National Technical University of Athens

School of Civil Engineering

Laboratory of Structural Analysis and Antiseismic Research

Master of Science Thesis

Evolutionary Computation and Swarm Intelligence Algorithms in Structural Optimization

By: Postgraduate Student/ Emad Abdalghaffar

Supervisor: Prof./ Nikos D. Lagaros

July 2021

Inspiration

Optimization is the heart of many natural processes, to name a few: the *natural selection* phenomenon in the biological evolution theory that is based on the survival-of-the-fittest principle, the *social swarming* behavior of birds or the *foraging strategies* of ants. Hence, the concept of simulating such natural phenomena into computational mechanics in form of computer algorithms may offer a very promising approach in the nowadays computer-aided engineering designs.

Preface

This Pharaonic-Grecian work aims to frame the art of the evolutionary and swarm based structural optimization. That is through five chapters. At the first chapter, the structural optimization problem is properly defined as well to elaborating its basic components. Then, an intensive review of the literature is introduced, through screening the most recent and most cited scientific articles in the topic during the first two decades of the 21st century. Afterwards, a set of the most promising and trending state-of-the-art algorithms is introduced through descriptive paragraphs that elaborate how each algorithm works. Following such elaboration of the literature and the state of the art, numerical tests assessing the performance of these most promising algorithms have been done in order to identify the capability of each algorithm, and eventually estimating the future trends in Structural Optimization.

Dedicated to dad.

Acknowledgement

Thanks to Professor Nikos Lagaros, the supervisor of this work, for the generous way to start it all up with me from scratch at the optimization topic, also for the valuable comments and feedback throughout this work. Thanks to Mr. Lagaros too for the indirect inspiration through his dedication and superiority in the field.

Thanks to Professor Vlasis Koumousis, who credited the very early interest sparks for me towards the topic. Mr. Koumousis smoothly managed to grab our attention towards the topic during that light Wednesday classes, back in 2019, in that coldish but stunning weather of Zografou's afternoon.

Thanks

To mama for the midnight juice and light sandwiches (sometimes Mahshi and Pastitsio nearly at 2 a.m.), as well to the cooking classes via video calls while being abroad. To brother and sisters who were/are/will always be a robust backbone all the way. Thanks a million to Amina Ramadan, the funniest and prettiest partner, literally nothing passed during these last two years, amongst homesick moments and stressful exam days, without her own unique way to help me healing such feelings. No one can bear what she does.

I must also thank Mr. Souvlaki (κοτόπουλο κοντοσούβλι) for the great company in Athens.

Content

Ch 1 Introduction to Optimization

1.1 Definition and Problem Formulation

"Optimization" term refers to obtaining the best fit (maximum/minimum) value of a certain single or multi-objective function (e.g., safety, cost or time, in structural optimization), corresponding to pre-defined variables within a feasible space that is bounded with some pre-defined constraints (e.g., stresses, displacements or frequencies, in structural optimization). Considering the computational cost and the consumed time as key performance indicators (KPIs) of the optimization process.

The general mathematical formulation of an optimization problem can be described as follow:

$$
\begin{aligned}
&Optimize\ f(X) \\
&g_i(X) \le 0 \quad , \quad i = 1, 2, \dots k \\
&h_i(X) = 0 \quad , \quad i = 1, 2, \dots p \\
&x_i^L \le x_i \le x_i^U \quad , \quad i = 1, 2, \dots n
\end{aligned}
$$

Where $f(X)$ is the objective function, $X = x_1, x_2, \dots x_n$ (design variables), $g_i(X)$ is an inequality constraint, $h_i(X)$ is an equality constraint, x_i^L and x_i^U are the lower and upper bounds of the ith design variable. k, p and n are the numbers of the inequality constraints, equality constraints and design variables, respectively.

The engineering optimization has conquered variant engineering fields and has witnessed a hot pace of scientific research. In contrast with the so-common manual (trial and error) approach in the engineering design, optimization techniques are employed to obtain optimal solutions faster and in high accuracy, as well to saving huge human efforts. Recently, equipping these optimization techniques with the trending AI features (e.g., Fuzzy Logic) gets the process done even faster and more open to handle general families of optimization problems.

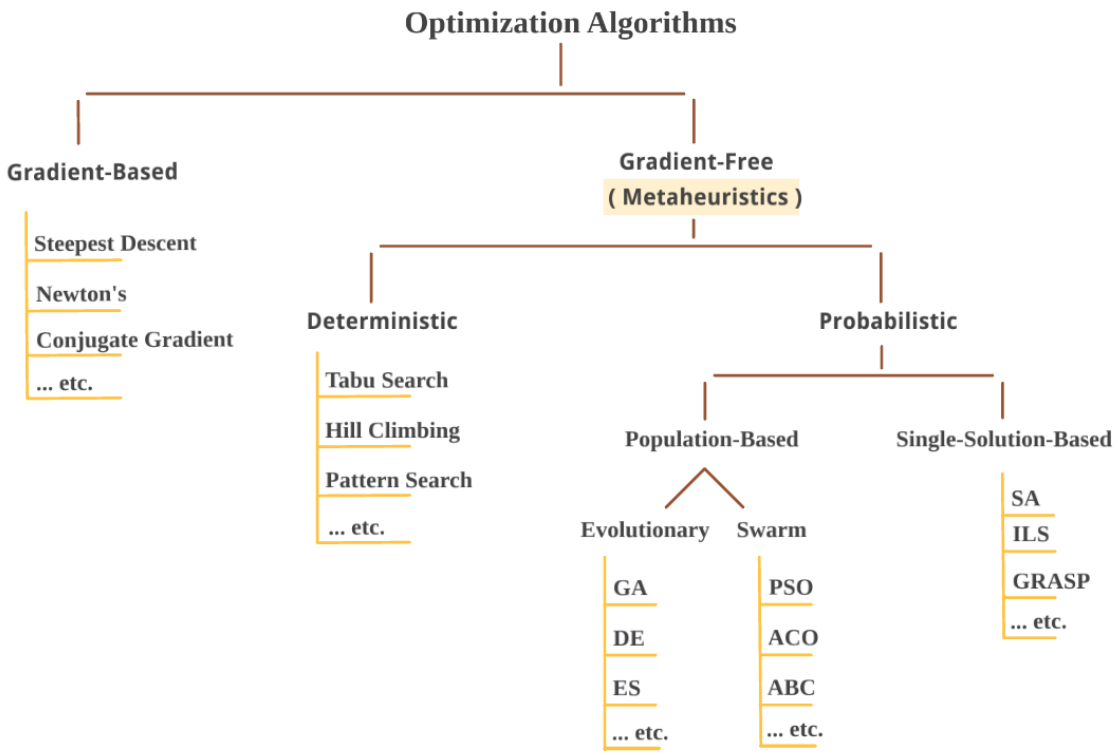
The main two classes of the optimization approaches are the derivative-based and the derivative-free. The former one tends to find the exact optimal solution, by definition, it uses information from the objective and/or constraints functions and derivates to proceed forward. Which requires continuous and differentiable functions. Although, this derivative-based class of approaches is generally faster and more capable to handle large number of design variables than that other class. However, it is competent in finding a local optimal, but not the global optimal. And also, it requires more mathematics as mentioned, which leads to prohibitive computational burdens consumption in case of high complexity problems. By time, many optimization problems have been issued as being non-deterministic polynomial-time hardness (NP-hardness) problems, that hold too much complexity in case of being handled by the derivative-based approaches. Basically, due to getting stuck in local optimum, discontinuity of the objective function or when the constraints and/or objective functions require complex simulations. Therefore, the so-called Heuristic approaches (derivative-free) has come to play thanks to its ease of implementation and light consumption of the computational burdens in handling high complexity real-world problems, but with relatively low number of design variables. But it worth mentioning that these derivative-free approaches tend to find a near-to-optimum solution, not the exact one.

The heuristic approaches were developed each-for-each problem. That is why, the theory behind this kind of approaches should had been promoted to introduce the nowadays called Metaheuristic approaches, as general frameworks that handle general families of problems. The population-based metaheuristic approaches in structural optimization is the main topic to be elaborated through this thesis.

