Archives of Computational Methods in Engineering (2022) 29:5457–5492
https://doi.org/10.1007/s11831-022-09773-0

REVIEW ARTICLE

The Mosaic of Metaheuristic Algorithms in Structural Optimization

Nikos D. Lagaros[1] · Vagelis Plevris[2] · Nikos Ath. Kallioras[1,3]

Received: 9 December 2021 / Accepted: 22 May 2022 / Published online: 11 July 2022

Abstract
Metaheuristic optimization algorithms (MOAs) represent powerful tools for dealing with multi-modal nonlinear optimization problems. The considerable attention that MOAs have received over the last decade and especially when adopted for dealing with several types of structural optimization problems can be mainly credited to the advances achieved in computer science and computer technology rendering possible, among others, the solution of real-world structural design optimization cases in reasonable computational time. The primal scope of the study is to present a state-of-the-art review of past and current developments achieved so far in structural optimization problems dealt with MOAs, accompanied by a set of tests aiming to examine the efficiency of various MOAs in several benchmark structural optimization problems. For this purpose, 24 population-based state-of-the-art MOAs belonging in four classes, (i) swarm-based; (ii) physics-based; (iii) evolutionary-based; and (iv) human-based, are used for solving 11 single objective benchmark structural optimization test problems of different levels of complexity. The size of the problems employed varies, with the number of unknowns ranging from 3 to 328 and the number of constraint functions ranging from 2 to 264, related to the structural performance of the design with reference to deformation and stress limits.

1 Introduction

During the last decades, the architectural, design and construction (ADC) industry has shown excessive innovation both in the theoretical and its practical directions [1]. These innovations could not be made possible without the advancements in fields related to computational mechanics, which played a critical role [2]. These developments made it possible not only to provide solutions to complex traditional problems in engineering, but also to propose novel mathematical formulations and solving techniques for practical applications, leading to innovative, unique, economic and more environmental-friendly structural systems [3]. Nowadays, modern numerical tools are available to provide enormous capabilities to architects and engineers, by fulfilling the demands of the analysis and design procedures.

During the last three decades, metaheuristic optimization algorithms (MOAs) have conquered many areas of engineering optimization, structural design optimization problems included, due to the easiness of implementation, their simplified nature, and mainly due to their efficiency in dealing with NP-complete problems. Structural design optimization (SDO) explains the procedure of proposing improved designs of structures with respect to material or construction cost, manufacturability, structural performance, among other design criteria. Several researchers have tried to make a systematic review and organize the broad research literature on optimization algorithms in general, or metaheuristics in particular, applied to structural optimization problems. Sahab et al. [4] performed a review on traditional and modern structural optimization problems and solution techniques. Kashani et al. [5] did a similar review work, focusing on population-based optimization methods applied in structural engineering, while Bekdaş et al. [6] presenting a review on metaheuristic algorithms and their applications in civil

✉ Nikos D. Lagaros
nlagaros@central.ntua.gr

Vagelis Plevris
vplevris@qu.edu.qa

Nikos Ath. Kallioras
kallioras.nikos@gmail.com; info@infersence.com

1 Institute of Structural Analysis & Antiseismic Research, School of Civil Engineering, National Technical University of Athens, 9, Heroon Polytechniou Str., Zografou Campus, 15780 Athens, Greece

2 Department of Civil and Architectural Engineering, College of Engineering, Qatar University, P.O. Box: 2713, Doha, Qatar

3 Infersence, 1, Georgiou Mpakou Str., 11524 Athens, Greece

engineering optimization problems, highlighting the recent progress and the state-of-the-art developments in the field. On the other hand, Yang et al. [7] focused their review on the applications of metaheuristic algorithms in civil engineering problems, particularly.

Most real-world structural design optimization problems are expressed in standard mathematical terms through highly nonlinear and multimodal expressions. A single objective non-linear programming (NLP) problem can be formulated as follows:

$$\begin{aligned}
&\text{Optimize } f(s), \\
&\textit{Subject to} \\
&g_a(s) \leq 0,\ a = 1, 2, \ldots, m_a \\
&h_b(s) = 0,\ b = 1, 2, \ldots, m_b \\
&lb_j \leq s_j \leq ub_j,\ j = 1, 2, \ldots, n
\end{aligned} \tag{1}$$

where $f(s)$ is the objective function (e.g. minimizing the weight of the structure that is related to the material requirements, improving the structural performance characterized for instance by the modal characteristics of the structural system, etc.), $g_a(s)$ is the ath inequality constraint, $h_b(s)$ is the bth equality constraint, while lb_j and ub_j denote the lower and upper limits, respectively, of the jth component of the design variable vector s of size n. It has to be noted that in the majority of structural optimization problems, no equality constraints are used for the problem formulation.

The scope of the present study is two-fold; first to review the achievements of the past and to present the future challenges through the state-of-the-art development of MOAs when used for solving structural optimization problems (SOPs). Then, in the second part of the study, several well-known MOAs are tested into typical and large-scale benchmark structural optimization problems, where the common characteristics and the similarities among the chosen MOAs are also presented. There are three categories of SOPs, namely (a) sizing; (b) shape; and (c) topology optimization. In addition, when uncertainty is involved into the problem formulation, two general types of problems can be described, namely reliability-based structural optimization and robust design optimization problems [8]. The structure of the work begins with the presentation of the history of the integration of MOAs with the various types of structural optimization problems. Subsequently, the 24 MOAs chosen for being tested into 11 benchmark structural optimization problems are briefly described. The selected MOAs cover a wide range of metaheuristic algorithms with different characteristics and nature, from well-known and well-established algorithms to the most recent and most promising ones that represent the latest trends in this research field. The implementation of these algorithms relies on the MATLAB codes provided by the developers of the corresponding MOAs, while the special features of each algorithm implementation together with the basis of comparison are provided in the next chapter of this study. In the last chapter, the numerical tests are presented, classified into two groups; in the first one three well known truss-structure problems are presented together with the welded beam, pressure vessel and the tension–compression string design problems. In the second group, five large-scale sizing-shape structural optimization problems are investigated, taken from the International Student Competition in Structural Optimization (ISCSO 2015 to 2019) [9–13].

2 The History of MOAs in Structural Optimization

Over the last few decades, the so-called "metaheuristic techniques" have been developed, to provide near-to-optimum solutions to various problems [14]. They are especially tailored to hard optimization problems that are difficult or even impossible to be optimized by the exact-optimal techniques, such as linear programming (LP) [15], non-linear programming (NLP) [16], integer programming (IP) [17], and dynamic programming (DP) [18]. Initially, these new techniques were called "Heuristics" and each such algorithm was exclusively developed to handle a specific problem [19], without having any "global" problem solving properties. After a while, the researchers started to generalize these algorithms, building wider solving frameworks that could be used to solve wider problems of any nature. This group of global solvers is nowadays known as "Metaheuristics" [20].

There are two main classifications of the fundamental mechanisms used in MOAs: (i) diversification and (ii) intensification [21]. The main difference between these two is that diversification tries to diverge the search in an attempt to explore the entire solution space, while intensification simply pushes the search towards the already found best solutions. Metaheuristics can be classified according to various criteria, such as (i) population-based or single-solution-based; (ii) trajectory or discontinuous; (iii) memoryless or using memory; (iv) local-oriented or global-oriented, among others.

Figure 1 displays Metaheuristics of various types where they have been first classified according to whether they use a population of solutions (population-based metaheuristics) or not (single-based metaheuristics). Single-based metaheuristics use a single solution at each run while population-based ones maintain a set of solutions (population) at each run. As shown in the figure, the class of population-based metaheuristics can be further classified into four main categories: (i) swarm-based, (ii) physics-based, (iii) evolutionary-based, and (iv) human-based. Swarm-based methods use swarm intelligence (SI) approaches that mimic the behaviour of swarms in nature. Evolutionary-based

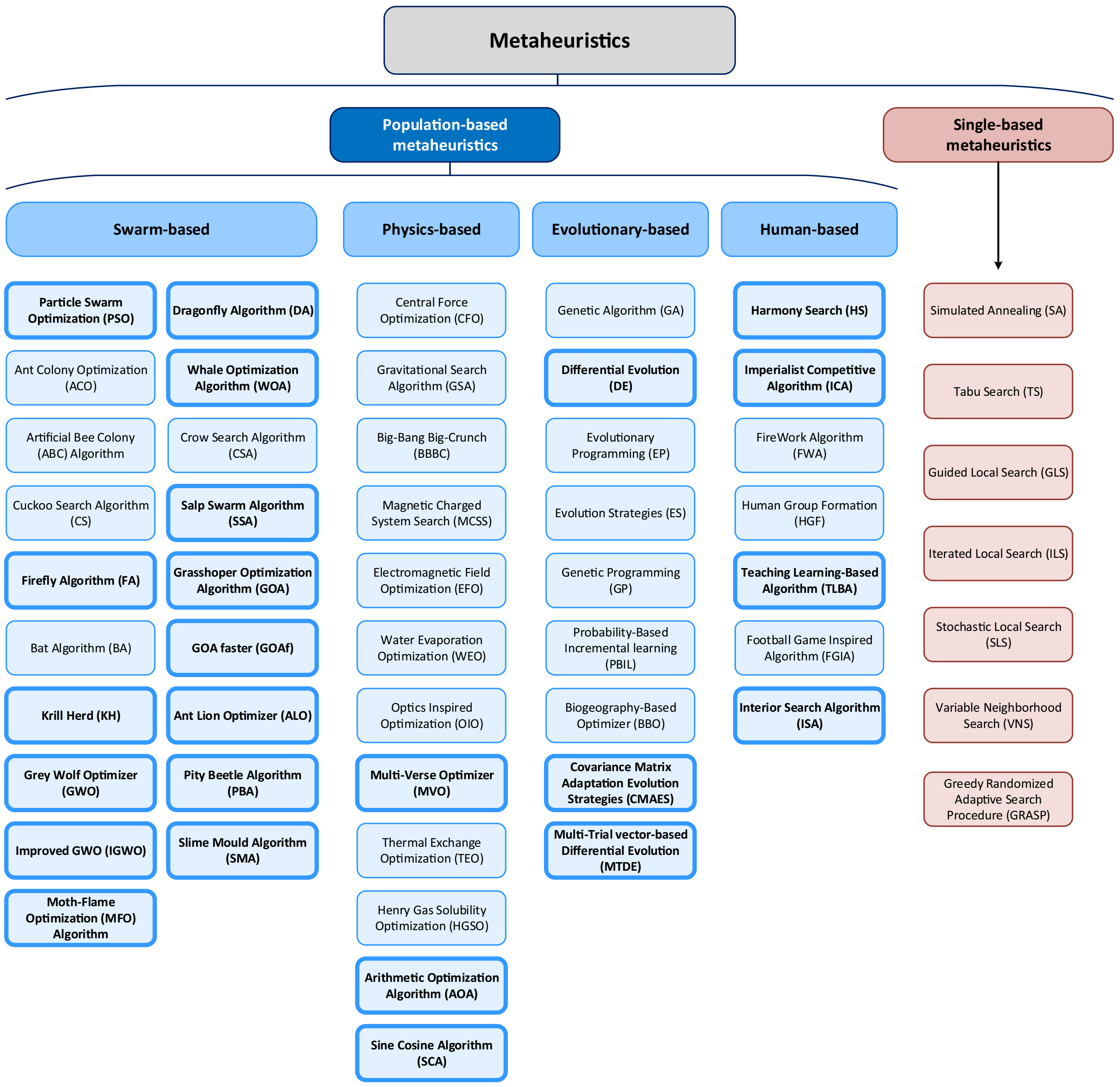

Metaheuristics
Population-based metaheuristics
Single-based metaheuristics
Swarm-based
Physics-based
Evolutionary-based
Human-based
Particle Swarm Optimization (PSO)
Dragonfly Algorithm (DA)
Ant Colony Optimization (ACO)
Whale Optimization Algorithm (WOA)
Artificial Bee Colony (ABC) Algorithm
Crow Search Algorithm (CSA)
Cuckoo Search Algorithm (CS)
Salp Swarm Algorithm (SSA)
Firefly Algorithm (FA)
Grasshoper Optimization Algorithm (GOA)
Bat Algorithm (BA)
GOA faster (GOAf)
Krill Herd (KH)
Ant Lion Optimizer (ALO)
Grey Wolf Optimizer (GWO)
Pity Beetle Algorithm (PBA)
Improved GWO (IGWO)
Slime Mould Algorithm (SMA)
Moth-Flame Optimization (MFO) Algorithm
Central Force Optimization (CFO)
Gravitational Search Algorithm (GSA)
Big-Bang Big-Crunch (BBBC)
Magnetic Charged System Search (MCSS)
Electromagnetic Field Optimization (EFO)
Water Evaporation Optimization (WEO)
Optics Inspired Optimization (OIO)
Multi-Verse Optimizer (MVO)
Thermal Exchange Optimization (TEO)
Henry Gas Solubility Optimization (HGSO)
Arithmetic Optimization Algorithm (AOA)
Sine Cosine Algorithm (SCA)
Genetic Algorithm (GA)
Differential Evolution (DE)
Evolutionary Programming (EP)
Evolution Strategies (ES)
Genetic Programming (GP)
Probability-Based Incremental learning (PBIL)
Biogeography-Based Optimizer (BBO)
Covariance Matrix Adaptation Evolution Strategies (CMAES)
Multi-Trial vector-based Differential Evolution (MTDE)
Harmony Search (HS)
Imperialist Competitive Algorithm (ICA)
FireWork Algorithm (FWA)
Human Group Formation (HGF)
Teaching Learning-Based Algorithm (TLBA)
Football Game Inspired Algorithm (FGIA)
Interior Search Algorithm (ISA)
Simulated Annealing (SA)
Tabu Search (TS)
Guided Local Search (GLS)
Iterated Local Search (ILS)
Stochastic Local Search (SLS)
Variable Neighborhood Search (VNS)
Greedy Randomized Adaptive Search Procedure (GRASP)

Fig. 1 The mosaic of metaheuristic optimization algorithms

algorithms or evolutionary algorithms (EA) are inspired by the evolutionary phenomena in nature and they usually use three operators: selection, recombination and mutation. Physics-based methods are inspired by and related to physical phenomena while Human-based methods are related to human activities and the human behaviour. The original figure with the classification of the algorithms, that appears in [22], includes 45 algorithms. In the new version of this figure (Fig. 1) we have added 9 additional algorithms and as a result the final scheme contains 54 algorithms in total. Of them, 24, denoted with bold letters and a thicker border in the figure, are examined in detail in the present study, as will be discussed in the next sections.

In this section we try to identify the various ways that MOAs have been used in structural optimization problems. These include benchmark structural optimization test problems for assessing the efficiency of new MOAs, implementations of existing MOAs for solving new formulations of structural optimization problems, methodologies for handling the excessive computational effort that MOAs require for solving structural optimization problems, and others.

Deterministic structural optimization problems that do not involve uncertainties of any kind can be classified into three broad categories: (i) sizing, (ii) shape, and (iii) topology optimization. In sizing optimization, usually the design variables have to do with some geometric characteristics of the cross sections of the members, for example in a truss or a frame structure, in 2D or 3D. In shape optimization, the optimizer can change the locations of the nodes or other control points of the geometry of the structure, changing this way the overall shape of the structure. In topology optimization, the optimizer can completely change the topology of the structure, by adding or removing elements, creating holes in slabs, etc. These three categories are not always discrete and clearly distinguishable from each other. Many times, we end up with mixed sizing-shape structural optimization problems, where both the geometric characteristics of the sections and the locations of some nodes are subject to change. In other cases, we have mixed shape-topology optimization problems where again the shape and the topology can be simultaneously changed by the optimizer.

2.1 New MOAs Assessed Through SOPs

The collection of metaheuristic algorithms is being continuously enriched with new methods and new, improved variations of existing methodologies. Researchers keep proposing new optimization schemes claiming that their performance is better than the one of other algorithms, at least in the optimization test examples examined. Many times, these test examples come from the field of structural engineering as these problems are usually some of the hardest to deal with. In other words, the performance of a new MOA is assessed through structural optimization problems, instead of the usual mathematical functions, or in addition to them.

Cuckoo search (CS) was originally presented by Yang and Deb [23] as a new metaheuristic based on the obligate brood parasitic behaviour of some cuckoo species combined with the Lévy flight behaviour of certain birds and fruit flies. The implementation of the algorithm involves Lévy flights with random steps, providing random walk capabilities to the method. In another study [24], the same authors applied the algorithm to solve engineering design optimisation problems, including the design of springs and welded beam structures. In addition, Gandomi et al. [25] assessed the validity and performance of the algorithm in handling SOPs. In particular, CS is assessed with several SOPs including the design of a pin-jointed plane frame with a fixed base, the minimization of the vertical deflection of an I-beam, the design of a piston component, the minimum-weight design of the corrugated bulkhead for a tanker, the design of a cantilever beam and a tubular column, a three-bar truss, a reinforced concrete beam design, and others. A multi-objective version of the CS algorithm was proposed in [26] by Yang and Deb, assessed again through of structural design optimization problems, such as beam design and disc brake design.

Cheng and Prayogo proposed the symbiotic organisms search (SOS) algorithm [27], a method which simulates the symbiotic interaction strategies adopted by organisms to survive and propagate in an ecosystem. To show the capabilities and the robustness of the algorithm, the authors used 26 mathematical problems as well as five engineering design problems including the design of a cantilever beam, the minimization of the vertical deflection of an I-beam, and two plane truss structures with 15 and 52 members. Yang proposed a Bat-inspired optimization algorithm [28], based on the echolocation behaviour of bats, in an attempt to combine the advantages of existing algorithms. The new algorithm is compared to GA and PSO in handling optimization problems. The method was first assessed through engineering optimization problems in the work of Yang and Gandomi [29] using eight nonlinear engineering optimization problems. Shadravan et al. [30] proposed the sailfish optimizer, a metaheuristic inspired by a group of hunting sailfish, tailored in solving constrained engineering optimization problems. The particularity of the algorithm is that it maintains two populations, one of sailfish for intensification of the search around the best so far and one of sardines for diversification of the search space. After evaluating the algorithm in 20 well known unimodal and multimodal mathematical functions, the authors proceed with testing it in different engineering optimization problems including an I-beam design problem, a Welded beam design problem, a Gear train design problem, a 3-bar truss design problem and a Circular antenna array design problem.

Heidari et al. [31] introduced Harris hawks optimization (HHO), simulating the hunting behaviour of Harris' Hawks. The algorithm is inspired by the cooperative behaviour and chasing style of Harris' hawks in nature, called surprise pounce, where hawks pounce a prey from different directions trying to surprise it. The effectiveness and performance of the method is tested on 29 benchmark problems and several real-world engineering design problems. Askarzadeh [32] introduced Crow search algorithm in 2016, for solving constrained engineering design optimization problems. The algorithm is based on the intelligent behaviour of crows and based on the idea that crows store their excess food in hidden places and retrieve it when it is needed. The method is applied to six engineering design problems with different natures and level of complexity. Eskandar et al. [33] proposed another nature-inspired metaheuristic, the so called water cycle algorithm. The algorithm is based on the observation of water cycle process and how rivers and streams flow to the sea in the real world. The algorithm comes with an embedded constraint handling mechanism and is suited to handling constrained engineering optimization problems.

2.2 MOAs for Solving New Formulations of Sizing SOPs

Sizing optimization is one of the most important and arguably the most widely applied structural optimization discipline because of the simplicity of the problem formulation and its practical significance for the design of real-world structures composed of linear elements, such as columns, beams and truss members. The class of sizing optimization problems was the first application of optimization algorithms in structural engineering.

Farshi and Alinia-ziazi [34] proposed a new methodology for the design of truss structures with optimum weight incorporating the force method based on the method of center points. The design variables of the optimization problem are the cross-sectional areas of the members. The method utilizes the largest hyperspheres inscribed within the feasible space and the analysis step is included in the optimization cycle. Kociecki and Adeli [35] developed GA with two phases to solve the problem of the design of space-frame roof structures with minimum weight (size optimization). They compared their results with the ones obtained with the commercial design software SAP2000, against the factors of: (a) convergence improvement, (b) computation time reduction, and (c) practicality of the optimum design. The advantages of this two-phase GA approach are the following: (a) the design process can be fully automated, even for one-of-a-kind structures, (b) the design time, no longer based on trial and error, can be drastically reduced, and c) a lighter and more economic design can be achieved.

Hasançebi et al. [36] investigated the use of genetic algorithms (GA), simulated annealing (SA), evolution strategies (ES), particle swarm optimizer, tabu search, ant colony optimization (ACO) and harmony search (HS) in the optimum design of real size pin jointed structures, where design limitations were imposed based on the allowable stress design code of American Institute of Steel Institution (ASD-AISC). The authors claim that HS and GA are characterized by slow convergence in large-scale problems, while SA and ES proved to be powerful techniques. Kaveh et al. [37] presented a performance-based optimal seismic design of frame structures using the ACO method. The structural response at various seismic performance levels, is simulated and evaluated using non-linear pushover analysis. The authors claim that ACO is more capable than GA for handling this type of problems and the relevant results are illustrated via two example steel frame structures. Moayyeri et al. [38] applied the PSO algorithm for the optimum design of reinforced concrete retaining walls taking into account both geotechnical and structural constraints for the optimization problem and considering different methods of the bearing capacity computation.

Gholizadeh and Milany [39] proposed an improved fireworks algorithm [40] (IFWA) for discrete sizing optimization of steel skeletal structures. The algorithm features the possibility of interaction among different solutions during the optimization process. IFWA is employed to deal with the discrete structural optimization problems of steel frames and trusses. Bureerat and Pholdee [41] proposed an adaptive differential evolution algorithm for the solution of optimal truss sizing problems. The method is based on DE while a strategically adaptive scheme is also employed, together with an effective constraint handling technique for dealing with constrained structural optimization problems. Hasançebi and Kazemzadeh [42] employed an exponential big bang-big crunch algorithm for the discrete design optimization of steel frames. Two real-world numerical design examples are used, including a 132-member unbraced steel frame and a 209-member industrial factory building. The method proved to be robust and efficient in tackling practical design optimization instances of steel frames. Another work on the optimum design of steel structures is the one by Lagaros et al. [43] where the optimum design of 3D steel structures with perforated I-section beams is examined. The problem is formulated as a combined sizing, shape and topology optimization problem where the cross-sectional dimensions of beams and columns are the sizing variables, while the number and size of web openings in the beams are the topology and shape design variables.

