Engineering with Computers
https://doi.org/10.1007/s00366-020-01077-w

ORIGINAL ARTICLE

Multiobjective meta-heuristic with iterative parameter distribution estimation for aeroelastic design of an aircraft wing

Kittinan Wansasueb[1] · Nantiwat Pholdee[1] · Natee Panagant[1] · Sujin Bureerat[1]

Received: 28 November 2019 / Accepted: 26 May 2020

Abstract
This paper proposes a new self-adaptive meta-heuristic (MH) algorithm for multiobjective optimisation. The adaptation is accomplished by means of estimation of distribution. The differential evolution reproduction strategy is modified and used in this dominance-based multiobjective optimiser whereas population-based incremental learning is used to estimate the control parameters. The new method is employed to solve aeroelastic multiobjective optimisation of an aircraft wing which optimises structural weight and flutter speed. Design variables in the aeroelastic design problem include thicknesses of ribs, spars and composite layers. Also, the ply orientation of the upper and lower composite skins are assigned as the design variables. Additional benchmark test problems are also use to validate the search performance of the proposed algorithm. The performance validation reveals that the proposed optimiser is among the state-of-the-art multiobjective meta-heuristics. The concept of using estimation of distribution algorithm for tuning meta-heuristic control parameters is efficient and effective and becomes a new direction for improving MH performance.

Keywords Aeroelasticity · Aircraft wing · Surrogate models · Multiobjective evolutionary algorithms · Estimation of distribution algorithm

1 Introduction

Aerospace industries nowadays have been highly competitive among leading countries around the world. With such a situation, the aircraft design phase becomes more vital for developing a prototype or improving existing aircraft. A good aircraft not only fulfils all avionic regulations requirements but also is expected to have high performance in many aspects such as fuel consumption, air pollution emission, manoeuvrability, and passenger comfort. This results in structural weight being reduced leading to more flexible aircraft wings and tails. In designing an aircraft with highly flexible wings, aeroelasticity must be included in the process. Aeroelastic optimisation is usually carried out for structural sizing of aircraft wings and tails at the preliminary design stage after having a certain aircraft configuration from the conceptual design phase. Often, it is taken into consideration in conceptual design [1] for more advanced design strategy. A design problem is commonly set to minimise mass subject to aeroelastic and other structural constraints. Design variables can be dimensions of ribs and spars, and skins thicknesses. In cases of wings made of carbon fibre, fibre ply orientations can also be used as design variables [2–6]. Multiobjective aeroelastic optimisation can also be performed [5–7] and leads to multiple optimal design solutions for further decision making. Unconventional design can be posed and found to be an efficient approach [8, 9].

Aeroelasticity is a study on mutual interaction between inertial, restoration, and aerodynamic forces. It can be classified as static and dynamic aeroelasticity. The former ignores the inertial forces while the latter considers all the three forces. The static aeroelastic phenomena used in aircraft structural design are divergence speed, control effectiveness, and lift effectiveness (sometimes referred to as the ratio of the cruise to jig shape lifts) while dynamic aeroelasticity is represented by flutter speed. Surely, other types of aeroelasticity e.g. aeroelastic freeplay, gust response, and aeroservoelasticity must be taken into account. Optimisation methods used to solve aeroelastic optimisation problems can be

✉ Natee Panagant
natepa@kku.ac.th

[1] Sustainable Infrastructure Research and Development Centre, Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand

Published online: 17 June 2020

gradient-based methods [10–12] or meta-heuristics (MHs) [5, 7]. The former is more powerful if design variables are structural sizing and all function derivative calculations are sufficiently accurate. For meta-heuristics, both single- [4, 13] and multi-objective [3, 5, 7] meta-heuristics have been employed to deal with aeroelastic optimisation. The meta-heuristic methods have relatively slower convergence rate if only sizing variables exist. However, they have undeniable features in that they are derivative-free, robust, and simple to use. More importantly, their multiobjective versions can explore a Pareto optimal set within one run. With the derivative-free feature, any type of design variables can be included in a design problem, therefore, unconventional design problems can be posed. Nevertheless, with the weak point in the slow convergence rate, it is always required to find a new powerful meta-heuristic algorithm.

Over the last few decades, there have been numerous meta-heuristic algorithms proposed in the literature for both single- and multiple- objective optimisation. Some recently published meta-heuristics include water evaporation optimization (WEO) [14], dolphin echolocation [15], ray optimization (RO) [16], tug of war optimization (TWO) [17], vibrating particles system (VPS) [18], dragonfly algorithm (DA) [19], ant lion optimiser (ALO) [20], etc. while some hybrid algorithms [21–25] between those existing algorithm have also been proposed. Beyond that, meta-heuristics for many-objective optimisation (having more than three objective functions) have been a new issue since such a design problem is considerably more difficult to solve [26]. For multiobjective meta-heuristics, some of the most popular algorithms are a non-dominated sorting genetic algorithm II [27], multiobjective evolutionary algorithm based on decomposition [28], and strength Pareto evolutionary algorithm 2 [29]. One of the main research issues for meta-heuristics is that they, most of the time, rely on the settings of control parameters such as crossover and mutation probabilities in a genetic algorithm. As a result, self-adaptive meta-heuristics have been introduced. The idea is to use many values of control parameters of a particular meta-heuristic while the algorithm adapts itself to use the best values during an optimisation run. This makes it easy for an inexperienced user to run the optimiser and commonly improves the performance of MHs.

This paper is concerned with introducing a new multiobjective meta-heuristic for solving multiobjective aeroelastic optimisation of an aircraft wing. The new method exploits the estimation of distribution algorithm from population-based incremental learning (PBIL) to iteratively estimate the best values of control parameters whereas its population reproduction is obtained from improving and modifying the reproduction concept of differential evolution. The new hybrid method is called multiobjective meta-heuristic with iterative parameter distribution estimation (MM-IPDE). The proposed algorithm is validated against a number of state-of-the-art algorithms while two problem sets, aeroelastic optimisation and standard benchmark test suites are used for performance evaluation. The hypervolume (HV), Inverted Generational Distance (IGD) are used as performance indicators whereas the statistical Friedman test is employed to rank the various optimisers. The rest of the paper is organised as Sect. 2 details the aeroelastic multiobjective optimisation problem formulation; Sect. 3 explains the new multiobjective meta-heuristic algorithm MM-IPDE; Numerical test, design demonstration as well as optimisation parameter setup are given in Sect. 4; the results and performance comparison are detailed in Sect. 5 whereas conclusions are drawn in Sect. 6.

2 Optimisation problem and function evaluations

Typically, a multiobjective optimisation problem can be formulated as:

$$\min_{\mathbf{x}} \left\{f_i(\mathbf{x})\right\}; i = 1, \ldots k \tag{1}$$

$$\text{subject to } g_i(x) \le 0, i = 1, \ldots, m$$
$$h_i(x) = 0, i = 1, \ldots, l$$
$$x_L \le x \le x_U$$

where $\{f_i(\mathbf{x})\}$ are k objective functions, which may be denoted as a vector $\mathbf{f}(\mathbf{x})$; $\{g_i(\mathbf{x})\}$ are m inequality constraints; $\{h_i(\mathbf{x})\}$ are l equality constraints; $\mathbf{x}_U and \mathbf{x}_U$ are respectively the lower and upper bounds of a design vector $\mathbf{x}$ size $n \times 1$.

Traditionally for optimisation with meta-heuristics (MHs), the problem is reduced to have only a vector of objective functions and the bound constraints of $\mathbf{x}$ while other constraints are dealt with using a penalty function. Often, it is convenient to normalise the bound constraints to be $0 \le \mathbf{x} \le 1$ where 0 and 1 are vectors full of zeroes and ones respectively. That means, in the function evaluation process, decoding of actual values for $\mathbf{x}$ must be operated. The normalised bound constraints are employed to ease in setting parameters in a reproduction process in meta-heuristics. In cases of single-objective optimisation, there is only one (global) optimum solution. Nevertheless, there are usually an infinite number of optimal solutions for multi-objective optimisation where a set of such optima is called a Pareto optimal set or a Pareto front if they are viewed in the objective function domain. Never-the-less, this will be explained further in the next section.

In this paper, a Goland wing (Fig. 1) is used for design demonstration [30]. The length of the root chord and half wing-span of the wing are 1.216 and 6.096 m, respectively

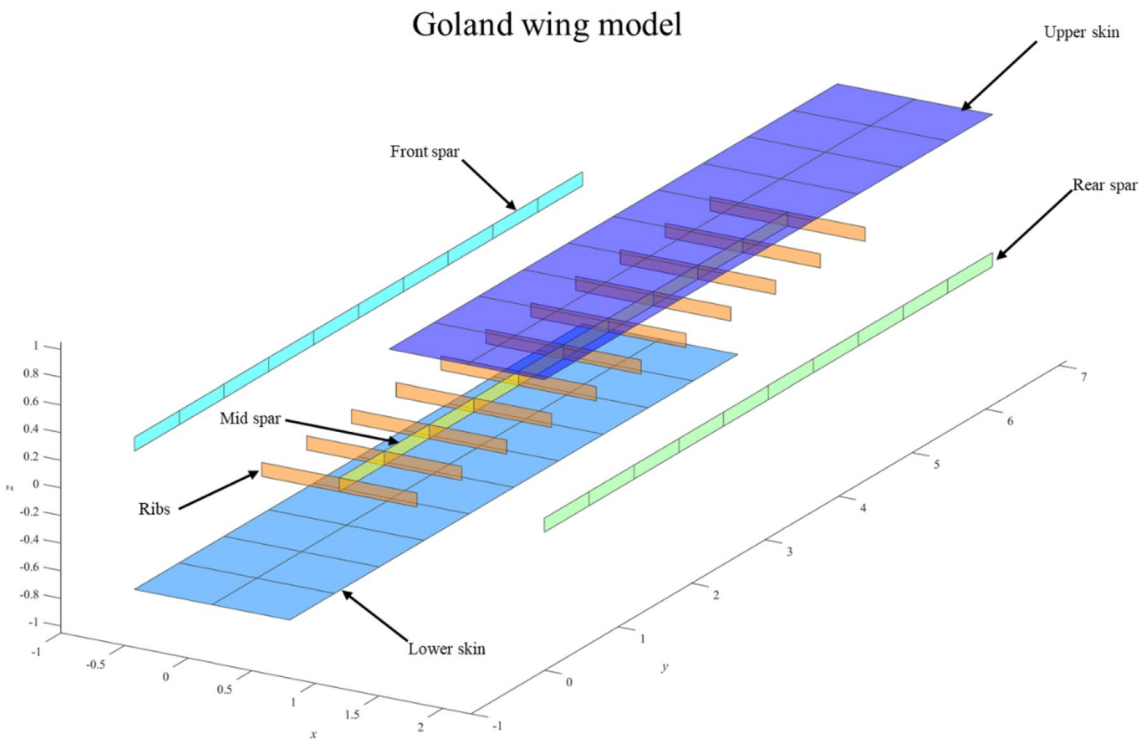

Fig. 1 Goland wing geometry model

Table 1 Mechanical properties used in Goland wing

Material	Properties name	Value	Unit
Aluminium	Young's modulus (E)	6.00E+10	Pa
	Poisson's ratio (ϑ)	0.3	–
	Density (ρ)	4000	kg/m^3
Carbon fibre	Young's modulus (E_{11})	1.49E+11	Pa
	Young's modulus (E_{22})	8.83E+09	Pa
	Shear modulus (G_{12})	5.38E+09	Pa
	Shear modulus (G_{13})	5.38E+09	Pa
	Shear modulus (G_{23})	2.98E+09	Pa
	Poisson's ratio (ν_{12})	0.342	–
	Density (ρ)	1800	kg/m^3

(4 and 20 ft. according to [31, 32]). Spars and ribs are made up of aluminium while the upper and lower skins are made of 3 layers of laminated carbon fibre. Material properties are given in Table 1. More details of the wing shape geometry can be found in [30].