1.2 Classification of The Optimization Approaches

The optimization approaches are either derivative-based or derivative-free approaches. The former class includes techniques of nonlinear, linear, geometric, quadratic, or integer programming. The other class includes the metaheuristic techniques, which are divided into two subcategories, defined as deterministic and probabilistic. The deterministic algorithms do not use any kind of randomness operators, it is more useful in searching for local optimal. On the other side, the probabilistic algorithms employ randomness operators either in the initialization phase or in setting some design parameters during the optimization process. The probabilistic approaches are either single-point-based or population-based approaches. The single-point-based could be either trajectory or discontinuous approaches. The population-based could be evolutionary computation, swarm intelligence or other nature-inspired algorithms.

The following figure illustrates the above-mentioned classification of the optimization approaches



1.3 Metaheuristic Algorithms

This kind of optimizers deals with discontinuous and non-differentiable functions, holds randomness in the initialization operator and in some internal operators (e.g., crossover and mutation operators in GA), as well to being highly efficient and dedicated to find near to global optimum in high complex real-world optimization problems. The two fundamental components of any metaheuristic optimizer are the exploration and exploitation, in another words, diversification and intensification. In the exploration phase, the optimizer diverges the search to scan and explore the entire search domain, this behavior is considered as an abrupt but produces a progressive movement of the candidate solutions. While in the exploitation phase, the optimizer scans certain promising regions in the explored domain, the regions that hold the best found solutions in the exploration phase. Enabling a suitable balance between exploration and exploitation highly increases the efficiency of the optimizer. As a result of this fact, many techniques are developed and equipped into the algorithms to maintain a proper balance. The most efficient and trending metaheuristic algorithms are population-based that follow either evolutionary computation or swarm intelligence techniques.

Most of the metaheuristic algorithms are basically inspired by the nature. Since the base of any natural process is indeed based on the optimization definition. To name but a few of these natural processes: the *natural selection* phenomenon in the biological evolution theory that is based on the survival-of-the-fittest principle, the *social swarming* behavior of birds or the *foraging strategies* of ants.

1.4 Constraint Handling Approaches

The optimization processes have been successfully applied to various optimization problems. However, they are not able to handle constrained optimization problems directly. In the past few decades, much work has been done to enhance the optimization algorithms performance in order to be able to deal with constrained optimization problems. Hence, many approaches are developed in order to ensure the feasibility of the provided optimum solution.

Following are presented a couple of words for some of the state-of-the-art most commonly used constraint handling approaches:

Static Penalty Function
The most common approach among the researchers. Where all the constraints of a problem are incorporated into its objective function, introducing the so-called fitness function. By changing the value of the penalty parameter, different solution could be achieved. Also, inappropriate values of that penalty factor may lead the search away from the global optimum.

Dynamic Penalty Function
The same as the Static Penalty Function, but in the Dynamic Penalty Function, the penalty factor is automatically updated during the optimization process according to the behavior of the swarm.

Fly-Back method

A trending constraint handling method, which considers the type of the violated constraints. In contrast with the Penalty Function approach. The main advantage of the Fly-Back technique is that it does not have any tunning parameters.

Improved Fly-Back method

The standard Fly-Back, but considering the type of the violated constraints. It works in three main steps: (i) determining whether the updated individual violates the constraints or not, (ii) for violation, finding which components cause the violation. Then, replacing them with corresponding components, and (iii) determining before updating the individual's current position if the new position is within the feasible space and provides better fitness function value than old particle.

Feasibility Criteria

Proposed by Deb [1], a constraint handling method that uses three feasible criteria as selection mechanisms: (i) Any feasible individual is chosen over any infeasible one. (ii) For two feasible individuals, the individual with the better objective function value is chosen. (iii) For two infeasible individuals, the individual with smaller constraint violation is chosen.

Stochastic Ranking

Proposed by Runarsson and Yao in 2000 [2], a constraint handling method where a pre-defined parameter controls the balance between the feasibility and infeasibility, as no penalty parameter is used. The preference between two individuals is based on both the objective function value and the value of the constraint violations.

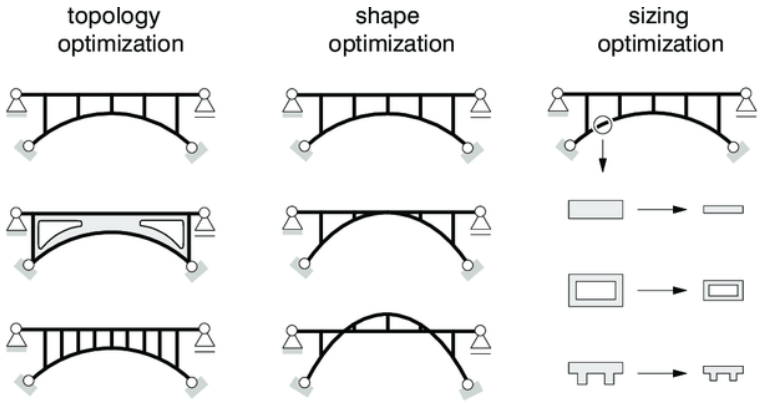
The ϵ-Constrained approach

In the ϵ-constrained method, the comparison between two individuals at a time is based on the objective function value and the constraint violation value as well. The parameter $\boldsymbol{\epsilon}$ controls the level of comparison. As in case of $\boldsymbol{\epsilon}$ is very large, the comparison is highly considering the objective function value rather than the constraint violation value. While if $\boldsymbol{\epsilon} = 0$, the ordering rule is supposed to precede the constraint violation value on the objective function value.

1.5 Structural Optimization

Structural Optimization can be traced back to 1904, when Michell [3] has derived formulae for weight minimizing of structures subjected to stress constraints. Afterwards during the era of the computational mechanics, the Architectural, Engineering and Construction (AEC) industry has experienced a hot pace of intensive developments in the structural optimization area. Where the designer's day-to-day issue of selecting proper systems that provide good distribution of the material over the design space, in a way that offers economical and stable structures, is never been handled before in a way like such nowadays computer-aided designs. Hence, the structural optimization current-state technology is an important output of the computational mechanics advances.

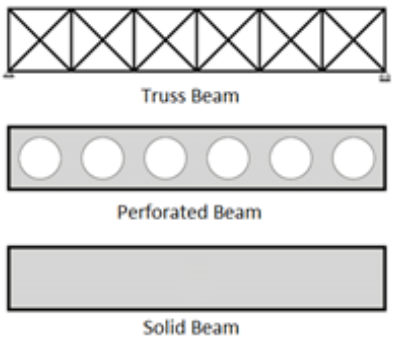
There are three aspects that could be structurally optimized in any structure: size, topology or shape.



- **Topology Optimization**

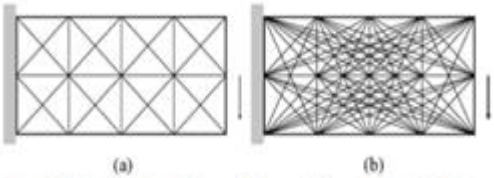
  This is the most general type of structural optimization, where the optimizer deals with the configuration of the members (placing and number of members) as the design variables.
  In another word, the topology optimization is the removal/placing material only where effective.
  "Topology" word is originated from the Greek word "τόπος (topos)" that means locus.

  The best examples of the topology optimization are the truss beam and also the so-called perforated beam, where the material is only placed where they perform optimally, instead of filling the whole design space with material, such like a solid beam.



  The most common approach for the truss structures topology optimization is the "ground structure approach", which was first introduced by Dorn et al. (1964) and is now widely used in all kinds of truss topology optimization. In this approach, the nodal locations are fixed and the ground structure is created by connecting any two nodes.

There are two kinds of the ground structure approach, either to be Partly Ground Structure where the members length is restricted to a certain value, or to be Fully Ground Structure where no length restriction is applied. Although in the Partly Ground the computational effort is reduced, but it may lead to an optimal solution which is not the global one.