Papadrakakis et al. [44] proposed the use of evolution strategies to perform structural sizing optimization of space frames under seismic loading conditions. In this work the authors used two methods for the dynamic analysis of the structure, namely the traditional design response spectrum approach and the direct integration approach [45] using artificial accelerograms compatible with the elastic design response spectrum. Fragiadakis et al. [46] went one step further in the optimum design of structures under dynamic loading, by proposing a performance-based optimum design methodology for steel structures subjected to seismic loading, considering the inelastic behavior (via pushover analysis) and the life-cycle cost of the structure. The life-cycle cost of the structure was also taken into account in the work of Mitropoulou et al. [47], for the assessment of optimally designed reinforced concrete buildings under seismic actions. In this work, the performance of the structure is evaluated in multiple earthquake hazard levels using incremental static and dynamic analyses, while the life-cycle cost is taken into account as an additional objective function, other than the initial weight of the structure.

2.3 MOAs for Solving New Formulations of Shape and Topology SOPs

Kociecki and Adeli, previously mentioned for their work in sizing optimization [35], extended their work to sizing and topology optimization also, where they used a two-phase genetic algorithm for solving the sizing and topology optimization problem of free-form steel space frame roof structures [48]. They applied the algorithm to two real-life space roof structures. The initial design in both cases is a real design performed by a design office iteratively using a general-purpose structural analysis software in a period of several days. The proposed method resulted in savings of 12% and 4% for the two example cases, respectively. The previously mentioned work of Kociecki and Adeli was further extended for handling sizing, topology, and shape optimization of free-form steel space-frame roof structures with complex geometries using evolutionary computing [49]. Two methods of changing the geometry of the structure are presented, a simple one for mostly regular geometries, as well as a more complex one. The aim was to achieve an optimal design by changing the geometry of the roof structure while simultaneously optimizing the roof member, the column dimensions and the roof topology. Additional constraints, having to do with esthetics have been added to the algorithm as heuristic limits to avoid undesirable changes in the architectural form. Efficiencies in the range of 10–16% have been achieved for the two examined example structures using the proposed methodology.

Amir [50] introduced a new computational approach for optimizing reinforced concrete structures. The major goal was to reduce the amount of material (weight) used in concrete structures, which is extremely desirable due to the negative environmental impact of cement production. Building lighter concrete structures can be considered an important step towards more sustainable architecture. The fundamental concept was to integrate realistic finite element modelling of reinforced concrete with topology optimization algorithms based on a sensitivity analysis. The strain softening response of concrete was treated as a continuum, with a nonlocal damage model used to account for it. Reinforcement was embedded in the continuum concrete domain and represented as a set of all allowable rebar placements. In a topology optimization approach combining truss-based and continuum-based approaches, both materials, concrete, and the steel reinforcement, were designed simultaneously. It was discovered that the optimized designs performed 20% to 30% better than the standard structures in terms of load-bearing capacity per unit weight. Lagaros et al. [51] also investigated the application of optimization methods in the design of 3D reinforced concrete buildings, where the aim was to minimize the eccentricity between the mass center and the rigidity center of each story, handled as a combined topology and sizing optimization problem. The optimized design led to a significant reduction in the structural cost of the building in the test examples considered. Zakian and Kaveh [52] conducted a research study on the topology optimization of shear walls considering material volume and displacement constraints. The aim was to optimize the structural compliance under seismic loads commonly applied to a shear wall. The one-field density approach of simplified isotropic material with penalization (SIMP) was employed and enriched with a penalty function for dealing with the drift constraint of shear walls. The optimality criteria method was incorporated for the solution of the optimization problem. Various heights are defined for shear walls to obtain optimized configurations under different circumstances. The shear wall-frame interaction that influences the single and coupled shear walls was assessed. The results of the investigation revealed the material distribution of shear walls and vital parts of the structure where openings or cut-outs should not be created.

Kaveh and Kalatjari [53] performed size/topology optimization of trusses using a genetic algorithm (GA), the force method and concepts of graph theory. The application of the force method, together with the theory of graphs, allowed the generation of a suitable initial GA population. If unstable trusses were identified during the process, they were penalized properly. The efficiency of the method was illustrated using numerical examples and comparisons to the corresponding results from previous studies. Tian et al. [54] carried out a study which aimed to apply topology optimization approaches in offshore platform structural design and examine how this might help produce better solutions and methods while reducing the design, deployment, and manufacturing costs. The methodology can be used at an early design stage, which helps in determining the initial structure and transmission path. The entire design space is selected as the available space for the design variables, and the objective is to maximize the structural stiffness. Deformation, stress and vibration-related constraints are imposed, all contributing to creating the constraints for a multi-criteria design assessment. The results of the optimization procedure were verified by FE analysis for static and dynamic performance.

De Souza et al. [55] optimized transmission line towers by dividing the structure into modules that can take various pre-determined topologies. Shape and size were optimized at the same time as the topology. Two example cases were considered, a tower with eight different load cases, and a self-supported tower that was subjected to a cable rupture scenario and a wind load. When compared to a classical topology optimization procedure, the obtained results indicated a reduction of up to 6.4% in the final structural weight. Jiang et al. [56] proposed four shape optimization problems in order to obtain reasonable shapes for free-form shell structures with both high static and dynamic performance.

Static performance was measured by strain energy under static loads, whereas dynamic performance was measured by the lower bound on the fundamental natural frequency or strain energy under seismic stresses. The self-weight and live loads are applied, and an optimization problem is initially developed for reducing the strain energy under frequency constraints. After that, the strain energy associated with the comparable seismic static load is minimized. In the cases where the design variables were dense, the surface curvature was added as the second objective function in a multi-objective optimization problem, to avoid ending up to an undesirable shape. The efficacy of the approach was demonstrated in several numerical examples.

Papadrakakis et al. [57] investigated the use of combinatorial optimization methods, in particular evolution strategies, for handling structural shape optimization problems. According to the study, the combination of ES with SQP gives very promising results in shape optimization problems, especially when also taking advantage of a parallel computing environment. The efficiency of the ES was confirmed in the work of Lagaros et al. [58], which investigated the optimum design of shell structures with stiffening beams using an ES optimization scheme, considering sizing, shape and topology design variables.

Belevičius et al. [59] developed a method for optimizing simultaneously the shape, size, and topology of tall, guyed masts. Strength, stability, and slenderness constraints were imposed while designing the mast structure against self-weight and wind loading. The guyed mast's nonlinear behavior was simplified by considering the nonlinear guys as approximate boundary conditions for the mast. Following the selection of the best solution from a set of Evolutionary Algorithm (EA) solutions, the pattern search algorithm was used to thoroughly investigate the solution's surroundings, after the best solution was chosen. A conventional 96 m steel guyed mast holding a standard antenna cluster was optimized using the method. The optimization of the mast with various sets of design parameters revealed that the most relevant mast schemes had three to five guys' clusters, with the optimal mat scheme being the one with five guys' clusters. Mam et al. [60] studied the optimization of the shape of a timber braced framed structure with dowel-type connections subjected to an overall drift restriction as well as strength requirements under wind and gravity loads. The primary goal of the study was to demonstrate the importance of joint flexibility in achieving the best possible solution for a truss-like construction. To establish a simpler relationship between joint stiffness and axial load-carrying capability, dowel-type connections are first investigated. The established local behavior rule is then used to the shape optimization and design of a discrete braced frame subjected to lateral drift constraints under wind load. A two-level optimization approach was developed, using the low-level optimization methods fully stressed design (FSD) and a rigorously determined optimality criteria (OC) for size optimization and a more general optimization approach for shape optimization. This approach provides more control over the optimization process as well as the use of specific optimization methods for each sub-problem. When compared to classical results, the semi-rigid behavior of connections results in a substantial increase in the volume of wood, but it also has an impact on the optimum form and the topology of the X-braced frame.

Pastore et al. [61] presented a novel optimization method for designing lightweight concrete structural components based on an integrated Risk-Factor and Stress-Constrained method, which can account for the asymmetrical compression and traction stresses that characterize concrete materials. In a simply supported beam arrangement, the algorithm was tested across a set of concrete material characteristics. When compared to a traditional Von Mises stress condition, the Risk Factor method showed to be more efficient in producing optimal beams when dealing with a variety of asymmetrical stress restrictions. The method was embedded in an iterative heuristic algorithm, which was then put to the test in a simply supported beam setup with a large number of concrete classes. The findings of the study indicate that the suggested approach can improve the traditional Von Mises paradigm by producing optimized beams that can withstand a variety of asymmetrical stress constraints. In an effort to support and advance sustainable architecture, Frangedaki et al. [62] investigated the design of tree-shaped structural systems using the advanced characteristics of a bamboo material native to South America, and to test its effectiveness by means of a structural parametric design optimization approach. Two structural systems, an elliptical-shaped one and a quadrangular one were parametrized and optimized.

2.4 Hybrid Methods Based on MOAs for Solving SOPs

Various hybrid methods have been proposed in the literature combining the advantages of different methodologies in an effort to achieve higher quality results and increased speed. Some of the hybrid approaches simply try to speed up the convergence of the optimization algorithm, usually combining different types of optimizers that work well either in searching the general search space or specialize mostly in local search. Other approaches combine different numerical methodologies, in an attempt to reduce the computational effort of the optimization scheme, especially when handling large-scale structural optimization problems.

Plevris and Papadrakakis [63, 64] presented a hybrid PSO-gradient algorithm for the global optimization of 2D and 3D truss structures. In this work, the Particle Swarm Optimization [65] method was enhanced with a gradient-based sequential quadratic programming (SQP) optimizer for handling constrained optimization problems of 2D and 3D trusses. The methodology proved to be better in finding optimal solutions for structural optimization problems compared to traditional (non-hybrid) optimization approaches. Similarly, Aydilek [66] proposed a hybrid firefly [67] and particle swarm optimization (HFPSO) algorithm for computationally expensive numerical problems. The algorithm combines the strongest points of firefly and particle swarm algorithms, while it mitigates the disadvantages of both methods. The hybrid algorithm is checked in several benchmark engineering mechanical design problems including the pressure vessel, welded beam, and tension and compression spring.

Using both GA and neural networks (NN), Gholizadeh et al. [68] proposed a method to find the optimal weight of structures subject to multiple natural frequency constraints. GA is used to find the optimal weight, through the virtual sub-population (VSP) method. NN is employed to evaluate the natural frequencies, through a wavelet radial basis function (WRBF). This is the first time WRBF has been employed to identify the natural frequencies of the structure as previously it was only employed to identify other structural characteristics. The test examples included a 10-bar aluminum truss and a 200-bar steel double layer grid. The result of the investigated algorithm (VSP & WRBF) is compared to an exact analysis result and an approximate one obtained by a single RBF neural network, and it is found that for an efficient trial structural optimization, the best results (in terms of weight & time) are obtained by VSP & WRBF. Nguyen and Vu [69] employed composite differential evolution (CoDE) for structural optimization, where the optimization scheme is accompanied with neural networks used as surrogate models for speeding up the process by rapidly evaluating the fitness of candidates. First, CoDE is used in the traditional way, but the fitness values of the possible solutions are saved to the database. After enough data have been generated, NN is trained with these data to provide inexpensive estimations of the fitness function value of other individuals. Three structural benchmark problems are used, the 10-bar truss, 25-bar truss, and 72-bar truss. The methodology achieves a significant reduction of the computational, by around 60%. In the same direction of employing neural networks in an optimization procedure, Papadrakakis et al. [70] investigated the application of NN models to substitute the time-consuming structural analysis phase in large-scale shape and sizing structural optimization problems, achieving significant computational advantages, especially in large-scale optimization problems. A similar approach was employed by the same group in [71], where this time the NN models were applied in a reliability-based structural optimization framework.

Lagaros et al. [72] proposed an adaptive neural network strategy for improving the computational performance of evolutionary structural optimization. In this work, NN is used to predict, the feasibility or infeasibility of structural designs in the framework of an ES optimization procedure. The NN is adaptive, in the sense that its configuration is updated incorporating knowledge about the search domain acquired during the optimization phase. Lagaros and Papadrakakis [73] assessed the performance of differential evolution, harmony search and particle swarm optimization, with reference to their efficiency and robustness for the optimum design of real-world structures with a large number of degrees of freedom. In addition, a neural network-based prediction scheme of the structural response was proposed for assessing the quality of each candidate design during the optimization procedure. The same authors [74] proposed a novel method to improve network training using an adaptive activation function with a properly updated gain parameter, where the efficacy of the methodology was examined in structural optimization problems with NN being used to replace the structural analysis phase.

Liao [75] proposed two hybrid differential evolution algorithms for dealing with engineering design optimization problems. One of them strengthens the exploitation ability by providing DE [76, 77] with a local search operator, i.e. a random walk with direction exploitation. The second hybrid approach enhances DE with its combination with harmony search to achieve a synergetic effect. The two hybrid algorithms are assessed with 14 engineering design optimization problems selected from different fields of engineering. Kaveh [78] proposed a hybrid scheme where swallow swarm optimization (SSO) [78] is implemented in the framework of particle swarm optimization (PSO) to form the hybrid particle swallow swarm optimization (HPSSO) algorithm, in an attempt to achieve a good balance between global and local search. The new scheme is evaluated by solving 11 mathematical optimization problems and 6 truss design engineering problems.

Carbas [79] used an enhanced firefly algorithm for the design optimization of steel frames under the load and resistance factor design–American Institute of Steel Construction (LRFD-AISC) steel design code provisions, where steel profiles for the members are selected from a given table of steel sections. The study proposes an enhancement of firefly algorithm by adding two new expressions for the attractiveness and randomness parameters. Two real-world design examples are successfully designed using the enhanced algorithm. Talatahari et al. [80] introduced a new hybrid scheme, ES-DE, of Eagle Strategy [81] with Differential Evolution, for the optimum design of frame structures. The performance of the hybrid algorithm is evaluated by solving four benchmark problems where the objective is to minimize the weight of steel frames. Khalilpourazari and

Khalilpourazary [82] proposed a hybrid algorithm based on water cycle [33] and moth-flame optimization (MFO) [83] algorithms for solving constrained engineering optimization problems. In particular, the spiral movement of moths in MFO is introduced into the water cycle algorithm in an attempt to enhance its exploitation ability. The efficiency of the hybrid scheme is evaluated with solving three well-known structural engineering problems and comparing the results with the ones of other optimizers in the literature.

2.5 MOAs for Solving Practical, Real-World SOPs

MOAs have been found very efficient for solving real world problems in various disciplines and currently they are used in the everyday professional practice with great success. In the case of SOPs, the economic and environmental benefits through the use of such algorithms are enormous [84, 85]. Although adopting optimization-based design procedures can have a drastic environmental impact and contribute to economic development, the architecture, engineering and construction (AEC) industry appears to be reluctant in adopting such procedures. Two of the reasons that justify the hesitance of AEC industry is the enormous computational effort required for solving real world SOPs and the issues of constructability encountered on the optimized solutions achieved. In this part of the investigation some attempts on dealing with these issues are reported.

To speed up the optimization time in a structural optimization framework based on ES, Papadrakakis et al. [86] employed a preconditioned conjugate gradient (PCG) solution algorithm that was proved as a computationally efficient iterative procedure for solving linear systems of equations resulting from the FEA discretization,. The numerical tests demonstrated the computational advantages of the methodology, especially in the case of large-scale optimization problems and in a parallel computing environment. The efficiency and benefits of parallel computational strategies in structural optimization are also exhibited in [87] with reference to ES and GA. In this work, parallel strategies are implemented first at the optimization algorithm level (i.e., dealing with several members of the population in parallel), and second at the structural model (FEM analysis) level, where the finite element analyses are performed with the help of the FETI domain decomposition method. In the same direction based on parallelism, Lagaros [8] proposed the implementation of parallel computing at the level of metaheuristic optimization, by exploiting the physical parallelization feature of the nondominated sorting evolution strategies method. The method is accompanied by an efficient dynamic load balancing algorithm for optimum exploitation of the available computing resources, achieving almost 100% speedup factors with respect to the sequential procedure.

On the constructability issue, Lagaros [88] proposed a generic real-world optimum design computing platform for civil structural systems, founded on advances achieved on MOAs, structural analysis and parallel computing. Five real-world design projects optimized using the proposed framework are presented. Lagaros and Karlaftis [89] investigated a design procedure for steel wind towers subject to constraints imposed by the Eurocode, formulated as a structural design optimization problem. In this work, five test examples are considered, in particular real-world steel wind towers with varying heights which are optimally designed with minimum cost.

3 Description of the 24 MOAs

As discussed in Sect. 2, MOAs can be categorized into four broad classes: (i) Swarm-based (SB); (ii) Physics-based (PB); (iii) Evolutionary-based (EB); and (iv) Human-based (HB). In this work, 24 MOAs are applied and tested into several structural optimization problems. 14 of them belong to the Swarm-based class, 3 to the Physics-based class, 3 to the Evolutionary-based class and 4 to the Human-based class, as shown in Table 1. In this section a short description of each algorithm is also provided.

An important characteristic of a MOA is the number of main parameters that need to be adjusted for the algorithm to work, a piece of information which is also provided in Table 1. In this table, the variables that are adjusted randomly or automatically into the range [0, 1], and are usually used during the search process, are not included. Based on the number of user defined parameters ALO, SSA, TLBO require only 2 basic parameters to be adjusted (the population size and the maximum function evaluations), while CMAES requires 3 parameters, the previous mentioned two and an extra one, since there is a distinction between the number of parents and offspring. The rest of the algorithms, require 1 up to 7 additional user defined parameters, on top of the two basic ones. MTDE and PBA are the most demanding cases, requiring 7 extra parameters to be defined, apart from population size and maximum function evaluations.

Although there are so many MOAs in the literature, it is worth mentioning that according to the no free lunch (NFL) theorem [90], there is no metaheuristic best suited for solving all optimization problems. In other words, there is no point in trying to find the "best" overall algorithm, as it simply does not exist for all cases and all possible problems. Different algorithms are better suited for different problems, while also their settings and the parameters used play a very important role in the efficiency of the algorithm for handling a specific problem. As far as structural optimization problems are concerned, it can be said that most established metaheuristics can be used for these problems, provided that

Table 1 The 24 investigated optimization algorithms

ID	Acronym	Name and references	Class	Year	Parameters
1	GWO	Grey wolf optimizer [91]	SB	2014	1+2[1]
2	IGWO	Improved GWO [92]	SB	2020	1+2[1]
3	WOA	Whale optimization algorithm [93]	SB	2016	2+2[1]
4	ALO	Ant lion optimizer [94]	SB	2015	0+2[1]
5	CMAES	Covariance matrix adaptation evolution strategies [95]	EB	2001	0+3[2]
6	MTDE	Multi-trial vector-based differential evolution [96]	EB	2020	7+2[1]
7	DA	Dragonfly algorithm [97]	SB	2016	1+2[1]
8	GOA	Grasshopper optimization algorithm [98]	SB	2017	2+2[1]
9	GOAf	Improved GOA [99]	SB	2020	2+2[1]
10	MFO	Moth-flame optimization [83]	SB	2015	2+2[1]
11	MVO	Multi-verse optimizer [100]	PB	2016	3+2[1]
12	SCA	Sine cosine algorithm [101]	PB	2016	1+2[1]
13	SSA	Salp swarm algorithm [102]	SB	2017	0+2[1]
14	PSO	Particle swarm optimization [65]	SB	1995	4+2[1]
15	FA	Firefly algorithm [67]	SB	2008	5+2[1]
16	ICA	Imperialist competitive algorithm [103]	HB	2007	4+2[1]
17	DE	Differential evolution [76, 77]	EB	1995	2+2[1]
18	HS	Harmony search [104]	HB	2001	4+2[1]
19	TLBO	Teaching–learning-based optimization [105]	HB	2011	0+2[1]
20	KH	Krill herd [106]	SB	2012	4+2[1]
21	ISA	Interior search algorithm [107]	HB	2014	1+2[1]
22	PBA	Pity beetle algorithm [108]	SB	2018	7+2[1]
23	SMA	Slime mould algorithm [109]	SB	2020	2+2[1]
24	AOA	Arithmetic optimization algorithm [110]	PB	2021	4+2[1]

[1]Population size and maximum function evaluations

[2]Number of parents and offspring plus maximum function evaluations

they can be equipped with an efficient constraint handling mechanism for dealing with the constraints.

The above mentioned 24 optimization algorithms are independent algorithms which have been published in distinct scientific papers, as shown in Table 1 and the relevant references for each algorithm. For this reason, in this study, they are treated individually, but similarities do exist among them in some cases, in terms of their formulation, the algorithmic description, and other characteristics. For example, the multi-verse optimizer (MVO) could be considered as a variant of differential evolution (DE), while moth-flame optimization (MFO) has important similarities with whale optimization algorithm (WOA). In addition, the flight equation used in dragonfly algorithm (DA) is based on the corresponding one of Cuckoo Search algorithm (CS) [23].

In the next sections, 3.1 to 3.24, a short description of each of the 24 metaheuristics is presented along with their distinct features and additional similarities with other methods and with each other. The following common notations and characteristics have been used:

- $s_i(g)$ is the position vector of the *ith* search agent, for the *gth* iteration,
- $s_{i,j}(g)$ is the *jth* element of the *ith* search agent,
- where $i = 1, 2, \cdots, N_{PopSize}$ and $j = 1, 2, \cdots, n$,
- $s_{gb}(g)$ is the global best solution achieved so far,
- ub_j and lb_j are the upper and lower bounds of the *jth* design variable (dimension),
- N is the maximum number of iterations,
- Random numbers are denoted as $r_k \sim U[0,1], (k = 1, 2, 3\ and\ 4)$,
- The initial population is generated randomly in the design space,
- The global best solution found so far is considered by many algorithms as the target to be chased by the agents.

