within the search space as follows:
$\boldsymbol{X}_i^0 \sim \boldsymbol{U}(\boldsymbol{X}^{\min}, \boldsymbol{X}^{\max}), i = 1, 2, \ldots, N_{\mathrm{H}}$;
Set $t = 0$;
while *termination criteria are not met* **do**
  Evaluate the fitness function values of the habitats $f(\boldsymbol{X}_i^t)$;
  Calculate the immigration (λ_i) and the emigration (μ_i) rates for each habitat based on the immigration model in Fig. 1;
    for $i \leftarrow 1$ **to** N_{H} **do**
      $\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t$;
      for $k \leftarrow 1$ **to** n **do**
        % Migration operator:
        if $rand < \lambda_i$ **then**
          Select the habitat j based on the emigration rates μ_i and roulette wheel selection method;
          Update the kth variable of the ith habitat: $x_{i,k}^{t+1} \leftarrow x_{j,k}^{t}$
        end
        % Mutation operator:
        If $rand < m_i$ **then**
          Mutate the kth variable of the ith habitat as follows: $x_{i,k}^{t+1} = U(X_k^{\min}, X_k^{\max})$;
        end
      end
    end
    Set $t = t + 1$;
end

2.5. League Championship Algorithm (LCA)

League Championship Algorithm (LCA) developed by Kashan [38,65] is a population-based meta-heuristic algorithm inspired by the sporting competitions between the teams in sports leagues. The algorithm simulates the sporting league by assuming each solution candidate as a team that competes with other teams to provide the best possible solution for the problem. The position and fitness function of each team represent its formation and playing strength, respectively. LCA performs a match analysis to generate new formations for teams, which simulates the process performed by coaches to find a suitable formation for their teams in real sporting competitions. In LCA, each *week* can be assumed as equivalent to one optimisation iteration.

The solution finding process in LCA is similar to the championship process in sports leagues, in which different teams play in pairs based on a league schedule in a given week. The outcome of each match is determined based on the strengths of the teams. Then, the teams perform match analysis and change their formation to enhance their strengths for the next week. This process is repeated until the termination criteria are met.

Let $\boldsymbol{X}_i^t = [x_{i,1}^t, x_{i,2}^t, \dots, x_{i,n}^t]$ be the position of the ith team at week t, where $i = 1, 2, \dots, N_{team}$ and N_{team} is the number of teams. Let us also assume that the best formation experience with the best strength obtained by the ith team until week t is represented by $\boldsymbol{B}_i^t = [b_{i,1}^t, b_{i,2}^t, \dots, b_{i,n}^t]$. LCA employs a *single round-robin* schedule to determine the teams that should play against each other. Fig. 2 illustrates how different matches in each season are arranged in LCA between the teams. For a league with N_{team} number of teams, each season will be consisted of $N_{team} \times (N_{team} - 1)/2$ matches. LCA continues the league championship for S seasons, which will result in a total of $S \times (N_{team} - 1)$ weeks of contests.

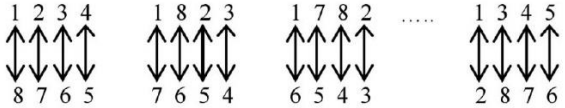

Fig. 2. League scheduling algorithm in LCA for the case of $N_{team} = 8$ [38]

When the teams play in pairs with each other, the outcome of the match for a given team can be a win, lose, or tie. To find the outcome of each match, LCA employs a probabilistic approach. Let us assume team i with the formation $\boldsymbol{X}_i^t$ plays against team j with the formation $\boldsymbol{X}_j^t$. Then, the winning probability of team i is expressed as follows:

$$p_i^t = \frac{f(\boldsymbol{X}_j^t) - \hat{f}}{f(\boldsymbol{X}_j^t) + f(\boldsymbol{X}_i^t) - 2\hat{f}} \tag{18}$$

where $\hat{f}$ is an arbitrary ideal value which is set to the best objective function value obtained by the teams until week t (i.e., $\hat{f} = min_{i=1,2,\dots,N_{team}} \{f(\boldsymbol{B}_i^t)\}$). In addition, the winning probability of team j against team i would be equal to $p_j^t = 1 - p_i^t$.

In real sporting competitions, coaches evaluate the strengths and weaknesses of their teams to improve their performance. Meanwhile, they need also to evaluate the opportunities and threats provided by the opponent teams. The coaches try to take the advantage of opportunities, while they have to monitor their opponents to protect their teams against possible external threats. The strengths and weaknesses are called internal factors, while the opportunities and threats are the external factors. To simulate these concepts, LCA performs a match analysis to update the formation of teams. Let us assume team i plays against team l based on the league schedule. The new formation of team i denoted by $\boldsymbol{X}_i^{t+1} = [x_{i,1}^{t+1}, x_{i,2}^{t+1}, \dots, x_{i,n}^{t+1}]$ depends on the previous experiences of both teams i and l at previous week t. Let j be the team that has played with team i at week t, and m be the team that has played with team l at week t. By considering these definitions, LCA updates the formation of team i as follows:

- If teams i and l both had won their opponents at previous week t:

$$x_{i,k}^{t+1} = b_{i,k}^t + y_{i,k}^t \left(\psi_1 rand\left(b_{i,k}^t - b_{m,k}^t\right) + \psi_1 rand\left(b_{i,k}^t - b_{j,k}^t\right)\right) \tag{19}$$

- If team i had won its previous match at week t, but team l was a loser at week t:

$$x_{i,k}^{t+1} = b_{i,k}^t + y_{i,k}^t \left(\psi_2 rand\left(b_{m,k}^t - b_{i,k}^t\right) + \psi_1 rand\left(b_{i,k}^t - b_{j,k}^t\right)\right) \tag{20}$$

- If team i had lost its previous match at week t, but team l was a winner at week t:

$$x_{i,k}^{t+1} = b_{i,k}^t + y_{i,k}^t \left(\psi_2 rand\left(b_{i,k}^t - b_{m,k}^t\right) + \psi_2 rand\left(b_{j,k}^t - b_{i,k}^t\right)\right) \tag{21}$$

- If teams i and l both had lost their matches at previous week t:

$$x_{i,k}^{t+1} = b_{i,k}^t + y_{i,k}^t \left(\psi_2 rand\left(b_{m,k}^t - b_{i,k}^t\right) + \psi_2 rand\left(b_{j,k}^t - b_{i,k}^t\right)\right) \tag{22}$$

In the above equations, $b_{i,k}^t$, $b_{m,k}^t$, and $b_{j,k}^t$ represent the kth variable of the best formation experienced by teams i, m and j until week t, respectively, ψ_1 is the approach coefficient that controls the acceleration of the team i toward the winner, and ψ_2 is the retreat coefficient that controls the retract team i from the loser. In equations (19-22), $y_{i,k}^t$ is a binary variable that

determines whether the kth variable of vector $\boldsymbol{B}_i^t$ should be changed or not. Let $\boldsymbol{Y}_i^t = [y_{i,1}^t, y_{i,2}^t, \dots, y_{i,n}^t]$ be the binary change array, in which the unit (zero) value for a given element of the array means the corresponding variable in $\boldsymbol{B}_i^t$ would (would not) change. LCA uses a truncated geometric distribution [66] to calculate the number of ones in array $\boldsymbol{Y}_i^t$, represented by q_i^t, as follows:

$$q_i^t = \left\lceil \frac{ln(1-(1-(1-p_c)^{n-q_0+1})rand)}{ln(1-p_c)} \right\rceil + q_0 - 1 \quad (23)$$

where p_c is a control parameter and q_0 is the lower bound for q_i^t. q_0 is typically taken as 1 [38]. The greater values for the parameter p_c will result in smaller changes in vector $\boldsymbol{B}_i^t$ and vice versa.

Algorithm 6 shows the detailed Pseudo code of LCA.

2.6. Optics Inspired Optimisation (OIO)

Optics Inspired Optimisation (OIO), recently developed by Kashan [44], is a meta-heuristic algorithm inspired by the optical characteristics of spherical mirrors in physics. In OIO, each solution vector is modelled as an artificial light point and the surface of the objective function is assumed as a spherical mirror that reflects the incident ray based on the governing equations in optics.