Two Kinds of Classic Truss Ground Structures: (a) Partly Ground Structure, (b) Fully Ground Structure

In the structural topology optimization, to achieve a feasible solution, the stability of the structure should be ensured during the optimization process. In this respect, several methods based on graph theory and algebraic approaches were provided to check this criterion (e.g., Geometrical Consistency Check [36], Evaluation of the condition number of the stiffness matrix [39])

- **Shape Optimization**

This is the most complex type of structural optimization, where the main task is to minimize the effective stresses at some local within the general layout of the structure. There are two approaches of the shape structural optimization. Either to be based on *FE models* where the optimizer deals with the nodes' coordinates of the structure's boundaries as the design variables, which usually leads to an enormous number of design variables and accordingly causing prohibitive computational burdens. Or to be based on *geometrics models* where the optimizer deals with the parameters of the geometry model as the design variables. So that, a link between the structural analysis (FE) model and the geometry model is maintained, as the modifications in the geometry model parameters lead to changes in the FE model. In another word, besides the structural analysis and the optimization algorithms modules, the shape optimization that is based on the geometric models operates also a third module that models the variant possible shapes, which is called the geometrical representation module [4]. The shape optimization is applied at the end of the design process when the general layout is determined through a topology optimization, where minor modifications are applicable on the design variables (nodes' coordinates).

- **Size optimization**

This is the easiest type of structural optimization, where the optimizer deals with the dimensions of the cross sections as the design variables. Consequently, any modifications in the section properties during the optimization results in changes in the structural analysis (FE) model. The size optimization is usually a discrete optimization where the design space consists of specific values of the design variables (e.g., standard cross sections of I-section steel member), however it could also be a continuous optimization where the section area of the members are limited within a continuous range (e.g., 1 < section area < 3 in^2). Similar to the shape optimization, the size optimization is applied at the end of the design process when the general layout is determined through a topology optimization, where minor modifications are applicable on the design variables (cross sections dimensions).

Ch 2 Literature Review of the PB-MOAs in Structural Optimization

The PB-MOAs are usually built based on evolutionary computations (e.g., GA), swarm intelligence (e.g., PSO) or based on other nature-inspired concepts (e.g., TLBO). In this chapter a review of the literature is introduced where 28 of the most recent and most cited scientific articles are reviewed and presented. These articles are presented in three sub-sections: Evolutionary Computation Metaheuristics, Swarm Intelligence Metaheuristics and Other Nature-Inspired Metaheuristics.

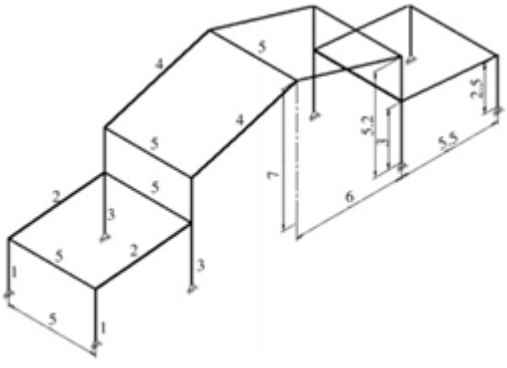
A. Evolutionary Computation Metaheuristics

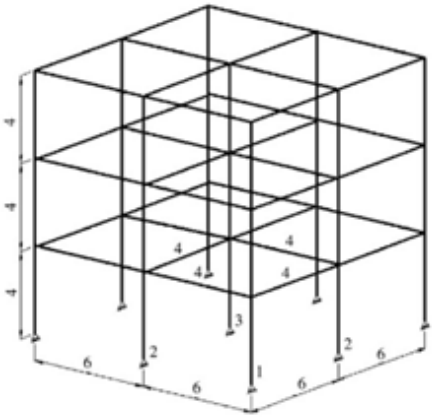
1) Design optimization of 3D steel structures: Genetic algorithms vs. classical techniques [5]

In this paper an elitist Genetic Algorithm (GA) developed by the authors is compared with a common commercial structural analysis program in a size optimization of spatial structures. The main implemented modifications in the standard GA to introduce the proposed algorithm are: phenotype crossover operator to select real commercial sections, new selection operator that removes the worst individuals from the population and new codification of the design variables that lets them all have the same initial probability of being selected. This paper does not consider any limitations for the displacements of the nodes. The tested benchmarks are discrete variables problems (2835 sections from the Spanish Basic Building Code). In order to represent the 2835 sections in binary form, a 12-bit chains are used. For the structural analysis, ESCAL3D software is used.

The optimization Process starts by assigning randomly initial sections to the members, a text file is generated called “start.exit”, which together with the file (“optimum.in”), enables the optimization module to run. The “start.exit” file contains the data needed to calculate the structure. The “optimum.in” file contains the code to apply. In this study, five complete evolutions were carried out, producing five solutions for each benchmark structure.

Two benchmark structural problems are tested in order to investigate the capability of the proposed algorithm: industrial portal frame and three-floors steel building. For the portal frame, the weight obtained by the proposed GA is 1507 kg, while the commercial software’s solution is 2160 kg. Also, in the steel building design the proposed GA shown better capability, where the obtained weight is 17910 kg, while being 19668 kg by the commercial software. Basically, this superiority of the proposed GA is due to the random nature of the section assignment process. However, that was at higher computational cost.





2) Structural optimization with frequency constraints by genetic algorithm using wavelet radial basis function neural network [6]

By combining Genetic Algorithm (GA) and Neural Networks (NN), this article is presented to find optimal weights of structures subjected to multiple natural frequency constraints.
The GA in the form of the Virtual Sub-Populations (VSP) method is employed to find the optimal weight of a structure. Using the VSP method is mainly to reduce the consumed computational efforts. As comparatively to the standard GA, the initial randomly selected population in VSP is much smaller, on which the GA operators are subsequently applied, then the best-found individuals in that initial small population are passed to the next same-size populations, iteratively. The remaining individuals of each new population are randomly selected. In another word, instead of implementing the mathematical models of the GA's natural selection operators on a relatively big population in one shot, like in the standard GA, these mathematical models are implemented on a randomly selected initial small population, which is iteratively upgraded to better consecutive populations. This is called virtual sup-population approach, which saves huge computational efforts.
In order to reduce the computational time and efforts that are consumed in evaluating the structures' natural frequencies using the common FE methods, a properly trained radial basis function neural network that called wavelet radial basis function (WRBF) is employed to predict these natural frequencies. In the WRBF, the activation function of the hidden layer neurons is substituted with the Cosine-Gaussian Morlet daughter wavelet function.
By this article, the WRBF has been used for the first time in the literature in evaluating the natural frequencies, as previously it was used commonly in setting up structural systems. The employed constraints handling approach in this study is the Penalty Function method.
Two benchmark structural problems are tested: 10-bar aluminium truss and 200-bar steel double layer grid. Both are discrete problems where the design variables are selected from a list of standard cross sections. The size optimization is performed combing the VSP-GA and WRBF, giving optimum weights 538 and 1483 kg for the 10-bar and 200-bar structures, respectively. While the VSP-GA and the standard RBF provided 548 and 1492 kg. And the optimum weights by the standard GA and WRBF are 550 and 1530 kg. While the standard GA with the standard RBF provided 556 and 1543 kg. As shown, the minimum obtained weight is by VSP-GA with WRBF.