3.1 Grey wolf Optimizer (GWO)

GWO refers to a swarm-based metaheuristic [91] inspired by the hunting mechanism and leadership hierarchy of grey wolves (Canis lupus). During the iterations of GWO, candidate solutions are classified into alpha, beta, delta and omega classes. The best one constitutes class alpha, the second and third best candidates belong to classes beta and delta, respectively, while the rest ones are part of the omega class. The three main operators of the algorithm are: *chasing the prey*, *encircling* and then *attacking* it. The position vectors $s_i(g+1)$ are defined as follows:

$$s_i(g+1) = \frac{s_{i,1}(g) + s_{i,2}(g) + s_{i,3}(g)}{3} \tag{2}$$

and

$$\begin{aligned} s_{i,1}(g) &= s_\alpha(g) - A_{i,1} \times D_\alpha \\ s_{i,2}(g) &= s_\beta(g) - A_{i,2} \times D_\beta \\ s_{i,3}(g) &= s_\delta(g) - A_{i,3} \times D_\delta \end{aligned} \tag{3}$$

where coefficient vectors $A_{i,1}, A_{i,2}\, and\, A_{i,3}$ are defined as $A = 2a \times r_1 - a$, where parameter a is decreased linearly from 2 to 0 to switch the values of A vectors outside and inside of $[-1, 1]$; $s_\alpha, s_\beta ands_\delta$ are the position vectors of the prey for each set, i.e. the first, second and third best agents obtained so far; $D_\alpha, D_\beta\, and\, D_\delta$ denote the distances of the agent of the corresponding set to the prey:

$$\begin{aligned} D_\alpha &= \left|C_1 \times s_\alpha(g) - s_i(g)\right| \\ D_\beta &= \left|C_2 \times s_\beta(g) - s_i(g)\right| \\ D_\delta &= \left|C_3 \times s_\delta(g) - s_i(g)\right| \end{aligned} \tag{4}$$

where $C_1, C_2\, and\, C_3$ are coefficient vectors $\left(C_i = 2 \times r_2\right)$. Alpha, beta, and delta are the agents who lead the search and omega is a follower. In every iteration, parameters a and C_i are defined, while vectors A and D are updated.

3.2 Improved GWO (IGWO)

An improvement of GWO algorithm [91] was proposed in 2020 [92], in an attempt to enhance the population diversity and to improve the equilibrium amid global and local search. The new algorithm (IGWO) can be considered as a variation of the existing GWO algorithm. In this variation, neighboring information can be shared amongst candidates through the *dimension learning-based hunting* (DLH) scheme, that aims to improve global search domain using multi neighbors learning. According to IGWO, candidates are selected either based on either GWO or DLH schemes depending on the quality of their new positions:

$$\begin{aligned} &s_{i,DLH}(g+1) = s_{i,j}(g) + r_1 \times \left(s_{n,j}(g) - s_{r,j}(g)\right), \\ &s_i(g+1) = \begin{cases} s_{i,GWO}(g+1)\, Eq.(2) & if\, f\left(s_{i,GWO}\right) < f\left(s_{i,DLH}\right) \\ s_{i,DLH}(g+1) & otherwise \end{cases} \end{aligned} \tag{5}$$

where $s_{n,i}(g)$ denotes the *ith* dimension of a random neighbor and $s_{r,d}(g)$ is an agent randomly chosen from the population.

3.3 Whale Optimization Algorithm (WOA)

WOA metaheuristic [93] depends on the hunting scheme of humpback whales, which is of spiral bubble-net type. The search process relies on three procedures: search (exploration), encircling, and bubble-net attacking (exploitation). The later one simulates the humpback swim type of whales around prey composed by two movements: spiral-shaped path towards the sea surface and shrinking circle, chosen with 50% probability each:

$$s_i(g+1) = \begin{cases} s_p(g) - A \times D & if\, r_1 < 0.5] \\ D' \times e^{b \times l} \times \cos(2\pi \times l) + s_i(g) & otherwise \end{cases} \tag{6}$$

where D denotes the distance to the prey of the *ith* search agent (whale) according to Eq. (6), A is a coefficient vector as defined for GWO and $D' = \left|s_p(g) - s_i(g)\right|$, parameter b controls the form of the logarithmic spiral, number l is randomly chosen in $[-1, 1]$. The position vector of the prey $s_p(g)$ contains the global best solution achieved so far, or a randomly chosen agent out of the current iteration, depending on whether the search purpose represents exploitation or exploration, respectively.

3.4 Ant Lion Optimizer (ALO)

ALO metaheuristic [94] simulates the synergy between ants and antlions during a hunting process. During the main procedure, the location of antlions and ants is renewed by means of five operators up to convergence: random walk of ants, traps building by antlions, ants entrapment, prey catching, and re-building traps for another prey. For numerically modelling the ants' random walk the following formula is used:

$$s(g) = \left[0, cs\left(2r\left(g_1\right) - 1\right), cs\left(2r\left(g_2\right) - 1\right), \ldots .\, cs\left(2r\left(g_N\right) - 1\right)\right] \tag{7}$$

where $cs(\cdot)$ stands for cumulative sum function of its vector input argument, N is the maximum number of allowed iterations, $r\left(g_1\right)$ is a stochastic function that returns 1 when $r_1 > 0.5$ and 0 otherwise. In order to mimic the sliding of ants towards the antlion in the trap, the radius of ants' random walk is shrunk. When a fitter prey is caught, the antlion renews its position to the prey. The random walk of each ant is affected by two antlions; one selected by the roulette wheel mechanism and another which is saved in the memory as the "elite" antlion.

3.5 Covariance Matrix Adaptation Evolution Strategies (CMAES)

CMAES method [95] represents a self-adaptation pattern which is entirely de-randomized, where the covariance of mutation is altered first aiming to enhance likelihood of generating the specific step. Subsequently, the modification degree is updated based on the amount of strategy parameters allowed to be altered. Then, according to a random selection scheme the expectation of the covariance matrix is stationary. In addition, the adaptation mechanism is independent of the coordinate system. In the $(g+1)^{th}$ iteration, λ new offspring are generated as follows:

$$s_i(g+1) \sim N\left(m(g), \sigma^2(t) \times C^{(g)}\right) \sim m(g) + \sigma^{(g)} N\left(0, C^{(g)}\right) \tag{8}$$

where $s_i(g+1) \in \mathfrak{R}^n$ $(i = 1, ..., \lambda)$, random numbers $N\left(m(g), C^{(g)}\right)$ are normally distributed where mean value vector $m(g) \in \mathfrak{R}^n$ and $C^{(g)}$ is the covariance matrix, while global step size $\sigma^{(g)} \in \mathfrak{R}_+$.

3.6 Multi-trial Vector-Based Differential Evolution (MTDE)

The performance of differential evolution (DE) algorithm is highly affected by the employed search strategy and the parameter settings. In order to cover a variety of problems, multiple search strategies should be combined [90]. In this direction, Nadimi-Shahraki et al. [96] proposed MTDE, a DE variant, where three different search strategies are combined, producing the so called multi-trial vector (MTV) approach. More specifically the trial vector procedures (TVP) are the representative based (R-TVP), one which maintains diversity, the local random based (L-TVP) one that ensures the balance between exploration and exploitation, and the global best history based (G-TVP) one that enhances the exploitation ability:

$$v_i(g+1) = \begin{cases} s_i(g) + F \times \left(s_{i,b}(g) - s_i(g)\right) + F \times \left(s_{i,w}(g) - s_i(g)\right) + a_1 \times \left(s_r(g) - s_i(g)\right), R-TVP \\ s_i(g) + F \times \left(s_{ri_1}(g) - s_{ri_2}(g)\right) + a_2 \times \left(s_r(g) - s_i(g)\right), L-TVP \\ s_{i,gbh}(g) + a_2 \times \left(s_{ri_1}(g) - s_{ri_2}(g)\right), G-TVP \end{cases} \tag{9}$$
$$i = 1, 2, ..., N_{R-TVP} \text{ or } N_{L-TVP} \text{ or } N_{G-TVP}$$

where indices ri_1 and ri_2 are random integers within the corresponding population range, mutually exclusive and different than index i. Mutation factor F controls the magnitude of variation and coefficients a_1 and a_2 are functions of seven user defined variables $(Win_{Iter}, H, ini, fin, \mu, \mu_f and \sigma)$ [96]. $s_{i,b}(g)$ and $s_{i,w}(g)$ are the best and worst members of $R-TVP$ sub-population while $s_{i,gbh}(g)$ is the ith member of history best archive. Subsequently, crossover operator is used based on a trial vector $u_i(g+1)$ and then the new population is generated by means of the search operator (see description of DE below). In contrast with DE variants that distribute the population into smaller subpopulations of the same size, MTV employs a winner-based policy, that distributes the subpopulation not in an equal manner, but using the approach "the better the search strategy, the larger the subpopulation it will handle". MTV approach introduces adaptive movement steps that rely on a life-time archive that preserves and shares information of the restored promising solutions while also maintaining population diversity.

3.7 Dragonfly Algorithm (DA)

DA is a swarm-intelligence metaheuristic that mimics the surviving swarm behavior of dragonflies [97]. The algorithm is implemented through five swarm movements. In nature, dragonflies follow this scheme either in static swarming (hunting), or in dynamic swarming (migration). The hunting swarming is simulated during the exploration phase of the optimization, in which the dragonflies create sub swarms and fly back and forth over different areas within the search domain. The dynamic migration swarming is simulated during the exploitation phase, where the dragonflies fly in larger swarms and along one direction. To simulate this scheme, the step vector is defined as follows:

$$\Delta s(g+1) = (s \times S_i + a \times A_i + c \times C_i + f \times F_i + e \times E_i) + w \times \Delta s(g) \tag{10}$$

where S_i, A_i, C_i, F_i, E_i are the five parameters of the swarm behavior defined as, *Separation* that maintains avoidance of individuals collision in a neighborhood, *Alignment* controlling the velocity matching between the neighboring agents, *Cohesion* referring to the individuals' tendency to the neighborhood, *Attraction* to food, and *Distraction* from enemies. respectively. The parameters s, a, c, f and e, are weighting factors and w is an inertia weight. The position vector is given by:

$$s(g+1) = \begin{cases} s(g) + \Delta s(g+1), & \textit{if} \text{ neighbors around} \\ s(g) + \text{L\`{e}vy(d)} \times s(g), & \textit{if} \text{ no neighbor around} \end{cases} \tag{11}$$

where

$$L\grave{e}vy(x) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{\frac{1}{\beta}}} \tag{12}$$

where β is a constant and σ is a function of β. It is worth mentioning that Lévy flights were first used by the Cuckoo Search (CS) algorithm [23].

3.8 Grasshopper Optimization Algorithm (GOA)

GOA metaheuristic [98] mimics the natural swarm behavior of grasshoppers. A grasshopper's movement is based mainly on the social interaction with its neighbors. It is described in terms of three zones: *attraction*, *repulsion*, or *comfort zone* state. In order for the swarm to converge to one optimal point eventually (global optimum), two mathematical terms are added to let the best-found solution at each iteration affect the next swarm direction, and also to shrink gradually the 3 zones that control each grasshopper's movement, in order to avoid being stuck at the comfort zone so early with no further movement (i.e. trapped in a local optimum). To model this behavior mathematically, the following equations are used. The position of *ith* grasshopper is given by:

$$\begin{gathered} s_{i,j}(g+1) = c \times \left(\sum_{k=1}^{N_{PopSIze}} c \times \frac{ub_j - lb_j}{2} \times SocInt \right) + s_{gb,j}(g) \\ SocInt = G\left(\left| s_{k,j}(g) - s_{i,j}(g) \right| \right) \frac{s_{k,j}(g) - s_{i,j}(g)}{d_{ik}} \end{gathered} \tag{13}$$

where shrinking coefficient $c \leq 1$, and *SocInt* defines the social interaction of the grasshopper with its neighbors, where $G()$ represents the social force, either attraction (during exploitation) or repulsion (during exploration), expressed as follows:

$$G(r) = f \times \mathrm{e}^{\frac{-r}{l}} - \mathrm{e}^{-r} \tag{14}$$

where the size of the attraction, repulsion and comfort zones can be adjusted by the intensity of attraction f and the attractive length scale l. Convergence of grasshoppers towards the target over the course of iterations, is actually due to decreasing c that is a function of its maximum and minimum values, and the target effect of pulling the swarm. The next position of a population member is affected by its current position, the target position, and the positions of all other members. This behavior is in contrast to PSO, in which the swarm particles' positions (except for the best individual) do not play a role in defining the next movement of a particle.

3.9 Improved GOA (GOAf)

GOAf metaheuristic [99] is a variant of the original GOA [98] with the change of a specific implementation feature, in particular the update expression of coefficient c, and the application of random walk. Given that coefficient c is used to balance between exploitation and exploration, it should remain close to unity during the first iterations and then be reduced at a lower value c_{min} in order to support exploitation, according to

$$c = \begin{cases} \exp\left(-0.5 \times \left[\frac{l}{L/\sigma} \right]^2 \right), & if\ c > c_{\min} \\ c_{\min}, & otherwise \end{cases} \tag{15}$$

where l is the attractive length scale (L is the maximum value of the length), σ controls the variation of c, as it defines the rate in which c will be decreased, with typical values $\sigma = 3 or 4$. Furthermore, in order to avoid premature convergence a biased random walk is also used, similar to the one of cuckoo search algorithm [23].

3.10 Moth-Flame Optimization (MFO)

MFO algorithm [83] relies on the transverse orientation navigation patterns of moths. In particular, the spiral movement of moths towards artificial lights is the part in the transverse orientation method that is simulated in the MFO's movement operator. In the transverse orientation approach, a moth flies by fixing a certain angle with respect to the light source, forming a spiral fly path, which ensures convergence. To maintain investigating the most promising areas of the search space, moths $s_{i,M}(g)$ (search agents) take flames $s_{i,F}(g)$ (best-found solutions) as the source of light and fly spirally around them. To preserve exploration and avoid local optima, each moth is allowed to alter its position using only one specific flame. The new positions of moths $s_{i,M}(g+1)$ are then defined as:

$$s_{i,M}(g+1) = S(s_{i,M}(g), s_{j,F}(g)) \tag{16}$$

where $S()$ is the spiral function calculated over the *ith* month and the *jth* flame as follows:

$$S(s_{i,M}(g), s_{j,F}(g)) = \left| s_{i,M}(g) - s_{j,F}(g) \right| \times e^{b \times g} \times cos(2\pi \times g) + s_{j,F}(g) \tag{17}$$

It is the same spiral-shaped path expression used by the WOA metaheuristic in Eq. (6), that was presented a year later than MFO. Constant b defines the shape of the logarithmic spiral, t is a random number in $[r, 1]$, and r is linearly decreased from -1 to -2 over the course of iterations to promote the exploitation proportional to the number of iterations (the lower g, the closer the distance to the flame). For further promotion of the exploitation proportional to the number of iterations, the number of flames is decreased gradually over the course of iterations, until all moths at the final step update their positions with respect to only one flame.

3.11 Multi-Verse Optimizer (MVO)

MVO metaheuristic [100] was inspired by the relevant multi-verse theory in physics. Three concepts of multi-universe are simulated, namely *white holes*, *black holes*, and *wormholes*, where the relevant models associated with these concepts are used for exploration, exploitation and local search, respectively. The white holes are the main component for the birth of a universe, while black holes will attract everything towards them with their enormous gravitational force. Wormholes, on the other hand, connect different parts of a universe and in the multi-verse theory they can also connect one universe to another. Objects (variables) travel from white holes (solution with high fitness function value) to black holes (solutions with low fitness function value), while searching for better fitness values. The white holes are selected using a roulette wheel mechanism. The exchange of variables through white and black holes maintains the exploration of the search space, while wormholes exist randomly in any universe (regardless of its fitness function value) to assist the MVO in exploiting the search space, through transporting a universe's objects within its space in a random manner. The process starts with the creation of a set of random solutions. In every iteration, variables in the solution agents with high objective values tend to move towards others with lower (better) objective values via the white and the black holes (exploration):

$$s_{i,j}(g+1) = \begin{cases} \begin{cases} s_{gb,j}(g) + TDR \times \left(\left(ub_j - lb_j\right) \times r_4 + lb_j\right), & if\, r_4 < 0.5 \\ s_{gb,j}(g) - TDR \times \left(\left(ub_j - lb_j\right) \times r_4 + lb_j\right), & if\, r_4 \geq 0.5 \end{cases}, & if\, r_2 < WEP \\ s_{i,j}(g), \; otherwise \end{cases} \tag{18}$$

where coefficients *TDR* and *WEP* are gradually modified (increased and decreased, respectively) over the iterations as functions of three variables (maximum and minimum values of *WEP* and parameter *p*). Meanwhile, all the members are moved towards the best solution randomly regardless of their own solution fitness value, which maintains the diversity.

3.12 Sine Cosine Algorithm (SCA)

SCA metaheuristic algorithm [101] relies on the concept of randomly generated agents that are forced to fluctuate either towards or outwards the best-found solution according to a sine–cosine behavior. The positioning of the agents is iteratively guided by a random-walk function:

$$s_i(g+1) = \begin{cases} s_i(g) + r_1 \times sin(r_2) \times \left| r_3 \times s_{gb}(g) - s_i(g) \right|, if\, r_4 < 0.5 \\ s_i(g) + r_1 \times cos(r_2) \times \left| r_3 \times s_{gn}(g) - s_i(g) \right|, otherwise \end{cases} \tag{19}$$

where random numbers r_1 indicates the region of the next position, either inside or outside the space between the solution and its destination, r_2 defines how far the movement will be, r_3 gives a random weight to control the effect of destination in defining the distance, and r_4 has the role of switching between the sine and cosine functions. The cyclic form of cosine and sine functions guarantee exploitation of the search space. While exploration is achieved by modifying the r_1 range of the sin/cosine function, where a solution will be able to move outwards its destination point. In order to promote the exploitation over exploration as the iteration number goes higher r_1 is defined as:

$$r_1 = a - g \times \frac{a}{N} \tag{20}$$

where g and N denote the current and maximum iterations allowed, respectively, and a is a constant.

3.13 Salp Swarm Algorithm (SSA)

SSA is a simple and easy-to-implement metaheuristic [102] that mimics the swarming behavior of the ocean salps travelling in form of a salp chain. The chain consists of a leader and a number of followers. The leader goes towards an artificial source of food, and the followers simply enjoy the ride behind the leader. In the optimization process, the best-found solution is considered as a target to be chased by the salps afterwards, iteratively. The algorithm is equipped with two movement equations, one for the leader and one for the followers. The leader walk is actually a random movement, but towards the source of food (best-found solution so far), which maintains investigating the most promising regions in the search space during the optimization process. On the other hand, the followers walk with respect to each other following the leader in a gradual movement based on Newton's law of motion. The following equation is used to update the position of the leader:

$$s_{1,j}(g+1) = \begin{cases} s_{gb,j}(g) + c_1 \times \left(\left(ub_j - lb_j\right) \times r_2 + lb_j\right) & r_3 \geq 0 \\ s_{gb,j}(g) - c_1 \times \left(\left(ub_j - lb_j\right) \times r_2 + lb_j\right) & r_3 < 0 \end{cases} \tag{21}$$

where $s_{1,j}(g+1)$ shows the *jth* dimension of the leader and coefficient c_1 balances exploration and exploitation. The position of the followers is updated as:

$$s_{i,j}(g+1) = \frac{1}{2} \times \left(s_{i,j}(g) + s_{i-1,j}(g)\right) \tag{22}$$

where $i \geq 2$.

3.14 Particle Swarm Optimization (PSO)

PSO [65] is an optimization algorithm that works with a population of solutions, called particles. Each particle has a position and a velocity in the design space while all the particles together form the so called "swarm". The method mimics the behaviour of birds and particles "fly" in the search space looking for the optimum solution. With iterations, the particles adjust their velocities and positions based on their own "experience", the experience of neighbouring particles and also the one of the "best" particle. The experience of a particle is about the best position they have seen, i.e. the best objective function value they have encountered in their path so far. This way PSO combines local and global search, balancing exploration and exploitation. The velocity and the position of every particle is updated as follows

$$v_i(g+1) = w \times v_i(g) + c_1 \times r_1 \times \left(s_{i,pb}(g) - s_i(g)\right) + c_2 \times r_2 \times \left(s_{gb}(g) - s_i(g)\right) \tag{23}$$

$$s_i(g+1) = s_i(g) + v_i(g+1) \tag{24}$$

where $v_i(g)$ is the velocity of *ith* particle and $s_{i,pb}(g)$ is the personal best (found by *ith* particle); c_1 and c_2 are the cognitive and social parameters (constant parameters of the method), respectively and coefficient w is a weight function.

3.15 Firefly Algorithm (FA)

FA is a metaheuristic inspired by the natural flashing behaviour of fireflies [67, 111]. Each firefly is considered to be attracted to brighter fireflies, while exploring and searching for prey. The brightness of each firefly is associated with the objective function value. The movement of the *ith* firefly which is attracted by a brighter *jth* firefly is determined by the formula:

$$s_i(g+1) = s_i(g) + \beta_0 \times e^{-\gamma \times r_{ij}^2} \times \left(s_j(g) - s_i(g)\right) + a_g \times \varepsilon_i(g) \tag{25}$$

where γ is the light absorption coefficient, a_g is the randomization parameter, $\varepsilon_i(g)$ is a vector of random numbers generated based on a Gaussian or uniform distribution defined as functions of $\delta \in [0,1]$, $\beta_0 \in [0,1]$ is the attractiveness when the Cartesian distance $r_{ij} = 0$, usually $\beta_0 = 1$, and $\gamma = \mathrm{O}(1)$ that characterizes the variation of attractiveness, usually varies from 0.001 to 1000. The randomization parameter a_g should ideally decrease with iterations. A simple scheme to achieve this is:

$$a_g = a_0 \times \theta^g \tag{26}$$

where the initial randomness $a_0 = 1$ and θ is the randomness reduction factor which is similar to the one used for cooling in simulated annealing. In its original version presented by Yang [67, 111], light absorption coefficient, attractiveness β_0 at r = 0 and the randomization parameter a_g, are the control parameters used. In the variant used for the purposes of this study, also presented by Yang [112], a_g is controlled by two parameters (a_0 and θ, see Eq. (26)), while the control parameter δ is used to generate the random vector $\varepsilon_i(g)$.