Let us assume NO light points represented by vectors $\boldsymbol{X}_i^t = [x_{i,1}^t, x_{i,2}^t, \dots, x_{i,n}^t]$ for a minimisation problem. For each light point $\boldsymbol{X}_i^t$, OIO randomly selects another solution represented by $\boldsymbol{X}_j^t = [x_{j,1}^t, x_{j,2}^t, \dots, x_{j,n}^t]$ from the population as an artificial mirror in which $j = 1, 2, \dots, NO$ and $j \neq i$. Depending on the objective function values of light points i and j, the solution $\boldsymbol{X}_j^t$ can be treated as a convex or concave mirror for light point $\boldsymbol{X}_i^t$. If $f(\boldsymbol{X}_i^t) > f(\boldsymbol{X}_j^t)$, then $\boldsymbol{X}_j^t$ would be a concave mirror for the light point $\boldsymbol{X}_i^t$. Otherwise, $\boldsymbol{X}_j^t$ would be a convex mirror. OIO finds the image position of the artificial light point $\boldsymbol{X}_i^t$ formed by the artificial mirror $\boldsymbol{X}_j^t$ as follows:

$$\boldsymbol{I}_i^t = \boldsymbol{X}_j^t - \frac{r_j^t}{2p_{i,j}^t - r_j^t}\left(\boldsymbol{X}_i^t - \boldsymbol{X}_j^t\right) \quad (24)$$

where $\boldsymbol{I}_i^t$ represents the image position of the artificial light point $\boldsymbol{X}_i^t$, $p_{i,j}^t$ indicates the distance between the artificial light point i and the artificial mirror j on the objective function axis, and r_j^t is the curvature radius of the artificial mirror $\boldsymbol{X}_j^t$. The parameter $p_{i,j}^t$ is defined as follows:

$$p_{i,j}^t = s_{i,j}^t - f(\boldsymbol{X}_j^t) \quad (25)$$

Algorithm 6: The detailed Pseudo code of LCA

Initialise parameters n, $\boldsymbol{X}^{\min}$, $\boldsymbol{X}^{\max}$, N_{team}, ψ_1, ψ_2, q_0, p_c;
Set $t = 1$;
Generate formations for N_{team} teams randomly within the search space:
$\boldsymbol{X}_i^1 \sim \boldsymbol{U}(\boldsymbol{X}^{\min}, \boldsymbol{X}^{\max}), i = 1, 2, \ldots, N_{\text{team}}$;
Evaluate the playing strengths of teams $f(\boldsymbol{X}_i^1)$;
Set the current formations of teams as their best formations: $\boldsymbol{B}_i^1 = \boldsymbol{X}_i^1$;
Generate the league schedule based on the single round-robin method as displayed in Fig. 2;
while *termination criteria are not met* **do**
  for $t \leftarrow 1$ **to** $N_{\text{team}} - 1$ **do**
    for $i \leftarrow 1$ **to** N_{team} **do**
      Find the team j that has played with team i at week t;
      Find the opponent team l based on the league schedule at week t;
      Find the team m that has played with team l at week t;
      Calculate the winning probability of team i as follows: $p_i^t = \frac{f(\boldsymbol{X}_j^t) - \hat{f}}{f(\boldsymbol{X}_j^t) + f(\boldsymbol{X}_i^t) - 2\hat{f}}$
      if $rand < p_i^t$ **then**
        The team i is the winner and the team l is the loser;
      else
        The team i is the loser and the team l is the winner;
      end
      for $k \leftarrow 1$ **to** n **do**
        if *teams i and l both had won their opponents at week t* **then**
          $x_{i,k}^{t+1} = b_{i,k}^t + y_{i,k}^t \left(\psi_1 rand(b_{i,k}^t - b_{m,k}^t) + \psi_1 rand(b_{i,k}^t - b_{j,k}^t)\right)$;
        else if *team i had won its previous match at week t, but team l was a loser at week t* **then**
          $x_{i,k}^{t+1} = b_{i,k}^t + y_{i,k}^t \left(\psi_2 rand(b_{m,k}^t - b_{i,k}^t) + \psi_1 rand(b_{i,k}^t - b_{j,k}^t)\right)$;
        else if *team i had lost its previous match at week t, but team l was a winner at week t* **then**
          $x_{i,k}^{t+1} = b_{i,k}^t + y_{i,k}^t \left(\psi_2 rand(b_{i,k}^t - b_{m,k}^t) + \psi_2 rand(b_{j,k}^t - b_{i,k}^t)\right)$;
        else
          $x_{i,k}^{t+1} = b_{i,k}^t + y_{i,k}^t \left(\psi_2 rand(b_{m,k}^t - b_{i,k}^t) + \psi_2 rand(b_{j,k}^t - b_{i,k}^t)\right)$;
        end
      end
    end
    Evaluate the playing strength of team i, $f(\boldsymbol{X}_i^{t+1})$;
    if $f(\boldsymbol{X}_i^{t+1}) < f(\boldsymbol{B}_i^t)$ **then**
      $\boldsymbol{B}_i^{t+1} = \boldsymbol{X}_i^{t+1}$;
    else
      $\boldsymbol{B}_i^{t+1} = \boldsymbol{B}_i^t$;
    end
  end
  Set $t = t + 1$;
end

where $s_{i,j}^t$ is the position of artificial light point i on the objective function axis. If the artificial mirror is concave, $s_{i,j}^t$ is randomly selected as $U[f(\boldsymbol{X}_i^t), f(\boldsymbol{X}_i^t) + d_\infty]$, where $U[a, b]$ represents a random value between a and b, and d_∞ indicates the physical infinity that can be any positive value. In OIO, the value of physical infinity is initially assumed as $d_\infty = |maxf(\boldsymbol{X}_l^t)|_{l=1,2,\ldots,NO}$. The physical infinity d_∞ would be updated in the artificial spherical aberration stage of the algorithm. For the case of a convex mirror, $s_{i,j}^t$ is randomly assigned as $U\left[f\left(\boldsymbol{X}_j^t\right), f\left(\boldsymbol{X}_j^t\right) + d_\infty\right]$.

The curvature radius of the artificial mirror $\boldsymbol{X}_j^t$ is calculated as follows:

$$r_j^t = m_j^t - f\left(\boldsymbol{X}_j^t\right) \tag{26}$$

In this equation, m_j^t is the position of the centre of curvature for artificial mirror j, which is randomly determined as $U[f(\boldsymbol{X}_i^t), f(\boldsymbol{X}_i^t) + d_\infty]$ for concave mirrors and $U[f(\boldsymbol{X}_i^t) - d_\infty, f(\boldsymbol{X}_i^t)]$ for convex mirrors.

The image vector $\boldsymbol{I}_i^t$ represents a new solution for the problem, which differs from the artificial light point $\boldsymbol{X}_i^t$ in all dimensions. OIO constructs a new light point (or solution) $\boldsymbol{X}_i^{t+1}$ by keeping the number of changes in vector $\boldsymbol{X}_i^t$ less than n. Let us assume $\boldsymbol{U}_i^t \leftarrow \boldsymbol{X}_i^t$. OIO uses equation (23) to determine the number of changes q_i^t. Then, q_i^t number of variables are randomly selected from $\boldsymbol{I}_i^t$, and their values are assigned to their corresponding variables in vector $\boldsymbol{U}_i^t$. Finally, if the objective function of $\boldsymbol{U}_i^t$ is better than $\boldsymbol{X}_i^t$, then $\boldsymbol{U}_i^t$ would be the new artificial light point, i.e., $\boldsymbol{X}_i^{t+1} = \boldsymbol{U}_i^t$. Otherwise, the algorithm keeps the previous light point for the next iteration, i.e., $\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t$.

It should be noted that the value of m_j^t should satisfy a certain condition. In some cases, the difference between the objective function values of artificial light point i and the artificial mirror j is significantly high. In these cases, the image of light point i on artificial mirror j would be a blurry image, which can cause premature convergence of OIO from the numerical viewpoint.

In OIO, to avoid the spherical aberration phenomenon, the algorithm repeatedly updates the value of the parameter m_j^t. The interested readers are referred to Ref. [44] for more details on the spherical aberration mechanism of OIO. Algorithm 7 shows the full detailed Pseudo code of the OIO algorithm.

Algorithm 7: The detailed Pseudo code of OIO

Initialise parameters n, $\boldsymbol{X}^{\min}$, $\boldsymbol{X}^{\max}$, NO, q_0, p_c;
Generate NO artificial light points randomly within the search space as follows:
$\boldsymbol{X}_i^0 \sim \boldsymbol{U}(\boldsymbol{X}^{\min}, \boldsymbol{X}^{\max}), i = 1, 2, \dots, NO$;
Evaluate the objective function values for each artificial light point $f(\boldsymbol{X}_i^0)$;
Set $t = 0$;
while *termination criteria are not met* **do**
  for $t \leftarrow 1$ **to** NO **do**
    Select the solution vector $\boldsymbol{X}_j^t$ from the population as the artificial mirror, where $j = 1, 2, \dots, NO$ and $j \neq i$;
    % Determine the type of the mirror:
    if $f(\boldsymbol{X}_i^t) > f(\boldsymbol{X}_j^t)$ **then**
      Assume $\boldsymbol{X}_j^t$ as a concave mirror;
      $s_{i,j}^t = U[f(\boldsymbol{X}_i^t), f(\boldsymbol{X}_i^t) + d_\infty]$;
      $m_j^t = U[f(\boldsymbol{X}_i^t), f(\boldsymbol{X}_i^t) + d_\infty]$;
    else
      Assume $\boldsymbol{X}_j^t$ as a convex mirror;
      $s_{i,j}^t = U[f(\boldsymbol{X}_j^t), f(\boldsymbol{X}_j^t) + d_\infty]$;
      $m_j^t = U[f(\boldsymbol{X}_i^t) - d_\infty, f(\boldsymbol{X}_i^t)]$;
    end
    Determine the image position of light point i as follows: $p_{i,j}^t = s_{i,j}^t - f(\boldsymbol{X}_j^t)$;
    Determine the radius of curvature of the mirror as follows: $r_j^t = m_j^t - f(\boldsymbol{X}_j^t)$;
    % Correct the spherical aberration:
    while $\frac{(r_j^t)^2}{2\sqrt{(r_j^t)^2 - (\|\boldsymbol{x}_i^t - \boldsymbol{x}_j^t\|)^2}} - \frac{|r_j^t|}{2} > 0.01$ *or* $\|\boldsymbol{X}_i^t - \boldsymbol{X}_j^t\| > |r_j^t|$ **do**
      Set $d_\infty = 2d_\infty$;
      if *artificial mirror is concave* **then**
        $m_j^t = m_j^t + d_\infty$;
      else
        $m_j^t = m_j^t - d_\infty$;
      end
      $r_j^t = m_j^t - f(\boldsymbol{X}_j^t)$;
    end
    % Generate the image position of the artificial light point $\boldsymbol{X}_i^t$:
    $\boldsymbol{I}_i^t = \boldsymbol{X}_j^t - \frac{r_j^t}{2p_{i,j}^t - r_j^t}(\boldsymbol{X}_i^t - \boldsymbol{X}_j^t)$;
    Set $\boldsymbol{U}_i^t = \boldsymbol{X}_i^t$;
    $q_i^t = \left\lceil \frac{ln(1 - (1 - (1 - p_c)^{n - q_0 + 1})rand)}{ln(1 - p_c)} \right\rceil + q_0 - 1$;
    Randomly select q_i^t number of variables from $\boldsymbol{I}_i^t$, and assign their values to their corresponding variables in $\boldsymbol{U}_i^t$;
    if $f(\boldsymbol{U}_i^t) < f(\boldsymbol{X}_i^t)$ **then**
      $\boldsymbol{X}_i^{t+1} = \boldsymbol{U}_i^t$;
    else
      $\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t$;
    end
  end
  Set $t = t + 1$;
end