3) Optimum design of shallow foundation using evolutionary algorithms [7]

This investigation article is proposed to implement a performance assessment of three famous evolutionary algorithms (Differential Evolution DE, Evolutionary Strategy ES and Biogeography-Based Optimization Algorithm BBO, in addition to four recent variations of that former-mentioned three (biogeography-based optimization with covariance matrix-based migration CMM-BBO, linear population size reduction success-history-based adaptive differential evolution algorithm L-SHADE, weighted differential evolution algorithm WDE and improved differential evolution IDE). This assessment is done on an optimization problem of RC foundation cost minimization. The assessment is done comparatively with the most famous evolutionary algorithm: Genetic Algorithm GA. The problem is optimized under three different cases: shallow footing design subjected to uniaxial load at

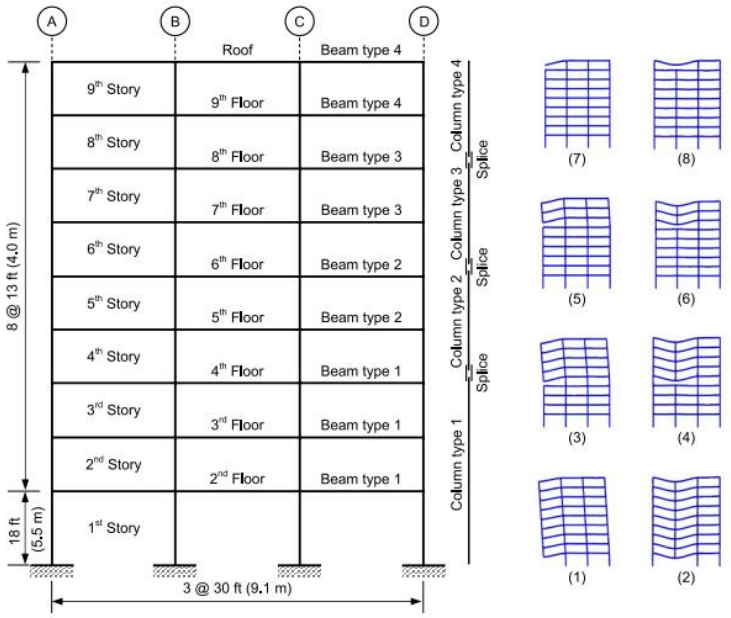
the center (case-1), effective moment is added to case-1 (case-2) and the impact of relocating the column is considered in case-2 (case-3).
This above-mentioned benchmark problem has both discrete and continuous design variable domains, for the number of steel bars variable (integer number) and for the concrete dimension variables, respectively. The Penalty Function approach is employed for constraint handling.
The results shown that: none of the assessed algorithm shows the best efficiency for the three cases, where L-SHADE algorithms provided the best optimum for case-1, WDE for case-2, and BBO for case-3. Also, BBO & WDE algorithms are able to deal successfully with the three cases.

4) Progressive collapse design of seismic steel frames using structural optimization [8]

This work uses the GA to cost-effectively design of steel moment frames against the possible progressive collapse modes, which is caused by a sudden removal of critical columns. The design satisfies both AISC seismic provisions and UFC progressive collapse requirements. It is not a goal of this study to obtain global optimal-weight design. This study aims to show the capability of optimization in obtaining cost-effective designs against the possible progressive collapse modes, especially within few GA generations.

The tested benchmark structural problem is a 2D nine-story three-bay immediate moment steel frame with a fixed base. It is a discrete problem, where the design variables are selected from a list of standard cross sections. The structural analysis is performed using the alternate path method provided in the 2009 edition of the United States Department of Defense United Facilities Criteria (UFC 4-023-03), considering each of the three analysis options: linear static, nonlinear static, and nonlinear dynamic. For the linear static procedure, an in-house linear elastic program is used. For the nonlinear static and nonlinear dynamic procedures, the DRAIN-2DX program is used to create a planar analytical structural model that accounts for both material and geometrical nonlinearities.

The results came that the steel weight provided by optimizing the frame considering progressive collapse modes is 476 kN in case of non-linear dynamic analysis, 498 kN in case of non-linear static analysis and 611 kN in case of linear static analysis. While in case of not considering the progressive collapse is 440 kN, which is less but as mentioned it was not a goal of this study to obtain global optimal-weight design but to show the capability of optimization in obtaining cost-effective designs against the possible progressive collapse modes.



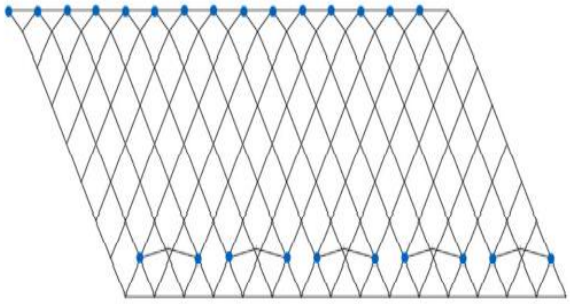
5) Two-phase genetic algorithm for size optimization of free-form steel space-frame roof structures [9]

In this investigation work, an actual design of two free-form steel space-frame roof structures that is performed in a design office by one of the authors (over a period of days), is used as an initial design in a size optimization using the investigated the two-phase GA. In phase one, an initial population of solutions is generated around that initial design for each design variable (depth, width and thickness of rectangular hollow sections), using a normal probability distribution. Then the GA's operators are applied producing a near-global but slightly infeasible optimal solution. Then after in phase two, that found solution by phase one is considered as an initial design for phase two, but after increasing the thickness dimension of the overstressed members by one increment at a time until all constraints are met and the solution is a feasible one. Afterwards, similar to phase one, an initial population is created around that initial design of phase two using a normal probability distribution, but with a smaller range of design variables. That created initial population in phase two is used as the initial population in phase one, repeatedly till one of the stopping criterions is met.

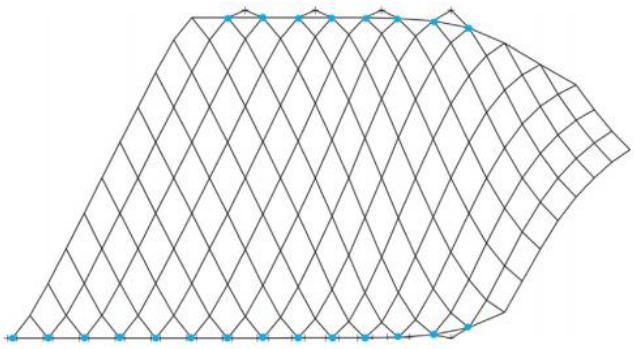
The investigated algorithm is developed through this article in seek of better convergence, less computational time and practical optimum weight (size optimization).

The tested benchmark structures are two free-form steel space-frame roof structures (Two of the thirteen train stations making up the Ottawa Light Rail Transit (OLRT) system in Ottawa, Canada, in 2018.). They are discrete problems, as the sections are selected from a list that holds commercially available rectangular hollow sections. For the structural analysis, SAP2000 is employed, considering snow, dead, wind, and earthquake loads assuming linearly elastic behavior.

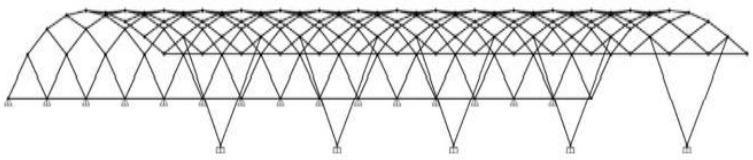
The investigated two-phase GA provided solutions less than the actual design of the author's design office, by 12% and 7% for station 1 and station 2, respectively.



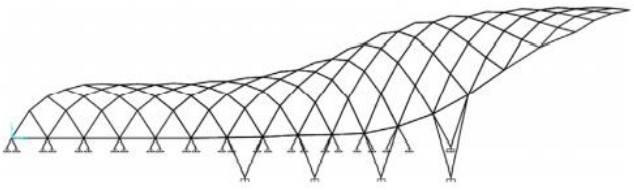
Top view of Station 1



Top view of Station 2



Side view, along the length, of Station 1

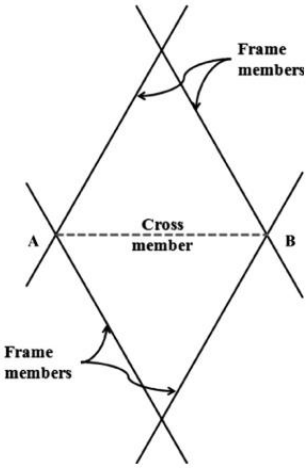


Side view, along the length, of Station 2

6) Two-phase genetic algorithm for topology optimization of free-form steel space-frame roof [10]

This article is an extension of article 5 [9], where a topology optimization features are equipped to the two-phase GA that performs size optimization for two free-form steel space-frame roof structures. Topology optimization is performed in phase one through randomly adding cross member in some of the grid's diamond panels, splitting the diamond into two triangles in shape. This is mainly to add resistance to the members against the in-plane buckling, and accordingly increasing the section capacity.