The wing model is illustrated in Fig. 1. The initial wing structure has 11 ribs and 3 spars. For simplicity, the high lift and control devices are fixed, thus, the whole structure is assembled as a wing box. The 3 spars are placed at the wing leading and trailing edges while another one is placed at the middle of the chord. The 11 ribs are equally placed in the span direction from the root to tip chords. In an optimisation process, the thicknesses of all wing parts are assigned as design variables. For the upper and lower skins, layer thicknesses and ply orientations are set as design variables.

The aeroelastic multiobjective optimisation problem is expressed as:

$$\min_{\mathbf{x}} f_1(\mathbf{x}) = \text{total mass}, \ \max_{\mathbf{x}} f_2(\mathbf{x}) = V_f \tag{2}$$

Subject to

$$u_{\max} - u_{\text{al}} \leq 0$$

$$\eta_L - \eta_{L,al} \leq 0$$

$$\mathbf{x}_l \leq \mathbf{x} \leq \mathbf{x}_u$$

where $\mathbf{x}$ is a vector of design variables having lower and upper bounds as $\mathbf{x}_l$ and $\mathbf{x}_u$. $u_{\max}$ is the maximum transverse displacement on the wing. u_{al} is an allowable value for the transverse displacement. η_L is wing lift effectiveness the ratio of flexible to rigid total lift forces. $\eta_{L,\text{al}}$ is an allowable value of lift effectiveness. V_f is a critical wind speed herein is a flutter speed.

Two groups of design variables are thicknesses and composite ply orientations. There are totally 16 design variables, which can be detailed as:

x_{1-3} = thicknesses of the 3 layers of the lower skin.
x_{4-6} = ply orientations of the lower skin.
x_{7-9} = thicknesses of the 3 layers of the upper skin.
x_{10-12} = ply orientations of the upper skin.
x_{13} = thickness of the mid spar.
x_{14} = thickness of the ribs.
x_{15} = thickness of the front spar.
x_{16} = thickness of the rear spar.

The design variables are assigned to be discrete where the bound constraints are set as follows:

- $x_i \in \{0.2, 0.3, 0.4, 0.5, 0.6 \text{ and } 0.7\}$ mm for $i = 1, 2, 3, 7, 8, 9$.
- $x_i \in \{1, 2, 3, 4 \text{ and } 5\}$ mm for $i = 13, 14, 15, 16$.
- $x_i \in \{-60, -45, -30, 0, 30, 45, 60 \text{ and } 90\}$ degree for $i = 4, 5, 6, 10, 11, 12$.

The aerodynamic conditions and structural constraints are set as:

- Air density is set to be $_{\text{air}} = 1.225\ \text{kg/m}^3$.
- Aerodynamic loads as static forces and Lift effectiveness are computed at wind speed $V_\infty = 100$ m/s.
- Allowable value for the transverse displacement is $u_{\text{al}} = 0.1$ m.
- Allowable lift effectiveness is set to be $_{L,\text{al}} = 0.9$.
- Compressibility effect is included in flutter calculation.

The actual function evaluations are carried out based on finite element analysis using quadrilateral Mindlin shell elements with drilling degree of freedom [33] for both static and dynamic analyses. Aerodynamic force is achieved using the vortex ring method [34] while integration between the finite element and aerodynamic models is carried out by means of surface spline interpolation [35]. Flutter analysis is performed using a quasi-steady aerodynamic approach [36], which gives acceptable results compared to commercial software. The structural mass is a usual objective function as lighter wing weight is advantageous in many ways such as less moment of inertia for the directional/lateral motion. Flutter speed is also assigned as the second objective function as a measure of structural performance. The ratio of flexible to rigid lift or lift effectiveness is imposed to guarantee that the wing still has sufficient aerodynamic performance when taking into account flexibility. Other constraints are added to guarantee the safety requirements in operating conditions.

3 Multiobjective meta-heuristic with iterative parameter distribution estimation (MM-IPDE)

Multiobjective evolutionary algorithms (MOEAs) or a more general term multiobjective meta-heuristics (MOMHs) have been developed and applied to a wide variety of applications particularly in the fields of science and engineering. Such optimisers gain a great deal of popularity due to their ability to explore a Pareto optimal front within one optimisation run. The methods are derivative-free, robust and difficult to stall. Up to the present time, there have been a number of MOMHs being proposed and upgraded where some of the most well-known are NSGAII, MPSO, and SPEA2. A multiobjective meta-heuristic herein is developed for tackling a multiobjective minimisation problem with a box or bound constraints. MOEAs basically cannot deal with other types of design constraints directly except for the bound constraints. In cases that such design constraints exist, a penalty function technique or constraint-based non-dominated sorting technique should be used with the algorithms.

For two particular design solutions, solution $\mathbf{x}$ is said to dominate solution $\mathbf{y}$ if all elements of $\mathbf{f}(\mathbf{x})$ are not worse than the corresponding elements from $\mathbf{f}(\mathbf{y})$ and at least one element of $\mathbf{f}(\mathbf{x})$ is strictly better than the same element of $\mathbf{f}(\mathbf{y})$. For a set of all feasible design solutions, solutions that are not dominated by any solution in the set are called non-dominated solutions. A set of non-dominated solutions is called a Pareto optimal set while, if mapped to the objective function domain, it is called a Pareto optimal front. These definitions are used for developing MOEAs which is often called a Pareto dominance approach [37]. Evolutionary Algorithms (EAs) normally use a set of design vectors traditionally called a population to search for optima. For multiobjective cases, MOEAs can benefit from having a population and using a non-dominated sorting operator to find out the non-dominated design solutions iteratively. This will be the concept to develop a multiobjective meta-heuristic in this work. The proposed MM-IPDE exploits the estimation of distribution algorithm of PBIL to iteratively compute optimisation parameters settings, thus, PBIL for multiobjective optimisation is detailed first.

3.1 Population-based incremental learning for multiobjective optimisation (PBILM)

Population-based incremental learning, a simple type of an estimation of distribution algorithm, was first proposed by Baluja [38] for single-objective optimisation. A PBIL search mechanism is based on a binary space similar to the

Table 2 Probability vectors and their corresponding populations

Probability vectors (**P**)	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.25	0.0	0.75
Binary populations (**B**) (each row represents a binary design solution)	1	0	1	0	0	1	0	1	1	0	0	1
	0	0	1	1	0	1	1	0	0	0	0	1
	1	1	0	0	1	0	1	0	0	0	0	0
	0	1	0	1	1	0	0	1	1	1	0	1

original genetic algorithm (GA). Unlike GA which manipulates a binary population directly, PBIL uses the so-called probability vector to represent a population during a search process. The idea on how to create a binary population from a probability vector is shown in Table 2. In the table, if one design solution is represented by a row of binary strings, the probability vector should be a row vector as shown. An *i*-th element of the vector determines the probability of having a digit '1′ on the *i*th column of a population. With such a concept, one probability vector can result in several different binary populations e.g. case 1 and 2 in the table. Basically, PBIL solves a problem starting with an initial probability vector being set to have all elements as 0.5. This means each column will contain an equal number of digits '0′ and '1′. A binary population is then generated with objective function values being evaluated. The best solution is then exploited to update the probability vector. This procedure is repeated until one of the termination criteria is fulfilled.

For multiobjective optimisation [25, 39–41], using one probability vector is not sufficient as population diversity is one of the main factors affecting MOMH search performance. Therefore, a set of probability vectors will be used instead and it is called a probability matrix in which one row (or column) is one probability vector. Each probability vector is used to generate a binary subpopulation. The whole binary population is the union of all those subpopulations [41]. Similar to single-objective cases, the search procedure of PBIL for multiobjective optimisation is started with an initial probability matrix having elements full of 0.5. In addition, an initial (empty) Pareto archive is initiated. During an optimisation process, each row (vector) of the matrix will be used to create a subpopulation consequently leading to a binary population. The function evaluations are performed and the non-dominated solutions of the combined solutions from the current external Pareto archive and the population are sorted and saved into the archive. In cases that a row of the matrix is a probability vector, the element $P_{i,j}$ of the probability matrix is then updated as

$$P_{i,j}^{\text{new}} = P_{i,j}^{\text{old}}\left(1 - L_R\right) + \overline{b}_j L_R \tag{3}$$

where L_R is the PBIL learning rate. The superscripts "old" and "new" are the original and updated values of $P_{i,j}$ respectively. The parameter $\bar{b}_j$ can be determined in such a way that several (say n_0) binary solutions from the Pareto archive are randomly selected, and it is the mean value of the *j*th column of those selected binary solutions. The learning rate value is assigned to be fluctuated around 0.5 as

$$L_R = 0.5 + \text{rand} \cdot (+0.1 \text{ or } -0.1) \tag{4}$$

where rand[0,1] is a uniform random number. To prevent a premature convergence, the value of $P_{i,j}$ in Eq. (2) will be bounded in the interval [0.1, 0.9] after updating (see Algorithm 1). This means that a mutation operator and its settings are not required. It should be noted that the interval of $P_{i,j}$ used herein was the best values obtained from testing the new optimiser with several test functions. Algorithm 1 displays the computational steps for PBIL for multiobjective optimisation used in this study. It should be noted that, in this study, only the total number of function evaluations is used as a stopping criterion. The so-called truncation algorithm with similar individuals (TASI) algorithm presented in [42], which is based on clustering, is used as an archiving technique for removing some members in the Pareto archive when the members exceed n_A solutions.