3. Problem Formulation

As it was mentioned earlier, buckling capacity maximisation for optimum layup of laminated composite plates is an important problem that has been widely investigated by researchers. Laminated composite plates subjected to in-plane compressive loads are potentially susceptible to lose their stability. Hence, the buckling capacity of laminated composites under in-plane compressive loads should be considered in the design process. Let us assume a simply supported laminated composite plate shown in Fig. 3 with dimensions a and b in x and y directions, respectively, in which N_x and N_y are the in-plane compressive loads in x and y directions, respectively. According to the classical laminated plate theory, the buckling load factor can be expressed as follows [67]:

$$\lambda_b(p,q) = \pi^2 \left[\frac{p^4 D_{11} + 2(D_{12} + 2D_{66})(rpq)^2 + (rq)^4 D_{22}}{(ap)^2 N_x + (raq)^2 N_y} \right] \tag{27}$$

where λ_b is the buckling load factor, r is the aspect ratio which represents the ratio of length to width, D_{ij} indicates the bending stiffness of composite plate, a and b are the dimensions of the laminate in *x* and *y* directions, respectively, p and q are the half-waves in the *x* and *y* directions, respectively, N_x and N_y are the in-plane compressive loads in *x* and *y* directions, respectively. It should be noted that the parameters D_{16} and D_{26} are neglected in Eq. (27), as they are zero for specially orthotropic laminates and very small for the symmetrically balanced laminates. Thus, the buckling load factor obtained by Eq. (27) is exact for specially orthotropic laminates and approximately correct for the symmetrically balanced laminates. It is clear from Eq. (27) that the buckling load factor is a function of the material property, length, width, stacking sequence, applied in-plane compressive loads, and value of parameters p and q. It is worth mentioning that various values for parameters p and q will result in different buckling load factors. The smallest buckling load factor yielded by Eq. (27) for different values of parameters p and q will be the critical buckling load factor for the laminated composite plate.

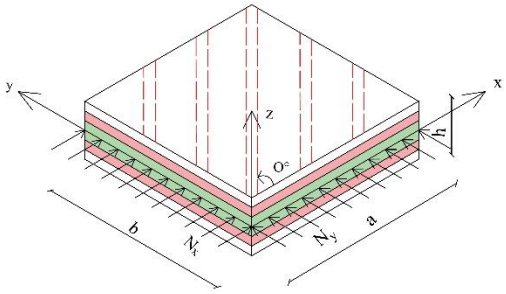

Fig. 3. Laminated composite plate

The optimum stacking sequence problem of the laminated composite plates under in-plane compressive loads for maximum buckling capacity can be mathematically expressed as follows:

$$\begin{aligned} &\text{Find } \boldsymbol{X} = [x_1, x_2, \dots, x_n] \\ &\text{To maximise: } f(\boldsymbol{X}) = \lambda_b \\ &\text{Subjected to:} \\ &x_i \epsilon \, \boldsymbol{S} = [s_1, s_2, \dots, s_p] \end{aligned} \tag{28}$$

where $f(.)$ represents the objective function of the problem, $\boldsymbol{X} = [x_1, x_2, \dots, x_n]$ indicates the vector of design variables, x_i is the fibre orientation for the ith ply which should be selected from a given discrete set, n is the number of design variables, $\boldsymbol{S}$ is the vector containing the allowable fibre orientations, and p is the number of allowable fibre orientations. It should be noted that for a composite laminate with m layers, the number of design variables will be equal to $m/2$ due to symmetry (i.e., $n = m/2$).

4. Numerical Results

In this section, the performance of different meta-heuristics in the buckling maximisation of a 64-layer laminated composite plate will be investigated. The plate is assumed as a symmetrically balanced laminated composite with simply supported boundary conditions. The design variables are the stacking sequences of layers, which should be selected from the discrete set of $[0_2,\ \pm15,\ \pm30,\ \pm45,\ \pm60, \pm75,\ 90_2]$. In previous studies, possible fibre orientations were taken as $[0_2, \pm45, 90_2\ ]$ [27,31]. However, this study considers more possible fibre orientations to make the problem more challenging for the algorithms. It is assumed that the laminated composite plate is made of graphite/epoxy material with mechanical properties presented in Table 1. The length of the plate is assumed to be 0.508 m with various aspect ratios and load cases listed in Table 2.

Table 1. Mechanical properties of the graphite/epoxy [27].

Property	Graphite/epoxy
Young's modulus, E_1(GPa)	127.55
Young's modulus, E_2(GPa)	13.03
Shear modulus, G_{12} (GPa)	6.41
Poisson's ratio, ν_{12}	0.3
Lamina thickness, t (mm)	0.127

Table 2. Different load cases, lengths, and widths considered for 64-layer laminated composite plate

Load cases	a (m)	b (m)	N_x	N_y
LC1	0.508	0.254	1	1
LC2	0.508	0.508	1	1
LC3	0.508	1.016	1	1
LC4	0.508	0.254	1	0.5
LC5	0.508	0.508	1	0.5
LC6	0.508	1.016	1	0.5
LC7	0.508	0.254	1	2
LC8	0.508	0.508	1	2
LC9	0.508	1.016	1	2

4.1. Internal parameters

According to Section 2, the algorithms have a set of internal parameters which can significantly affect their performance. To find the best possible values for these parameters, several performance sensitivity analyses have been performed by considering different values for internal parameters. The sensitivity analyses were performed based on a trial and error approach. Table 3 lists the possible values for internal parameters of each algorithm alongside their suitable values obtained from the sensitivity analyses. From different combinations of possible values, the suitable parameter values were selected based on the best performance exhibited by each algorithm. To keep the article in a manageable size, only the final results of sensitivity analyses were presented in Table 3. In this study, the obtained internal parameters from the sensitivity analyses will be used to assess the performance of different algorithms.

Table 3. Internal parameters of different algorithms and their appropriate values obtained from the sensitivity analyses

Algorithm	Possible values	Selected values from sensitivity analysis
PSO	$N_p \epsilon \{20, 30, 40\}$	$N_p = 40$
	$c_1, c_2 \epsilon \{1, 2\}$	$c_1 = 2, c_2 = 1$
	$\omega_{\text{damp}} \epsilon \{0.95, 0.99\}$	$\omega_{\text{damp}} = 0.99$
DE\best\1	$N_I \epsilon \{20, 30, 40\}$	$N_I = 40$
	$CR \epsilon \{0.8, 0.9\}$	$CR = 0.8$
	$F \epsilon \{0.5, 0.7, 0.9\}$	$F = 0.7$
DE\rand-to-best\1	$N_I \epsilon \{20, 30, 40\}$	$N_I = 20$
	$CR \epsilon \{0.8, 0.9\}$	$CR = 0.9$
	$F \epsilon \{0.5, 0.7, 0.9\}$	$F = 0.5$
DE\current-to-rand\1	$N_I \epsilon \{20, 30, 40\}$	$N_I = 20$
	$CR \epsilon \{0.8, 0.9\}$	$CR = 0.8$
	$F \epsilon \{0.5, 0.7, 0.9\}$	$F = 0.5$
DE\current-to-best\1	$N_I \epsilon \{20, 30, 40\}$	$N_I = 20$
	$CR \epsilon \{0.8, 0.9\}$	$CR = 0.8$
	$F \epsilon \{0.5, 0.7, 0.9\}$	$F = 0.5$
CA	$N_S \epsilon \{20, 30, 40\}$	$N_S = 40$
	$N_{\text{Accept}} \epsilon \{0.1, 0.2, 0.3, 0.4\}$	$N_{\text{Accept}} = 0.2$
	$\beta \epsilon \{0.1, 0.2, 0.3, 0.4, 0.5\}$	$\beta = 0.1$
BBO	$N_H \epsilon \{20, 30, 40\}$	$N_H = 40$
OIO	$NO \epsilon \{9, 21, 30\}$	$NO = 30$
	$p_c \epsilon \{0.0001, 0.5, 0.999\}$	$p_c = 0.0001$
LCA	$N_{\text{team}} \epsilon \{10, 20, 30\}$	$N_{\text{team}} = 20$
	$\psi_1, \psi_2 \epsilon \{1.\ 1.5, 2\}$	$\psi_1 = \psi_2 = 1$
	$p_c \epsilon \{0.0001, 0.5, 0.999\}$	$p_c = 0.0001$

4.2. Performance comparisons

To investigate the performance of different algorithms in optimum design of 64-layer laminated composite, 5000 function evaluations was defined as the termination criterion. The algorithms were repeated for 30 independent runs, and their best, average, and worst results alongside the standard deviations for different load cases were reported.