The results shown that for station 1, the optimum weight is 220 kips, while the design office weight is 214 kips. And for station 2, the optimum weight is 252 kips, while the design office weight is 275 kips. Which means that the design office weight is lighter that of the investigated algorithm for station 1, while the opposite happens for station 2. Which leads to a conclusion that: in less complexity of such roof structures (like station 1), the diamond pattern is recommended than using cross members.

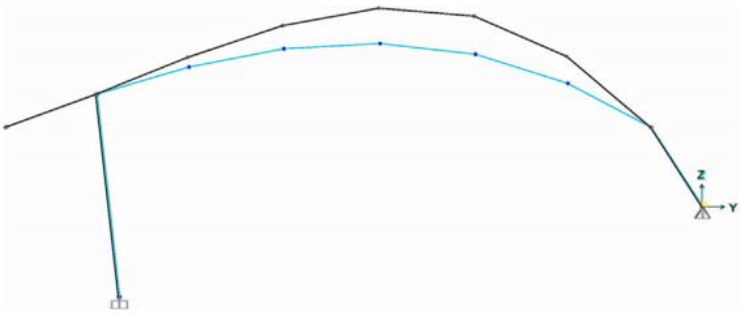


7) Shape optimization of free-form steel space-frame roof structures with complex geometries using evolutionary computing [11]

This article is an extension of articles 5 and 6 [9] [10], where extra features are equipped to perform shape optimization as well to the already existing features of the size and topology optimization. So, presented in this article a new methodology which handles all of sizing, topology, and shape optimization of free-form steel space-frame roof structures with complex geometries using the two-phase GA. Both topology and shape optimization are performed during phase one, in phase two the unfeasible solution of phase one is just moved towards the feasible search space.
The shape optimization is achieved through randomly changing only the vertical coordinates of the roof joints.

As deduced from article 5) that no topology optimization is recommended for such roof structures with simple geometrics, so station 1 in this article has no cross members added, only size and shape optimization are applied. While by performing the shape optimization, it comes out with better solution for both stations.

The final optimal solutions by the two-phase GA optimization are less than the design office solution by 25% for both stations.



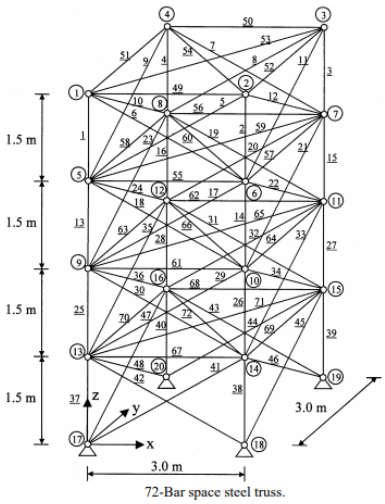
Side view of the final optimal shape (dark line) along with the initial shape (light line) for Station 1

8) Dynamic Optimization of Structures Subjected to Earthquake [12]

Employing Genetic Algorithm (GA) and Neural Networks (NN), this article is proposed to reduce the computational time of structural weight optimization of a space truss structure subjected to the El Centro (S-E 1940) earthquake loads, that occurred in California. This investigated algorithm seeks reducing the computational time of the optimization process through using:

- Self-Organizing Neural System (SONS), in the approximate predictions of the time history responses. Which is an improved version by this work's authors of the Intelligent Neural System INS [13], through substituting the original classification neural networks of the INS by a self-organizing map (SOM). (SOM is developed by Kohonen [14]). This improvement resulted in a new neural system that consists of two main components: a smart classification component (SOM) that classifies the input space data (natural frequencies) into subspaces, as each subspace have similar natural frequencies. And a prediction component which is a set of some parallel RBF networks, each of them is trained to cover one of the classified subspaces to accordingly predict its corresponding time history responses in terms of node displacements and element stresses, which is the target space of the SONS, and the optimization constraints as well.
- The GA in the form of the Virtual Sub-Populations (VSP) method is employed to find an optimal weight. Which is comparatively less consumer of both computational efforts and time than the standard GA, due to the less number of the implemented mathematical models. As in the VSP, instead of implementing the mathematical models of the GA's natural selection operators on one relatively big population in one shot, like in the standard GA, these mathematical models are implemented on a randomly selected initial small population, which is iteratively upgraded to better consecutive populations till reaching an optimal solution.

The tested benchmark is a 72-bar space truss structure, subjected to the El Centro (S-E 1940) earthquake. It is a discrete variables problem (specific cross sections). ANSYS is employed to provide the training data after performing the time history analyses. While MATLAB is employed to design the neural networks. As the structure is subjected to dynamic loads, then the stress and displacement constraints are functions of time. So that, the constrains are transformed from time-dependent to time-independent through the conventional method of dynamic constraints [15]. The structural analysis is performed three times with three different methods: exact analysis (EA) -FE analysis-, approximate analysis by a single RBF neural network and approximate analysis by SONS neural network.



72-Bar space steel truss.

The optimum design weight obtained using SONS is better than that obtained using RBF network (without SOM). However, SONS's solution not better than the exact analysis solution. But as the objective of this article is to reduce the computational time, so that is fulfilled already as the computational time consumed by SONS and EA are 7 and 2538 min, respectively.

9) Sizing and topology optimization of truss structures using genetic programming [16]

This paper presents a structural optimization genetic programming (SOGP) approach for simultaneous sizing and topology optimization of truss structures. Genetic Programming is a prominent stochastic evolutionary algorithm.

In contrast with the GA that utilizes fixed-length representation of the population, the GP adopts variable-length computer programs in the form of binary trees. Which empowers GP to identify redundant truss members in complex problems. Each GP individual represents a truss. GP is more efficient in exploring the search space than other EAs. As well to simultaneously synthesizing its structure and tuning its parameters, which makes it capable of simultaneously performing size and topology optimization.

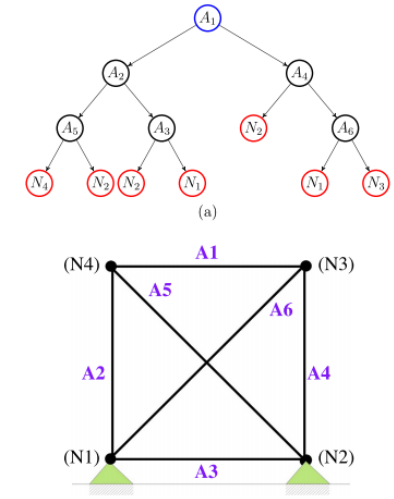


Illustration of a sample tree decoded into a truss structure

The three tested benchmark structural problems are: 10-element 6-joint truss, 17-element 9-joint truss and 39-element 12-joint truss. They have a continuous domain for the design variables. The penalty function is employed for the constraint handling.

The results shown that the investigated approach SOGP shown higher capability in optimizing the three benchmarks structures, comparatively to eight famous optimization algorithms in the literature: Conventional methods [17,18,19], Genetic algorithm (GA) [20], Harmony Search algorithm (HS) [21], Heuristic Particle Swarm Optimization (HPSO) [22], Ant Algorithm [23], Adaptive Multi-Population Differential Evolution (AMPDE) [24], Grammatical Evolution (GE) and Dual-Optimization in Grammatical Evolution (DO-GE) [25], Hybrid Particle Swarm and Swallow Optimization algorithm (HPSSO) [26] and Water Evaporation Optimization (WEO) [27].

10) Guided genetic algorithm for dome optimization against instability with discrete variables [28]

As the “stability” is a decisive factor in the design of domes, this article is presented in order to enhance the stability and accordingly to improve the buckling capacity of space domes. Through a guided genetic algorithm GGA. This article considers the joints as rigid.

As an index to represent the stability of a dome from the perspective of joint well-formedness, the relative gradient of joint well-formedness (gra_r) is defined: the gra_r is a simple scalar that indicates the tendency to lose stability quantitatively rather than the ability to keep stable (gra_r is identified from the structural response). The lowest value of gra_r (gra_r_{min}) is considered as an indication of the buckling capacity (Pcr). The higher the value of gra_r_{min}, the greater the Pcr that the dome provides. Accordingly, the optimization objective is set as the maximization of (gra_r_{min}), in order to get an optimum (maximum) buckling capacity of the domes.