Algorithm 1 MPBIL

Input: number of generation (n_{iter}), number of sub-populations (n_{sp}), binary length for a design vector (n_B), maximum archive size (n_A), population size (*Pop_Size*), number of non-dominated solutions used to update a probability vector (n_O)
Output: $\mathbf{x}^{\text{best}}$, $\mathbf{f}^{\text{best}}$ (Approximate Pareto front)
Initialisation:
: Assign the values to the elements of the probability matrix size $n_{sp} \times n_B$ as $P_{ij} = 0.5$ (this implies that one row of the matrix is one probability vector)
: Generate an empty Pareto archive $\mathbf{A} = \{\}$.

MPBIL main iterations
1: For $iter = 1$ to n_{iter}
2: Generate a binary population **B** from **P** (refer to Table 2).
3: Find non-dominated solutions of $\mathbf{A} \cup \mathbf{B}$.
4: Replace the members in **A** by those non-dominated solutions.
5: If the number of members in **A** exceeds n_A, remove some of the solutions in **A** using a Pareto archive technique.
6: Update the probability matrix $P_{ij}^{new} = update(P_{ij}^{old}, \mathbf{A}, n_O)$.
6.1: For $i = 1$ to n_{sp}
6.2: Update the i-th row of P_{ij} using Equations (2-3)
6.3: If $P_{ij}^{new} < 0.1$, set $P_{ij}^{new} = 0.1$. If $P_{ij}^{new} > 0.9$, set $P_{ij}^{new} = 0.9$.
6.4 End
7: End

3.2 MM-IPDE algorithm

It has long been found that optimisation parameter settings for MHs such as crossover and mutation rates in GA greatly affect their performance. Meta-heuristics with parameter adaptation, also known as self-adaptive MHs, are usually more powerful than their constant-parameter counterparts. Self-adaptive MHs are optimisers that can automatically tune the values of their key parameters for any optimisation problem being solved. There have been several concepts used for parameter adaptation such as SaNSDE-WEE [43], SHADE [44], and fuzzy logics [45]. Self-adaptive strategies for MHs are a simple type of learning algorithms, which can be roughly grouped as strategies for control or optimisation parameters settings and that for selection reproduction operators. For the first group, the probability of using a particular value of a control parameter is based on the history of its successful/failed use. Generation of optimisation parameter can be based on randomisation with a particular type of statistical distribution [43, 44, 46], interval selection [47], fitness-based adaptation [48], and fuzzy logic decision making [49]. The self-adaptive selection of reproduction operators, on the other hand, can be carried out using a statistical outlier detection method [50], and a history-based approach [43, 44, 46]. To our knowledge, there has never been a study on using an estimation of distribution algorithm (EDA), one of the learning algorithms, for parameter self-adaptation of MHs. This work is probably the first paper to investigate this issue. EDA used herein is PBIL which works on the binary searching spaces. The algorithm is embedded into the main search of differential evolution while the dominance-based approach is used for dealing with multiple objective functions.

The proposed multiobjective optimiser, as it is named, exploits and modifies part of the reproduction of differential evolution (DE) for population generation while PBILM is used to iteratively estimate DE parameters. Modified multi-objective DE (MODE) and PBILM are operated in parallel whereas MODE searches for the best design solutions of the original design problem but PBILM searches for a set of acceptable optimisation parameters successfully used by MODE. As a result, two (high and low) levels of design variables will also be used. The design variables for the high-level denoted as $\mathbf{x}_h$ will have elements representing the control or optimisation parameters similar to those used for generating a design vector in DE procedures. For this study, the high-level design variables consist of:

$$\mathbf{x}_h = \left\{F_{\text{scheme}}, F_1, F_2, C_R, I_{\text{reprod}}\right\}^T \tag{5}$$

where F_{scheme} is the choices for creating the value of a scaling factor F. F_1 and F_2 will be used for generating the scaling factor. C_R is used for generating the value of a binomial crossover parameter (C_R). I_{reprod} is the choices for a reproduction scheme. The elements of the high-level design vector will be used to create a vector of low-level design variables denoted by $\mathbf{x}_l$, which is the design variables of

the optimisation problem being solved. These two types of variables are coupled in such a way that the high-level design variable controls the reproduction of the low-level design variables. All the variables in Eq. (5) will be properly bounded. The decoding of $\mathbf{x}_h$ into $\mathbf{x}$ or $\mathbf{x}_l$ is carried out in a similar manner to the reproduction of DE. In this work as our first study, the number of schemes for generating F will be

$$F = \begin{cases} n\text{rand}(F_1, F_2); F_{\text{scheme}} = 1 \\ \text{rand}(F_2, F_1); F_{\text{scheme}} = 2 \end{cases} \tag{6}$$

This means that $F_{\text{scheme}}\{1, 2\}$. $n\text{rand}(x, y)$ is a normally distributed random number having the mean value of x and the standard deviation of y. The variable $\text{rand}(a, b)$ is a uniform random number generated in the interval $[a, b]$. From the first and second schemes, the value of F_2 will be constrained in an interval such that it is always lower than that of F_1. We use F_1 and F_2 for the two schemes to reduce the number of design variables of the high-level. The first choice for generating a scaling factor is based on an opposition-based operator as F can be negative or positive. The second scheme is similar to scaling factor generation used in some self-adaptive DE [43].

For the reproduction part in this work, there are five reproduction schemes in which three of them emulate DE reproduction. The schemes are given in Eq. (7) as:

$$\boldsymbol{u}_i = \begin{cases} \text{Algorithm2}(\mathbf{x}_{\text{best},1}, \boldsymbol{x}_{\text{best},2}, 1 - C_R); I_{\text{reprod}} = 1 \\ \boldsymbol{x}_i + F(\text{rand}\boldsymbol{x}_{\text{best},1} + \text{rand}\boldsymbol{x}_{\text{best},2} - \boldsymbol{x}_i); \text{I}_{\text{reprod}} = 2 \\ \text{rand}\boldsymbol{x}_{\text{best},1} + \text{rand}\boldsymbol{x}_{\text{best},2} + F(\text{rand}\boldsymbol{x}_{r_1} + \text{rand}\boldsymbol{x}_{r_2} - \text{rand}\boldsymbol{x}_{r_3} - \text{rand}\boldsymbol{x}_{r_4}) I_{\text{reprod}} = 3 \\ \boldsymbol{x}_i + F(\boldsymbol{x}_{\text{best},1} - \boldsymbol{x}_i) I_{\text{reprod}} = 4 \\ \text{Simple\_mutate}(\mathbf{x}_{\text{best},1}); I_{\text{reprod}} = 5 \end{cases} \tag{7}$$

where $\text{rand}[0, 1]$ is a uniform random number. $\boldsymbol{x}_i$ is the solution i in the current population. Four $\boldsymbol{x}_{ri}$ are randomly selected solutions from the population whereas $\boldsymbol{x}_{\text{best},i}$ are two individuals randomly selected from an external Pareto archive. The random number *rand* is used in Eq. (7) mainly to maintain or increase search diversity or exploration. The first reproduction scheme is detailed in Algorithm 2 where the probability of mutation is set to be $1 - C_R$. This scheme is successfully used in [51]. The number 1.15 in steps 8 and 10 are used based on the idea that some Pareto optimal solutions can possibly be located at the bounds of some design variables. Using the step-length value of 1.15 will sometimes cause variables to be outside of the bounds. As a result, those elements that violate the bound constraints will be repaired in such a way that the elements exceeding the lower or upper bounds will be respectively reassigned the values of those corresponding lower and upper bounds. This is given as the final step in Algorithm 2. This implies that the algorithm will have a chance to capture those solutions whose elements may be located on the bounds.

The second scheme will be called current − to − weightedbest/1 meaning the best solution is obtained from a randomly weighted sum of two randomly selected non-dominated solutions from the archive. The third is weightedbest/2 standing for a mutant vector being generated from the randomly weighted sum of two randomly selected non-dominated solutions and two of randomly weighted solutions of four solutions randomly selected from the current population. The motivation to implement reproduction schemes 2 and 3 instead of usual mutation schemes of DE is to increase population diversity which is very important for multiobjective optimisation by means of dominance-based search. It has been tested that these two reproduction schemes are powerful for many test problems. In addition, they are part of the main contribution of this paper. The fourth scheme is called current − to − best/1, which is traditionally used in many DE versions. For the last scheme, the simple mutation in Algorithm 2 (command lines 6–10) is activated to create a new solution. Moreover, a binomial crossover with parameter C_R needs to be operated for solutions from the second, third, and fourth schemes to have complete offspring $\mathbf{v}_i$ as.

$$v_j^i = \begin{cases} u_j^i; \text{rand} \le C_R \\ x_j^i; \text{otherwise} \end{cases} \tag{8}$$

where j represents the indices of vector elements. Algorithm 3 displays how the high-level design vector is transformed into a low-level design vector.

Algorithm 2 Reproduction scheme 1

Input: $\mathbf{x_1}$, $\mathbf{x_2}$, $p_m = 1 - C_R$
Output: $\mathbf{u}$ a low-level design vector
Main procedure
$\lambda = -0.25 + 1.5rand$
For $i = 1$ to n
1: Assign $P_R = rand.$
2: If $P_R \in [0,0.33)$, set $v_i = x_{1,i}$ *or* $x_{2,i}$ with equal probability
3: If $P_R \in [0.33,0.66)$, set $v_i = x_{1,i} + \lambda(x_{2,i} - x_{1,i})$.
4: If $P_R \in [0.66,1.00]$, Regenerate $\lambda_i = -0.25 + 1.5rand$ and set $v_i = x_{1,i} + \lambda_i(x_{2,i} - x_{1,i})$.
5: Otherwise, set $\mathbf{x}_{l,i} = \mathbf{u}_i$.
End

If $rand < p_m$
$Simple\_mutate(\boldsymbol{v})$: Remark, a simple mutation scheme will also be used as the fifth reproduction scheme.
6: $I_m =$ randomly choose a number from $\{1, 2, ..., n\}$, randomly choose an element to be mutated.
7: if $rand < 0.5$
8: set $v_{I_m} = x_{L,I_m} + 1.15rand(v_{I_m} - x_{L,I_m})$
9: otherwise
10: set $v_{I_m} = v_{I_m} + 1.15rand(x_{U,I_m} - v_{I_m})$
end if
Repair all elements that violate the bound constraints.