For different load cases, Tables 4-12 present the optimum stacking sequences corresponding to the maximum buckling load factors obtained by each algorithm alongside the best, mean, standard deviation, and worst results. At first glance, it is observable from the results in Tables 4-12 that none of the algorithms is capable of exhibiting better performance than others in all load cases and the efficiency of each algorithm differs from a given load case to another one. The reason behind this contradiction can be explained based on the "No Free Lunch" theorem [68] which states that it is almost impossible to develop a general strategy to solve different problem types in an equally efficient manner. With this introductory statement, the performance of the algorithms will be investigated in more detail in this section to find out which algorithms are capable of providing the most promising results for the different load cases of the laminated composite layup optimisation problem.

From Tables 4-12, it can be seen that the algorithms exhibit quite similar performances for the second, fifth, and eighth load cases, whereas their performances seem to be different for other load cases. The numerical results in Tables 4-12 can be interpreted in different ways. In terms of the best results, for the first load case, it is turn out that PSO, CA, OIO, and LCA can find the maximum buckling load factor. LCA found better best solution than all other algorithms in the third case. On the other hand, PSO and BBO exhibit better performances than others for the fourth load case in terms of the best solution. Moreover, the best solutions obtained by PSO, BBO, and LCA are better than those yielded by other algorithms in the ninth load case. For the rest of the load cases, all algorithms were able to find the optimum buckling load factors.

If the results reported in Tables 4-12 are compared in terms of standard deviations, it can be seen that the standard deviations yielded by BBO, OIO, and LCA algorithms for the second, fifth, and eighth load cases are equal to zero, which indicate that these algorithms are capable of finding the optimum solutions in each independent run. Comparison of the standard deviations yielded by different algorithms for other load cases suggests that LCA and OIO algorithms are capable of providing the lowest standard deviations for the almost rest of the load cases except the seventh

load case, which make their performance more impressive than others. Comparing with other algorithms, the efficiency of LCA and OIO may stem from the fact that their search operators can keep the diversity of the population at a high level to avoid premature convergence to the local optimum points in the search space. Although the DE\current-to-rand\1 algorithm yielded the lowest standard deviation for the seventh load case, the LCA and OIO are still competitive in this load case as well.

The worst results obtained from 30 independent runs for each algorithm is also another important criterion, which can show how the algorithms are capable of generating better results in the worst-case scenarios. For the second, fifth, and eighth load cases, the BBO, OIO, and LCA performed better than other algorithms in terms of worst results. For the rest of the load cases except the seventh and last load cases, LCA obtained better worst results than all other algorithms. However, DE\current-to-rand\1 and OIO algorithms provided better worst buckling load factors for the seventh and last load cases, respectively.

Table 4. Optimum stacking sequences and statistical results obtained by different algorithms for load case 1

Algorithm	Statistical results				Stacking Sequence
	Best	Mean	Std	Worst	
PSO	720616.44	720489.75	134.23	720093.61	$[\pm75_6/\pm60/\pm75_2/\pm60_5/\pm75/\pm60]_s$
DE\best\1	720498.49	718577.92	1658.90	712666.84	$[\pm75_3/\pm60/\pm75_6/\pm60_2/90_2/\pm75_2/\pm45]_s$
DE\rand-to-best\1	720493.99	719645.94	509.35	718451.13	$[\pm75_5/(\pm60,\pm75)_2/\pm75_2/\pm60/\pm75_2/90_2/\pm15]_s$
DE\current-to-rand\1	720503.07	719554.38	535.65	717826.67	$[\pm75_4/\pm60/\pm75_3/\pm60/\pm75_2/\pm60/90_2/\pm75/90_4]_s$
DE\current-to-best\1	720381.08	719546.57	638.74	718158.63	$[\pm75_6/\pm60/\pm75/\pm60_2/\pm75/\pm60/\pm75/90_6]_s$
CA	720616.44	719891.49	719.38	717953.80	$[\pm75_6/\pm60/\pm75_2/\pm60_5/\pm75/\pm60]_s$
BBO	720602.77	719355.15	854.85	717381.11	$[\pm75_6/\pm60_2/(\pm75_2,\pm60)_2/\pm60/\pm45]_s$
OIO	720616.44	720482.28	140.91	720077.26	$[\pm75_6/\pm60/\pm75_2/\pm60_5/\pm75/\pm60]_s$
LCA	720616.44	720578.74	29.63	720521.13	$[\pm75_6/\pm60/\pm75_2/\pm60_5/\pm75/\pm60]_s$

Table 5. Optimum stacking sequences and statistical results obtained by different algorithms for load case 2

Algorithm	Statistical results				Stacking Sequence
	Best	Mean	Std	Worst	
PSO	242823.08	242814.40	27.83	242715.57	$[\pm45_{16}]_s$
DE\best\1	242823.08	242187.80	857.37	239382.59	$[\pm45_{16}]_s$
DE\rand-to-best\1	242823.08	242745.37	196.58	241985.59	$[\pm45_{16}]_s$
DE\current-to-rand\1	242823.08	242822.33	4.13	242800.45	$[\pm45_{16}]_s$
DE\current-to-best\1	242823.08	242785.36	111.73	242393.02	$[\pm45_{16}]_s$
CA	242823.08	242822.89	1.03	242817.42	$[\pm45_{16}]_s$
BBO	242823.08	242823.08	0.00	242823.08	$[\pm45_{16}]_s$
OIO	242823.08	242823.08	0.00	242823.08	$[\pm45_{16}]_s$
LCA	242823.08	242823.08	0.00	242823.08	$[\pm45_{16}]_s$

Table 6. Optimum stacking sequences and statistical results obtained by different algorithms for load case 3

Algorithm	Statistical results				Stacking Sequence
	Best	Mean	Std	Worst	
PSO	180150.69	180101.05	73.71	179813.21	$[\pm15_5/\pm30/\pm15_3/(\pm30_2,\pm15)_2/\pm45]_s$
DE\best\1	180140.92	179737.47	263.91	179004.20	$[\pm15_7/\pm30_3/\pm15/\pm30_2/\pm15/0_2/\pm30]_s$
DE\rand-to-best\1	180124.62	179911.88	144.75	179546.03	$[\pm15_4/(\pm30,\pm15_3)_2/\pm30/\pm15_2/\pm60]_s$
DE\current-to-rand\1	180109.63	179863.70	138.42	179445.58	$[\pm15/(\pm15_3,\pm30)_2/\pm15_2/\pm30/0_2/\pm15_2/\pm30]_s$
DE\current-to-best\1	180087.22	179851.20	176.28	179154.11	$[\pm15_5/(\pm30,\pm15)_2/\pm15/\pm30/\pm15/0_4/\pm15/0_2]_s$
CA	180150.69	179949.77	188.70	179490.83	$[\pm15_4/\pm30/\pm15_5/\pm30_4/\pm15/\pm45]_s$
BBO	180147.01	179916.47	187.43	179452.23	$[\pm15_7/\pm30_3/\pm15/\pm30_2/\pm15/0_4]_s$
OIO	180149.46	180110.70	40.01	179977.96	$[\pm15_2/\pm30/\pm15_8/\pm30/\pm15_4]_s$
LCA	180150.90	180140.06	15.63	180073.69	$[\pm15_2/\pm30/\pm15_8/\pm30/\pm15_3/0_2]_s$

Table 7. Optimum stacking sequences and statistical results obtained by different algorithms for load case 4

Algorithm	Statistical results				Stacking Sequence
	Best	Mean	Std	Worst	
PSO	1119540.28	1119250.08	318.47	1117991.76	$[\pm 60_4/\pm 75/\pm 60/(\pm 75,\pm 60_2)_2/\pm 75_3/90_2]_s$
DE\best\1	1119452.35	1118513.39	1308.49	1112891.38	$[\pm 60_5/(\pm 75,\pm 60)_2/\pm 75_3/\pm 60/\pm 75/90_4]_s$
DE\rand-to-best\1	1119445.95	1118916.13	316.33	1118355.16	$[\pm 60_4/\pm 75/\pm 60_2/(\pm 75,\pm 60)_2/\pm 75_3/\pm 60/\pm 45]_s$
DE\current-to-rand\1	1119437.74	1118883.15	347.43	1118123.88	$[\pm 60_4/(\pm 75,\pm 60)_2/(\pm 60,\pm 75)_2/\pm 60_4]_s$
DE\current-to-best\1	1119441.92	1118298.54	1798.76	1109502.65	$[\pm 60_4/\pm 75_2/\pm 60_5/\pm 75_2/90_2/\pm 75/\pm 45]_s$
CA	1119530.70	1118057.95	1946.47	1112408.51	$[\pm 60_4/\pm 75/\pm 60_2/\pm 75/\pm 60/\pm 75_2/\pm 60_2/\pm 75_3]_s$
BBO	1119540.28	1119249.54	244.54	1118452.16	$[\pm 60_4/\pm 75/\pm 60/(\pm 75,\pm 60_2)_2/\pm 75_3/\pm 90_2]_s$
OIO	1119530.70	1119363.52	181.27	1118716.08	$[\pm 60_4/\pm 75/\pm 60_2/(\pm 75,\pm 60)_2/\pm 75_3/\pm 60/\pm 75]_s$
LCA	1119514.33	1119461.82	66.75	1119193.36	$[\pm 75/\pm 60_8/\pm 75/\pm 60_4/90_4]_s$

Table 8. Optimum stacking sequences and statistical results obtained by different algorithms for load case 5