The GGA works on guided mutation rather than stochastic mutation of the standard genetic algorithm, to realize oriented evolution for rapid search, as follow:

- In the random mutation, the members to be altered are randomly selects. While in the guided mutation, only the critical members are altered.
- In the random mutation, the sections may be strengthened or weakened. While in the guided mutation, the risky members are strengthened and the stiffness-redundant members are weakened.
- In the random mutation, section's size may be changed greatly. While in the guided mutation, a critical section's size is just slightly changed. Thus, the guided mutation guarantees the continuity in terms of both chromosome and phenotype, which leads to a robust optimization algorithm.

Two benchmark structures are optimized in order to investigate the capability of the GGA, which are two space domes: dome-1 (22m span) and dome-2 (50m span). The two size optimization problems are discrete variables problems (specific cross sections).

The results shown that the GGA optimum buckling capacity is 94.36 and 32.55 kN/m^2, while by the standard GA is 93.84 and 28.44 kN/ m^2 for dome-1 and dome-2, respectively. In addition to that the solutions by the GGA are obtained in less computational time and number of generations.

11) Design optimization of domes against instability considering joint stiffness [29]

This article is an extension of articles 10 [28], where a single-layer space dome is optimized to improve the buckling capacity of the dome by a guided Genetic Algorithms (GGA), while joint flexibility is further considered in this article in the optimization process. Thus, in this paper the instability mechanism could be caused by the failure of members or joints as well.
The relative gradient of the well-formedness of flexible joints, denoted as gra_r, is defined as an index that represents the stability of the dome from the perspective of joint well-formedness that is achieved from both members and joints stability, as joints are not considered rigid as in article 10 [28].

The objective function is still to maximize the minimum gra_r, in order to get an optimum (maximum) buckling capacity of the dome. The benchmark structures are the same two space dome as in articles 10 [28]. They are size optimization discrete problems (section variables: specific cross sections, joint variables: diameter and stiffness of the joints).

The results shown that the buckling capacity obtained by the GGA while considering the joints as flexible is 92.55 and 32.11 kN/m^2, while being 94.22 and 32.55 kN/m^2 when the joints are considered rigid, for dome-1 and dome-2, respectively.

12) Shape optimization of cold-formed steel beam-columns with practical and manufacturing constraints [30]

This study aims to present a practical method for optimization of symmetric cold-formed steel (CFS) beam-column members using Genetic Algorithm (GA). A framework that aims to maximize the ultimate capacity of the CFS beam-columns under the combined effects of axial and bending stresses. The used CFS beam-column members are: short, intermediate and long in lengths (1000, 2000, and 3000 mm). Which are subjected to axial compressive loads with eccentricities that vary from e = 0, 10, 20 and 30 mm, in order to create different levels of bending moment about the X-axis. The cross-sectional shapes were based on using variations number of rollers (4, 6, 8, 10 or 12) and lips (1, 2 or 3). A roller is a joint between two segments. One lip is considered for the 4 rollers sections, up to two lips for the 6 and 8 rollers sections, up to three lips for the 10 and 12 rollers sections. This forms eleven cross-sectional shapes, which can be identified by a two-digit numbers standing for the number of rollers and the lip strips, e.g. 4-1 shape: 4 rollers and one lip.

132 different cases of beam-column member are shape-optimized (different in shape (11 shapes), length (1000, 2000 and 3000 mm) and load eccentricities (0, 10, 20 and 30mm)). The design variables domain is continuous, for the variables: the angles between the segments and the lengths of the segments.

Based on the results, sections with shape 10-2, 12-3 and 10-3 (as shown in [30]) are the optimum design solutions for the 1000, 2000 and 3000mm beam-column members, respectively, under the four eccentricities. The results shown also that as the load eccentricity increases, the shape of the optimized beam-column members changes from lumped to more spread shapes.

13) Coupled element and structural level optimisation framework for cold-formed steel frames [31]

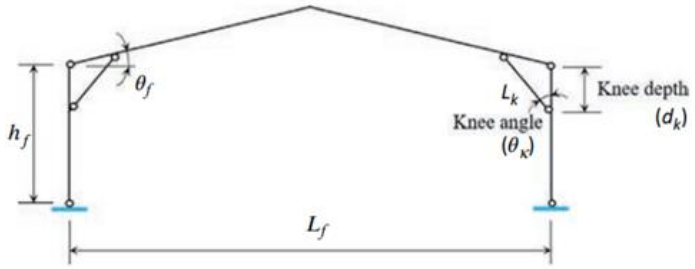
In this article, a coupled framework is presented for both size optimization and structural-performance optimization of CFS portal frames.
A Real-Coded Genetic Algorithm (RC-GA) was programmed to solve the objective functions. The main advantage of RC-GA compared to conventional binary GA methods is that the genetic operators are directly applied to the design variables without coding and decoding.

At first, a size optimization is performed for the standard CFS lipped-channel beam sections with respect to their flexural capacity. Where the design variables domain is continuous for each of the height, width and lip length. Then, structural-performance optimization is performed to find the best configuration of a pitched-frame with knee braces, in terms of the weight per unit area. This structural-performance optimization is performed twice, once using the standard CFS lipped-channel beam sections, and once else using the size-optimized sections. Where the design variables domains are discrete and continuous among the variables: knee depth, knee angel, frame spacing (2 to 20 m) and frame pitch (6° to 30°). A MATLAB code was developed to provide a link between ANSYS and the RC-GA optimization code.

For the size optimization, a simply-supported beam subjected to three different uniform distributed load (UDL) cases: 4, 6 and 8 kN/m, and in three different spans: 4, 6 and 8 m, and for the structural-

performance optimization, a pitched-frame with 13.6m span and 5.4m eave height, are considered as the benchmark problems in investigating the proposed algorithm.

The results shown that the optimum weight per unit area of the frame after implementing the two optimization kinds is 10.19 kg/m^2, while being 11.85 kg/m^2 when only size optimization is considered, and 12.75 kg/m^2 when neither considering size nor structural-performance optimization (using standard CFS sections and reference frame geometry and knee brace configuration).



Geometry of long-span CFS portal frame with knee braces

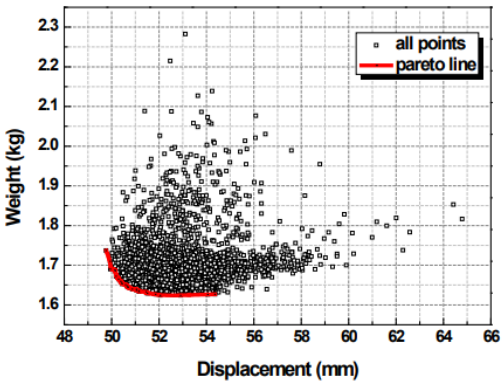
14) Multi-Objective Optimization of Spatially Truss Structures Based on Node Movement [32]

This article introduces and improved Multi-objective Evolutionary Algorithm (MOEA) method in topology optimization of spatially discrete structures. The innovation of this study is that it uses the coordinate of the nodes as the optimization variable. Considering the weight of the structure in addition to the displacement of the nodes as the objective functions, producing not a single solution, but a pareto frontier [33].

During the very few steps of the optimization, relatively few Pareto front points are existing, and often obvious inflection points. As the number of the iterations increase, the points on the pareto frontier line become evenly more distributed, and then inflection points become more blurred. As the process further continue, the pareto frontier does not stop heading forward until a convergence criterion is met.

Two benchmark structures are used in order to investigate the capability of the introduced method, which are: space truss and space tower.

The proposed algorithm in this article is tested and the results shown the MOEA's relatively fast rate of convergence and good diversity.