3.16 Imperialist Competitive Algorithm (ICA)

ICA is an evolutionary socio-politically inspired metaheuristic [103]. The idea is to consider the countries as possible solutions, where the best ones are the *imperialists* (N_{imp}) and the rest are the *colonies* (N_{col}). Each imperialist is supposed to possess a portion of the colonies, thus forming an empire. The evolutionary improvement of the solutions is implemented through assimilation, revolution, title exchange and empires' survival/collapse operators. The normalized power of an imperialist, i.e. the elements of the solution vector, is given by:

$$\begin{gathered} s_i(g+1) = \left| \frac{C_i}{\sum_{k=1}^{N_{imp}} C_k} \right| \\ C_i = f\left(s_i(g)\right) - \max_l \left(f\left(s_l(g)\right)\right) \end{gathered} \tag{27}$$

where C_i denotes the normalized cost of an imperialist. Given that N_{imp} refers to the number of imperialists, $N_{col} = N_{PopSIze} - N_{imp}$ is the number of the colonies. The initial number of colonies of an empire will be:

$$N_{col,i} = round\left(s_i(g+1) \times N_{col}\right) \tag{28}$$

and they will be randomly chosen. During the *Assimilation* process (defined as function of variable θ, β), the power of each colony approaches gradually the one of its respective imperialist. The colonies move in random distances, along directions towards their respective imperialist, maintaining both exploration and exploitation capabilities. The *Revolution* operator maintains better exploration, in which some colonies resist to be ruled by the imperialists, jumping out of the empire, thus exploring new promising areas within the search space. The *Title Exchange* operator is performed to promote a colony to be an imperialist in the next iterations, and vice versa. *Empires' survival/collapse* occurs after performing assimilation, revolution and title exchanging processes, when the empires get either weaker or more powerful (the power is defined through variable ξ). The weak empires collapse leaving behind their colonies that will be taken over by the stronger ones. *Convergence* in ICA occurs when either one empire finally survives (the "grand empire") while all the rest have collapsed, or when another convergence criterion is met.

3.17 Differential Evolution (DE)

Since its inception by Storn and Price in 1995 [76, 77], DE has proven to be a powerful optimization algorithm. The method generates a new vector through the weighted difference of two population members to a third one. According to the "DE/rand/1" scheme, a donor vector $v_i(g+1)$ is defined as:

$$v_i(g+1) = s_{ri_1}(g) + F \times (s_{ri_2}(g) - s_{ri_3}(g)) \tag{29}$$

where the indices ri_1, ri_2 and ri_3 are random integers within the population range, mutually exclusive and different than index i. The mutation factor $F \in [0,2]$ controls the magnitude of differential variation. The crossover operator is used based on a trial vector $u_i(g+1)$ which is defined from the components of vectors $s_i(g)$ and $v_i(g+1)$:

$$u_{i,j}(g+1) = \begin{cases} v_{i,j}(g+1), & \mathit{if}\ \text{rand}_{i,j} \le CR \text{ or } j = I_{rand} \\ s_{i,j}(g), & \mathit{if}\ \text{rand}_{i,j} > CR \text{ or } j \ne I_{rand} \end{cases} \tag{30}$$

where $rand_{i,j} \sim U[0,1]$ and random integer $I_{rand} \in [1,n]$ ensures that $v_i(g+1) \ne s_i(g)$. The last step of the generation procedure is the implementation of the selection operator:

$$s_i(g+1) = \begin{cases} u_i(g+1), \mathit{if}\, f(u_i(g+1)) \le f(s_i(g)) \\ s_i(g), \text{ otherwise} \end{cases} \tag{31}$$

Several successful variations of DE have been reported and investigated in the literature, for general optimization problems as well as structural optimization problems [113–115]. It is worth mentioning that MVO metaheuristic can be considered as a variant of DE, since the derivation of the new design implemented by Eq. (18) represents a combination of Eqs. (29 and 30).

3.18 Harmony Search (HS)

HS is an algorithm inspired by music [104] which aims to mimic the improvisation process of Jazz musicians. Every musician (saxophonist, bassist, guitarist etc.) represents a design variable, while the pitch range of each musical instrument corresponds to a value of a design variable. The Musical harmony has to do with a solution vector at a given iteration, and the objective function is expressed by the audience's aesthetics. Given this algorithmic concept, HS has the following five steps: parameter initialization; harmony memory initialization; new harmony improvisation; harmony memory update; and termination criteria check. A new harmony vector is defined following three rules: usage of harmony memory, pitch adjustment and randomization. The harmony memory has a function similar to the mutation operator in GA. Randomization is employed to increase the diversity of the solutions. In the case that the new generated harmony vector is better (having a better objective function value) than the worst harmony vector already in the harmony memory (HM), then the new harmony vector replaces the worst harmony. In the original variant of HS [104], the harmony memory consideration rate (HMCR) was the basic control parameter, while parameters including pitch adjustment rate (PAR), and fret width (FW) were fixed. In the current version [116, 117], pitch adjustment rate, fret width and fret width damping ratio (FWDR) are also considered as control parameters. Thus, the main control parameters that need to be adjusted by the user are the harmony memory consideration rate, pitch adjustment rate, fret width and fret width damping ratio.

3.19 Teaching–Learning-Based Optimization (TLBO)

TLBO is a population-based metaheuristic inspired by the human teaching and learning behavior [105]. Using two main operators, *Teacher Phase* and *Learner Phase*, the students (solutions) get improved in terms of their grades (fitness function value). The taught subjects are represented by the design variables. The indication of the students' level of knowledge in a specific subject, is the mean value of their grades in the subject. The best candidate (the one having the best fitness) $s_{gb}(g)$ is set as a teacher, and for the rest of the candidates, the mean value of each design variable is calculated: $s_{mean,j}(g) = mean\left[s_{i,j}(g)\right]$. Then, the *Teacher Phase* (TF) starts, by enhancing the students' level of knowledge, through pulling the mean value of each design variable to the corresponding one in the teacher's solution:

$$\begin{gathered} s_{new,i}(g+1) = s_i(g) + r_1 \times (s_{gb}(g) - T_f \times s_{mean}(g)) \\ s_{TF,i}(g+1) = \begin{cases} s_{new,i}(g+1), & \mathit{if}\, f\left(s_{new,i}(g+1)\right) \mathit{better}\, f\left(s_i(g)\right) \\ s_i(g), & \mathit{otherwise} \end{cases} \end{gathered} \tag{32}$$

where integer T_f is randomly set as 1 or 2, with equal probability. The second operator (*Learner Phase)* also provides an improvement for the solutions through the interaction between the candidates themselves:

$$s_i(g+1) = \begin{cases} s_{TFi}(g+1) + r_1 \times \left(s_{ri_1}(g) - s_{ri_2}(g)\right), \mathit{if}\, f\left(s_{ri_1}(g)\right) \mathit{better}\, f\left(s_{ri_2}(g)\right) \\ s_{TFi}(g+1) + r \times \left(s_{ri_2}(g) - s_{ri_1}(g)\right), \mathit{otherwise} \end{cases} \tag{33}$$

where $s_{ri_1}(g)$ and $s_{r_{i2}}(g)$ are two randomly chosen solutions.

3.20 Krill Herd (KH) Algorithm

KH is a metaheuristic inspired by the krill herding in nature [106], in which the movement of each individual of the swarm has three main pillars to determine its time-dependent position: the whole swarm movement, seeking food and random spread. The objective of Krill herd is to minimize the distances of each individual krill from the food location. For modelling the motion of the individuals mathematically, the following formula is employed:

$$\frac{ds_i(g)}{dt} = N_i + F_i + D_i \tag{34}$$

where N_i is the motion induced by the swarm movement (function of its user defined maximum value N_{max}), F_i is the foraging motion (function of user defined variable v_f in $[0, 1]$), and D_i is the physical diffusion of the i^{th} krill [106] (function of its user defined maximum value D_{max}). While the position of the krill is updated as follows:

$$s_i(g+1) = s_i(g) + \Delta t \times \frac{ds_i(g)}{dt} \tag{35}$$

$$\Delta t = C_t \times \sum_{j=1}^{n} (ub_j - lb_j) \tag{36}$$

where constant C_t is a real number in [0,2]. Subsequently the well-known crossover and mutation operators are implemented over the Krills' locations. One of the advantages of the algorithm, according to the authors of the original study [106], is that only one parameter, the time interval (C_t) needs to be fine-tuned.

3.21 Interior Search Algorithm (ISA)

ISA metaheuristic [107] was inspired by the architectural process of the interior design and decoration. In the interior design and decoration process, there are two main concepts used to find the best view and decoration; *composition* and *mirror* concepts. Composition refers to the process of replacing the items' position until getting the best view, while mirror denotes the concept of placing mirrors near the most beautiful items in order to emphasize their beauty. These two concepts are followed in ISA, where the candidates (with the exception of the fittest candidate) are randomly divided into two groups: the composition group in which the candidates change their position only when it gives fitter values, and the mirror group in which some mirrors are placed near the fittest candidates giving them higher weights among the population. The position vector is defined as follows:

$$s_i(g+1) = \begin{cases} lb(g) + (ub(g) - lb(g)) \times r_2, \; if \; r_1 \le a(\text{composition group}) \\ r_3 \times s_i(g) + (1 + r_3) \times s_{gb}(g), \; otherwise(\text{mirror group}) \end{cases} \tag{37}$$

where $ub(g)$ and $lb(g)$ denote the upper and lower bounds of the composition group at the gth iteration, while parameter a needs to be fine-tuned. The location of $s_{gb}(g)$ is slightly changed by means of random walk using a variable λ.

3.22 Pity Beetle Algorithm (PBA)

PBA is a metaheuristic optimization algorithm [108] inspired by the behaviour of the six-toothed bark beetle (pityogenes chalcographus beetle) when searching for food. This beetle feeds on the bark of the trees. PBA simulates the searching for food behavior of this bark beetle, with three main stages; *initialization* of a population consisting of males and females, *regeneration* of new populations, and *location update* stage. In the first stage, an initial population consisting of males and females is randomly located within the search space. Some males act as pioneers as they search for the most suitable host, aggregating into it by producing pheromone that attracts the other males and females. The initial population in PBA should be well diversified in order to avoid premature convergence. To ensure diversification, the initial population is generated by means of a random sampling technique. In the second stage, every particle will look for a better position in the search space to create its own population. This is done through five types of hyper volume selection patterns: neighboring search volume, global-scale search volume, large-scale search volume, mid-scale search volume and memory consideration search volume. In the last type, the best-found positions are saved and used. In the third stage, the position of each mating male and female is updated, removing the previous positions except those that are kept in the memory for the memory consideration search volume. By experiments, it is proved that PBA can handle NP-hard optimization problems.

3.23 Slime Mould Algorithm (SMA)

SMA is a stochastic metaheuristic [109] that simulates the slime mold process that Physarum polycephalum forages in a way that leads to the food through optimal paths, producing positive and negative indications out of the propagation wave that is resulted from the bio-oscillator. The formula for updating the location of the slime mould (wrap food) is defined as follows:

$$s_{i,j}(g+1) = \begin{cases} r_1 \times (ub_j - lb_j) + lb_j, & if\ r_2 < z \\ s_{gb,j}(g) + vb \times (W \times s_{A,j}(g) - s_{B,j}(g)), & if\ r_2 < p \\ vc \times s_{i,j}(g), & otherwise \end{cases} \tag{38}$$

where $z = 0.03$ based on a parametric study, parameter vc is decreased linearly from 1 to 0 and $vb \in [-a, a]$, $s_A(g)$ and $s_B(g)$ represent two individuals, randomly selected from slime mould, W represents the weight of the slime mould defined as a function of the best and worst solutions currently in the iterative process, and parameter p is given by

$$\begin{aligned} p &= tanh\left(\left|f(s_i(g)) - f(s_{gb}(g))\right|\right) \\ a &= arctanh\left(1 - \frac{g}{N}\right) \end{aligned} \tag{39}$$

3.24 Arithmetic Optimization Algorithm (AOA)

AOA is a newly presented population-based metaheuristic [110], inspired by the four main mathematical operators, i.e. addition, division, multiplication and subtraction. In order to switch between exploration and exploitation phases, a random number r_1 is used. If $r_1 > moa(g)$ then exploration phase is activated, otherwise, exploitation phase is implemented, where $moa(g)$ stands for math optimizer accelerated function of the g^{th} iteration:

$$moa(g) = min + g \times \frac{max - min}{N} \tag{40}$$

where min and max are minimum the maximum values of the accelerated function and N is the maximum number of iterations. For the exploration phase:

$$s_{i,j}(g+1) = \begin{cases} best(s_j) \div (mop + \epsilon) \times ((ub_j - lb_j) \times \mu + lb_j) & if\ r_2 < 0.5 \\ best(s_j) \times (mop) \times ((ub_j - lb_j) \times \mu + lb_j) & otherwise \end{cases} \tag{41}$$

while for the exploitation phase:

$$s_{i,j}(g+1) = \begin{cases} best(s_j) - (MOP) \times ((ub_j - lb_j) \times \mu + lb_j) & if\ r_3 < 0.5 \\ best(s_j) + (MOP) \times ((ub_j - lb_j) \times \mu + lb_j) & otherwise \end{cases} \tag{42}$$

where match optimizer probability (mop) coefficient is defined as:

$$mop(g) = 1 - \frac{g^{1/\alpha}}{N^{1/\alpha}} \tag{43}$$

where ϵ is a small integer number, control parameter μ aims to emphasize on exploration not only during the first steps of the search procedure, control parameter a is used to emphasize on exploitation accuracy during the optimization.

4 Additional Features of MOAS' Implementation for Solving Structural Optimization Problems

MOAs represent randomized search procedures where computing is combined with concepts from physical and biological sciences like the imitation of the evolution process, the social behaviour of species etc., and were developed originally for solving unconstrained NP-complete problems, while so far, they have been used for solving any type of problems, ranging from engineering design to economics, routing problems, among others. For implementing MOAs into problems related to structural optimization, there are some special features that need to be integrated into their implementation, such as the handling of constraints, either related to structural performance or box-type constraints for the bounds of the design variables. Before presenting the results obtained through the implementation of the 24 state-of-the-art MOAs, the authors need to underline that although an optimized objective function value is provided for each problem found in the literature, the scope of this study is not to achieve better results compared to the literature, since the conditions of the implementation of the algorithms and the characteristics of the problems' formulations are not always clear. The main scope is to present the efficiency of these algorithms, all assessed on a common framework and a common basis of comparison. In order to define the common basis of comparison, all algorithms were implemented in MATLAB using the guidelines provided by their own developers in the original work where they were first presented. In addition, the same stopping criterion corresponding to a specific number of function evaluations and common technique for dealing with the problem constraints were used, for both performance and box-type constraints. Last but not least, the same procedure has been implemented also for the discrete and the integer design variables.

A feasibility rules-based procedure is used for handling the constraints. In order to calculate the fitness function of an infeasible individual, $p_{violation}$ factor is introduced which is the individual's normalized maximum constraint violation multiplied by a term that takes into account the number of the violated constraint functions of a solution. This factor is defined as:

$$p_{violation} = \|max\{max\{0, g_j(s)\}\}\| \times \left(1 + \frac{n_{constviol}}{n_{const}}\right) > 1 \tag{44}$$

where the term $n_{constviol}$ denotes the number of violated constraint functions and n_{const} stands for the total number of constraints. Then, in order to calculate the individual's fitness function, the $p_{violation}$ factor is multiplied with the maximum objective function value between the best feasible individual found until now and the individual itself. The fitness function is formulated as:

$$F(s) = \begin{cases} f(s) & \text{if } g_a(s) \leq 0 \ \forall a = 1, 2, ..., m_a \\ max\left(f_{bestfeasible}, f(s)\right) \times p_{violation} & \text{otherwise} \end{cases} \tag{45}$$

where $f_{bestfeasible}$ is the global best feasible solution found so far. The constraints of Eq. (1) can be divided into two broad categories; function constraints and bound constraints. The first category includes the inequality and equality constraints and represents a more complex type of constraints, defined as functions. The second category concerns the variable's upper and lower limits (bounds) which restrict the possible values of the problem's design variables. Most researches try to optimize constrained problems using techniques that handle the function constraints while only few have put significant effort to handle properly the limits of the decision variables implementing boundary constraint handling methods (BCHMs). These methods are controlling formulations that try to modify and correct the position of an infeasible variable solution vector of a problem and set it again inside the search space in order to become feasible. Some of the most known boundary constraints handling methods, where y_j is the new corrected variable vector, are the following:

$$Projection: y_j = \begin{cases} s_j & \text{if } lb_j \leq s_j \leq ub_j \\ lb_j & \text{if } s_j < lb_j \\ ub_j & \text{if } s_j > ub_j \end{cases} \tag{46}$$

It has to be noted that in the case of structural optimization problems, the design variables are not always continuous as many of them can only take integer or discrete values. These variables, for every algorithm examined in this study, are treated as equivalent continuous variables, using the correction function of the following simple expression:

$$Correction: y_j = \begin{cases} floor(s_j) & \text{for the integer variables} \\ floor(s_j \times 10)/10 & \text{for discrete variables of } 0.1 \text{ step size} \end{cases} \tag{47}$$

For the case of discrete variables where there is no constant step size, an integer variable is employed instead, used as a pointer denoting the discrete value to be assigned to the design variable.

5 Numerical Tests

In this section, 11 benchmark test examples are investigated, aiming to test the efficiency of the 24 MOAs presented previously. Each problem was solved by each algorithm 20 times (i.e. in 20 independent runs), in order to remove any random bias and to obtain the probabilistic characteristics of the results. In total, $11 \times 24 \times 20 = 5280$ optimization runs were conducted. The parameters that need to adjusted and were used during the implementation of the 24 algorithms can be found in Table 2, they refer to those of Table 1 and correspond to the suggested values provided by the developers of each algorithm.

The constraints handling mechanism used for all the employed optimization algorithms and test cases is such that ensures that in the end of the of the optimization process all constraints will be satisfied and there will be no constraint violations, at all.

5.1 Six Benchmark Structural Optimization Problems

In this section, six benchmark structural optimization problems are studied. First, three well-known benchmark structural truss sizing optimization problems in 2D and 3D are investigated, namely the 10-bar truss [34], the 25-bar truss [34] and the 72-bar truss structures [34]. All three problems refer to steel truss structures, that are formulated as sizing structural optimization problems, with their size in terms of design variables ranging from 8 to 16 design variables. The sizing design variables are continuous values denoting the cross-section area that is to be assigned to the specific structural element, or group of elements. For all three problems the weight of the structure is used as the objective function, to be minimized. Next, three well-known benchmark structural optimization problems having an analytical expression of the corresponding problem formulation are studied, namely the Welded beam design problem [118], the Pressure vessel design problem [118], and the Tension–compression string problem [118].

For all six cases, the number of function evaluations allowed, for all algorithms, was equal to the dimensionality n of the problem times 10,000. This value for the maximum number of function evaluations may not be optimal for each individual problem, but it provides a common base of comparison for problems with different levels of complexity, while also ensuring that the number of function evaluations will be large enough to accommodate even the most difficult cases.