Algorithm	Statistical results				Stacking Sequence
	Best	Mean	Std	Worst	
PSO	323764.11	323759.08	26.16	323620.75	$[\pm 45_{16}]_s$
DE\best\1	323764.11	323568.19	269.84	322949.26	$[\pm 45_{16}]_s$
DE\rand-to-best\1	323764.11	323761.09	9.21	323733.93	$[\pm 45_{16}]_s$
DE\current-to-rand\1	323764.11	323755.31	38.83	323552.85	$[\pm 45_{16}]_s$
DE\current-to-best\1	323764.11	323723.62	145.07	323190.69	$[\pm 45_{16}]_s$
CA	323764.11	323758.83	28.93	323605.67	$[\pm 45_{16}]_s$
BBO	323764.11	323764.11	0.00	323764.11	$[\pm 45_{16}]_s$
OIO	323764.11	323764.11	0.00	323764.11	$[\pm 45_{16}]_s$
LCA	323764.11	323764.11	0.00	323764.11	$[\pm 45_{16}]_s$

Table 9. Optimum stacking sequences and statistical results obtained by different algorithms for load case 6

Algorithm	Statistical results				Stacking Sequence
	Best	Mean	Std	Worst	
PSO	208148.56	208132.26	17.15	208092.62	$[0_4/(\pm 15,0_2)_2/0_{10}/\pm 15_3/0_4]_s$
DE\best\1	208148.56	207918.78	1024.87	202532.00	$[0_2/\pm 15/0_2/(0_6,\pm 15)_2/\pm 15/0_4/\pm 15/0_2]_s$
DE\rand-to-best\1	208148.56	208144.68	7.93	208109.91	$[0_2/\pm 15/0_2/(0_6,\pm 15)_2/\pm 15/0_4/\pm 15/0_2]_s$
DE\current-to-rand\1	208148.56	208145.36	7.46	208109.91	$[0_2/\pm 15/0_2/(0_6,\pm 15)_2/\pm 15/0_4/\pm 15/0_2]_s$
DE\current-to-best\1	208148.56	208124.08	117.56	207502.00	$[0_2/\pm 15/0_2/(0_6,\pm 15)_2/\pm 15/0_4/\pm 15/0_2]_s$
CA	208148.56	208099.28	83.45	207756.63	$[0_2/(0_4,\pm 15)_2/(0_2,\pm 15)_2/\pm 15/0_8]_s$
BBO	208148.56	208126.44	31.79	207985.48	$[\pm 15/0_{12}/(0_2,\pm 15)_2/(\pm 15,0_2)_2/0_2]_s$
OIO	208148.56	208138.52	11.88	208094.36	$[\pm 15/0_2/(0_8,\pm 15)_2/0_4/\pm 15_2]_s$
LCA	208148.56	208144.69	5.65	208126.77	$[0_2/\pm 15/0_2/(0_6,\pm 15)_2/\pm 15/0_4/\pm 15/0_2]_s$

Table 10. Optimum stacking sequences and statistical results obtained by different algorithms for load case 7

Algorithm	Statistical results				Stacking Sequence
	Best	Mean	Std	Worst	
PSO	416297.12	416246.54	87.70	415833.10	$[90_2/\pm 75/90_8/\pm 75/90_6/\pm 75_2/90_4/\pm 75/90_2]_s$
DE\best\1	416297.12	416045.22	672.26	413665.23	$[90_4/\pm 75/90_2/\pm 75/90_{12}/\pm 75_3/90_4]_s$
DE\rand-to-best\1	416297.12	416290.44	16.47	416207.77	$[90_2/\pm 75/90_6/\pm 75/90_8/\pm 75/90_6/\pm 75_2]_s$
DE\current-to-rand\1	416297.12	416293.75	5.89	416275.78	$[90_4/(\pm 75,90_2)_2/90_4/(90_4,\pm 75)_2/\pm 75/90_2]_s$
DE\current-to-best\1	416297.12	416293.52	8.36	416265.21	$[90_2/\pm 75/90_2/(90_6,\pm 75)_2/\pm 75/90_4/\pm 75/90_2]_s$
CA	416297.12	416211.30	100.16	415883.44	$[90_4/\pm 75/90_6/\pm 75_2/90_{10}/\pm 75_2/90_2]_s$
BBO	416297.12	416243.94	49.67	416093.90	$[90_2/\pm 75/90_6/\pm 75/90_{10}/\pm 75_2/90_4/\pm 75]_s$
OIO	416297.12	416285.32	11.43	416243.21	$[\pm 75/90_2/(90_8,\pm 75)_2/\pm 75/90_4/\pm 75_2]_s$
LCA	416297.12	416290.63	9.60	416258.83	$[90_6/\pm 75/90_4/\pm 75/(90_2,\pm 75)_2/\pm 75/90_8]_s$

Table 11. Optimum stacking sequences and statistical results obtained by different algorithms for load case 8

Algorithm	Statistical results				Stacking Sequence
	Best	Mean	Std	Worst	
PSO	161882.05	161881.17	4.82	161855.65	$[\pm 45_{16}]_s$
DE\best\1	161882.05	161386.10	518.11	159950.55	$[\pm 45_{16}]_s$
DE\rand-to-best\1	161882.05	161836.78	122.78	161323.73	$[\pm 45_{16}]_s$
DE\current-to-rand\1	161882.05	161882.05	0.00	161882.05	$[\pm 45_{16}]_s$
DE\current-to-best\1	161882.05	161848.35	87.96	161595.35	$[\pm 45_{16}]_s$
CA	161882.05	161872.50	52.35	161595.35	$[\pm 45_{16}]_s$
BBO	161882.05	161882.05	0.00	161882.05	$[\pm 45_{16}]_s$
OIO	161882.05	161882.05	0.00	161882.05	$[\pm 45_{16}]_s$
LCA	161882.05	161882.05	0.00	161882.05	$[\pm 45_{16}]_s$

Table 12. Optimum stacking sequences and statistical results obtained by different algorithms for load case 9

Algorithm	Statistical results				Stacking Sequence
	Best	Mean	Std	Worst	
PSO	139942.53	139917.16	26.46	139831.96	$[\pm 30_4/\pm 15/\pm 30_2/(\pm 15, \pm 30)_2/\pm 15_3/\pm 30/0_2]_s$
DE\best\1	139915.42	139535.79	512.21	138043.81	$[\pm 30_2/\pm 15/\pm 30_4/\pm 15/\pm 30_3/\pm 15/\pm 30/\pm 15_2/0_2]_s$
DE\rand-to-best\1	139930.05	139880.17	33.40	139794.40	$[\pm 30_6/\pm 15_4/\pm 30_2/\pm 15/0_6]_s$
DE\current-to-rand\1	139930.35	139848.15	67.12	139623.36	$[\pm 30_5/\pm 15_2/\pm 30_2/\pm 15/\pm 30/\pm 15_3/0_2/\pm 15]_s$
DE\current-to-best\1	139931.54	139873.65	47.68	139733.88	$[\pm 30_3/\pm 15/\pm 30_4/\pm 15/\pm 30/\pm 15_4/0_4]_s$
CA	139939.29	139860.59	81.75	139639.57	$[\pm 15/\pm 30_8/\pm 15/\pm 30_4/0_4]_s$
BBO	139942.53	139901.46	42.23	139767.83	$[\pm 30_4/\pm 15/\pm 30_2/\pm 15/\pm 30/\pm 15_2/\pm 30_2/\pm 15_2/0_2]_s$
OIO	139938.09	139901.46	15.45	139880.39	$[\pm 30/(\pm 15, \pm 30_5)_2/\pm 30/0_2/\pm 15]_s$
LCA	139942.53	139929.94	21.31	139824.37	$[\pm 30_4/\pm 15/\pm 30_2/\pm 15/\pm 30/\pm 15_2/\pm 30_2/\pm 15_2/0_2]_s$

Fig. 4 presents the boxplots to provide statistical intuitive performance comparisons between different algorithms for the first and fourth load cases. The reason behind the selection of these load cases stems from the fact that these load cases are challenging enough to clearly show the statistical differences between the algorithms. For the first load case, it can be seen from Fig. 4 that the ranges and variances of the results obtained by the PSO, OIO, and LCA are significantly smaller than those for other algorithms, which show the stability of these algorithms in finding maximum buckling load factors. For the fourth load case, the superiority of PSO, OIO, and LCA is still observable. However, the BBO is also exhibited competitive performance in the fourth load case. It can also be seen that the mean values obtained by the PSO, OIO, and LCA for the first and fourth load cases are well distributed near the optimum buckling load factors. The overall conclusion that can make from Fig. 4 is that the LCA is statistically more stable than all other algorithms. However, the statistical performances of PSO and OIO are also remarkable.

To investigate the convergence properties of different algorithms, the average convergence diagrams for the first and fourth load cases are illustrated in Fig. 5. From Fig. 5, it can be observed that the LCA and PSO exhibit faster convergence rates than other algorithms. Although the convergence rate of OIO is slower than others, this algorithm can find higher quality solutions than most of the other algorithms as the iterations proceed. This may reflect the fact that the trade-offs between the exploration and exploitation phases in LCA and PSO are more appropriately implemented and they are computationally more efficient. It seems that OIO switches from the exploration phase to the exploitation phase with a significant delay in comparison to the LCA and PSO. However, despite this delay, OIO is still capable of converging to better final solutions in comparison to most of the investigated algorithms. It is also observable from Fig. 5 that the convergence behaviours of different variants of DE seem to be somehow similar.