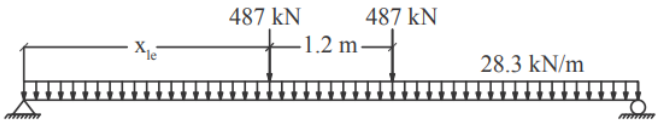


15) Optimizing the steel girders in a high strength steel composite bridge [34]

Using the Genetic Algorithm (GA), this study aimed at identifying the potential benefits of hybridizing a steel girder with different steel grades within its two flanges and web. These benefits considered as the weight, the material cost and the environmental impact (CO2 emission). Eighteen different combinations of different steel grades considered to perform size optimization of the girder. The size optimization was performed on a continuous domain of the variables: thickness, width and height of the I-section girder.

$10 \leqslant t_{uf} \leqslant 60$, $10 \leqslant t_{lf} \leqslant 60$, $10 \leqslant t_w \leqslant 40$, $200 \leqslant b_{uf} \leqslant 100$, $200 \leqslant b_{lf} \leqslant 1000$ and $1000 \leqslant h_w \leqslant 2800$

The considered benchmark structural problem in this study is an I-section steel bridge's girder with 32.1 m length. The results are analyzed as normalized values against the homogeneous S355 solution. The optimum solution with respect to the weight and CO_2 is achieved by the solution of a homogeneous combination of S690 grade for the web and both flanges. And the optimum solution with respect to the cost is achieved by the solution of a hybrid combination of S460/S460/S690 grades, for the web, upper flange and lower flange, respectively. By comparing these solutions with the solution of the conventional homogeneous combination of S355 grade, this proposed method offers 33% less in weight, 28.3% less in CO_2 and 16.4% less in cost. As a result's notice, the highest steel grade in such cases should be placed in the lower flange. Furthermore, the steel grade of the web and upper flange seems to be of equal importance.



B. Swarm Intelligence Metaheuristics

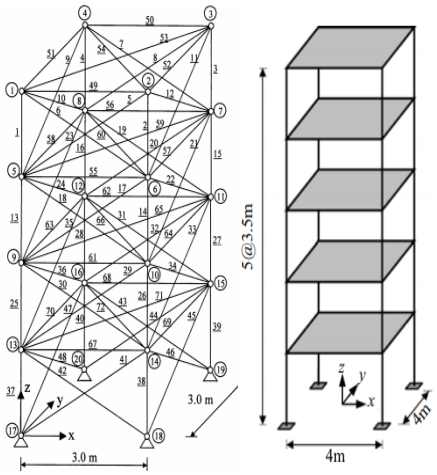
1) Optimal design of structures for earthquake loads by a hybrid RBF-BPSO method [35]

Employing each of swarm-based search technique (binary particle swarm optimization BPSO) and Neural System (RBF NN), this article is proposed to reduce the computational time of structural weight optimization of space truss and steel shear frame. This investigated algorithm seeks reducing the computational time of the optimization process through using:

- Radial Basis Function neural network (RBF) in the approximate predictions of the time history responses in terms of inter-story drifts as the target space of the RBF, which is also the optimization constraints. While the input space of the RBF-NN is the design variables (cross-sectional properties of the columns).
- Binary Particle Swarm Optimization (BPSO) for finding an optimal weight.

The tested benchmarks are 72-bar space truss structure subjected to the Californian El Centro (S-E 1940) earthquake loads and a 5-story steel shear frame structure subjected to the Chile (N10E-1985) earthquake loads. Both are discrete variables problems (specific cross sections). ANSYS is employed to provide the training data after performing the time history analyses. While MATLAB is employed to design the neural networks. As the structure is subjected to dynamic loads, then the stress and displacement constraints are functions of time. So that, the constrains are transformed from time-dependent to time-independent through the conventional method of dynamic constraints [15]. The structural analysis is performed two times with two different methods: exact analysis (EA) -FE Analysis-,
approximate analysis by the RBF neural network.

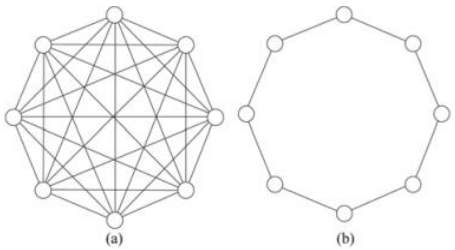
The optimum design weight obtained using RBF-BPSO is not better than that obtained by the exact analysis. But as the objective of this article is to reduce the computational time, so that is met already as the consumed computational time by RBF and EA in the optimization process of the 72-bar space truss are 6.4 and 2538 min, respectively. While for the 5-story steel shear frame, the consumed computational time by RBF and EA are 5.3 and 1342 min, respectively.



2) Solving Truss Topological Optimization with Discrete Design Variables via Swarm Intelligence [36]

The main purpose of this article is to display the potential of a modified Lbest-based PSO (MLPSO), while being employed to perform topology optimization, in cooperation with a geometrical consistency check that is tightly connected to the ground structure approach of topology optimization. By this work, the proposed method is investigated in finding the optimum layout of a truss structure with minimization of compliance (maximization of stiffness).
The proposed method employs the ground structure approach for constituting the design domain, then when the optimization algorithm (MLPSO) is applied, which searches at random for possible solutions within that design space, and before the structural analysis operator runs, a geometrical consistency check is applied in order to avoid computing any proposed non-realistic solutions (solutions with a mechanism or with redundant members), thereby reducing the computational effort and time.

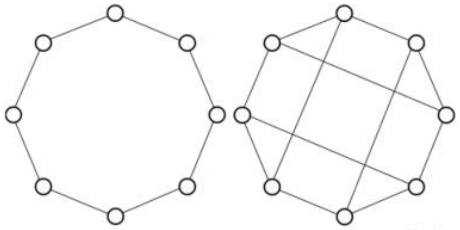
The L-best model of PSO is a ring-topology model of PSO, where each individual is not influenced by all the neighbor individuals but only the best-performance ones, in contrast with the other G-best model of PSO.



Topologies of Particle Swarm Optimizer:
(a) Gbest Model, (b) Ring Lbest Model

Three cases of a benchmark truss structure are considered: single-load wheel, single-load cantilever and single-load Michell beam. All are discrete-design-variable problems.

The benchmark structures are optimized two times with two different ring topologies of the L-best model (k=2 and k=3), with two different domains for the design variable (member volume) as well: [0,1] or [0,5]. So, four times in total for each structure.


Ring topology with $k=2$ Ring topology with $k=3$

The displacement is considered during the optimization as a state variable. The Quadratic Penalty Function is employed for the constraints handling. And for the discrete variable handling, the Rounding-Off strategy is employed.

The obtained results are not better than but same as the best-found solution in the literature of the same structures, in (Achtziger and Stolpe, 2007) [37]. For the above-mentioned four times optimization of each structure.

3) An integrated particle swarm optimizer for optimization of truss structures with discrete variables [38]

This study presents an integrated particle swarm optimization (iPSO) algorithm, through integrating the weighted particle concept to enhance the search capability overriding the local optimal traps within the search space. As in the standard PSO algorithm, when a particle stands very close to its own previous best position and/or to the global best particle, the role of one of this two guidance particles can be highly reduced or even be vanished. So that, the iPSO uses the weighted particle which is indeed a particle at the gravity center of the Pbest's swarm, in order to improve the flight path of these particles that are flying excessively close to either their own prior best point, that is stored in Pbest, or close to the global best point, that is stored in Gbest. The iPSO also integrates the Fly-Back technique to handle the optimization constraints. The constrained handling approach that is followed in this article in an improved version of the Fly-Back technique (He et al. 2004), which considers the type of the violated constraints. In contrast with the well-known Penalty Function approach, this Fly-Back technique does not have any tunning parameters. This Fly-Back technique guarantees the feasibility of the final solution as well, as it keeps all the particle in the feasible region during the optimization.

Four benchmarks truss structures are optimized in order to investigate the potential of the proposed iPSO in the size optimization of truss structures: 10-bar planar truss, 25-bar space truss, 72-bar space truss and 244-bar space-truss tower. All of them are discrete problems (standard cross sections). The results are compared with corresponding optimum solutions by other different algorithms in the literature, and the obtained weights by the proposed iPSO after size optimization show the best capability among the solutions of the other algorithms, for the four benchmark structures except only for the 25-bar truss structure where the iPSO's solution came as secondly-rated after an obtained solution by the well-known Genetic Algorithm.