5.1.1 10-Bar Truss

For the 10-bar truss problem, an independent design variable is employed for each bar, resulting into a 10 design variables problem, that are treated as continuous variables in the range [0.1, 33.5] in^2. The constraint functions imposed

Table 2 Parameters of the 24 algorithms

Global Parameters

gpopsize_short=30
gpopsize_long=100
maxFEs_mult=10,000 (for problems 1 to 6)
maxFEs_mult=1,000 (for problems 7 to 11)
maxFEs=dim*maxFEs_mult

Algorithm	Parameters	Algorithm	Parameters

GWO	SearchAgents=gpopsize_long,
	MaxIter=floor(maxFEs/SearchAgents)
	a linearly decreased from 2 to 0
		SSA	SearchAgents_no=gpopsize_short,
			MaxIter=floor(maxFEs/SearchAgents_no)

IGWO	Same with GWO	PSO	noP=gpopsize_short,
			maxIter=floor(maxFEs/noP), wMax=0.9,
			wMin=0.2, c1=2, c2=2

WOA	SearchAgents=gpopsize_short,
	MaxIter=floormaxFEs/SearchAgents),
	parameter a like in GWO, b = 1
		FA	nPop=gpopsize_short,
			maxIter=floor(maxFEs/nPop), gamma=1,
			beta0=1, alpha0=0.2, theta=0.98,
			delta=0.05

ALO	SearchAgents=gpopsize_long,
	MaxIter=floor(maxFEs/SearchAgents)
		ICA	nPop=gpopsize_long,
			maxIter=floor(maxFEs/nPop), Nimp=10,
			beta=0.5, theta=pi/4, xi=0.1

CMAES	PopSize=2*dim+10,
	maxIter=floor(MaxFunEvals/PopSize),
	lambda=floor(PopSize/2),
	sigma=0.5*(UBounds-LBounds)
		DE	nPop=gpopsize_long,
			MaxIter=floor(maxFEs/nPop), F=0.5,
			CR=0.2

MTDE	PopSize=gpopsize_long,
	MaxGen=floor(MaxFES/PopSize),
	WinIter=20, H=5, initial=0.001,
	final=2, Mu=log(D), μf=0.5, σ=0.2
	where D is the problem dimension.
		HS	HMS=gpopsize_short,
			MaxIter=floor((maxFEs-HMS)/nNew),
			HMCR=0.9, PAR=0.1, FW=0.02,
			FWDR=0.995

DA	SearchAgents=gpopsize_short,
	MaxIter=floor(maxFEs/SearchAgents),
	beta=3/2
		TLBO	nPop = gpopsize_long,
			MaxIter=floor(maxFEs/(2*nPop))

GOA	SearchAgents=gpopsize_long,
	MaxIter=floor(maxFEs/SearchAgents),
	cMax=1, cMin=0.00001
		KH	NK = gpopsize_short,
			maxIter=floor(maxFEs/NK), Vf=0.02,
			Dmax=0.005, Nmax=0.01, Ct=1

GOAf	Same with GOA	ISA	n=gpopsize_long,
			maxIter=floor(maxFEs/n), lambda=0.01

MFO	SearchAgents=gpopsize_short,
	MaxIter=floor(maxFEs/SearchAgents),
	b=1, r linearly decreased from -1 to -2
		PBA	nPop = gpopsize_short
			maxIter=floor(maxFEs/nPop),
			fnear=0.08, fact=3, flast=0.015,
			fnear2=0.90, frand2=100, fact3=0.35,
			fmem=0.20

MVO	Universes=gpopsize_short,
	MaxIter=floor(maxFEs/Universes),
	WEPmax=1, WEPmin=0.2, p=6
		SMA	SearchAgents=gpopsize_short,
			maxIter=floor(maxFEs/SearchAgents),
			z=0.03, vc linearly decreased from 1 to 0

SCA	SearchAgents=gpopsize_short,
	MaxIter=floor(maxFEs/SearchAgents),
	a=2
		AOA	Solution=gpopsize_short,
			maxIter=floor(maxFEs/Solution),
			max=1, min=0.2, alpha=5, mu=0.499

refer to (i) stress constraints, where the stress of the truss members should not exceed the stress limit of 25 ksi, and (ii) displacement constraints where the absolute value of the displacement of all nodes should not exceed the limit of 2.0 in; more details on the problem formulation can be found in [34]. The reference objective function value found in the literature that refers to the weight of the structure is equal to 5057.88 lb [34]. The results obtained for the 10-bar truss problem are reported in Table 3, where it can be seen that most algorithms achieved excellent results; the best result for this problem was achieved by CMAES, FA, TLBO and SMA algorithms resulting to the *Best* optimized value lower than 5061 lb, while the least variance on the results obtained out of the 20 independent optimization runs carried out for each algorithm corresponds to IGWO, MTDE, MVO, FA, DE and TLBO algorithms, as denoted by the coefficient of variation with values lower than 0.10%. This problem proved to be easy to handle for most optimizers, with very few exceptions.

5.1.2 25-Bar Truss

For the 25-bar truss problem, the structural elements are grouped, resulting into a problem with 8 design variables, that are treated as continuous variables in the range [0.01, 3.4] in^2. The constraint functions imposed refer to (i) stress constraints, where for tension members the stress should not exceed the stress limit of 35 ksi and for compression members the stress is limited according to AISC code, and (ii) displacement constraints where the absolute value of the displacement of all nodes should not exceed the threshold of 0.35 in; more details on the problem formulation can be found in [34]. The reference objective function value found in the literature, referring to the weight of the structure, is equal to 545.175 lb [34]. The results obtained for the 25-bar truss problem are provided in Table 4, where it can be seen that most algorithms achieved very good results. The best result for this problem was achieved by CMAES, MTDE, FA and TLBO algorithms resulting to the *Best* optimized value lower than 545.20 lb, while the least variance on the results obtained out of the 20 independent optimization runs corresponds to CMAES, MTDE, MVO, FA, DE and TLBO algorithms, with coefficient of variation values lower than 0.10%.

5.1.3 72-Bar Truss

For the 72-bar truss problem, the structural elements are grouped, resulting into a 16 design variables problem, that are treated as continuous variables in the range [0.1, 3.0] in^2. The constraint functions imposed refer to (i) stress constraints, where the stress of the truss members should not exceed the stress limit of 25 ksi in general, and (ii) displacement constraints, where the absolute value of the displacement of the uppermost nodes should not exceed the limit of 0.25 in; more details on the problem formulation can be found in [34]. The reference objective function

Table 3 10-Bar truss example—collective results

Algorithm	Best	Median	Worst	Mean	CoV (%)
GWO	5062.28	5069.68	5086.28	5071.16	0.14
IGWO	5061.55	5062.45	5063.21	5062.46	**0.01**
WOA	5171.54	5606.54	6987.53	5823.45	9.55
ALO	5062.28	5072.33	5097.44	5074.88	0.18
CMAES	**5060.85**	5060.85	5076.67	5063.23	0.11
MTDE	**5060.86**	5060.90	5060.95	5060.90	**0.00**
DA	5090.99	5235.04	5575.38	5257.89	2.67
GOA	5065.29	5088.91	5102.23	5087.51	0.17
GOAf	5077.83	5193.91	5633.45	5210.60	2.84
MFO	5062.74	5081.20	5429.34	5108.44	1.64
MVO	5061.76	5065.67	5072.91	5066.22	**0.06**
SCA	5158.26	5238.47	5307.14	5234.16	0.81
SSA	5061.99	5067.16	5087.15	5069.10	0.13
PSO	5061.27	5087.17	6279.92	5173.62	5.14
FA	**5060.87**	5060.95	5061.48	5061.00	**0.00**
ICA	5077.56	5134.42	5549.57	5175.55	2.30
DE	5062.10	5063.48	5070.28	5064.05	**0.04**
HS	5071.60	5102.72	5621.60	5128.90	2.28
TLBO	**5060.87**	5060.90	5076.71	5061.69	**0.07**
KH	5061.01	5061.96	5077.59	5068.60	0.16
ISA	5707.11	5896.29	7116.73	6007.23	5.41
PBA	5265.65	5410.69	5680.10	5422.12	1.82
SMA	**5060.95**	5061.37	5077.27	5063.78	0.11
AOA	5096.51	5299.32	5610.07	5320.36	2.43

Some algorithms achieved an optimum value close to or better than the reference value reported in the literature; these results are denoted with bold in Tables [3–13] containing the results of the investigation performed. Accordingly, some algorithms depict low coefficient of variation (CoV) values, denoting robustness on their performance; these results are also denoted with bold

Table 4 25-Bar truss example—collective results

Algorithm	Best	Median	Worst	Mean	CoV (%)
GWO	545.58	546.59	549.06	546.81	0.18
IGWO	545.33	545.46	545.64	545.46	**0.01**
WOA	550.61	579.29	617.16	580.24	3.01
ALO	545.31	549.50	575.83	552.17	1.44
CMAES	**545.16**	545.16	545.16	545.16	**0.00**
MTDE	**545.16**	545.16	545.17	545.16	**0.00**
DA	545.49	562.41	662.15	572.15	5.02
GOA	545.44	547.30	611.04	554.85	3.51
GOAf	545.50	548.67	602.03	556.57	2.70
MFO	545.28	546.73	551.74	547.05	0.26
MVO	545.23	545.38	546.00	545.43	**0.04**
SCA	551.99	558.09	564.37	557.92	0.64
SSA	545.20	549.11	556.31	549.20	0.61
PSO	545.18	545.45	547.07	545.82	0.14
FA	**545.16**	545.19	545.28	545.20	**0.01**
ICA	545.45	549.52	554.11	549.42	0.39
DE	545.33	545.40	545.88	545.46	**0.03**
HS	545.92	549.43	567.12	550.39	0.86
TLBO	**545.16**	545.18	545.22	545.18	**0.00**
KH	545.22	545.53	549.14	545.79	0.17
ISA	557.59	557.91	592.64	562.97	1.74
PBA	550.95	563.09	576.02	562.34	1.08
SMA	545.19	545.35	546.25	545.45	0.05
AOA	553.75	563.42	601.81	570.16	2.38

Some algorithms achieved an optimum value close to or better than the reference value reported in the literature; these results are denoted with bold in Tables [3–13] containing the results of the investigation performed. Accordingly, some algorithms depict low coefficient of variation (CoV) values, denoting robustness on their performance; these results are also denoted with bold value found in the literature, referring to the weight of the structure, is equal to 379.66 lb [34]. The results obtained for the 72-bar truss problem are provided in Table 5. It can be seen that most of the algorithms achieved very good results. The best result for this problem was achieved by GWO, IGWO, CMAES, MTDE, PSO, FA, TLBO, KH and SMA algorithms resulting to the *Best* optimized value lower than 379.70 lb, while the algorithms GWO, IGWO, CMAES, MTDE, FA, DE, TLBO and SMA achieved the least variance, with values of the coefficient of variation lower than 0.10%.

5.1.4 Welded Beam Design Problem

The well-known welded beam design problem [118] can be formulated as follows:

Table 5 72-Bar truss example—collective results

Algorithm	Best	Median	Worst	Mean	CoV (%)
GWO	**379.69**	379.78	379.93	379.80	**0.02**
IGWO	**379.69**	379.75	379.90	379.75	**0.01**
WOA	411.27	472.21	536.62	477.52	6.97
ALO	381.05	385.22	421.12	390.83	3.07
CMAES	**379.61**	379.61	379.61	379.61	**0.00**
MTDE	**379.62**	379.62	379.62	379.62	**0.00**
DA	385.36	421.09	508.78	427.72	8.11
GOA	380.86	434.26	540.58	446.20	12.55
GOAf	381.30	403.39	579.48	423.11	11.84
MFO	379.86	380.53	447.17	391.68	4.99
MVO	379.94	381.61	387.65	382.00	0.46
SCA	415.92	428.07	445.85	429.14	2.08
SSA	380.64	383.68	412.35	387.53	2.23
PSO	**379.62**	379.63	467.15	391.63	6.41
FA	**379.62**	379.62	379.63	379.62	**0.00**
ICA	388.63	400.48	503.47	459.63	13.76
DE	379.66	379.69	379.74	379.69	**0.01**
HS	380.25	381.95	386.68	382.33	0.45
TLBO	**379.62**	379.62	379.62	379.62	**0.00**
KH	**379.68**	379.78	430.68	385.96	4.16
ISA	451.79	755.51	947.13	727.08	21.18
PBA	416.96	439.93	453.68	439.38	2.11
SMA	**379.65**	379.70	379.77	379.70	**0.01**
AOA	482.52	506.23	525.30	505.17	2.82

Some algorithms achieved an optimum value close to or better than the reference value reported in the literature; these results are denoted with bold in Tables [3–13] containing the results of the investigation performed. Accordingly, some algorithms depict low coefficient of variation (CoV) values, denoting robustness on their performance; these results are also denoted with bold

$$
\begin{aligned}
&\text{Minimize} f(s) = 1.10471 s_1^2 s_2 + 0.04811 s_3 s_4 \left(14.0 + s_2\right)\\
&\text{Subject to}\\
&g_1(s) = \tau(s) - \tau_{\max} \le 0\\
&g_2(s) = \sigma(s) - \sigma_{\max} \le 0\\
&g_3(s) = s_1 - s_4 \le 0\\
&g_4(s) = 1.10471 s_1^2 s_2 + 0.04811 s_3 s_4 \left(14.0 + s_2\right) - 5.0 \le 0\\
&g_5(s) = 0.125 - s_1 \le 0\\
&g_6(s) = \delta(s) - \delta_{\max} \le 0\\
&g_7(s) = P - P_c(s) \le 0\\
&\text{where}\\
&\tau(s) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{s_2}{2R} + (\tau'')^2}\\
&\tau' = \frac{P}{\sqrt{2}s_1 s_2}\\
&\tau'' = \frac{MR}{J}\\
&M = P\left(L + \frac{s_2}{2}\right)\\
&R = \sqrt{\frac{s_2^2 + \left(s_1 + s_3\right)^2}{4}}\\
&J = 2\left\{\sqrt{2}s_1 s_2\left[\frac{s_2^2}{12} + \frac{\left(s_1 + s_3\right)^2}{4}\right]\right\}\\
&\sigma(s) = \frac{6PL}{s_4 s_3^2}\\
&\delta(s) = \frac{4PL^3}{E s_3^3 s_4}\\
&P_c(s) = \frac{4.013E\sqrt{\frac{s_3^2 s_4^6}{36}}}{6L^2}\left(1 - \frac{s_3}{2L}\sqrt{\frac{E}{4G}}\right)
\end{aligned}
\tag{48}
$$

where $P = 6000lb$, $L = 14in$, $E = 30 \times 10^6 psi$, $G = 12 \times 10^6 psi$, $\tau_{max} = 13600psi$, $\sigma_{max} = 30,000psi$, $\delta_{max} = 0.25in$. More details on the problem and its formulation can be found in [118]. The reference objective function value found in the literature is equal to 1.72485084 [118]. The results obtained for the welded beam problem are provided in Table 6. It can be seen that most of the algorithms achieved very good results managing to reach the vicinity of the optimum. The best result for this problem was achieved by CMAES, MTDE, MFO, PSO, FA and TLBO algorithms resulting to the Best optimized value lower than 1.7249, while the least variance on the results obtained out of the 20 independent optimization runs carried out for each algorithm corresponds to IGWO, CMAES, MTDE, FA and TLBO algorithms as denoted by the coefficient of variation with values lower than 0.10%.

5.1.5 Pressure Vessel Design Problem

The pressure vessel problem [119] is formulated as follows:

Table 6 Welded beam design problem—collective results

Algorithm	Best	Median	Worst	Mean	CoV (%)
GWO	1.725815	1.727201	1.732929	1.727801	0.10
IGWO	1.725205	1.726402	1.727412	1.726407	**0.03**
WOA	1.764439	2.066289	3.805661	2.320700	24.75
ALO	1.738265	2.033519	2.270306	2.077232	6.62
CMAES	**1.724852**	1.724852	1.724852	1.724852	**0.00**
MTDE	**1.724861**	1.724871	1.724927	1.724879	**0.00**
DA	1.736235	1.808789	2.497832	1.949844	13.34
GOA	1.758468	2.240731	3.395531	2.330782	16.02
GOAf	1.842398	2.275301	3.370820	2.353270	17.04
MFO	**1.724852**	1.967988	3.051153	1.994456	19.11
MVO	1.761176	1.964720	2.239683	1.953782	7.37
SCA	1.792420	1.869048	1.943238	1.865548	2.15
SSA	1.936579	2.037333	2.316735	2.089118	5.61
PSO	**1.724852**	1.724852	1.974449	1.737488	3.21
FA	**1.724852**	1.724852	1.724852	1.724852	**0.00**
ICA	1.725042	2.151993	4.197115	2.347707	28.54
DE	1.761492	1.833467	2.142021	1.875381	5.79
HS	2.356651	3.160089	4.310701	3.179208	17.22
TLBO	**1.724852**	1.724852	1.724852	1.724852	**0.00**
KH	1.731836	2.124365	3.097703	2.249855	18.47
ISA	1.890158	3.304555	4.262717	3.264702	20.62
PBA	1.797781	1.904672	2.038480	1.912669	3.46
SMA	1.725136	1.725794	1.839148	1.732070	1.46
AOA	1.950500	2.225528	2.617353	2.283393	8.51

Some algorithms achieved an optimum value close to or better than the reference value reported in the literature; these results are denoted with bold in Tables [3–13] containing the results of the investigation performed. Accordingly, some algorithms depict low coefficient of variation (CoV) values, denoting robustness on their performance; these results are also denoted with bold

Table 7 Pressure vessel design problem—collective results

Algorithm	Best	Median	Worst	Mean	CoV (%)
GWO	**5851.70**	5852.63	5854.93	5853.15	**0.02**
IGWO	**5850.65**	5850.99	5851.88	5851.23	**0.01**
WOA	6254.77	7463.27	8490.07	7481.39	12.61
ALO	**5850.38**	6410.09	7332.84	6415.42	8.84
CMAES	6059.71	6370.78	7119.97	6492.28	7.16
MTDE	**5850.38**	5850.38	5850.39	5850.38	**0.00**
DA	6185.52	6198.52	7544.49	6482.06	9.19
GOA	**5850.54**	5940.26	6516.42	6043.91	4.48
GOAf	**5850.41**	6059.73	7050.68	6255.45	7.47
MFO	6073.18	6379.40	6410.56	6269.42	2.72
MVO	6092.04	6372.21	7333.59	6532.62	8.07
SCA	6352.29	6664.72	7734.11	6786.73	8.12
SSA	6068.85	6820.41	7273.51	6652.20	7.22
PSO	6059.71	6090.53	7544.49	6375.17	10.26
FA	**5850.38**	6090.53	6370.78	6098.55	3.02
ICA	6063.30	6074.21	6130.17	6083.42	0.46
DE	**5883.62**	5929.28	5991.73	5934.27	0.74
HS	6069.95	6432.58	6853.87	6453.31	4.31
TLBO	**5850.38**	5850.39	5850.42	5850.39	**0.00**
KH	6090.72	6410.60	6820.90	6437.27	5.49
ISA	7486.10	10400.00	13200.00	10150.69	23.22
PBA	6084.00	6422.30	6878.42	6438.08	4.45
SMA	**5850.38**	6090.53	7332.84	6421.01	9.60
AOA	7044.90	8574.39	14736.28	9940.79	30.76

Some algorithms achieved an optimum value close to or better than the reference value reported in the literature; these results are denoted with bold in Tables [3–13] containing the results of the investigation performed. Accordingly, some algorithms depict low coefficient of variation (CoV) values, denoting robustness on their performance; these results are also denoted with bold

$$\begin{aligned}
&\text{Minimize} f(s) = 0.6224 s_1 s_3 s_4 + 1.7781 s_2 s_3^2 + 3.1661 s_1^2 s_4 + 19.84 s_1^2 s_3 \\
&\text{Subject to} \\
&g_1(s) = -s_1 + 0.0193 s_3 \le 0 \\
&g_2(s) = -s_2 + 0.00954 s_3 \le 0 \\
&g_3(s) = -\pi s_3^2 s_4 - \tfrac{4}{3}\pi s_3^3 + 1296000 \le 0 \\
&g_4(s) = s_4 - 240 \le 0
\end{aligned} \tag{49}$$

where $s_{1,} s_2$ design variables are integer multipliers of 0.0625. More details on the problem formulation can be found in [119]. The reference objective function value found in the literature is equal to 5888.3400 [92]. The results obtained for the welded beam problem are provided in Table 7, where it is shown that while some algorithms achieved very good results, others failed to do so. The best result for this problem was achieved by ALO, MTDE, FA, TLBO and SMA algorithms. Some algorithms achieved a better (smaller) optimum value than the reference value reported in the literature and these results are denoted with bold in the table. The least variation of the results was exhibited by GWO, IGWO, MTDE and TLBO algorithms, with values of the coefficient of variation lower than 0.10%.

5.1.6 Tension–Compression String Problem

The tension–compression string problem [118] can be formulated as follows:

$$\begin{aligned}
&\text{Minimize} f(s) = \left(s_3 + 2\right) s_2 s_1^2 \\
&\text{Subject to} \\
&g_1(s) = 1 - \frac{s_2^3 s_3}{71785 s_1^4} \le 0 \\
&g_2(s) = \frac{4s_2^2 - s_1 s_2}{12566\left(s_2 s_1^3 - s_1^4\right)} + \frac{1}{5108 s_1^2} - 1 \le 0 \\
&g_3(s) = 1 - \frac{140.45 s_1}{s_2^2 s_3} \le 0 \\
&g_4(s) = \frac{s_2 + s_1}{1.5} - 1 \le 0
\end{aligned} \tag{50}$$

Table 8 Tension–compression string problem—collective results

Algorithm	Best	Median	Worst	Mean	CoV (%)
GWO	0.012694	0.012730	0.012761	0.012729	0.12
IGWO	0.012671	0.012687	0.012703	0.012689	**0.07**
WOA	**0.012665**	0.013369	0.016708	0.013967	9.18
ALO	**0.012667**	0.012719	0.017741	0.013407	10.94
CMAES	**0.012665**	0.012665	0.012665	0.012665	**0.00**
MTDE	**0.012667**	0.012671	0.012678	0.012672	**0.03**
DA	0.012719	0.012869	0.017854	0.013240	8.61
GOA	0.012706	0.014500	0.024514	0.015786	19.62
GOAf	0.012864	0.015688	0.022771	0.016179	16.86
MFO	**0.012665**	0.012719	0.015753	0.012992	5.51
MVO	0.012739	0.014010	0.018147	0.015168	15.62
SCA	0.012840	0.013027	0.013820	0.013077	1.61
SSA	0.012678	0.012884	0.016380	0.013303	8.17
PSO	0.012685	0.012746	0.013298	0.012849	1.54
FA	0.012684	0.012719	0.012821	0.012720	0.20
ICA	**0.012665**	0.012670	0.012987	0.012698	0.56
DE	0.012680	0.012722	0.012825	0.012737	0.33
HS	0.012671	0.015082	0.017776	0.015229	11.47
TLBO	**0.012667**	0.012680	0.012706	0.012682	**0.09**
KH	0.012673	0.012857	0.017374	0.013204	8.46
ISA	**0.012665**	0.012711	0.014808	0.012823	3.70
PBA	0.012689	0.012892	0.013365	0.012942	1.49
SMA	0.012734	0.015367	0.017813	0.015145	13.29
AOA	0.012810	0.013265	0.027160	0.016390	32.74

Some algorithms achieved an optimum value close to or better than the reference value reported in the literature; these results are denoted with bold in Tables [3–13] containing the results of the investigation performed. Accordingly, some algorithms depict low coefficient of variation (CoV) values, denoting robustness on their performance; these results are also denoted with bold

More details on the problem formulation can be found in [118]. The reference objective function value found in the literature is equal to 0.012665 [118]. The results obtained for the tension–compression string problem are provided in Table 8 where it can be seen that most of the algorithms achieved excellent results. The best result for this problem was achieved by WOA, ALO, CMAES, MTDE, MFO, ICA, TLBO and ISA algorithms resulting to the *Best* optimized value lower than 0.012668. The least variance of the results was exhibited by IGWO, CMAES, MTDE and TLBO algorithms with a value of the coefficient of variation lower than 0.10%.