Algorithm 3 From $\mathbf{x_h}$ to $\mathbf{x_l}$ ($\mathbf{x_l}$ is original $\mathbf{x}$)

Input: $\{\mathbf{x}_{h,i}\}$ population of high-level design vectors for next generation, $\{\mathbf{x}_i\}$ a current population, population size (Pop_Size)
Output: $\{\mathbf{x}_{l,i}\}$ population of low-level design solutions for next generation
Main procedure
For $i = 1$ to Pop_Size
1: Assign values of parameters for creating a new solution, $F_{scheme} = x_{h,1}$, $F_1 = x_{h,2}$, $F_2 = x_{h,3}$, $C_R = x_{h,4}$, $I_{reprod} = x_{h,5}$.
2: Generate the value of F using Equation (6).
3: Find a mutant vector $\mathbf{u}_i$ using Equation (7).
4: If $I_{reprod} = 1$ *or* 5, set $\mathbf{v}_i = \mathbf{u}_i$, otherwise, perform binomial crossover on $\mathbf{u}_i$ as per Equation (8).
5: Set $\mathbf{x}_{l,i} = \mathbf{u}_i$.
End

The computational procedure for MM-IPDE is given in Algorithm 4. The optimiser can be classified as a Pareto dominance-based optimiser which exploiting a non-dominated sorting scheme to create an approximate Pareto front from a population of design solutions. The external non-dominated archive will be used to save all non-dominated solutions found. Since there are two levels of design variables, therefore, there will be two types of the non-dominated archive. For the initialisation stage, it starts with an initial probability matrix of the high-level whose elements are full of 0.5. A binary population $\{\mathbf{b}_h^i\}$ and its corresponding high-level real-code population $\{\mathbf{x}_h^i\}$ are generated according to the initial probability matrix. It is noted that the superscript i means individual i in the population. A solution set of the low-level variables $\{\mathbf{x}_l^i\}$ is then generated based on $\{\mathbf{x}_h^i\}$ following Algorithm 3. Since at the beginning, the non-dominated archive is empty, the terms $\mathbf{x}_{best,i}$ in Eq. (7) are replaced by randomly selected solutions. Having, evaluated the objective function values of $\{\mathbf{x}_l^i\}$, the non-dominated solutions sorted from $\{\mathbf{x}_l^i\}$ will be saved to the initial low-level Pareto archive $\mathbf{A}_l$. Meanwhile, binary solutions in $\{\mathbf{b}_h^i\}$ whose indices correspond to those in $\mathbf{A}_l$ will be saved to the

high-level external archive $\mathbf{A}_h$. It should be noted that initially the low-level and high-level variables will be loosely related, however, as the algorithm proceeds, their correlation will be increased. The algorithm iterates, whilst updating the high- and low-level archives iteratively, until a termination criterion is met. This is arguably the first self-adaptive meta-heuristic that uses an estimation of distribution algorithm for controlling optimisation parameters.

Algorithm 4 Procedure of MM-IPDE

Input: number of generation (n_{iter}), number of sub-populations of $\mathbf{x}_h$ (n_{sp}), binary length for a design vector $\mathbf{x}_h$ (n_B), maximum archive size (n_A), high-level bounds $\mathbf{x}_{h,L} \le \mathbf{x}_h \le \mathbf{x}_{h,U}$, population size ($n_{sol}$), number of non-dominated solutions used to update a probability vector (n_0)
Output: $\mathbf{x}_l^{\text{best}}$, $\mathbf{f}^{\text{best}}$ (Approximate Pareto front)
Initialisation:
0.1: Assign $P_{ij} = 0.5$ sized $n_{sp} \times n_B$.
0.2: Randomly generate $\{\mathbf{b}_h^i\}$.
0.3: Generate $\{\mathbf{x}_l^i\}$ using a Latin hypercube sampling technique.
0.4: Calculate objective functions $\mathbf{f} = fun(\{\mathbf{x}_l^i\})$ where fun is an objective function evaluation.
0.5: Find non-dominated solutions of $\{\mathbf{x}_l^i\}$, and save to the external archive $\mathbf{A}_l$.
0.6: Find the members in $\{\mathbf{b}_h^i\}$ that have corresponding indices to the solutions in $\mathbf{A}_l$ and save to the high-level archive $\mathbf{A}_h$.

Main iterations
1: For $iter = 1$ to n_{iter}
2: Update P_{ij}
  For $i = 1$ to n_{sp}
    2.1: Update the i-th row of P_{ij} using Equations (2-3) and solution members in $\mathbf{A}_h$.
    2.2: If $P_{ij}^{new} < 0.1$, set $P_{ij}^{new} = 0.1$. If $P_{ij}^{new} > 0.9$, set $P_{ij}^{new} = 0.9$.
  End
3: Generate $\{\mathbf{b}_h^i\}$ and then $\{\mathbf{x}_h^i\}$ from P_{ij}.
4: Generate $\{\mathbf{x}_l^i\}$ from $\{\mathbf{x}_h^i\}$ using Algorithm 3.
5: Calculate objective functions $\mathbf{f} = fun(\{\mathbf{x}_l^i\})$.
6: Find non-dominated solutions of $\{\mathbf{x}_l^i\} \cup \mathbf{A}_l$, and save to the external archive $\mathbf{A}_l$.
7: Find the members in $\{\mathbf{b}_h^i\}$ that are corresponding to the solutions in $\mathbf{A}_l$ and save to $\mathbf{A}_h$.
8: If the number of members in $\mathbf{A}_l$ and $\mathbf{A}_h$ exceeds n_A, remove some of the solutions using an archiving technique.
9: End

4 Numerical test

To examine the performance of the newly proposed multiobjective optimiser for aeroelastic optimisation of an aircraft wing, its performance in exploring the Pareto front of the aeroelastic multiobjective optimisation problem is tested and compared to other popular multiobjective meta-heuristics. The details of the implemented optimisers with control parameter settings are:

- Multi-objective grey wolf optimiser (MGWO) [52]: parameters including grid inflation (α), leader selection pressure (β) and extra repository member selection pressure (γ) are set as 0.1, 4 and 2 respectively.
- Multi-objective Salp Swarm Algorithm (MSSA) [53]: code and parameter setting are provided by author
- Multi-objective evolutionary algorithm based on decomposition (MOEA/D) [28]: the crossover parameter (γ) is set as 0.5.
- Non-dominated sorting genetic algorithm version II (NSGA-II) [27]: crossover percentage (pC), mutation percentage (pM) and mutation rate (μ) are set as 0.5, 0.5 and 0.02 respectively.
- Non-dominated sorting genetic algorithm version III (NSGA-III) [54]: crossover percentage (pC), mutation percentage (pM) and mutation rate (μ) are set as 0.5, 0.5 and 0.02 respectively.
- Multi-objective meta-heuristic with iterative parameter distribution estimation (MM-IPDE): lower and upper bound of adaptive parameters; F_1, F_u, CR_1, CR_u and *MutationSchemeno*. Are set as [1 2], [0.75 1.5], [0.1 0.5], [0.75 1.0] and [1 5] respectively.

Table 3 IGD results of benchmark tests

IGD		MGWO	MSSA	MOEA/D	NSGA-II	NSGA-III	MM-IPDE
ZDT1 (2,300)	Mean	0.00315	0.19601	0.12454	0.15657	0.16957	**0.00045**
	Std	0.00106	0.00956	0.03050	0.00503	0.00528	**1.30E−05**
	Rank	2	6	3	4	5	**1**
	Friedman's Rank	2	5.96667	3.23333	3.9	4.9	**1**
ZDT2 (2,300)	Mean	0.06408	0.27889	0.33391	0.24357	0.22137	**0.00289**
	Std	0.02405	0.02376	0.02171	**0.00715**	0.01151	0.01331
	Rank	2	5	6	4	3	**1**
	Friedman's Rank	1.98333	4.83333	5.96667	4.03333	3.16667	**1.01667**
ZDT3 (2,300)	Mean	0.00233	0.18111	0.10752	0.13970	0.13335	**0.00056**
	Std	0.00109	0.01413	0.02436	0.00876	0.00315	**0.00002**
	Rank	2	6	3	5	4	**1**
	Friedman's Rank	2	6	3.3	4.66667	4.03333	**1**
ZDT4 (2,100)	Mean	96.60904	37.43138	125.63168	92.72423	16.71107	**8.06025**
	Std	11.79122	15.15860	10.77063	9.66952	**1.46524**	1.81456
	Rank	5	3	6	4	2	**1**
	Friedman's Rank	4.76667	2.9	5.96667	4.26667	2.1	**1**
ZDT6 (2,100)	Mean	0.01088	0.65720	0.72171	0.60898	0.53404	**0.00034**
	Std	0.00631	0.01606	0.01673	0.01104	0.01654	**0.00004**
	Rank	2	5	6	4	3	**1**
	Friedman's Rank	2	5.0	6.0	4	3	**1**
DTLZ1 (3,12)	Mean	3.28894	2.86708	1.47970	1.17621	0.04863	**0.02258**
	Std	0.38619	0.66362	0.48534	0.26570	**0.01440**	0.02278
	Rank	6	5	4	3	2	**1**
	Friedman's Rank	5.7	5.2	3.8	3.3	1.8	**1.2**
DTLZ2 (3,12)	Mean	0.03765	0.04351	**0.00460**	0.16060	0.00626	0.00491
	Std	0.00581	0.00693	0.00035	0.01809	**0.00027**	0.00101
	Rank	4	5	**1**	6	3	2
	Friedman's Rank	4.23333	4.76667	**1.4**	6.0	2.9	1.7
DTLZ3 (3,12)	Mean	10.17099	7.35801	5.41782	2.85469	0.14484	**0.01481**
	Std	1.31088	2.37738	1.73314	0.66636	0.08726373	**0.015554**
	Rank	6	5	4	3	2	**1**
	Friedman's Rank	5.93333	4.73333	4.16667	3.16667	2	**1**
DTLZ4 (3,120)	Mean	0.12596	0.09004	**0.00531**	0.12741	0.01954	0.00767
	Std	0.02257	0.04765	**0.00098**	0.06611	0.01123	0.00244
	Rank	5	4	**1**	6	3	2
	Friedman's Rank	5.43333	4.53333	**1.23333**	5.03333	2.73333	2.03333
DTLZ5 (3,120)	Mean	0.03878	0.02992	**0.00106**	0.09881	0.00322	0.00303
	Std	0.00605	0.00600	0.00032	0.01691	**0.00027**	0.00119
	Rank	5	4	**1**	6	3	2
	Friedman's Rank	4.96667	4.03333	**1.03333**	6	2.46667	2.5
DTLZ6 (3,120)	Mean	0.02219	2.94127	0.51680	5.45116	4.56592	**0.00013**
	Std	0.03555	0.10544	0.33909	0.07536	0.11351	**1.14E−05**
	Rank	2	4	3	6	5	**1**
	Friedman's Rank	2.03333	4	2.96667	6	5	**1**
DTLZ7 (3,120)	Mean	0.01395	0.35887	0.23067	0.25214	0.20879	**0.00266**
	Std	0.01726	0.02286	0.04396	0.02289	0.00633	**0.00007**
	Rank	2	6	4	5	3	**1**
	Friedman's Rank	2	6	4.03333	4.63333	3.33333	**1**