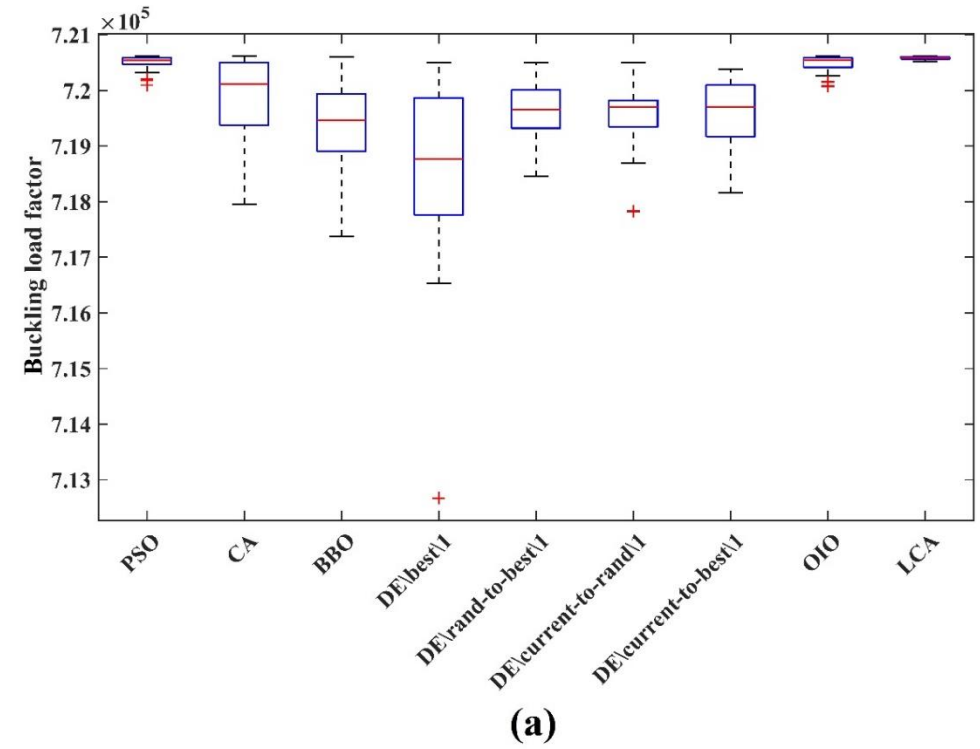

Fig. 4. Box plots of different algorithms: a) first load case, b) fourth load case

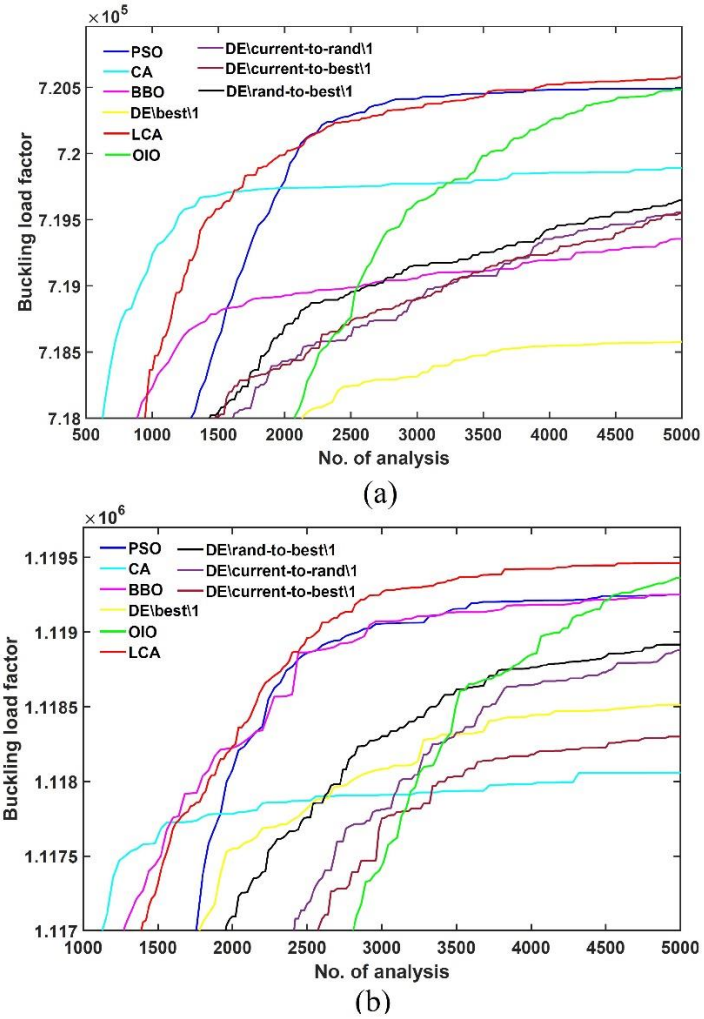

Fig. 5. Average convergence diagrams of different algorithms: a) first load case, b) fourth load case

4.3. Deep statistical comparison

Recent research trends in meta-heuristics revealed that the comparison of the efficiency of algorithms only based on the basic statistical parameters, such as best, mean, worst, and standard deviation, is not statistically enough to make proper conclusions [69,70]. Various statistical tests in the literature are applicable to evaluate, compare, and rank the performance of meta-heuristics for a given problem [69]. Recently, Eftimov et al. [70] proposed a deep statistical comparison (DSC) method for the performance comparison of algorithms. The approach uses the two-sample Kolmogorov–Smirnov (KS) test to pair-wise performance comparison between each pair of algorithms. Then, the algorithms are ranked based on the results obtained from the two-sample KS test. In this study, in order to provide a fair comparison, the DSC method is employed to rank the

performance of investigated algorithms in the optimum layup problem of laminated composite plates.

The two-sample KS test is a non-parametric statistical test that determines whether two sets of data come from the same continuous distribution or not. The test assumes the null hypothesis that the results obtained from each pair of algorithms come from the same continuous distribution. In other words, the KS test considers the null hypothesis that the performances of each pair of algorithms are statistically equivalent. Let us assume α_{KS} be the significance level used by the KS test, which indicates the probability threshold for acceptance or rejection of the null hypothesis. For the selected significance level α_{KS}, the KS test computes p_{value} for each pair of algorithms. If p_{value} is smaller than the significance level α_{KS}, the null hypothesis would be rejected. Otherwise, the null hypothesis would be accepted. The acceptance of the null hypothesis means that there is no significant difference between the two algorithms and they perform statistically the same. However, if the null hypothesis is rejected, it means that the two algorithms perform statistically different. The DSC approach use the p_{value} for ranking the performance of different algorithms. For more details about the ranking formulations, the interested readers are referred to Ref. [70].

The selection of α_{KS} plays an important role in the acceptance or rejection of the null hypothesis. Inappropriate values for this parameter could result in the wrong conclusion about the performances of different algorithms. If α_{KS} is taken as 1, it would result in the rejection of the null hypothesis for all values of p_{value}, which would mean the performances of all algorithms are always statistically different. The value of α_{KS} is typically considered between 0.05 and 0.1 [70]. In this study, the significance level is taken as $\alpha_{\mathrm{KS}} = 0.05$. The p_{value} obtained from the KS test for each pair of algorithms in different load cases are illustrated in Fig. 6. It is obvious that the diagonal elements in Fig. 6 should be equal to one, which means that each algorithm comes from the same distribution in comparison to itself. Fig. 6 illustrates some interesting information about the difference between algorithms in each load case. Firstly, the null hypothesis is accepted for all of the algorithms except DE\best\1 in load cases 2, 5, and 8. This means that there are significant differences between the performances of the DE\best\1 algorithm and others in these load cases. It also implies that there are no significant differences between most of the algorithms in load cases 2, 5, and 8, and their performances are statistically equivalent. In the rest of the load cases which are more challenging than load cases 2, 5, and 8, the null hypotheses are rejected and there are significant differences between the performances of different algorithms.