5.1.7 Comparative Results

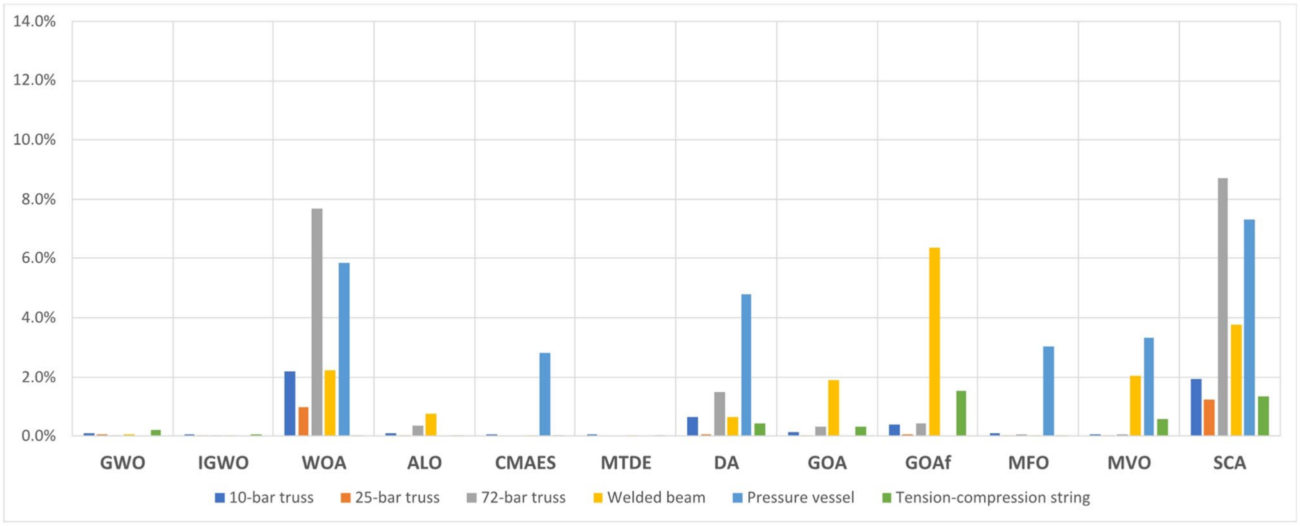
In order to present the globality of the algorithms' efficiency, Fig. 2 shows the variation (or relative error value) of the best achieved optimum solution by each one of the 24 MOAs, in comparison to the reference (best reported) solution found in the literature, for each problem. In this diagram, lower bars represent better solutions and ideally a zero-height bar (i.e. zero error) would mean that the algorithm has achieved the same optimum as the one found in the literature. 10 out of 24 MOAs (GWO, IGWO, ALO, CMAES, MTDE, MFO, PSO, FA, TLBO, SMA) managed to give excellent solutions with error values less than 1% for all the problems examined, while 12 of them (GWO, IGWO, ALO, CMAES, MTDE, DA, MFO, PSO, FA, TLBO, KH, SMA) managed to end up to very good solutions with error values less than 2% in all examined problems. The best three overall performances were the ones of CMAES, MTDE and TLBO, with average error values (average over all 6 problems) less than 0.16%, followed by MFO, IGWO, PSO and FA with average error values less than 0.2%. These excellent results show the clear potential of MOAs in handling structural optimization problems.

5.2 International Student Competition in Structural Optimization (ISCSO 2015 to 2019)

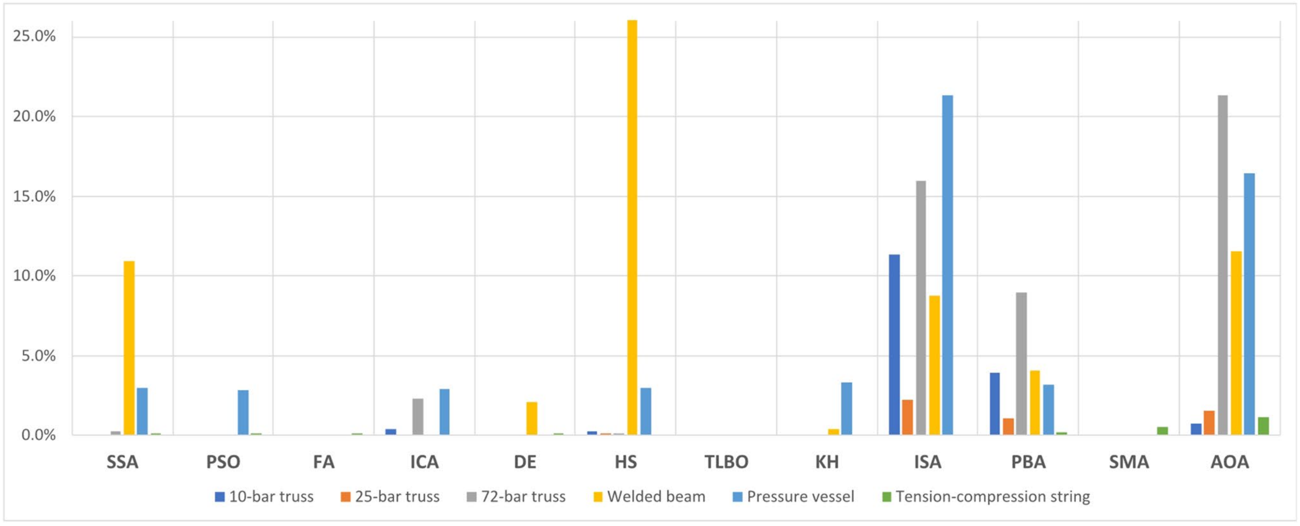
In this section, five test examples taken from the recent International Student Competition in Structural Optimization events (i.e. ISCSO2015 to ISCSO2019, [9–13]), are used for further challenging the efficiency of the 24 MOAs. These five problems refer to steel truss structures, they are formulated as combined sizing-shape structural optimization problems and their size, in terms of design variables, range from 54 to 328 design variables. The sizing design variables are integer values denoting the discrete standardized cross-section that is to be assigned to the specific 2D or 3D truss structural element, while the shape design variables are continuous denoting the value of the specific node coordinate. For each problem, we report a table which presents the results obtained by the 20 independent optimization runs, performed for each problem with the same algorithm. In particular, each table reports the best objective function value found in 20 runs, the median, worst and mean value, as well as the coefficient of variation which is a standardized measure of dispersion of the results, defined as the ratio of the standard deviation to the mean. The number of function evaluations allowed for the six problems, for all algorithms, was equal to the dimension *n* of each problem times 1000.

5.2.1 ISCSO 2015 Problem

The test example of ISCSO2015 [9] is formulated as a sizing and shape optimization of the 45-bar 2D truss structure shown in Fig. 3, that is discretised with 45 sizing and 9 shape design variables. The sizing variables denote the



(a)



(b)

Fig. 2 Performance of the 24 algorithms in the group of the 6 benchmark test problems: **a** Algorithms 1–12, **b** Algorithms 13–24

cross-sectional areas of the truss elements (in groups) and take values in the range 0.1 to 15 in^2 with increments of 0.1 in^2. The constraint functions imposed refer to (i) stress constraints, where the stress of the truss members should not exceed the stress limit of 30 ksi, and (ii) displacement constraints where the absolute value of the displacement of all nodes should not exceed the limit of 2.0 in. More information about the problem formulation (including loading conditions, design variables grouping etc.) and how it can be implemented through a simple MATLAB function for the structural analysis and design of this particular truss is provided in [9].

The best value achieved in the framework of the competition was equal to 3861.1045 lb and it is taken as the reference value for comparison in the present study. The results obtained from the 24 MOAs, for the ISCSO2015 two-dimensional truss optimization problem, are presented in Table 9. In this test example, DE outperformed the other algorithms resulting to the *Best* optimized value of 5046.70 lb, followed by PBA, SSA and SCA. The worst performance in terms of final best objective value is the one of GWO (7847.57 lb) followed by IGWO, ICA and GOAf. IGWO exhibited the least variation on the results obtained out of 20 independent optimization runs (4.07%), but its performance was overall poor when we consider the

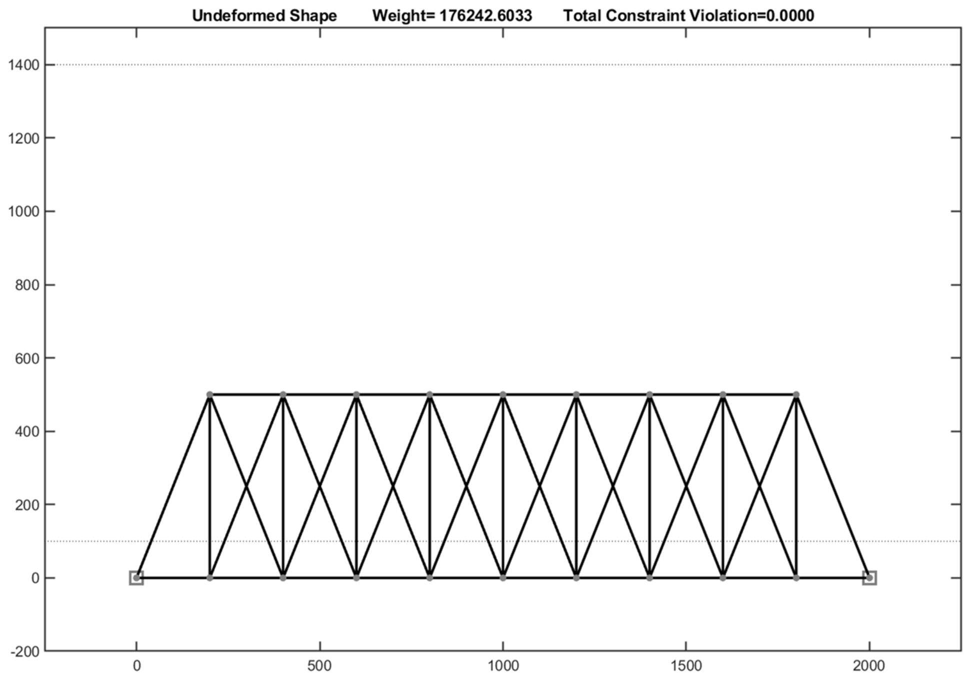

Undeformed Shape Weight= 176242.6033 Total Constraint Violation=0.0000

Fig. 3 The ISCSO2015 two-dimensional truss problem (dimensions in in)

Table 9 ISCSO2015 test example—collective results (objective function values in lb)

Algorithm	Best	Median	Worst	Mean	CoV (%)
GWO	7847.57	8667.70	10011.98	8695.66	6.77
IGWO	7322.22	8098.97	8473.70	8051.59	**4.07**
WOA	5153.90	8488.70	12175.60	8777.50	26.66
ALO	5208.14	7458.92	11850.75	7940.61	24.95
CMAES	5633.38	6038.90	6680.38	6077.00	**4.77**
MTDE	5469.02	8735.15	10092.62	8587.29	14.10
DA	5209.22	7823.33	12532.20	8530.07	28.06
GOA	5170.39	7970.05	12574.50	8254.99	27.90
GOAf	5844.59	9264.72	12128.22	9078.35	24.41
MFO	5423.47	8495.52	11808.38	8455.49	21.34
MVO	5205.39	8247.57	12432.26	8462.21	22.52
SCA	5128.14	9778.64	12139.61	8903.87	28.97
SSA	5126.45	9893.93	11772.47	9441.13	20.60
PSO	5373.44	8828.13	11243.41	8391.92	19.99
FA	5226.31	5797.08	6868.02	5862.77	8.19
ICA	6014.06	7301.66	8567.34	7387.79	11.08
DE	**5046.70**	8960.40	12341.89	9023.18	26.10
HS	5247.47	9043.99	10581.26	8655.61	16.99
TLBO	5692.23	6499.32	9156.10	6846.41	14.23
KH	5136.65	8579.90	12542.30	8753.96	30.60
ISA	5471.80	8978.11	12088.31	8932.18	24.23
PBA	5113.99	8548.61	12559.71	8780.52	27.83
SMA	5666.07	6559.16	7910.83	6730.15	9.36
AOA	5193.87	9100.84	12234.46	9072.24	26.50

Some algorithms achieved an optimum value close to or better than the reference value reported in the literature; these results are denoted with bold in Tables [3–13] containing the results of the investigation performed. Accordingly, some algorithms depict low coefficient of variation (CoV) values, denoting robustness on their performance; these results are also denoted with bold

value of the objective function achieved. From the top-5 performers in terms of best objective value achieved (DE, PBA, SSA, SCA and KH), SSA showed a good balance between best value and variation, with a best value of 5126.45 lb and a coefficient of variation equal to 20.60%. Interestingly, when the median or the mean values are taken into account, things are different with the top performers being FA, CMAES, TLBO, SMA and ICA. So, the top-5 performers in terms of best value achieved are completely different than the top-5 performers when the median value or the average value is taken into account.

5.2.2 ISCSO 2016 Problem

The test example of ISCSO2016 [10] refers to the steel cantilever 3D truss structure shown schematically in Fig. 4. The structure consists of 117 members and 30 nodes in total. The problem is formulated as a combined sizing and shape optimization problem, with 117 sizing and 7 shape design variables. The sizing design variables can only take integer values ranging from 1 to 37 representing the section ID from a database of 37 pipe sections. The shape variables have to do with the vertical coordinates of the 14 top nodes of the structure, grouped in pairs. The structure is designed according to AISC-LRFD 1994 regulations, and each member is assessed considering the limit states of tensile yielding and compressive buckling. Thus, the constraint functions imposed refer to (i) stress constraints where the truss members should satisfy the stress requirements of the code, and (ii) displacement constraints where the absolute value of the displacement of all nodes should not exceed the limit of 4.0 cm. More information about the problem formulation

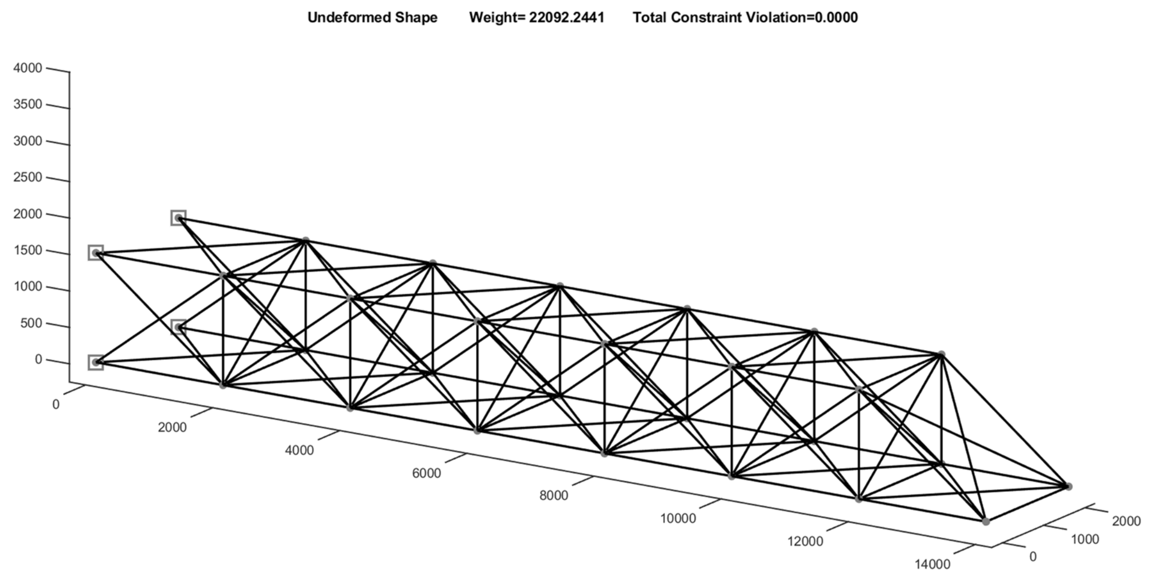


Fig. 4 The ISCSO2016 three-dimensional truss problem (dimensions in mm)

Table 10 ISCSO2016 test example – collective results (objective function values in kg)

Algorithm	Best	Median	Worst	Mean	CoV (%)
GWO	4965.04	5898.39	6722.70	5774.91	8.42
IGWO	4382.93	4515.26	4688.40	4508.56	**1.64**
WOA	4417.05	6969.41	8608.95	6862.18	19.89
ALO	4709.22	7095.62	8400.86	6842.96	13.12
CMAES	**3862.19**	3913.07	3941.95	3910.77	**0.55**
MTDE	4776.83	5063.14	5489.69	5071.87	3.90
DA	3891.69	6405.33	9296.56	6550.99	24.52
GOA	4294.78	7411.09	8637.58	7245.68	12.24
GOAf	3971.69	6084.66	9502.94	6235.25	25.00
MFO	4050.96	7309.54	9150.50	6812.72	23.94
MVO	5530.12	6299.50	7532.81	6363.06	8.35
SCA	3972.65	7334.26	9557.15	6877.93	27.51
SSA	4245.66	7044.14	9472.31	6878.10	21.04
PSO	5976.08	6904.59	8628.51	6931.45	11.01
FA	4215.88	4510.99	4939.68	4514.09	4.56
ICA	4339.00	4642.72	5083.93	4645.91	4.56
DE	5770.26	7499.07	9532.21	7439.93	18.37
HS	5303.45	5506.09	5716.85	5539.99	**2.70**
TLBO	5040.97	5387.82	6267.89	5465.42	7.13
KH	4441.11	5921.93	7531.86	5846.85	19.77
ISA	4361.39	6878.25	9234.16	6696.96	22.67
PBA	4181.79	7895.46	9269.37	7278.02	26.80
SMA	4306.85	4412.69	4934.29	4482.34	4.21
AOA	4979.00	6433.52	9327.49	6943.43	24.16

Some algorithms achieved an optimum value close to or better than the reference value reported in the literature; these results are denoted with bold in Tables [3–13] containing the results of the investigation performed. Accordingly, some algorithms depict low coefficient of variation (CoV) values, denoting robustness on their performance; these results are also denoted with bold

(including loading conditions, design variables etc.) and how it can be implemented through a simple MATLAB function for the structural analysis and design of this particular truss is provided in [10].

The best value achieved in the framework of the competition is equal to 2816.0281 kg and it is taken as the reference value for comparisons. The results obtained from the 24 MOAs, for the ISCSO2016 three-dimensional truss optimization problem, are presented in Table 10. In this test example, CMAES outperformed the other algorithms resulting to the *Best* optimized value of 3862.19 kg, followed by DA, GOAf, SCA and MFO. A similar trend is seen when the median or average values are taken into account. In the median case criterion, the top-5 performers are CMAES, SMA, FA, IGWO and ICA. In this test example, CMAES has consistently shown the best performance, in terms of both the best value achieved and also the median value and the mean value over the 20 independent runs. The worst performances in terms of the median value are the ones of PBA, DE, GOA, SCA and MFO, while the worst performers in terms of the best value are PSO, DE, MVO, HS and TLBO. Interestingly, CMAES also exhibited the least variation on the results obtained out of 20 independent optimization runs (0.55%), followed by IGWO, HS, MTDE and SMA in this criterion.

5.2.3 ISCSO 2017 Problem

The test example of ISCSO2017 [11] refers to the 3D steel truss structure shown in Fig. 5. It consists of 198 members and 52 nodes. The problem is formulated as a sizing and

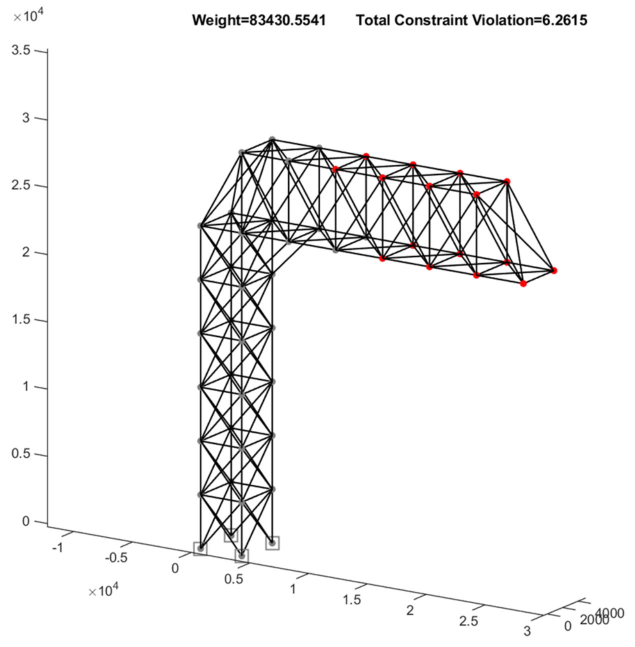


Fig. 5 The ISCSO2017 three-dimensional truss problem (horizontal dimensions in mm)

shape optimization problem with 198 sizing variables (one for each member) and 13 shape design variables, resulting in 211 design variables in total. The sizing variables can only take integer values ranging from 1 to 37 representing the section ID from a database of 37 pipe sections. The structure is designed according to AISC-LRFD 1994 regulations, considering the limit states of tensile yielding and compressive buckling for each member. The constraint functions imposed refer to (i) stress constraints, where the truss members should satisfy the stress requirements of AISC-LRFD 1994, and (ii) displacement constraints where the absolute value of the displacement of all nodes should not exceed the limit of 100.0 mm. More information about the problem formulation (loading conditions, design variables etc.) and how it can be implemented through a simple MATLAB function for the structural analysis and design of this particular truss is provided in [11].