Table 3 (continued)

IGD		MGWO	MSSA	MOEA/D	NSGA-II	NSGA-III	MM-IPDE
UF1 (2,30)	Mean	0.01366	0.01443	0.01699	**0.01206**	0.01333	0.01304
	Std	0.00067	0.00077	0.00555	0.00058	**0.00022**	0.00035
	Rank	4	5	6	**1**	3	2
	Friedman's Rank	3.9	5.0	5.1	**1.10000**	3.26667	2.63333
UF2 (2,30)	Mean	0.00926	0.01351	0.01002	0.00825	0.00711	**0.00601**
	Std	0.00086	0.00144	0.00218	0.00111	**0.00054**	0.00100
	Rank	4	6	5	3	2	**1**
	Friedman's Rank	4.30000	5.83333	4.23333	3.26667	2.13333	**1.23333**
UF3 (2,30)	Mean	0.03016	0.05212	0.04632	0.02699	0.03793	**0.01521**
	Std	0.00706	0.00552	0.00754	0.00416	**0.00193**	0.00434
	Rank	3	6	5	2	4	**1**
	Friedman's Rank	2.86667	5.76667	5.03333	2.36667	3.9	**1.06667**
UF4 (2,30)	Mean	0.01537	0.01828	0.01490	0.01599	0.00980	**0.00954**
	Std	0.00115	0.00188	0.00109	0.00068	0.00034	**0.00026**
	Rank	4	6	3	5	2	**1**
	Friedman's Rank	3.90000	5.96667	3.63333	4.5	1.73333	**1.26667**
UF5 (2,30)	Mean	0.31097	0.22589	0.27759	0.24947	0.18509	**0.04395**
	Std	0.07738	0.03641	0.08602	0.03525	0.02079	**0.00471**
	Rank	6	3	5	4	2	**1**
	Friedman's Rank	5.06667	3.6	4.43333	4.4	2.5	**1**
UF6 (2,30)	Mean	0.04581	0.04897	0.07883	0.04653	0.02290	**0.01514**
	Std	0.00720	0.00321	0.02355	0.00780	0.01087	**0.00218**
	Rank	3	5	6	4	2	**1**
	Friedman's Rank	3.86667	4.5	5.8	3.7	1.76667	**1.36667**
UF7 (2,30)	Mean	0.01061	0.01165	0.01397	0.00917	0.01067	**0.00915**
	Std	0.00088	0.00087	0.01258	0.00096	**0.00036**	0.00053
	Rank	3	5	6	2	4	**1**
	Friedman's Rank	4.06667	5.43333	3.66667	1.93333	4.13333	**1.76667**
UF8 (2,30)	Mean	0.01765	0.01960	0.01789	0.01889	0.02016	**0.01050**
	Std	0.00197	0.00151	0.00228	0.00111	0.00035	**0.00021**
	Rank	2	5	3	4	6	**1**
	Friedman's Rank	2.86667	4.46667	3.46667	3.96667	5.23333	**1**
UF9 (2,30)	Mean	0.01122	0.01776	0.01120	0.01586	0.01205	**0.00947**
	Std	0.00030	0.00125	0.00251	0.00192	0.00044	**0.00006**
	Rank	3	6	2	5	4	**1**
	Friedman's rank	2.93333	5.7	2.33333	5.16667	3.86667	**1**
UF10 (2,30)	Mean	0.04062	0.11780	0.05652	0.09886	0.02829	**0.01677**
	Std	0.01724	0.02175	0.01062	0.01244	0.00490	**0.00010**
	Rank	3	6	4	5	2	**1**
	Friedman's Rank	2.93333	5.73333	3.86667	5.23333	2.23333	**1**

The best statistical results of each problem are highlighted in bold

Moreover, to investigate the ability of the proposed optimiser for general-purpose multiobjective optimisation, the method along with the abovementioned established algorithms are used to solve several standard test suites for multiobjective optimisation. These include the ZDT [55], DTLZ [56] and UF [57] test instances. The total numbers of function evaluations (FEs) are set as a termination condition for the optimisers, which are 50,000 FEs for ZDTs and UF1-7, 100,000 FEs for DTLZs and UF8-10, and 40,000 FE for the aeroelastic optimisation problem. The population size and the archive size are equally set as 100 for ZDTs and UF1-7, 200 for DTLZs and UF8-10, and 50 for the aeroelastic case study. Each optimisation algorithm is performed 30 and 10 independent runs for standard test problems and the aeroelastic optimisation problem respectively. The conditions and bound constraints for the standard test problems are set

Table 4 Summarise of IGD rank result of test problem with MOMHs

IGD Rank	MGWO	MSSA	MOEA/D	NSGA-II	NSGA-III	MM-IPDE
ZDT1	2	6	3	4	5	1
ZDT2	2	5	6	4	3	1
ZDT3	2	6	3	5	4	1
ZDT4	5	3	6	4	2	1
ZDT6	2	5	6	4	3	1
DTLZ1	6	5	4	3	2	1
DTLZ2	4	5	1	6	3	2
DTLZ3	6	5	4	3	2	1
DTLZ4	5	4	1	6	3	2
DTLZ5	5	4	1	6	3	2
DTLZ6	2	4	3	6	5	1
DTLZ7	2	6	4	5	3	1
UF1	4	5	6	1	3	2
UF2	4	6	5	3	2	1
UF3	3	6	5	2	4	1
UF4	4	6	3	5	2	1
UF5	6	3	5	4	2	1
UF6	3	5	6	4	2	1
UF7	3	5	6	2	4	1
UF8	2	5	3	4	6	1
UF9	3	6	2	5	4	1
UF10	3	6	4	5	2	1
Mean	3.5455	5.0455	3.9545	4.1364	3.1364	**1.1818**
Rank	3	6	5	4	2	**1**

The best mean rank and overall rank are highlighted in bold

following [58–61]. Two performance indicators, a hypervolume (HV) and an inverted generational distance (IGD) are used for performance evaluation. The hypervolume determines the area (in 2-dimensional cases) or hypervolume (in M-dimensional cases) covered by a Pareto front obtained from running an optimiser with respect to a predefined reference point. The HV value of a front P can be computed as

$$\mathrm{HV} = \bigcup_{i=1}^{|P|} \mathrm{HV}_i \tag{9}$$

where $|P|$ is the number of solution members in front P. HV_i is a hypervolume of the ith solution and the reference point computed on the objective function domain. The references points for the standard test problems can be found in their references. The reference point is defined so that it must not dominate any points in the compared Pareto fronts. The use of union hypervolume implies that some of the hypervolumes can overlap each other while the overlapped parts will count only once. For the indicator, the higher HV the better front for both front advancement and extension.

For IGD, the true Pareto front is needed for its computation, thus, this indicator cannot be used in the case of aeroelastic optimization as the true Pareto front is not provided. The indicator can be computed as

$$\mathrm{IGD} = \frac{\sum_{\mathbf{f} \in P^*} \mathrm{Dist}(\mathbf{f}, P)}{|P^*|} \tag{10}$$

where P^* is the true Pareto front and P is the measured Pareto front. Dist($\mathbf{f}$, P) is the minimum distance from $\mathbf{f}$, a member in P^*, to all of the members in P. IGD can measure both front advancement and extension as with HV, however, the lower IGD the better front. The optimisers are statistically compared. The mean values of HV and IGD are used to measure their convergence rate and consistency. In cases that the mean values are insignificantly different, the standard deviation is used to decide the better optimiser. In addition, the Friedman rank-sum test is used to rank the optimisers for all case studies.

5 Results and discussion

Before assessing the performance of MM-IPDE for wing aeroelastic optimisation, all of the abovementioned algorithms are used to solve 3 sets of the standard test problems

Table 5 HV results of benchmark tests

HV		MGWO	MSSA	MOEA/D	NSGA-II	NSGA-III	MM-IPDE
ZDT1 (2,300)	Mean	0.90957	0.31708	0.49208	0.41919	0.39810	**0.91952**
	Std	0.01021	0.01449	0.09002	0.01234	0.00635	**5.4E−05**
	Rank	2	6	3	4	5	**1**
	Friedman's Rank	2	6	3.43333	3.86667	4.7	**1**
ZDT2 (2,300)	Mean	0.78299	0.24086	0.10943	0.30947	0.37700	**0.85330**
	Std	0.01496	0.06545	0.04629	0.01607	0.01538	**0.01289**
	Rank	2	5	6	4	3	**1**
	Friedman's Rank	1.98333	4.83333	5.96667	4.1	3.1	**1.01667**
ZDT3 (2,300)	Mean	0.82818	0.28222	0.40870	0.38825	0.36529	**0.83369**
	Std	0.00499	0.01144	0.07330	0.01521	0.00523	**0.00008**
	Rank	2	6	3	4	5	**1**
	Friedman's Rank	2	6	3.8	3.56667	4.63333	**1**
ZDT4 (2,100)	Mean	0.54370	0.83482	0.17684	0.56237	0.94136	**0.98310**
	Std	0.05870	0.07830	0.11219	0.04794	**0.00729**	0.00919
	Rank	5	3	6	4	2	**1**
	Friedman's Rank	4.73333	2.9	6	4.26667	2.1	**1**
ZDT6 (2,100)	Mean	0.88470	0.20477	0.11605	0.25993	0.35277	**0.93550**
	Std	0.03260	0.02385	0.03645	0.01330	0.01957	**0.00016**
	Rank	2	5	6	4	3	**1**
	Friedman's Rank	2	5.03333	5.96667	4	3	**1**
DTLZ1 (3,12)	Mean	0.95243	0.96339	0.99185	0.99771	1.00000	**1.00000**
	Std	0.00659	0.00842	0.00462	0.00119	6.7871E−07	**2.74E−07**
	Rank	6	5	4	3	2	**1**
	Friedman's Rank	5.83333	5.16667	3.9	3.1	1.9	**1.1**
DTLZ2 (3,12)	Mean	0.95997	0.99298	0.99969	0.99191	0.99978	**0.99986**
	Std	0.01639	0.00274	0.00045	0.00158	3.1703E−05	**9.03E−06**
	Rank	6	4	3	5	2	**1**
	Friedman's Rank	5.93333	4.33333	2.1	4.73333	2.6	**1.3**
DTLZ3 (3,12)	Mean	0.90081	0.95069	0.98639	0.99669	1.00000	**1.00000**
	Std	0.03164	0.01335	0.00981	0.00178	1.0058E−06	**2.2E−08**
	Rank	6	5	4	3	2	**1**
	Friedman's Rank	6	4.96667	3.93333	3.1	2	**1**
DTLZ4 (3,120)	Mean	0.99746	0.98828	0.99981	0.99666	0.99217	**0.99992**
	Std	0.00133	0.00475	0.00034	0.00368	0.01403	**0.00001**
	Rank	3	6	2	4	5	**1**
	Friedman's Rank	4.46667	5.66667	1.93333	4	3.66667	**1.26667**
DTLZ5 (3,120)	Mean	0.93619	0.95679	0.98180	0.94922	0.98840	**0.98883**
	Std	0.04023	0.01062	0.00356	0.00852	**0.00009**	0.00029
	Rank	6	4	3	5	2	**1**
	Friedman's Rank	5.03333	4.63333	2.96667	5.3	1.96667	**1.1**
DTLZ6 (3,120)	Mean	0.99948	0.91121	0.99743	0.55264	0.70252	**0.99991**
	Std	0.00029	0.00325	0.00240	0.01108	0.01337	**1.62E-09**
	Rank	2	4	3	6	5	**1**
	Friedman's Rank	2.06667	4	2.93333	6	5	**1**
DTLZ7 (3,120)	Mean	0.77285	0.16270	0.24295	0.30567	0.37313	**0.81891**
	Std	0.03053	0.03265	0.06894	0.03387	0.01134	**0.00037**
	Rank	2	6	5	4	3	**1**
	Friedman's Rank	2	5.9	4.9	4.13333	3.06667	**1**