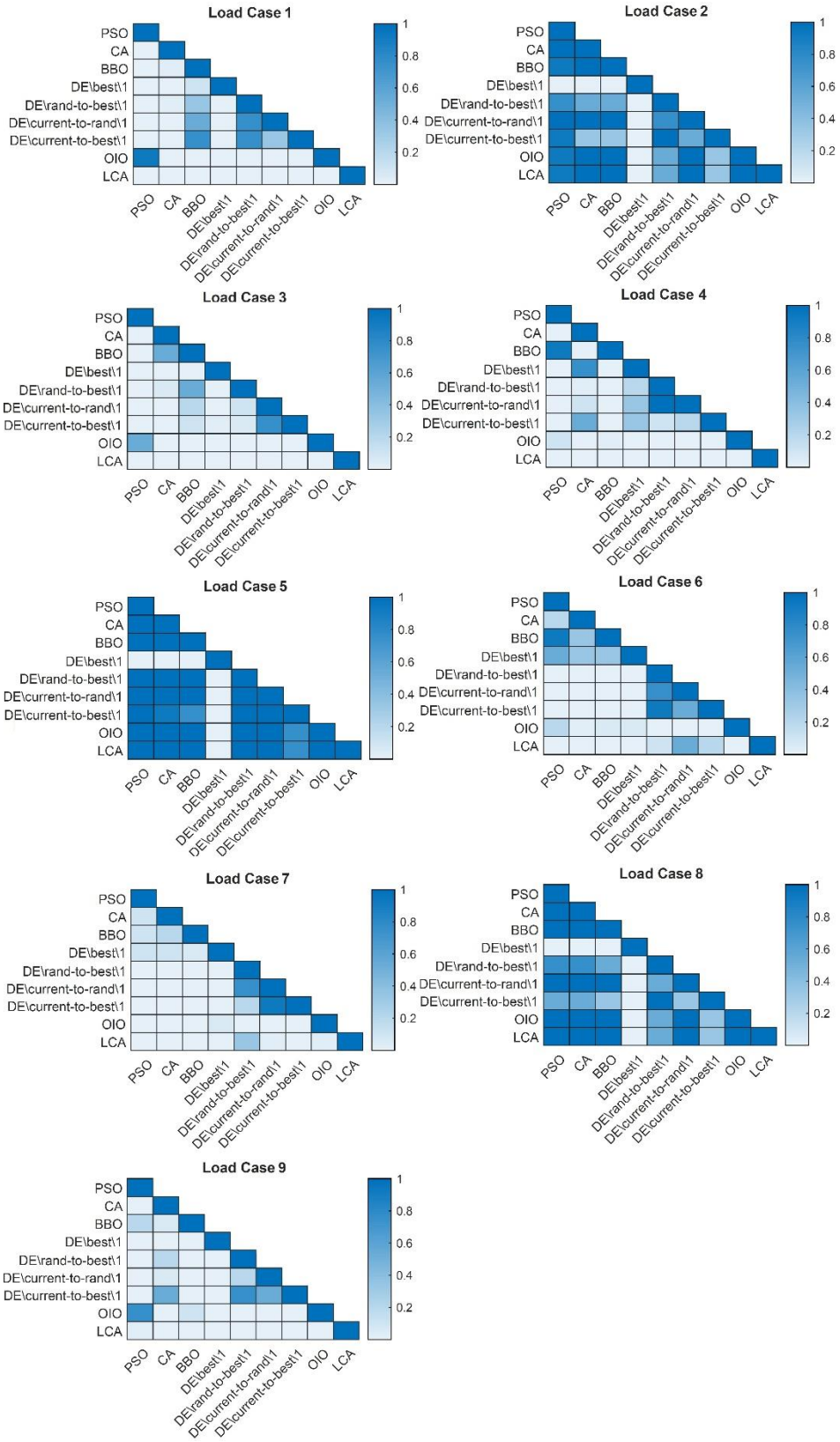

Fig. 6. p_{value} obtained from the two-sample Kolmogorov-Smirnov (KS) test for each pair of algorithms in different load cases

Finally, Table 13 lists the performance rankings of different algorithms for different load cases. The algorithms with the same rank values in each load case reveal the fact that their performances are statistically equivalent. From this table, it can be concluded that the LCA algorithm performs better or equal in comparison to other algorithms in different load cases, except load case 7. Among different variants of the DE algorithm, the DE\current-to-rand\1 performs better than others as it is ranked as the first algorithm in load cases 6 and 7. The last column of Table 13 shows the overall ranking of algorithms calculated based on the overall scores obtained in different load cases, in which the LCA and OIO algorithms are ranked as the two most efficient algorithms between the investigated nine algorithms.

Table 13. Ranking of different algorithms obtained by the DSC approach for different load cases

	Load cases									Overall
	LC1	LC2	LC3	LC4	LC5	LC6	LC7	LC8	LC9	
PSO	2	6	3	3	5	5	6	5	3	3
DE\best\1	9	9	9	7	9	9	9	9	9	9
DE\rand-to-best\1	5	8	6	5	4	3	4	8	5	6
DE\current-to-rand\1	6	5	7	6	7	1	1	2.5	8	5
DE\current-to-best\1	7	7	8	8	8	7	2	7	6	8
CA	4	4	4	9	6	8	8	6	7	7
BBO	8	2	5	4	2	6	7	2.5	4	4
OIO	3	2	2	2	2	4	5	2.5	2	2
LCA	1	2	1	1	2	2	3	2.5	1	1

5. Concluding remarks

The performance of nine meta-heuristic algorithms, including PSO, different variants of DE, CA, BBO, OIO and LCA, were assessed for the optimum layup problem of laminated composite plates. The buckling capacity maximisation of a 64-layer laminated composite plate under various load cases has been investigated as the benchmark problem, in which the design variables are the stacking sequences of layers. The performances of algorithms in finding maximum buckling load factors were evaluated in terms of the basic statistical parameters, including best, mean, standard deviation, and worst results. The numerical results revealed that the ranges and variances of the results obtained by the PSO, OIO, and LCA are significantly smaller than those for other algorithms, which show the stability of these algorithms in finding maximum buckling load factors.

To provide a fair comparison between the algorithms, a deep statistical comparison (DSC) method was employed to rank the performance of different algorithms. The DSC approach uses a non-parametric two-sample Kolmogorov–Smirnov (KS) test for pair-wise performance comparison between each pair of algorithms. The KS test assumes the null hypothesis that the

performances of each pair of algorithms are statistically equivalent. The results obtained from the KS test with the significance level of $\alpha_{\mathrm{KS}} = 0.05$ revealed that there are significant differences between the performances of different algorithms for most of the load cases. The performance rankings obtained from the DSC method suggested that the LCA algorithm performs better or equal in comparison to other algorithms in most of the load cases. The overall ranking of algorithms calculated based on the overall scores obtained from different load cases was as follows: LCA>OIO>PSO>BBO>DE\current-to-rand\1>DE\rand-to-best\1>CA>DE\current-to-best\1>DE\best\1. The convergence diagrams obtained from 30 independent runs revealed that the PSO and LCA exhibit faster convergence rates than other algorithms. This may reflect the fact that the trade-offs between the exploration and exploitation phases in LCA and PSO are more adequately implemented. Despite its remarkable performance in terms of final results, the convergence rate of OIO is slower than other algorithms. It was observed that OIO switches from the exploration phase to the exploitation phase with a significant delay in comparison to the LCA and PSO.

Conflict of interest

The authors declare that they have no conflict of interest.

References:

[1] Chung, D.L.D (2010) Composite materials: science and applications. Springer Science & Business Media.

[2] Nikbakt S., Kamarian S., Shakeri, M. A review on optimization of composite structures Part I: Laminated composites. Composite Structures. 2018. 195. 158–185.

[3] Riche, R.L., Haftka, R.T. Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm. AIAA journal. 1993. 31, 5. 951–956.

[4] Erdal, O., Sonmez, F.O. Optimum design of composite laminates for maximum buckling load capacity using simulated annealing. Composite Structures. 2005. 71, 1. 45–52.

[5] Soykasap, O., Karakaya, S. Structural optimization of laminated composite plates for maximum buckling load capacity using genetic algorithm. Key Engineering Materials. 348. 2007. 725–728.

[6] Holland, J.H. Genetic algorithms. Scientific American. 1992. 267, 1. 66–73.

[7] Van Laarhoven, P.J.M, Aarts, E.H.L. Simulated annealing. Simulated annealing: Theory and applications. 1987. 7–15.

[8] Price, K.V. Differential evolution. Handbook of optimization. 2013. 187–214.

[9] Dorigo, M., Di Caro, G. Ant colony optimization: a new meta-heuristic. Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). 2. 1999. 1470–1477.

[10] Kennedy, J., Eberhart, R. Particle swarm optimization. Proceedings of ICNN'95-international conference on neural networks. 4. 1995. 1942–1948.

[11] Reynolds, R.G. An introduction to cultural algorithms. Proceedings of the third annual conference on evolutionary programming. 24. 1994. 131–139.

[12] Maheri, A., Jalili, S., Hosseinzadeh, Y., Khani, R., Miryahyavi, M. A comprehensive survey on cultural algorithms. Swarm and Evolutionary Computation. 2021. 62. 100846.

[13] Simon, D. Biogeography-based optimization. IEEE transactions on evolutionary computation. 2008. 12, 6. 702–713.

[14] Geem Z.W., Kim, J.H., Loganathan, G.V. A new heuristic optimization algorithm: harmony search. simulation. 2001. 76, 2. 60–68.

[15] Wu. Y. A survey on population-based meta-heuristic algorithms for motion planning of aircraft. Swarm and Evolutionary Computation. 2021. 62. 100844

[16] Gao, K.Z., He, Z.M., Huang, Y., Duan, P.Y., Suganthan, P.N. A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing. Swarm and Evolutionary Computation. 2020. 57. 100719

[17] Abu Arqub, O., Abo-Hammour, Z. Numerical solution of systems of second-order boundary value problems using continuous genetic algorithm. Information Sciences. 2014. 279. 396-415.

[18] Abo-Hammour, Z., Abu Arqub, O., Alsmadi, O., Momani, S., Alsaedi, A. An optimization algorithm for solving systems of singular boundary value problems. Applied Mathematics Information Sciences. 2014. 8, 6. 2809.

[19] Abo-Hammour, Z., Alsmadi, O., Momani, S., Abu Arqub, O. A genetic algorithm approach for prediction of linear dynamical systems. Mathematical Problems in Engineering. 2013. 2013.

[20] Abu Arqub, O., Abo-Hammour, Z., Momani, S., Shawagfeh, N. Solving singular two-point boundary value problems using continuous genetic algorithm. Abstract and Applied Analysis. 2012. 2012.

[21] Jalili, S., Nallaperuma, S., Keedwell, E., Dawn, A., Oakes-Ash, L. Application of metaheuristics for signal optimisation in transportation networks: A comprehensive survey. Swarm and Evolutionary Computation. 2021. 63. 100865

[22] Aymerich, F., Serra, M. Optimization of laminate stacking sequence for maximum buckling load using the ant colony optimization (ACO) metaheuristic. Composites Part A: Applied Science and Manufacturing. 2008. 39, 2. 262–272.