The best value achieved in the framework of the competition is equal to 44090.5356 kg and it is considered the reference value for comparison in the present study. The results obtained from the 24 MOAs, for the ISCSO2017

Table 11 ISCSO2017 test example—collective results (objective function values in kg)

Algorithm	Best	Median	Worst	Mean	CoV (%)
GWO	67002.68	117756.09	153277.42	113644.64	31.33
IGWO	**62172.46**	120165.52	148519.88	114437.05	27.58
WOA	62472.37	88053.99	145342.98	98860.32	28.94
ALO	64920.22	86122.89	141814.45	98483.01	31.43
CMAES	79235.84	112207.46	154717.85	116872.75	23.58
MTDE	87374.94	105761.09	110090.33	102065.62	9.50
DA	76446.15	82123.40	122174.65	95670.80	23.69
GOA	88676.07	142898.32	152849.15	129227.54	21.77
GOAf	68000.43	118675.90	145884.71	116821.88	25.47
MFO	105213.86	111712.35	136461.41	117210.47	10.77
MVO	73336.95	109988.18	131288.65	103132.39	22.24
SCA	85895.53	116980.14	149377.89	122556.67	21.81
SSA	75099.34	125820.59	147015.58	118558.58	22.81
PSO	69185.98	101626.58	152751.45	102722.99	33.70
FA	99473.62	114855.25	126806.13	112697.90	9.05
ICA	84976.11	129741.57	152394.73	127943.81	21.58
DE	95835.68	98722.06	99984.35	98104.99	**1.92**
HS	91799.40	123280.33	138668.79	116235.70	15.96
TLBO	71876.54	90500.69	143171.23	98291.52	29.46
KH	109902.55	125320.46	150197.83	129067.13	12.23
ISA	107746.38	120473.00	154049.94	127515.43	15.31
PBA	63947.01	84796.79	113317.97	87974.23	24.34
SMA	72368.74	92391.89	129049.47	94824.73	23.61
AOA	66280.21	77992.87	111039.46	84010.68	20.00

Some algorithms achieved an optimum value close to or better than the reference value reported in the literature; these results are denoted with bold in Tables [3–13] containing the results of the investigation performed. Accordingly, some algorithms depict low coefficient of variation (CoV) values, denoting robustness on their performance; these results are also denoted with bold

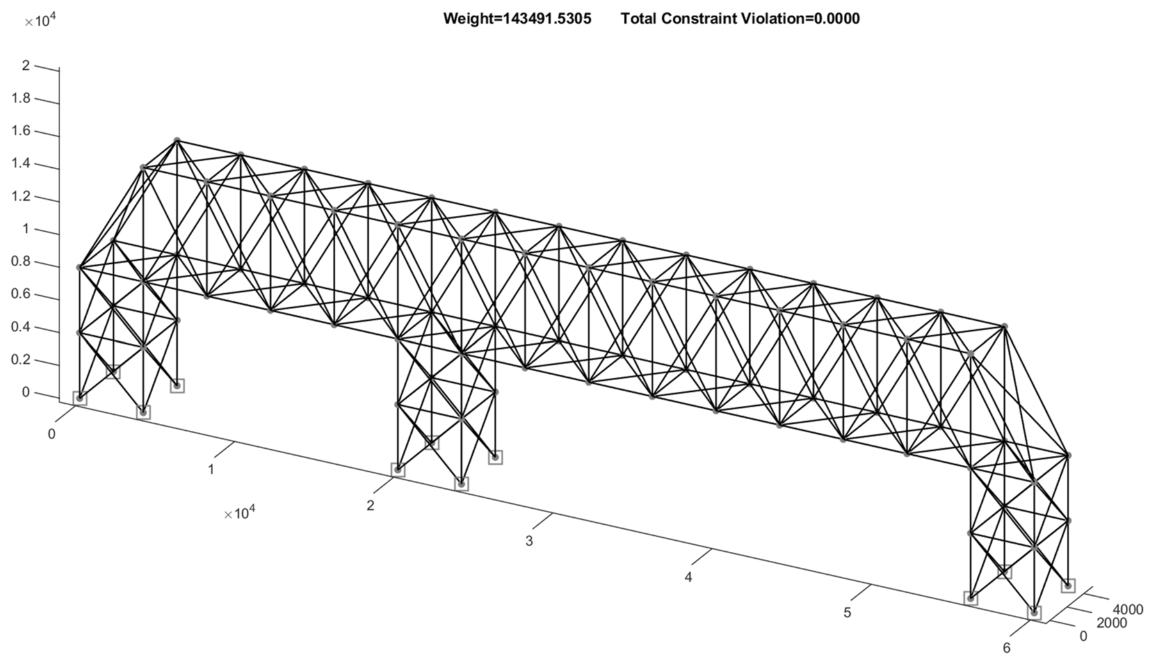


Fig. 6 The ISCSO2018 three-dimensional truss problem (horizontal dimensions in mm)

three-dimensional truss optimization problem, are presented in Table 11.

In this test example, IGWO outperformed the other algorithms resulting to the *Best* optimized value of 62172.46 kg, followed by WOA, PBA, ALO and AOA. When the median value is taken into account, the top-5 performers are AOA, PBA, SMA, DA and DE while when the average value is taken into account, the top-5 performers are AOA, PBA, SMA, DA and DE. The worst performances in terms of the median values are the ones of GOA, ICA, SSA, KH and HS, while the worst performers in terms of the best value achieved are KH, ISA, MFO, FA and DE. DE achieved the least variation with a CoV value of 1.92%, followed by FA, MTDE, MFO and KH in this criterion.

5.2.4 ISCSO 2018 Problem

The test example of ISCSO2018 [12] refers to the 3D steel truss structure shown in Fig. 6. The structure is composed of 314 members and 84 nodes. The problem is formulated as a combined sizing and shape optimization problem having 314 sizing variables (representing the cross-sectional areas of the truss members) and 14 shape design variables (representing the z-coordinates of the 28 top nodes, grouped in pairs). The sizing variables can only take integer values ranging from 1 to 37 representing the section ID from a database of 37 pipe sections. The structure is designed according to the regulations of AISC-LRFD 1994, considering the limit states of tensile yielding and compressive buckling for each member. The constraint functions refer to (i) stress constraints, according to the stress requirements of AISC-LRFD 1994, and (ii) displacement constraints, where the absolute value of the displacement of any node should not exceed the limit of 50.0 mm. More information about the problem formulation (including loading conditions, design variables etc.) is provided in [12].

The best value achieved in the framework of the competition is equal to 14425.0973 kg, taken as the reference value for comparison in the present study. The relevant results obtained from the 24 MOAs, for the ISCSO2018 three-dimensional truss optimization problem, are presented in Table 12. In this test example, SCA outperformed the other algorithms resulting to the *Best* optimized value of 21,341.16 kg, followed by PBA, FA, MVO and GOA. When the median value is used as a criterion, the top-5 performers are MTDE (27521.62 kg), PBA, GOA, DE and WOA. Exactly the same are the top-5 performers if the average value is used. In terms of the CoV value and the least variation of the results, the top-5 performers are CMAES (16.30%), SSA, DE, TLBO and HS. The worst performers in terms of best objective value achieved are TLBO (30754.40 kg), SSA, MFO, ISA and CMAES. If we use the median value as the ranking criterion, the worst performers become MFO (49296.43 kg), GWO, FA, ISA and ALO.

5.2.5 ISCSO 2019 Problem

The test example of ISCSO2019 [13] is formulated as a sizing and shape optimization of the 260-member 3D truss

Table 12 ISCSO2018 test example—collective results (objective function values in kg)

Algorithm	Best	Median	Worst	Mean	CoV (%)
GWO	24645.09	47708.26	50620.89	39478.24	32.85
IGWO	24230.62	40219.53	46544.21	36938.99	28.40
WOA	26642.16	31526.44	46282.57	34898.86	23.37
ALO	25012.71	43013.61	50819.41	40600.24	23.40
CMAES	28244.96	38030.40	44975.47	37699.79	**16.30**
MTDE	22056.32	27521.62	48896.31	30044.00	36.66
DA	23280.23	41183.75	52325.09	38791.98	27.45
GOA	21708.44	31615.69	48482.15	33132.88	30.54
GOAf	23339.12	40642.34	50265.65	38557.28	29.24
MFO	28990.20	49296.43	51651.67	42989.19	23.48
MVO	21563.40	35968.90	44347.80	35213.84	24.26
SCA	**21341.16**	40146.07	51055.73	36135.45	34.78
SSA	29832.12	38218.37	45152.10	37106.54	18.65
PSO	23106.30	36025.75	50528.95	37060.66	27.86
FA	21560.03	47297.80	50730.87	42635.56	28.52
ICA	24707.18	38728.63	48458.06	38647.73	23.00
DE	26323.23	31355.72	42524.15	33843.16	18.76
HS	26401.04	40513.03	45500.21	38936.50	20.23
TLBO	30754.40	38302.58	47858.18	38270.10	19.07
KH	21811.50	38319.41	47824.84	35474.90	28.73
ISA	28387.39	43368.48	50945.76	41972.37	22.80
PBA	21492.56	30504.49	42378.49	32449.64	24.55
SMA	26122.25	40447.14	49090.18	37155.12	26.01
AOA	21854.95	42083.95	51312.41	39463.85	27.88

Some algorithms achieved an optimum value close to or better than the reference value reported in the literature; these results are denoted with bold in Tables [3–13] containing the results of the investigation performed. Accordingly, some algorithms depict low coefficient of variation (CoV) values, denoting robustness on their performance; these results are also denoted with bold

Table 13 ISCSO2019 test example—collective results (objective function values in kg)

Algorithm	Best	Median	Worst	Mean	CoV (%)
GWO	18991.54	31293.86	37716.59	29968.59	23.05
IGWO	20211.42	34348.39	38904.45	32829.80	16.17
WOA	21919.76	31809.16	41527.48	32074.53	21.05
ALO	19003.75	26326.31	40925.86	27628.12	25.68
CMAES	18323.61	28785.49	43560.18	29852.75	26.81
MTDE	19421.50	31409.96	43620.59	32289.22	24.28
DA	21065.37	35332.21	43677.06	33192.22	25.72
GOA	18916.25	24385.98	41997.75	27869.24	31.01
GOAf	18696.02	29519.40	43455.02	29504.43	26.10
MFO	19721.84	31964.70	40919.61	32277.63	20.36
MVO	18597.36	30259.26	40929.23	30138.30	24.34
SCA	19942.10	34103.17	43611.44	33461.70	21.57
SSA	18331.28	24859.55	40043.42	27088.33	27.64
PSO	19287.75	27286.90	43925.87	28455.32	25.19
FA	21272.41	33280.35	41658.88	32481.87	18.97
ICA	23973.61	34020.50	43820.49	34345.02	20.15
DE	18052.67	24598.27	42599.65	28165.39	30.50
HS	21954.27	27126.90	37994.61	28601.30	20.15
TLBO	17735.41	23380.96	33541.89	25144.96	24.68
KH	18147.48	33993.82	44084.55	32554.39	28.90
ISA	19863.61	30503.88	38508.74	28701.59	24.00
PBA	18051.75	29984.21	40554.94	29450.65	20.99
SMA	29440.10	34039.64	38908.84	34063.71	**6.86**
AOA	**17697.21**	31951.31	39946.53	30512.30	22.71

Some algorithms achieved an optimum value close to or better than the reference value reported in the literature; these results are denoted with bold in Tables [3–13] containing the results of the investigation performed. Accordingly, some algorithms depict low coefficient of variation (CoV) values, denoting robustness on their performance; these results are also denoted with bold

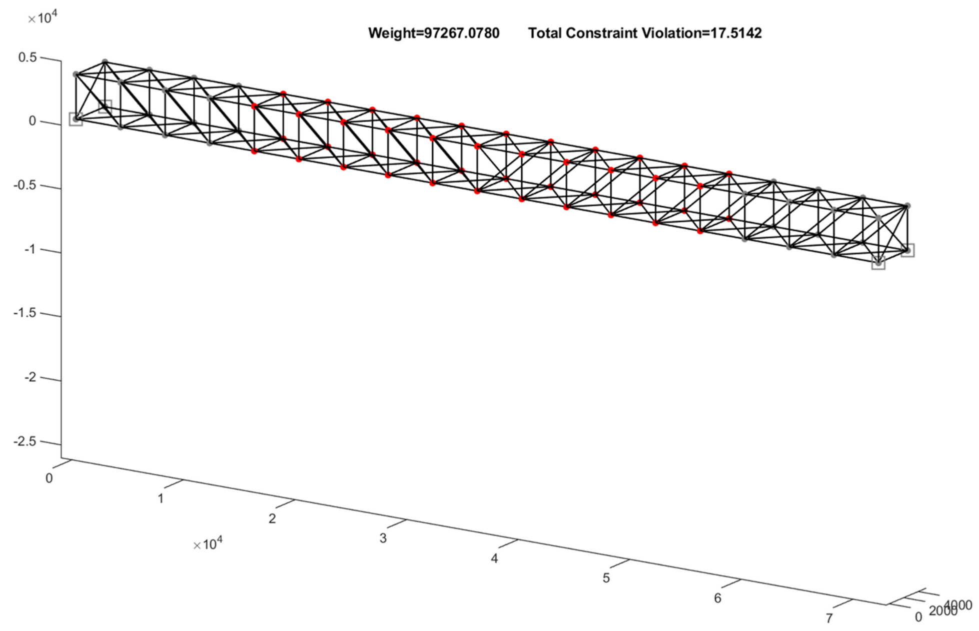
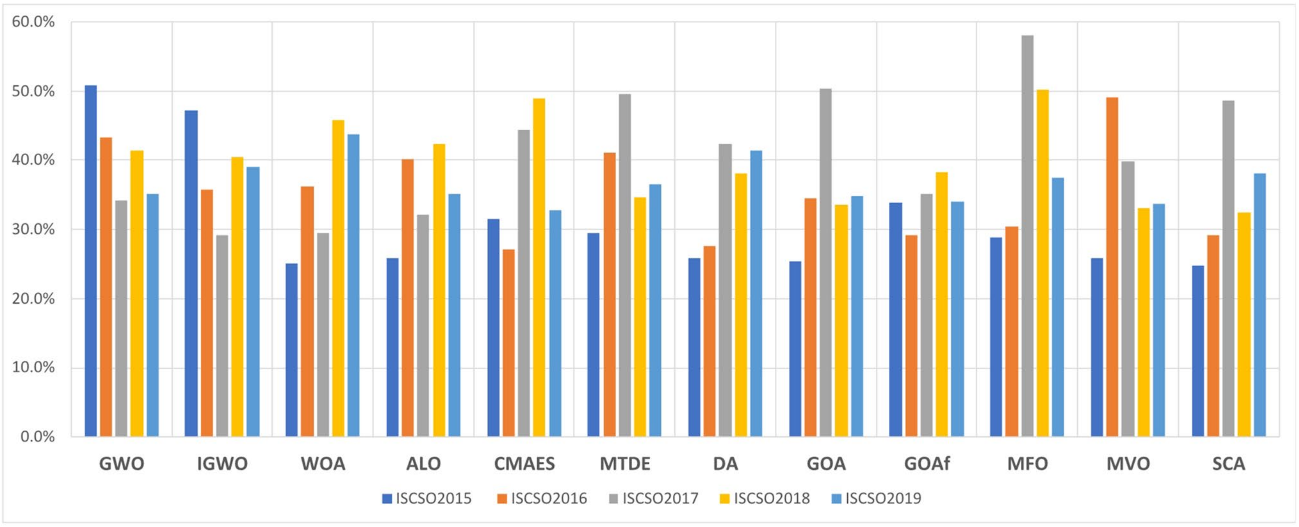


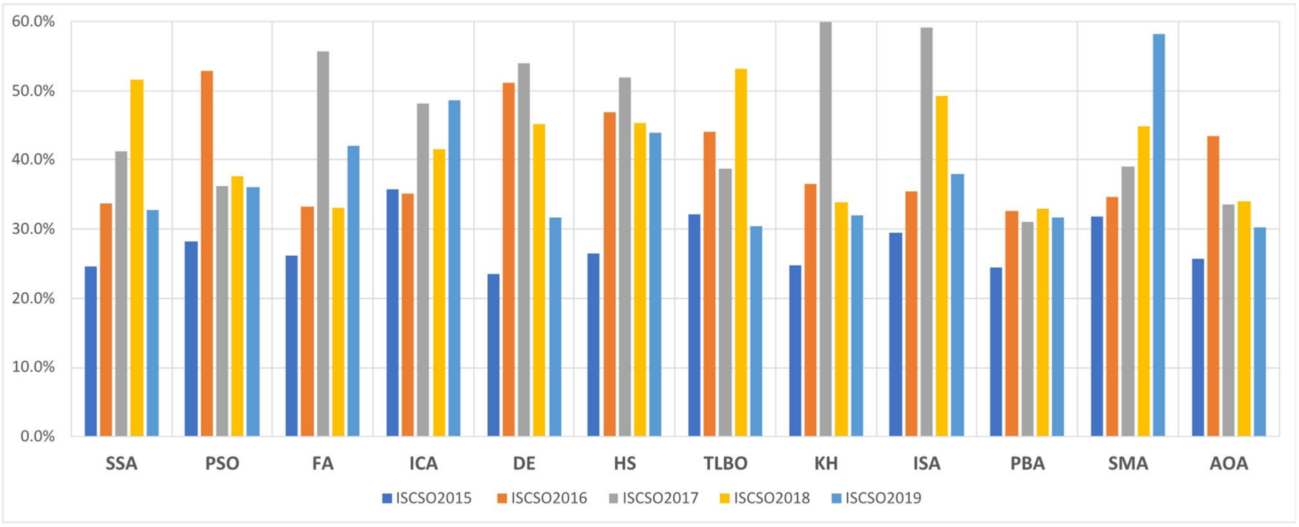
Fig. 7 The ISCSO2019 three-dimensional truss problem (horizontal dimensions in mm)

structure shown in Fig. 7. The structure is composed of 260 members and 76 nodes. The optimization problem consists of 260 sizing design variables (representing the cross-sectional areas of the truss members) and 10 shape design variables (representing 10 characteristic z-coordinates of the structure affecting the locations of 38 nodes). The sizing variables are discrete, taking integer values ranging from 1 to 37 representing the section ID from a database of 37 pipe sections. The structure is designed according to AISC-LRFD 1994 regulations, where each member is assessed considering the limit states of tensile yielding and compressive buckling. The constraint functions refer to (i) stress constraints where the truss members should satisfy the stress requirements of the code, and (ii) displacement constraints where the absolute value of the displacement of any node should not exceed the limit of 25.0 mm. More information about the problem formulation (including loading conditions, design variables, etc.) can be found in [13].

The best objective value achieved in the framework of the competition is equal to 12329.1302 kg, taken as the reference value for comparison. The results obtained from the 24 MOAs, for the ISCSO2019 problem, are presented in Table 13. In this test example, AOA outperformed the other algorithms resulting to the *Best* optimized value of 17,697.21 kg, followed by TLBO, PBA, DE and KH. When the median value is taken into account, the top-5 performers are TLBO (23380.96 kg), GOA, DE, SSA and ALO while for the average value, the relevant ranking is TLBO



(a)



(b)

Fig. 8 Performance of the 24 algorithms in the group of the 5 ISCSO test problems: **a** Algorithms 1–12, **b** Algorithms 13–24

(25144.96 kg), SSA, ALO, GOA and DE. The least coefficient of variation is exhibited by SMA (6.86%), followed by IGWO, FA, HS and ICA. Nevertheless, the result of SMA in terms of best value is very poor (29440.1 kg, the worst of all algorithms). The worst performers in terms of the median value achieved are DA (35332.21 kg), IGWO, SCA, SMA and ICA, while if the best achieved value is taken into consideration the algorithms with the worst performances are SMA (29440.1 kg), ICA, HS, WOA and FA.

5.2.6 Comparative Results

Figure 8 shows the variation (or relative error value) of the best achieved optimum solution by each one of the 24 MOAs, in comparison to the reference (best) solution found in framework of the competitions, for each problem. Overall, the error values vary from the lowest value of 23.49% (DE optimizer, ISCSO2015 problem) to the highest value of 59.88% (KH optimizer, ISCSO2017 problem). A general finding is that these structural optimization problems are hard and much more demanding than the ones examined in the previous section where most of the algorithms did an excellent job in finding solutions very close to the known global optimum.

Considering the difficulty and overall complexity of each problem, it appears that the first problem of ISCSO2015 was the least demanding, with the optimizers managing an average error value of 29.48% (median value 26.27%) altogether and the best (minimum) error value of 23.49% (DE optimizer). The most demanding problem appears to be the one of ISCSO2017, with an average error value of 43.40% for the 24 MOAs altogether (median value 41.81%) and the best (minimum) error value of 29.08% (IGWO optimizer). No optimizer managed to give results with error values less than 20% in comparison to the reference (best found) solution, in any of the examined problems. This is a clear indications that these problems are very complex and hard to deal with.

6 Conclusions

Metaheuristic optimization algorithms (MOAs) have proved to be very efficient, able to handle various optimization problems in several scientific fields during the last decades. The study presented a state-of-the-art review of past and current developments achieved so far in structural optimization problems dealt with MOAs. In addition, 24 well-known MOAs are presented in short in a unified description framework aiming to identify their differences and similarities, while they are also investigated in several structural optimization problems of varying complexity and difficulty. The numerical tests belong to two groups. The first six problems are benchmark structural optimization problems taken from the literature, while the next five problems are taken from the International Student Competition in Structural Optimization (2015–2019). The investigated MOAs exhibited excellent performance in handling the first six problems. Most of the algorithms managed to find the vicinity of the optimum in the majority of the problems rather easily, while 12 of them achieved optimal results leading to error values less than 2% in all problems examined. The top-3 performers managed to end up to solutions with average values (i.e. average over all 6 problems) less than 0.16% in all problems examined, combined. These results show the great potential of MOAs in handling structural optimization problems.