Table 5 (continued)

HV		MGWO	MSSA	MOEA/D	NSGA-II	NSGA-III	MM-IPDE
UF1 (2,30)	Mean	0.70746	0.70166	0.66815	0.71655	0.73502	**0.73906**
	Std	0.01364	0.00981	0.03881	0.00806	**0.00108**	0.00273
	Rank	4	5	6	3	2	**1**
	Friedman's Rank	4.2	4.83333	5.5	3.46667	1.96667	**1.03333**
UF2 (2,30)	Mean	0.88260	0.84186	0.84521	0.87846	0.88656	**0.89511**
	Std	0.00383	0.01201	0.02387	0.01465	**0.00592**	0.00764
	Rank	3	6	5	4	2	**1**
	Friedman's Rank	3.13333	5.46667	5.26667	3.3	2.4	**1.43333**
UF3 (2,30)	Mean	0.93765	0.78395	0.68276	0.80819	0.76090	**0.95754**
	Std	0.02010	0.02647	0.04068	0.03668	0.01172	**0.00477**
	Rank	2	4	6	3	5	**1**
	Friedman's Rank	1.93333	3.86667	5.93333	3.46667	4.73333	**1.06667**
UF4 (2,30)	Mean	0.36419	0.32375	0.36202	0.36071	0.45056	**0.45099**
	Std	0.01537	0.01148	0.01281	0.00811	0.00600	**0.00386**
	Rank	3	6	4	5	2	**1**
	Friedman's Rank	3.96667	6	4	4.03333	1.56667	**1.43333**
UF5 (2,30)	Mean	0.55842	0.68095	0.52602	0.57949	0.62304	**0.93900**
	Std	0.10303	0.03609	0.14535	0.05789	0.03947	**0.02380**
	Rank	5	2	6	4	3	**1**
	Friedman's Rank	4.46667	2.53333	4.86667	4.5	3.63333	**1**
UF6 (2,30)	Mean	0.64076	0.64825	0.44621	0.67484	0.80189	**0.84101**
	Std	0.03582	0.01815	0.12200	0.06684	0.06082	**0.01898**
	Rank	5	4	6	3	2	**1**
	Friedman's Rank	4.16667	4.2	5.9	3.63333	1.83333	**1.26667**
UF7 (2,30)	Mean	0.66237	0.63993	0.62087	0.67358	0.68458	**0.69722**
	Std	0.01473	0.01215	0.09017	0.00746	0.00893	**0.00271**
	Rank	4	5	6	3	2	**1**
	Friedman's Rank	3.9	5.53333	5	3.16667	2.36667	**1.03333**
UF8 (2,30)	Mean	0.84419	0.81292	0.85461	0.85540	0.88050	**0.89119**
	Std	0.05539	0.01023	0.01276	0.00630	0.00222	**0.00023**
	Rank	5	6	4	3	2	**1**
	Friedman's Rank	3.93333	5.86667	4	4.13333	2.06667	**1**
UF9 (2,30)	Mean	0.96508	0.91342	0.95292	0.95084	0.95951	**0.98900**
	Std	0.00638	0.01248	0.01772	0.01008	0.00667	**0.00026**
	Rank	2	6	4	5	3	**1**
	Friedman's Rank	2.5	5.93333	3.83333	4.4	3.33333	**1**
UF10 (2,30)	Mean	0.85617	0.81416	0.78581	0.74626	0.91962	**0.98566**
	Std	0.04190	0.02769	0.05559	0.03537	0.03712	**0.00008**
	Rank	3	4	5	6	2	**1**
	Friedman's Rank	3.26667	4.23333	4.83333	5.53333	2.13333	**1**

The best statistical results of each problem are highlighted in bold

for multiobjective optimisation including ZDT1-4 and ZDT6, DTZL1-7 and UF1-10 with 30 independent runs for each problem. The results of the MM-IPDE are compared to the competitors to ensure flexibility of the algorithm. The benchmark test numbers, for example, ZDT1 (M, n), mean the test problem having M objective functions and n design variables. The performance metrics for measuring convergence of the optimisers are Inverted Generational Distance (IGD) [19, 53] and hypervolume (HV) indicators [39, 52, 62].

The IGD metric determines the average distance between a Pareto front obtained by a particular algorithm and the true Pareto front with the former being the reference front. Table 3 shows the comparative IGD results of MM-IPDE

Table 6 Summarise of HV rank result of test problem with MOMHs

HV Rank	MGWO	MSSA	MOEA/D	NSGA-II	NSGA-III	MM-IPDE
ZDT1	2	6	3	4	5	1
ZDT2	2	5	6	4	3	1
ZDT3	2	6	3	4	5	1
ZDT4	5	3	6	4	2	1
ZDT6	2	5	6	4	3	1
DTLZ1	6	5	4	3	2	1
DTLZ2	6	4	3	5	2	1
DTLZ3	6	5	4	3	2	1
DTLZ4	3	6	2	4	5	1
DTLZ5	6	4	3	5	2	1
DTLZ6	2	4	3	6	5	1
DTLZ7	2	6	5	4	3	1
UF1	4	5	6	3	2	1
UF2	3	6	5	4	2	1
UF3	2	4	6	3	5	1
UF4	3	6	4	5	2	1
UF5	5	2	6	4	3	1
UF6	5	4	6	3	2	1
UF7	4	5	6	3	2	1
UF8	5	6	4	3	2	1
UF9	2	6	4	5	3	1
UF10	3	4	5	6	2	1
Mean	3.6364	4.8636	4.5455	4.0455	2.9091	**1**
Rank	3	6	5	4	2	**1**

The best mean rank and overall rank are highlighted in bold

Table 7 Hypervolumes of Pareto fronts from MOMHs

Algorithm	Mean	Std	Friedman's rank	Rank
Multi-objective grey wolf optimiser (MGWO)	0.89411	0.00562	1.8	2
Multi-objective Salp Swarm Algorithm (MSSA)	0.88321	1.1E−16	3.4	3
Multi-objective evolutionary algorithm based on decomposition (MOEA/D)	0.30780	0.06782	6	6
Non-dominated sorting genetic algorithm version II (NSGA-II)	0.61019	0.03143	5	5
Non-dominated sorting genetic algorithm version III (NSGA-III)	0.83772	0.05315	3.2	4
Multi-objective meta-heuristic with iterative parameter distribution estimation (MM-IPDE)	**0.89652**	0.00384	**1.6**	**1**

The best statistical results of each problem are highlighted in bold

with all competitors. From the table, the mean IGD and Friedman's rank results are coincided i.e. MM-IPDE provided the best mean results in 18 out of 22 test problems. The second best is MOEA/D with 3 of 22 best mean IGD results in DTLZ2, DTLZ4 and DTLZ5. The third best is NSGA-II being the best for solving the UF1 problem. For the standard deviation of IGD, MM-IPDE achieves 12 best standard deviations (Std) while NSGA-III, MOEA/D and NSGA-II provided 8, 1 and 1 best Stds, respectively. The ranks based on the mean IGD of all optimisers for all test problems are provided in Table 4. From the mean rank results at the bottom of the table, MM-IPDE is obviously the best algorithm in these benchmark test sets. The best mean rank provided by MM-IPDE is 1.1818 while the second and the third-best ranks found by NSGA-III and MGWO are 3.1364 and 3.5455, respectively.

Moreover, the performance of MOMHs according to the convergence rate can be found based on HV. The comparative results of HV are given in Table 5. The mean and Friedman's ranks of the HV results give a similar conclusion where the best mean HV of all problems are achieved by MM-IPDE. MM-IPDE also provided 18 out of 22 best