[23] Wang, W., Guo, S., Chang, N., Zhao, F., Yang, W. A modified ant colony algorithm for the stacking sequence optimisation of a rectangular laminate. Structural and Multidisciplinary Optimization. 2010. 41, 5. 711–720.

[24] Deveci, H.A., Aydin, L., Secil, A.H. Buckling optimization of composite laminates using a hybrid algorithm under Puck failure criterion constraint. Journal of Reinforced Plastics and Composites. 2016. 35, 16. 1233–1247.

[25] Lakshmi, K., Rao, A., Rama, M. Optimal design of laminate composite plates using dynamic hybrid adaptive harmony search algorithm. Journal of Reinforced Plastics and Composites. 2015. 34, 6. 493–518.

[26] Hosseinzadeh, Y., Jalili, S., Khani, R. Investigating the effects of flax fibers application on multi-objective optimization of laminated composite plates for simultaneous cost minimization and frequency gap maximization. Journal of Building Engineering. 2020. 32. 101477

[27] Karakaya, S., Soykasap, O. Buckling optimization of laminated composite plates using genetic algorithm and generalized pattern search algorithm Structural and Multidisciplinary Optimization. 2009. 39, 5. 477.

[28] Chang, N., Wang, W., Yang, W., Wang, J. Ply stacking sequence optimization of composite laminate by permutation discrete particle swarm optimization. Structural and Multidisciplinary Optimization. 2010. 41, 2. 179–187.

[29] Jalili, S., Khani, R., Hosseinzadeh, Y. On the performance of flax fibres in multi-objective design of laminated composite plates for buckling and cost. 2021. 33. 3094-3106

[30] Karakaya, S., Soykasap, O. Natural frequency and buckling optimization of laminated hybrid composite plates using genetic algorithm and simulated annealing. Structural and Multidisciplinary Optimization. 2011. 43, 1. 61–72.

[31] Kaveh, A., Dadras, A., Malek, N.G. Buckling load of laminated composite plates using three variants of the biogeography-based optimization algorithm. Acta Mechanica. 2018. 229, 4. 1551–1566.

[32] Almeida, F.S. Stacking sequence optimization for maximum buckling load of composite plates using harmony search algorithm. Composite Structures. 2016. 143. 287–299.

[33] Akcair, M., Savran, M., Aydın, L., Ayakdas¸ O., Ozturk. S., Kucukdogan, N. Optimum design of anti-buckling behavior of graphite/epoxy laminated composites by differential evolution and simulated annealing method. Research on Engineering Structures and Materials. 2019. 5, 2.

[34] Kaveh, A., Dadras, A., Malek, N.G. Robust design optimization of laminated plates under uncertain bounded buckling loads. Structural and Multidisciplinary Optimization. 2019. 59, 3. 877–891.

[35] Rao, A.R.M., Arvind N. A scatter search algorithm for stacking sequence optimisation of laminate composites. Composite Structures. 2005. 70, 4. 383–402.

[36] Kaveh, A., Hashemi, S.B., Sheikholeslami, R. Optimal design of laminated composite structures via hybrid charged system search and particle swarm optimization. Asian Journal of Civil Engineering (Building Engineering). 2013.

[37] Kaveh, A., Dadras, A., Malek, N.G. Optimum stacking sequence design of composite laminates for maximum buckling load capacity using parameter-less optimization algorithms. Engineering with Computers. 2019. 35, 3. 813–832.

[38] Kashan, A.H. An efficient algorithm for constrained global optimization and application to mechanical engineering design: League championship algorithm (LCA). Computer-Aided Design. 2011. 43, 12. 1769–1792.

[39] Alimoradi, M.R., Kashan, A.H. A league championship algorithm equipped with network structure and backward Q-learning for extracting stock trading rules. Applied Soft Computing. 2018. 68. 478–493.

[40] Jalili, S., Kashan, A.H., Hosseinzadeh, Y. League championship algorithms for optimum design of pin-jointed structures. Journal of Computing in Civil Engineering. 2017. 31, 2. 04016048.

[41] Bouchekara, H.R.E.H., Abido, M.A., Chaib, A.E., Mehasni, R. Optimal power flow using the league championship algorithm: a case study of the Algerian power system. Energy conversion and management. 2014. 87. 58–70.

[42] Kashan, A.H., Jalili, S., Karimiyan, S. Optimum structural design with discrete variables using league championship algorithm. Civil Engineering Infrastructures Journal. 2018. 51, 2. 253-275.

[43] Kashan, A.H., Jalili, S., Karimiyan, S. Premier league championship algorithm: A multi-population-based algorithm and its application on structural design optimization. Socio-cultural Inspired Metaheuristics. 2019. 828. Springer.

[44] Kashan, A.H. An effective algorithm for constrained optimization based on optics inspired optimization (OIO). Computer-Aided Design. 2015. 63. 52–71.

[45] Lalwani, P., Banka, H., Kumar, C. CRWO: Clustering and routing in wireless sensor networks using optics inspired optimization. Peer-to-Peer Networking and Applications. 2017. 10, 3. 453–471.

[46] Ozdemir, M.T., Ozturk, D. Comparative performance analysis of optimal PID parameters tuning based on the optics inspired optimization methods for automatic generation control. Energies. 2017. 10, 12. 2134.

[47] Jalili, S., Kashan, A.H. Optimum discrete design of steel tower structures using optics inspired optimization method. The Structural Design of Tall and Special Buildings. 2018. 27, 9. e1466.

[48] Jalili, S., Kashan, A.H. An optics inspired optimization method for optimal design of truss structures. The Structural Design of Tall and Special Buildings. 2019. 8, 6. e1598.

[49] Mavrovouniotis Michalis, Li Changhe, Yang Shengxiang. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. Swarm and Evolutionary Computation. 2017. 33. 1–17.

[50] Storn Rainer, Price Kenneth. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization. 1997. 11, 4. 341–359.

[51] Das Swagatam, Suganthan Ponnuthurai Nagaratnam. Differential evolution: A survey of the state-of-the-art // IEEE transactions on evolutionary computation. 2010. 15, 1. 4–31.

[52] Qin A Kai, Huang Vicky Ling, Suganthan Ponnuthurai N. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE transactions on Evolutionary Computation. 2008. 13, 2. 398–417.

[53] Richerson Peter J, Boyd Robert. Not by genes alone: How culture transformed human evolution. 2008.

[54] Laland Kevin N. Exploring gene–culture interactions: insights from handedness, sexual selection and niche-construction case studies. Philosophical Transactions of the Royal Society B: Biological Sciences. 2008. 363, 1509. 3577–3589.

[55] Ali, M.Z., Awad, N.H. A novel class of niche hybrid cultural algorithms for continuous engineering optimization, Information Science. 2014. 267. 158–190.

[56] Awad, N.H., Ali, M.Z., Suganthan, P.N., Reynolds, R.G., CADE: a hybridization of cultural algorithm and differential evolution for numerical optimization, Information Science. 2017. 378. 215–241.

[57] Jalili, S., Hosseinzadeh, Y. A cultural algorithm for optimal design of truss structures. Latin American Journal of Solids & Structures. 2015. 12, 2. 1721-1747.

[58] Yan, X., Song, T. & Wu, Q. An improved cultural algorithm and its application in image matching. Multimedia Tools & Applications. 2017. 76, 14951–14968.

[59] Jalili, S., Hosseinzadeh, Y., Rabczuk, T. Simultaneous size and shape optimization of dome-shaped structures using improved cultural algorithm, in: Socio-cultural Inspired Metaheuristics, Springer, 2019, pp. 93–120.

[60] MacArthur Robert H, Wilson Edward O. The theory of island biogeography. 1. 2001.

[61] Guo, W., Ming, C., Wang, L., Mao, Y., Wu, Q. A survey of biogeography-based optimization. Neural Computing & Applications. 2017. 25. 1909–1926.

[62] Ma, H., Simon, D., Siarry, P., Yang, Z., Fei, M. Biogeography-based optimization: a 10-year review. IEEE Transactions on Emerging Topics in Computational Intelligence. 2017. 1, 5. 391-407.

[63] Carbas, S. Optimum structural design of spatial steel frames via biogeography-based optimization. Neural Computing & Applications. 2017. 28. 1525–1539.

[64] Jalili, S., Hosseinzadeh, Y., Taghizadieh, N. A biogeography-based optimization for optimum discrete design of skeletal structures. Engineering Optimization. 2015. 48, 9. 1491-1514.

[65] Kashan Ali Husseinzadeh. League championship algorithm: a new algorithm for numerical function optimization. Soft Computing and Pattern Recognition, 2009. SOCPAR'09. International Conference of. 2009. 43–48.

[66] Chattopadhyay, S., Murthy, C. A., Pal, S. K. (2014). Fitting truncated geometric distributions in large scale real world networks. Theoretical Computer Science, 551, 22-38.

[67] Soremekun G, G¨ urdal Z, Haftka RT, Watson LT. Composite laminate design optimization by genetic algorithm with generalized elitist selection // Computers & structures. 2001. 79, 2. 131–143.

[68] Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82

[69] Carrasco Jacinto, Garc´ıa Salvador, Rueda MM, Das S, Herrera Francisco. Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review // Swarm and Evolutionary Computation. 2020. 54. 100665

[70] Eftimov Tome, Korosec Peter, Seljak Barbara Korousic. A novel approach to statistical comparison of meta-heuristic stochastic optimization algorithms using deep statistics. Information Sciences. 2017. 417. 186–215.