The results of MOAs were not so impressive in the case of the five problems taken from the International Student Competition in Structural Optimization. It appears that these problems are extremely hard, incorporating a large number of design variables. The examined MOAs were not able to provide solutions with error values less than 20% (in comparison to the reference solution) in any of the examined problems. The best performance was 23.49% far from the optimum reference value, which is not an impressive result, but from an engineer point of view it is not a bad result, also. Practically the algorithms were unable to find the vicinity of the optimum in the huge, multi-dimensional search space of these problems. At this point, it has to be noted that the optimizers were simply run with random initialization of the design variables without having any particular knowledge or guidance on the specific optimization problem at hand. There were no heuristic rules or tips that the optimizers could use to facilitate their search; they faced the problems "blindly". In a real-life situation, an experienced engineer may be able to help the optimizer by providing tips and guidance based on experience and intuition. For example, the engineer can facilitate the search by appropriately grouping variables based on existing symmetries on the structure, or can guide the optimizer towards specific areas of the search space based on the expected shape of the optimal structure, or other expected outcomes. This can boost the optimization procedure as it can quickly guide the optimizer near the neighbourhood of the global minimum and thus drastically reduce the size of the search space in practice, especially in cases with a large number of design variables, such as the competition problems examined in this study. In this sense, it can be said that in structural optimization problems, an optimization algorithm is a powerful tool in the hands of an experienced engineer, rather than an expert system that can provide solutions merely on its own. In other words, the expert needs the optimizer, but the optimizer also needs the expert, in order to achieve the best possible results.

Acknowledgements This research has been co-financed by the European Union and Greek national funds through the Operational Program

Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH-CREATE-INNOVATE (project code: T1EDK-05603).

References

1. Dulaimi MF et al (2002) Enhancing integration and innovation in construction. Build Res Inf 30(4):237–247. https://doi.org/10.1080/09613210110115207
2. Plevris V, Tsiatas G (2018) Computational structural engineering: past achievements and future challenges. Front Built Environ 4(21):1–5. https://doi.org/10.3389/fbuil.2018.00021
3. Slaughter ES (1998) Models of construction innovation. J Constr Eng Manage 124:226–231. https://doi.org/10.1061/(ASCE)0733-9364(1998)124:3(226)
4. Sahab MG, Toropov VV, Gandomi AH (2013) A review on traditional and modern structural optimization: problems and techniques. In: Gandomi AH et al (eds) Metaheuristic applications in structures and infrastructures. Elsevier, Oxford, pp 25–47. https://doi.org/10.1016/B978-0-12-398364-0.00002-4
5. Kashani AR et al (2022) Population-based optimization in structural engineering: a review. Artif Intell Rev 55(1):345–452. https://doi.org/10.1007/s10462-021-10036-w
6. Bekdaş G et al (2019) Optimization in civil engineering and metaheuristic algorithms: a review of state-of-the-art developments. In: Platt GM, Yang X-S, Silva Neto AJ (eds) Computational intelligence, optimization and inverse problems with applications in engineering. Springer, Cham, pp 111–137. https://doi.org/10.1007/978-3-319-96433-1_6
7. Yang X-S, Bekdaş G, Nigdeli SM (2016) Review and applications of metaheuristic algorithms in civil engineering. In: Yang X-S, Bekdaş G, Nigdeli SM (eds) Metaheuristics and optimization in civil engineering. Modeling and optimization in science and technologies. Springer, Berlin. https://doi.org/10.1007/978-3-319-26245-1_1
8. Lagaros ND (2014) An efficient dynamic load balancing algorithm. Comput Mech 53(1):59–76. https://doi.org/10.1007/s00466-013-0892-1
9. International Student Competition in Structural Optimization (2015) (ISCSO 2015). https://www.brightoptimizer.com/problem_iscso2016/. Accessed 25 May 2021
10. International Student Competition in Structural Optimization (2016) (ISCSO 2016). http://www.brightoptimizer.com/optimization-problem-of-iscso-2016/. Accessed 25 May 2021
11. International Student Competition in Structural Optimization (2017) (ISCSO 2017). https://www.brightoptimizer.com/problem_iscso2017/. Accessed 25 May 2021
12. International Student Competition in Structural Optimization (2018) (ISCSO 2018). https://www.brightoptimizer.com/problem_iscso2018/. Accessed 25 May 2021
13. International Student Competition in Structural Optimization (2019) (ISCSO 2019). https://www.brightoptimizer.com/problem-iscso2019/. Accessed 25 May 2021
14. Kaveh A (2021) Advances in metaheuristic algorithms for optimal design of structures, 3rd edn. Springer, Cham
15. Brockett RW (1991) Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems. Linear Algebra Appl 146:79–91. https://doi.org/10.1016/0024-3795(91)90021-N
16. Lyamin AV, Sloan SW (2002) Lower bound limit analysis using non-linear programming. Int J Numer Meth Eng 55(5):573–611. https://doi.org/10.1002/nme.511
17. Yokota T, Gen M, Li Y-X (1996) Genetic algorithm for non-linear mixed integer programming problems and its applications. Comput Ind Eng 30(4):905–917. https://doi.org/10.1016/0360-8352(96)00041-1
18. Dadebo SA, McAuley KB (1995) Dynamic optimization of constrained chemical engineering problems using dynamic programming. Comput Chem Eng 19(5):513–525. https://doi.org/10.1016/0098-1354(94)00086-4
19. Wang F-S, Chen L-H (2013) Heuristic Optimization. In: Dubitzky W et al (eds) Encyclopedia of systems biology. Springer, New York, NY, pp 885–885. https://doi.org/10.1007/978-1-4419-9863-7_411
20. Sörensen K, Glover FW (2013) Metaheuristics. In: Gass SI, Fu MC (eds) Encyclopedia of operations research and management science. Springer, Boston, MA, pp 960–970. https://doi.org/10.1007/978-1-4419-1153-7_1167
21. Glover F, Samorani M (2019) Intensification, diversification and learning in metaheuristic optimization. J Heuristics 25(4):517–520. https://doi.org/10.1007/s10732-019-09409-w
22. Meraihi Y et al (2021) Grasshopper optimization algorithm: theory, variants, and applications. IEEE Access 9:50001–50024. https://doi.org/10.1109/ACCESS.2021.3067597
23. Yang X, Suash D (2009) Cuckoo Search via Lévy flights. In 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)
24. Yang X-S, Deb S (2010) Engineering optimisation by cuckoo search. Int J Math Model Numer Optim 1(4):330–343. https://doi.org/10.1504/IJMMNO.2010.03543
25. Gandomi AH, Yang X-S, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29(1):17–35. https://doi.org/10.1007/s00366-011-0241-y
26. Yang X-S, Deb S (2013) Multiobjective cuckoo search for design optimization. Comput Oper Res 40(6):1616–1624. https://doi.org/10.1016/j.cor.2011.09.026
27. Cheng M-Y, Prayogo D (2014) Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput Struct 139:98–112. https://doi.org/10.1016/j.compstruc.2014.03.007
28. Yang X-S (2010) A new metaheuristic bat-inspired algorithm. In: González JR et al (eds) Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, Berlin. https://doi.org/10.1007/978-3-642-12538-6_6
29. Yang XS, Gandomi AH (2012) Bat algorithm: a novel approach for global engineering optimization. Eng Comput 29(5):464–483. https://doi.org/10.1108/02644401211235834
30. Shadravan S, Naji HR, Bardsiri VK (2019) The sailfish optimizer: a novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. Eng Appl Artif Intell 80:20–34. https://doi.org/10.1016/j.engappai.2019.01.001
31. Heidari AA et al (2019) Harris hawks optimization: algorithm and applications. Futur Gener Comput Syst 97:849–872. https://doi.org/10.1016/j.future.2019.02.028
32. Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput Struct 169:1–12. https://doi.org/10.1016/j.compstruc.2016.03.001
33. Eskandar H et al (2012) Water cycle algorithm—a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct 110–111:151–166. https://doi.org/10.1016/j.compstruc.2012.07.010
34. Farshi B, Alinia-ziazi A (2010) Sizing optimization of truss structures by method of centers and force formulation. Int J Solids Struct 47(18):2508–2524. https://doi.org/10.1016/j.ijsolstr.2010.05.009

35. Kociecki M, Adeli H (2013) Two-phase genetic algorithm for size optimization of free-form steel space-frame roof structures. J Constr Steel Res 90:283–296. https://doi.org/10.1016/j.jcsr.2013.07.027
36. Hasançebi O et al (2009) Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures. Comput Struct 87(5):284–302. https://doi.org/10.1016/j.compstruc.2009.01.002
37. Kaveh A et al (2010) Performance-based seismic design of steel frames using ant colony optimization. J Constr Steel Res 66(4):566–574. https://doi.org/10.1016/j.jcsr.2009.11.006
38. Moayyeri N, Gharehbaghi S, Plevris V (2019) Cost-based optimum design of reinforced concrete retaining walls considering different methods of bearing capacity computation. Mathematics 7(12):1–21. https://doi.org/10.3844/jcssp.2018.1351.1362
39. Gholizadeh S, Milany A (2018) An improved fireworks algorithm for discrete sizing optimization of steel skeletal structures. Eng Optim 50(11):1829–1849. https://doi.org/10.1080/0305215X.2017.1417402
40. Tan Y, Zhu Y (2010) Fireworks algorithm for optimization. In: Tan Y, Shi Y, Tan KC (eds) Advances in swarm intelligence. ICSI 2010. Lecture notes in computer science. Springer, Berlin. https://doi.org/10.1007/978-3-642-13495-1_44
41. Bureerat S, Pholdee N (2016) Optimal truss sizing using an adaptive differential evolution algorithm. J Comput Civ Eng 30(2):04015019. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000487
42. Hasançebi O, Azad SK (2012) An exponential big bang-big crunch algorithm for discrete design optimization of steel frames. Comput Struct 110–111:167–179. https://doi.org/10.1016/j.compstruc.2012.07.014
43. Lagaros ND et al (2008) Optimum design of steel structures with web openings. Eng Struct 30(9):2528–2537
44. Papadrakakis M, Lagaros ND, Plevris V (2001) Optimum design of space frames under seismic loading. Int J Struct Stab Dyn 1(1):105–123. https://doi.org/10.1142/S0219455401000093
45. Papazafeiropoulos G, Plevris V (2018) OpenSeismoMatlab: a new open-source software for strong ground motion data processing. Heliyon 4(9):1–39. https://doi.org/10.1016/j.heliyon.2018.e00784
46. Fragiadakis M, Lagaros ND, Papadrakakis M (2006) Performance-based multiobjective optimum design of steel structures considering life-cycle cost. Struct Multidiscip Optim 32(1):1–11
47. Mitropoulou CC, Lagaros ND, Papadrakakis M (2011) Life-cycle cost assessment of optimally designed reinforced concrete buildings under seismic actions. Reliab Eng Syst Saf 96(10):1311–1331. https://doi.org/10.1016/j.ress.2011.04.002
48. Kociecki M, Adeli H (2014) Two-phase genetic algorithm for topology optimization of free-form steel space-frame roof structures with complex curvatures. Eng Appl Artif Intell 32:218–227. https://doi.org/10.1016/j.engappai.2014.01.010
49. Kociecki M, Adeli H (2015) Shape optimization of free-form steel space-frame roof structures with complex geometries using evolutionary computing. Eng Appl Artif Intell 38:168–182. https://doi.org/10.1016/j.engappai.2014.10.012
50. Amir O (2013) A topology optimization procedure for reinforced concrete structures. Comput Struct 114:46–58
51. Lagaros ND, Papadrakakis M, Bakas N (2006) Automatic minimization of the rigidity eccentricity of 3D reinforced concrete buildings. J Earthq Eng 10(4):533–564
52. Zakian P, Kaveh A (2020) Topology optimization of shear wall structures under seismic loading. Earthq Eng Eng Vib 19(1):105–116. https://doi.org/10.1007/s11803-020-0550-5
53. Kaveh A, Kalatjari V (2003) Topology optimization of trusses using genetic algorithm, force method and graph theory. Int J Numer Meth Eng 58(5):771–791. https://doi.org/10.1002/nme.800
54. Tian X et al (2019) Topology optimization design for offshore platform jacket structure. Appl Ocean Res 84:38–50. https://doi.org/10.1016/j.apor.2019.01.003
55. de Souza RR et al (2016) A procedure for the size, shape and topology optimization of transmission line tower structures. Eng Struct 111:162–184
56. Jiang B, Zhang J, Ohsaki M (2021) Shape optimization of free-form shell structures combining static and dynamic behaviors. Structures 29:1791–1807. https://doi.org/10.1016/j.istruc.2020.12.045
57. Papadrakakis M, Tsompanakis Y, Lagaros ND (1999) Structural shape optimization using evolution strategies. Eng Optim 31(4):515–540
58. Lagaros ND, Fragiadakis M, Papadrakakis M (2004) Optimum design of shell structures with stiffening beams. AIAA J 42(1):175–184
59. Belevičius R et al (2017) Optimization of rigidly supported guyed masts. Adv Civ Eng. https://doi.org/10.1155/2017/4561376
60. Mam K et al (2020) Shape optimization of braced frames for tall timber buildings: influence of semi-rigid connections on design and optimization process. Eng Struct 216:110692. https://doi.org/10.1016/j.engstruct.2020.110692
61. Pastore T et al (2019) Topology optimization of stress-constrained structural elements using risk-factor approach. Comput Struct 224:106104. https://doi.org/10.1016/j.compstruc.2019.106104
62. Frangedaki E, Sardone L, Lagaros ND (2021) Design optimization of tree-shaped structural systems and sustainable architecture using bamboo and earthen materials. J Archit Eng 27(4):04021033. https://doi.org/10.1061/(ASCE)AE.1943-5568.0000492
63. Plevris V, Papadrakakis M (2011) A hybrid particle swarm—gradient algorithm for global structural optimization. Comput-Aided Civ Infrastruct Eng 26(1):48–68. https://doi.org/10.1111/j.1467-8667.2010.00664.x
64. Plevris V (2009) Innovative computational techniques for the optimum structural design considering uncertainties. National Technical University of Athens, Athens, p 312
65. Kennedy J, Eberhart R (1995) Particle swarm optimization. In IEEE International Conference on Neural Networks, Piscataway, NJ, pp 1942–1948
66. Aydilek İB (2018) A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems. Appl Soft Comput 66:232–249. https://doi.org/10.1016/j.asoc.2018.02.025
67. Yang X-S (2008) Nature-inspired metaheuristic algorithms. Luniver Press, ISBN: 1905986106
68. Gholizadeh S, Salajegheh E, Torkzadeh P (2008) Structural optimization with frequency constraints by genetic algorithm using wavelet radial basis function neural network. J Sound Vib 312(1):316–331. https://doi.org/10.1016/j.jsv.2007.10.050
69. Nguyen T-H, Vu A-T (2021) Speeding up composite differential evolution for structural optimization using neural networks. J Inf Telecommun. https://doi.org/10.1080/24751839.2021.1946740
70. Papadrakakis M, Lagaros ND, Tsompanakis Y (1998) Structural optimization using evolution strategies and neural networks. Comput Methods Appl Mech Eng 156(1–4):309–333
71. Papadrakakis M, Lagaros ND (2002) Reliability-based structural optimization using neural networks and Monte Carlo simulation. Comput Methods Appl Mech Eng 191(32):3491–3507
72. Lagaros ND, Charmpis DC, Papadrakakis M (2005) An adaptive neural network strategy for improving the computational

performance of evolutionary structural optimization. Comput Methods Appl Mech Eng 194(30–33):3374–3393
73. Lagaros ND, Papadrakakis M (2012) Applied soft computing for optimum design of structures. Struct Multidiscip Optim 45(6):787–799. https://doi.org/10.1007/s00158-011-0741-9
74. Lagaros ND, Papadrakakis M (2004) Learning improvement of neural networks used in structural optimization. Adv Eng Softw 35(1):9–25
75. Liao TW (2010) Two hybrid differential evolution algorithms for engineering design optimization. Appl Soft Comput 10(4):1188–1199. https://doi.org/10.1016/j.asoc.2010.05.007
76. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359. https://doi.org/10.1023/a:1008202821328
77. Storn R, Price K (1995) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. J Global Optim
78. 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
79. Carbas S (2016) Design optimization of steel frames using an enhanced firefly algorithm. Eng Optim 48(12):2007–2025. https://doi.org/10.1080/0305215X.2016.1145217
80. Talatahari S et al (2015) Optimum design of frame structures using the eagle strategy with differential evolution. Eng Struct 91:16–25. https://doi.org/10.1016/j.engstruct.2015.02.026
81. Yang X-S, Deb S (2010) Eagle strategy using Lévy walk and firefly algorithms for stochastic optimization. In: González JR et al (eds) Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, Berlin. https://doi.org/10.1007/978-3-642-12538-6_9
82. Khalilpourazari S, Khalilpourazary S (2019) An efficient hybrid algorithm based on Water Cycle and Moth-Flame Optimization algorithms for solving numerical and constrained engineering optimization problems. Soft Comput 23(5):1699–1722. https://doi.org/10.1007/s00500-017-2894-y
83. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl-Based Syst 89:228–249. https://doi.org/10.1016/j.knosys.2015.07.006
84. Lagaros ND (2018) The environmental and economic impact of structural optimization. Struct Multidiscip Optim 58(4):1751–1768. https://doi.org/10.1007/s00158-018-1998-z
85. Mavrokapnidis D, Mitropoulou CC, Lagaros ND (2019) Environmental assessment of cost optimized structural systems in tall buildings. J Build Eng 24:100730. https://doi.org/10.1016/j.jobe.2019.100730
86. Papadrakakis M et al (1998) Advanced solution methods in structural optimization based on evolution strategies. Eng Comput 15(1):12–34
87. Papadrakakis M, Lagaros ND, Fragakis Y (2003) Parallel computational strategies for structural optimization. Int J Numer Meth Eng 58(9):1347–1380
88. Lagaros ND (2014) A general purpose real-world structural design optimization computing platform. Struct Multidiscip Optim 49(6):1047–1066. https://doi.org/10.1007/s00158-013-1027-1
89. Lagaros ND, Karlaftis MG (2016) Life-cycle cost structural design optimization of steel wind towers. Comput Struct 174:122–132. https://doi.org/10.1016/j.compstruc.2015.09.013
90. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82. https://doi.org/10.1109/4235.585893
91. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
92. Nadimi-Shahraki MH, Taghian S, Mirjalili S (2021) An improved grey wolf optimizer for solving engineering problems. Expert Syst Appl 166:113917. https://doi.org/10.1016/j.eswa.2020.113917
93. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
94. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98. https://doi.org/10.1016/j.advengsoft.2015.01.010
95. Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9(2):159–195. https://doi.org/10.1162/106365601750190398
96. Nadimi-Shahraki MH et al (2020) MTDE: an effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems. Appl Soft Comput 97:106761. https://doi.org/10.1016/j.asoc.2020.106761
97. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27(4):1053–1073. https://doi.org/10.1007/s00521-015-1920-1
98. Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. Adv Eng Softw 105:30–47. https://doi.org/10.1016/j.advengsoft.2017.01.004
99. Mishra P, Goyal V, Shukla A (2020) An improved grasshopper optimization algorithm for solving numerical optimization problems. In: Mohanty MN, Das S (eds) Advances in intelligent computing and communication. Springer, Singapore
100. Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 27(2):495–513. https://doi.org/10.1007/s00521-015-1870-7
101. Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120–133. https://doi.org/10.1016/j.knosys.2015.12.022
102. Mirjalili S 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
103. Atashpaz-Gargari E, Lucas C (2007) Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In 2007 IEEE Congress on Evolutionary Computation
104. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. SIMULATION 76(2):60–68. https://doi.org/10.1177/003754970107600201
105. Rao RV, Savsani VJ, Vakharia DP (2011) Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315. https://doi.org/10.1016/j.cad.2010.12.015
106. Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845. https://doi.org/10.1016/j.cnsns.2012.05.010
107. Gandomi AH (2014) Interior search algorithm (ISA): a novel approach for global optimization. ISA Trans 53(4):1168–1183. https://doi.org/10.1016/j.isatra.2014.03.018
108. Kallioras NA, Lagaros ND, Avtzis DN (2018) Pity beetle algorithm—a new metaheuristic inspired by the behavior of bark beetles. Adv Eng Softw 121:147–166. https://doi.org/10.1016/j.advengsoft.2018.04.007
109. Li S et al (2020) Slime mould algorithm: a new method for stochastic optimization. Futur Gener Comput Syst 111:300–323. https://doi.org/10.1016/j.future.2020.03.055

110. Abualigah L et al (2021) The arithmetic optimization algorithm. Comput Methods Appl Mech Eng 376:113609. https://doi.org/10.1016/j.cma.2020.113609
111. Yang X-S (2009) Firefly algorithms for multimodal optimization. In: Watanabe O, Zeugmann T (eds) Stochastic algorithms: foundations and applications. Springer, Berlin
112. Yang X-S (2014) Chapter 8—firefly algorithms. In: Yang X-S (ed) Nature-inspired optimization algorithms. Elsevier, Oxford, pp 111–127. https://doi.org/10.1016/B978-0-12-416743-8.00008-7
113. Georgioudakis M, Plevris V (2020) A comparative study of differential evolution variants in constrained structural optimization. Front Built Environ 6(102):1–14. https://doi.org/10.3389/fbuil.2020.00102
114. Georgioudakis M, Plevris V (2020) On the performance of differential evolution variants in constrained structural optimization. Procedia Manuf 44:371–378. https://doi.org/10.1016/j.promfg.2020.02.281
115. Georgioudakis M, Plevris V (2018) A combined modal correlation criterion for structural damage identification with noisy modal data. Adv Civ Eng 2018(3183067):20. https://doi.org/10.1155/2018/3183067
116. Tuo S, Geem ZW, Yoon JH (2020) A new method for analyzing the performance of the harmony search algorithm. Mathematics 8(9):1421
117. Ocak A et al (2022) Optimization of tuned liquid damper including different liquids for lateral displacement control of single and multi-story structures. Buildings 12(3):377
118. Tsipianitis A, Tsompanakis Y (2020) Improved Cuckoo Search algorithmic variants for constrained nonlinear optimization. Adv Eng Softw 149:102865. https://doi.org/10.1016/j.advengsoft.2020.102865
119. Kannan BK, Kramer SN (1994) An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. J Mech Des 116(2):405–411. https://doi.org/10.1115/1.2919393