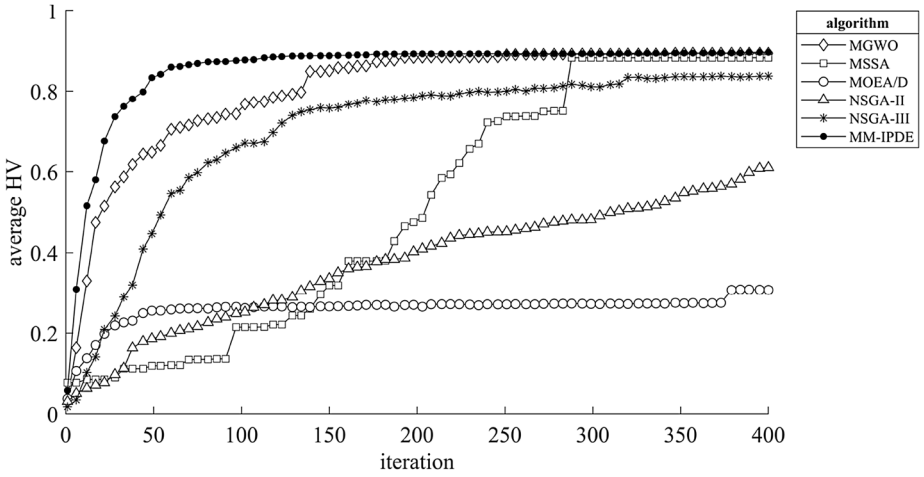

Fig. 2 Average hypervolume history

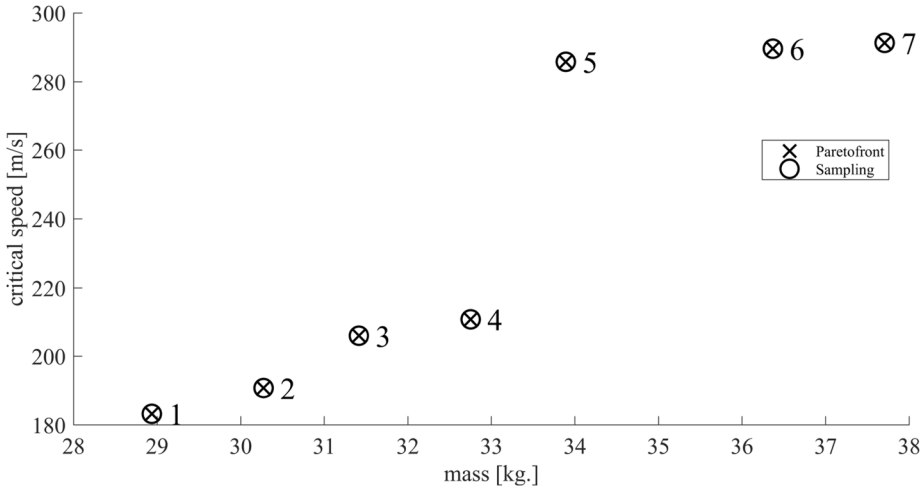

Fig. 3 Pareto front solution of MM-IPDE

standard deviations of HV while the best standard deviation of the rest is found by NSGA-III. The ranks of mean HV of all optimisers for all problems are provided in Table 6. The best average rank found by MM-IPDE is 1 while the average ranks of the runner-up and the third best are 2.9091 (NSGAIII) and 3.6364 (MGWO) respectively. Overall results of IGD and HV are similar, the results of both metrics confirm that the search performance, precision and flexibility of MM-IPDE are superior to some state-of-the-art algorithms. MM-IPDE can outperform all optimisers in these benchmark test sets.

For the aeroelastic optimisation problem, 10 independent runs of six optimisers including MGWO, MSSA, MOEA/D, NSGA-II, NSGA-III and the proposed MM-IPDE are evaluated. The comparative results based on a hypervolume are shown in Table 7 where the reference point for computing a hypervolume is $f_1 = 58.16972$ and $f_2 = -118.9267$ as the total mass and a critical speed. The search histories of all the optimisers as the plots of average HV from 10 runs versus iteration numbers are displayed in Fig. 2. The MM-IPDE algorithm proposed in this work can reach the optimum approximately after 100 iterations. MGWO and MSSA converge at around 200 and 300 iterations respectively. It can be seen that the rest fail to converge or reach the highest HV. It is shown that the proposed optimiser has a high convergence rate for the Goland wing aeroelastic optimisation.

In Table 7, the HV results of the problem from 10 independent runs of all algorithms are summarised. The rankings by means of HV averages and the Friedman's rank are coincided. The MM-IPDE is the best algorithm with best mean HV and Friedman's rank. The results from Table 7 also coincide with convergence results in Fig. 2 while MGWO and MSSA are the runner-up and the third best, respectively. MM-IPDE is proven to be the best algorithm in both speed and consistency aspects in both test problem sets.

The best Pareto front obtained from MM-IPDE is displayed in Fig. 3. Obtain optimum solution is present, the geometries of the wing structures of the solutions in the front are detailed in Table 8 by related to the number of solution 1–7 in Fig. 3. It should be noted that, according to the search history, the

Table 8 Design variables of optimum solutions from best Pareto front obtained by MM-IPDE

Optimum solution	No. 1		No. 2		No. 3	
	Thickness [m]	Orientation [deg]	Thickness [m]	Orientation [deg]	Thickness [m]	Orientation [deg]
Upper skin layer 1	0.0002	− 60	0.0003	− 60	0.0002	− 60
Upper skin layer 2	0.0002	60	0.0002	60	0.0002	45
Upper skin layer 3	0.0002	60	0.0002	60	0.0002	60
Lower skin layer 1	0.0002	− 45	0.0002	− 45	0.0002	− 45
Lower skin layer 2	0.0002	60	0.0002	60	0.0002	60
Lower skin layer 3	0.0002	− 45	0.0002	− 45	0.0002	− 45
Front spar	0.001	–	0.001	–	0.002	–
Mid spar	0.001	–	0.001	–	0.001	–
Rear spar	0.001	–	0.001	–	0.001	–
Ribs	0.001	–	0.001	–	0.001	–

Optimum solution	No. 4		No. 5		No. 6		No. 7	
	Thickness [m]	Orientation [deg]	Thickness [m]	Orientation [deg]	Thickness [m]	Orientation [deg]	Thickness [m]	Orientation [deg]
Upper skin layer 1	0.0002	− 60	0.0002	− 60	0.0002	− 60	0.0002	− 60
Upper skin layer 2	0.0002	45	0.0002	60	0.0002	60	0.0002	60
Upper skin layer 3	0.0002	60	0.0002	60	0.0002	90	0.0003	60
Lower skin layer 1	0.0002	− 45	0.0002	− 60	0.0002	− 30	0.0002	− 60
Lower skin layer 2	0.0003	60	0.0002	45	0.0002	45	0.0002	45
Lower skin layer 3	0.0002	− 45	0.0002	− 45	0.0002	− 45	0.0002	− 45
Front spar	0.002	–	0.003	–	0.004	–	0.004	–
Mid spar	0.001	–	0.001	–	0.001	–	0.001	–
Rear spar	0.001	–	0.001	–	0.001	–	0.001	–
Ribs	0.001	–	0.001	–	0.001	–	0.001	–

Table 9 Objective and constraint functions of optimum solutions from best Pareto front obtained by MM-IPDE (continue)

Optimum Solution	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7
Mass [kg]	28.9362	30.2740	31.4136	32.7514	33.8911	36.3685	37.7063
Critical speed [m/s]	183.227	190.761	205.952	210.702	285.829	289.621	291.319
Maximum deflection [m]	0.0926	0.0866	0.0964	0.0868	0.0925	0.0956	0.0860
Lift effectiveness [–]	0.9329	1.2344	1.2670	0.9760	2.4645	1.6355	2.0061

proposed optimiser converges since the early stage of an optimisation run. The reason there are a few Pareto optimal solutions is because the design variables are set to be discrete for manufacturability. If the design variables are set to be continuous, the higher number of Pareto optimal solutions can be expected. Those are regarded as optimal solutions. Therefore, a decision making will be used to select some of them for the next design stages. The details of the objective and constraint functions of all solutions in the front are provided in Table 9. The optimum solutions have a total weight between 28.93623 and 37.70629 kg while the critical speed is in between 183.227 and 291.319 m/s. The constraints, lift effectiveness and maximum of transverse displacement, are fulfilled for all solutions. In the wing that lifting surface do not have the control surface, the lift effectiveness can be over 1. For this reason, lift effectiveness of observed solutions are between 0.93291 and 2.4645. The transverse displacements of the optimum wing are in between 0.08604 and 0.09637 m.

6 Conclusions

A new multiobjective metaheuristic, MM-IPDE is proposed to solve aeroelastic optimisation of an aircraft wing. The design problem is posed to minimise structural mass and maximise wing flutter speed subject to structural safety and

aerodynamics performance constraints. The new optimiser, based on the dominance approach, exploits PBILM to iteratively estimate the control parameters for the reproduction of modified differential evolution. The idea of using EDA for estimating optimisation parameter of meta-heuristics is probably first presented in this work. Two new DE mutation schemes are also presented. The results from the 3 sets of the standard test problems show that MM-IPDE can outperform the other MOMHs. For the composite Goland wing optimisation results, MM-IPDE is still superior to the other MOMHs. This means that the newly proposed method is a powerful optimiser for both design of a real wing structure and a variety of benchmark test problems. The concept of using PBILM to iteratively estimate control parameters instead of other learning algorithms is effective and should have further investigation. For future work, MM-IPDE will be upgraded and used in more real engineering applications.

Acknowledgements The authors are grateful for the support from the Thailand Research Fund (RTA6180010) and the Royal Golden Jubilee Ph.D. program (PHD/0182/2559).

References

1. Cavagna L, Ricci S, Travaglini L (2011) NeoCASS: an integrated tool for structural sizing, aeroelastic analysis and MDO at conceptual design level. Prog Aerosp Sci 47:621–635. https://doi.org/10.1016/J.PAEROSCI.2011.08.006
2. Pathan MV, Patsias S, Tagarielli VL (2018) A real-coded genetic algorithm for optimizing the damping response of composite laminates. Comput Struct 198:51–60. https://doi.org/10.1016/J.COMPSTRUC.2018.01.005
3. Stanford BK, Jutte CV (2017) Comparison of curvilinear stiffeners and tow steered composites for aeroelastic tailoring of aircraft wings. Comput Struct 183:48–60. https://doi.org/10.1016/J.COMPSTRUC.2017.01.010
4. Nielsen MWD, Johnson KJ, Rhead AT, Butler R (2017) Laminate design for optimised in-plane performance and ease of manufacture. Compos Struct 177:119–128. https://doi.org/10.1016/J.COMPSTRUCT.2017.06.061
5. Li J, Narita Y (2014) Multi-objective design for aeroelastic flutter of laminated shallow shells under variable flow angles. Compos Struct 111:530–539. https://doi.org/10.1016/J.COMPSTRUCT.2014.01.026
6. Werter NPM, De Breuker R (2016) A novel dynamic aeroelastic framework for aeroelastic tailoring and structural optimisation. Compos Struct 158:369–386. https://doi.org/10.1016/J.COMPSTRUCT.2016.09.044
7. Caixeta PR, Marques FD (2018) Multiobjective optimization of an aircraft wing design with respect to structural and aeroelastic characteristics using neural network metamodel. J Braz Soc Mech Sci Eng 40:17. https://doi.org/10.1007/s40430-017-0958-7
8. Sleesongsom S, Bureerat S (2013) New conceptual design of aeroelastic wing structures by multi-objective optimization. Eng Optim 45:107–122. https://doi.org/10.1080/0305215X.2012.661728
9. Sleesongsom S, Bureerat S, Tai K (2013) Aircraft morphing wing design by using partial topology optimization. Struct Multidiscip Optim 48:1109–1128. https://doi.org/10.1007/s00158-013-0944-3
10. Stodieck O, Cooper JE, Neild SA et al (2018) Slender-wing beam reduction method for gradient-based aeroelastic design optimization. AIAA J 56:4529–4545. https://doi.org/10.2514/1.J056952
11. Kennedy G, Kenway G, Martins J (2014) Towards gradient-based design optimization of flexible transport aircraft with flutter constraints. In: 15th AIAA/ISSMO multidisciplinary analysis and optimization conference, American Institute of Aeronautics and Astronautics, Reston, Virginia. https://doi.org/10.2514/6.2014-2726
12. Zhu W, Miao K, Li D (2019) Static aeroelastic models with integrated stiffness-contributing shell structures built by additive manufacturing. Eng Struct 187:352–361. https://doi.org/10.1016/J.ENGSTRUCT.2019.02.066
13. Huo SH, Wang FS, Yuan Z, Yue ZF (2013) Composite wing elastic axis for aeroelasticity optimization design. Aircr Eng Aerosp Technol 85:10–15. https://doi.org/10.1108/00022661311294030
14. Kaveh A, Bakhshpoori T (2016) Water evaporation optimization: a novel physically inspired optimization algorithm. Comput Struct 167:69–85. https://doi.org/10.1016/J.COMPSTRUC.2016.01.008
15. Kaveh A, Farhoudi N (2013) A new optimization method: dolphin echolocation. Adv Eng Softw 59:53–70. https://doi.org/10.1016/J.ADVENGSOFT.2013.03.004
16. Kaveh A, Khayatazad M (2012) A new meta-heuristic method: ray optimization. Comput Struct 112–113:283–294. https://doi.org/10.1016/J.COMPSTRUC.2012.09.003
17. Kaveh A, Zolghadr A (2016) A novel meta-heuristic algorithm: tug of war optimization. Iran Univ Sci Technol 6:469–492
18. Kaveh A, Kaveh A, Ilchi Ghazaan M (2017) A new meta-heuristic algorithm: vibrating particles system. Sci Iran 24:551–566. https://doi.org/10.24200/sci.2017.2417
19. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27:1053–1073. https://doi.org/10.1007/s00521-015-1920-1
20. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98. https://doi.org/10.1016/J.ADVENGSOFT.2015.01.010
21. Kaveh A, Talatahari S (2009) Hybrid algorithm of harmony search. Particle Swarm and Ant Colony for Structural Design Optimization. Springer, Berlin, Heidelberg, pp 159–198
22. Kaveh A, Talatahari S (2008) A hybrid particle swarm and ant colony optimization for design of truss structures. ASIAN J Civ Eng (Build Hous) 9:329–348
23. Kaveh A, Talatahari S (2010) Optimal design of Schwedler and ribbed domes via hybrid Big Bang-Big Crunch algorithm. J Constr Steel Res 66:412–419. https://doi.org/10.1016/J.JCSR.2009.10.013
24. Kaveh A, Bakhshpoori T, Afshari E (2014) An efficient hybrid Particle Swarm and Swallow Swarm Optimization algorithm. Comput Struct 143:40–59. https://doi.org/10.1016/J.COMPSTRUC.2014.07.012
25. Pholdee N, Bureerat S, Yıldız AR (2017) Hybrid real-code population-based incremental learning and differential evolution for many-objective optimisation of an automotive floor-frame. Int J Veh Des 73:20. https://doi.org/10.1504/IJVD.2017.082578
26. Wan Z, Zhang B, Du Z, Yang C (2014) Aeroelastic two-level optimization for preliminary design of wing structures considering robust constraints. Chin J Aeronaut 27:259–265. https://doi.org/10.1016/J.CJA.2014.02.018
27. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197. https://doi.org/10.1109/4235.996017
28. Zhang Q, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11:712–731. https://doi.org/10.1109/TEVC.2007.892759

29. Zitzler E, Laumanns M, Thiele L (2001) SPEA2: improving the strength pareto evolutionary algorithm. TIK-Report. https://doi.org/10.3929/ETHZ-A-004284029
30. Goland M (1945) The flutter of a uniform cantilever wing. J Appl Mech Asme 12:A197–A208
31. Beran PS, Strganac TW, Kim K, Nichkawde C (2004) Studies of store-induced limit-cycle oscillations using a model with full system nonlinearities. Nonlinear Dyn 37:323–339. https://doi.org/10.1023/B:NODY.0000045544.96418.bf
32. Beran PS, Khot NS, Eastep FE et al (2004) Numerical analysis of store-induced limit-cycle oscillation. J Aircr 41:1315–1326. https://doi.org/10.2514/1.404
33. Kefal A, Oterkus E, Tessler A, Spangler JL (2016) A quadrilateral inverse-shell element with drilling degrees of freedom for shape sensing and structural health monitoring. Eng Sci Technol an Int J 19:1299–1313. https://doi.org/10.1016/J.JESTCH.2016.03.006
34. Katz J, Plotkin A (2004) Low-speed aerodynamics. Second Edition J Fluids Eng 126:293. https://doi.org/10.1115/1.1669432
35. Harder RL, Desmarais RN (1972) Interpolation using surface splines. J Aircr 9:189–191. https://doi.org/10.2514/3.44330
36. Haddadpour H, Firouz-Abadi RD (2006) Evaluation of quasi-steady aerodynamic modeling for flutter prediction of aircraft wings in incompressible flow. Thin-Walled Struct 44:931–936. https://doi.org/10.1016/J.TWS.2006.08.020
37. Li B, Li J, Tang K, Yao X (2015) Many-objective evolutionary algorithms. ACM Comput Surv 48:1–35. https://doi.org/10.1145/2792984
38. Baluja S (1994) Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Carnegie Mellon University, Pittsburgh, PA
39. Pholdee N, Bureerat S (2013) Hybridisation of real-code population-based incremental learning and differential evolution for multiobjective design of trusses. Inf Sci (Ny) 223:136–152. https://doi.org/10.1016/j.ins.2012.10.008
40. Noilublao N, Bureerat S (2013) Simultaneous topology, shape, and sizing optimisation of plane trusses with adaptive ground finite elements using MOEAs. Math Probl Eng 2013:1–9. https://doi.org/10.1155/2013/838102
41. Bureerat S, Srisomporn S (2010) Optimum plate-fin heat sinks by using a multi-objective evolutionary algorithm. Eng Optim 42:305–323. https://doi.org/10.1080/03052150903143935
42. Gao J, Wang J (2010) WBMOAIS: a novel artificial immune system for multiobjective optimization. Comput Oper Res 37:50–61. https://doi.org/10.1016/J.COR.2009.03.009
43. Wansaseub K, Pholdee N, Bureerat S (2017) Optimal U-shaped baffle square-duct heat exchanger through surrogate-assisted self-adaptive differential evolution with neighbourhood search and weighted exploitation-exploration. Appl Therm Eng 118:455–463. https://doi.org/10.1016/j.applthermaleng.2017.02.100
44. Tanabe R, Fukunaga A (2013) Evaluating the performance of SHADE on CEC 2013 benchmark problems. In: 2013 IEEE congress on evolutionary computation, CEC 2013, Cancun, Mexico, pp 1952–1959. https://doi.org/10.1109/CEC.2013.6557798
45. Niknam T, Azadfarsani E, Jabbari M (2012) A new hybrid evolutionary algorithm based on new fuzzy adaptive PSO and NM algorithms for Distribution Feeder Reconfiguration. Energy Convers Manag 54:7–16. https://doi.org/10.1016/j.enconman.2011.09.014
46. Piotrowski AP (2018) L-SHADE optimization algorithms with population-wide inertia. Inf Sci (Ny) 468:117–141. https://doi.org/10.1016/j.ins.2018.08.030
47. Bureerat S, Sleesongsom S (2020) Constraint handling technique for four-bar linkage path generation using self-adaptive teaching–learning-based optimization with a diversity archive. Eng Optim. https://doi.org/10.1080/0305215X.2020.1741566
48. Neri F, Tirronen V (2009) Scale factor local search in differential evolution. Memetic Comput 1:153–171. https://doi.org/10.1007/s12293-009-0008-9
49. Niknam T, Mojarrad HD, Nayeripour M (2010) A new fuzzy adaptive particle swarm optimization for non-smooth economic dispatch. Energy 35:1764–1778. https://doi.org/10.1016/j.energy.2009.12.029
50. Whitacre JM, Pham TQ, Sarker RA (2006) Use of statistical outlier detection method in adaptive evolutionary algorithms. In: Proceedings of the 8th annual conference on genetic and evolutionary computation - GECCO '06. ACM Press, New York, USA, pp 1345–1352. https://doi.org/10.1145/1143997.1144205
51. Srisomporn S, Bureerat S (2008) Geometrical design of plate-fin heat sinks using hybridization of MOEA and RSM. IEEE Trans Compon Packag Technol 31:351–360. https://doi.org/10.1109/TCAPT.2008.916799
52. Mirjalili S, Saremi S, Mirjalili SM, Coelho LDS (2016) Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. Expert Syst Appl 47:106–119. https://doi.org/10.1016/j.eswa.2015.10.039
53. Mirjalili S, Gandomi AH, Mirjalili SZ et al (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191. https://doi.org/10.1016/j.advengsoft.2017.07.002
54. Ibrahim A, Rahnamayan S, Martin MV, Deb K (2016) EliteNSGA-III: an improved evolutionary many-objective optimization algorithm. In: 2016 IEEE congress on evolutionary computation (CEC). IEEE, Vancouver, Canada, pp 973–982. https://doi.org/10.1109/CEC.2016.7743895
55. Zitzler E (1999) Evolutionary algorithms for multiobjective optimization: methods and applications. Swiss Federal Institute of Technology Zurich
56. Deb K, Thiele L, Laumanns M, Zitzler E (2002) Scalable multi-objective optimization test problems. In: Proceedings of the 2002 congress on evolutionary computation. CEC'02 (Cat. No.02TH8600), pp 825–830. https://doi.org/10.1109/CEC.2002.1007032
57. Zhang Q, Zhou A, Zhao S et al (2008) Multiobjective optimization test instances for the CEC 2009 special session and competition. https://www3.ntu.edu.sg/home/epnsugan/index_files/CEC09-MOEA/CEC09-MOEA.htm
58. Shim VA, Tan KC, Tan KK (2012) A hybrid adaptive evolutionary algorithm in the domination-based and decomposition-based frameworks of multi-objective optimization. In: 2012 IEEE congress on evolutionary computation, CEC 2012. IEEE. https://doi.org/10.1109/CEC.2012.6256485
59. Shim VA, Tan KC, Tang H (2015) Adaptive memetic computing for evolutionary multiobjective optimization. IEEE Trans Cybern 45:610–621. https://doi.org/10.1109/TCYB.2014.2331994
60. Liang Z, Song R, Lin Q et al (2015) A double-module immune algorithm for multi-objective optimization problems. Appl Soft Comput J 35:161–174. https://doi.org/10.1016/j.asoc.2015.06.022
61. Chen B, Lin Y, Zeng W et al (2015) Modified differential evolution algorithm using a new diversity maintenance strategy for multi-objective optimization problems. Appl Intell 43:49–73. https://doi.org/10.1007/s10489-014-0619-9
62. Emmerich M, Beume N, Naujoks B (2005) An EMO algorithm using the hypervolume measure as selection criterion. In: Lecture notes in computer science. Springer, Berlin, Heidelberg, pp 62–76. https://doi.org/10.1007/978-3-540-31880-4_5