4) Weight minimization of truss structures with sizing and layout variables using integrated particle swarm optimizer [39]

This article is an extension the done work in article 3 [38], where additional features are added to implement topology optimization as well to the size optimization. Which required also to add a stability check to the iPSO algorithm, as the topology optimization process sometimes leads to unstable systems due to the nature of randomly picking possible solutions. So that, an evaluation technique of the condition number of the stiffness matrix is equipped into the algorithm. Such that, if the condition number of the stiffness matrix is greater than a predefined large number, system is determined as unstable and accordingly ignored, saving computer's effort and time.

Five benchmarks truss structures are optimized in order to investigate the potential of the proposed iPSO in the size and topology optimization of truss structures: 15-bar planar truss, 18-bar planar truss, 25-bar space truss, 39-bar space truss and 47-bar space-truss tower. All of them are discrete problems (standard cross sections). The results are compared with corresponding optimum solutions by other different algorithms in the literature, and the obtained weights by the proposed iPSO algorithm after size and topology optimization came as secondly-rated solutions after the obtained solutions by the sequential cellular particle swarm optimization (SCPSO) algorithm (Gholizadeh 2013), for the first four benchmark structures. While the best weight for the fifth benchmark structure is obtained by the proposed iPSO algorithm.

5) Comparison of Two Metaheuristic Algorithms on Sizing and Topology Optimization of Trusses and Mathematical Functions [40]

The current study intends to compare the performances of two different metaheuristic algorithms in size-topology optimization, the Integrated Particle Swarm Optimizer (iPSO) (article 4 [39]) and the Teaching-Learning Based Optimizer (TLBO). It worthy to mention that in the TLBO, the objective function evaluations (OFEs) is the most time-consuming process. As the TLBO is a two-phase algorithm, so the objective function is evaluated twice in each iteration, once in the teaching phase and another in the learning phase.

Two truss structures are considered to demonstrate the feasibility and validity of the iPSO and the TLBO in handling the size and topology optimization of trusses: 11-bar and 39-bar truss structures. The two problems have continuous domain for the design variables (cross sections and nodes' coordinates). The iPSO provided a slightly better solutions than the TLBO, as the obtained weight of the 11-bar truss by the iPSO is 21709 N, while being 21784 N by the TLBO. And for the 39-bar truss, 867 N by the iPSO, and 877 by the TLBO. Furthermore, the superiority of the iPSO (single-phase: one OFE/iteration) over the TLBO (double-phase: two OFEs/iteration) is obviously determined regards the consumption of the computer burdens, in terms of the consumed time and the number of the objective function evaluations. As shown in the following table.

11-bar Truss	Time (s)	OFEs	Weight (N)
iPSO	210.22	780	21708.91
TLBO	552.91	1100	21784.36
39-bar Truss	Time (s)	OFEs	Weight (N)
iPSO	640.55	16000	867.35
TLBO	3025.59	60000	1089.05

6) Interactive fuzzy search algorithm: A new self-adaptive hybrid optimization algorithm [41]

The proposed method combines the affirmative features of the Integrated Particle Swarm Optimizer (iPSO) (article 4 [39]) and the Teaching and Learning Based Optimizer (TLBO) with a nine-rule fuzzy decision mechanism. Named as Interactive Fuzzy Search Algorithm (IFSA). The proposed algorithm utilizes two main navigation models to search the domain: tracking, which is a feature of iPSO, and interacting, which is a feature of TLBO. The balance between these two navigation models is achieved through a factor called "tendency factor", as a tuning parameter. IFSA also utilizes the inertia weight also as a tuning parameter, which is a feature of iPSO to adjust the exploration level of the tracking model. Both tendency factor and inertia weight are adaptively determined thanks to an employed fuzzy module. In addition to that, the IFSA provides a self-adaptive synchronization between local and global search strategies by achieving a balance on the rate of exploration and exploitation behaviors. Both of these tuning parameters should be adjusted via implementing a series of sensitivity analyses if there is no such fuzzy module. So, to avoid such time-consuming sensitivity analyses, the fuzzy module is equipped to the ISA, introducing through this paper the proposed IFSA. Such fuzzy module does permanently monitor the search process and automatically designate the proper values for the tuning parameters through employing linguistic terms.

Two size optimization problems are considered to assess the search capability of the proposed IFSA: 72-bar truss structure and 160-bar pyramid space truss, both are discrete problems (standard cross sections).

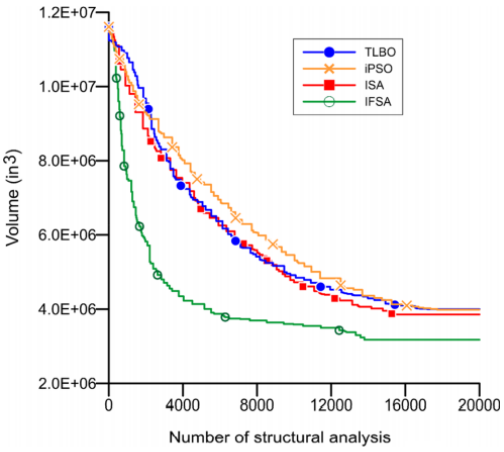
The results shown that the proposed IFSA provided fair optimum weights comparatively to solutions of other algorithms found in the literature. But in so much less number of objective function evaluations (OFEs).

7) Large-scale structural optimization using a fuzzy reinforced swarm intelligence algorithm [42]

This article displays an assessment of the search performance of the interactive fuzzy search algorithm IFSA (article 6 [41]) in the size optimization of large-scale structures, comparatively to different techniques (Stochastic: PSO, TLBO, iPSO, ISA) and (deterministic gradient-based: simultaneous analysis and design SAND).

Three benchmark structures are considered in the investigation of the IFSA's capability: 582-bar, 1262-bar and 4666-bar truss towers. The first-mentioned one is a discrete problem, where the design variables (cross sections area) are selected randomly from the standard list of AISC code. While the second and third problems are continuous, as the cross sections are selected randomly from the range [1,100] and [1,300] in^2, respectively.

Among the results of the different deterministic and stochastic techniques, IFSA provided the best solutions for all the benchmark structures, in terms of the weight and the computational cost (number of objective function evaluations).

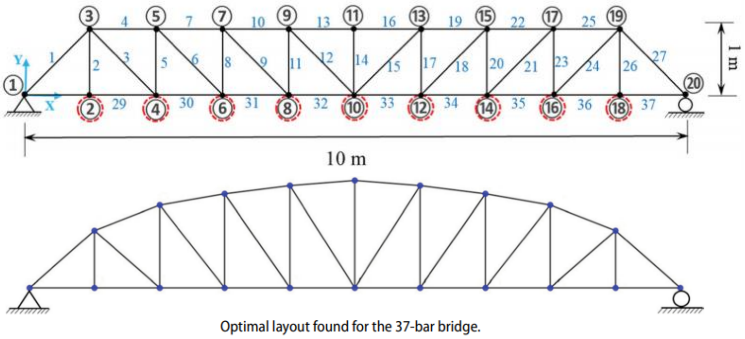


Optimization history for 1262-bar tower

8) Size and layout optimization of truss structures with dynamic constraints using the interactive fuzzy search algorithm [43]

This investigation article assesses the potential of the interactive fuzzy search algorithm IFSA (article 6 [41]), throughout a size and layout optimization of trusses while considering the natural frequency as a problem constraint, which are very sensitive to any configuration changes in the system. The Penalty Function method is used in this work for the constraint handling. Five benchmark structural problems are optimized in order to investigate the IFSA's performance: 37-bar truss bridge, 52-bar dome, 72-bar space truss, 120-bar dome and 200-bar truss structure. For the first-two-mentioned problems, size and topology optimization is considered. While only size optimization is performed for the other three problems.

The results shown that the IFSA provided fair optimum weights comparatively to solutions of other algorithms found in the literature. But in so-much less number of objective function evaluations (OFEs), thanks to the fuzzy logic which leads the algorithm to ignore the useless iterations that has no progress throughout the optimization process.

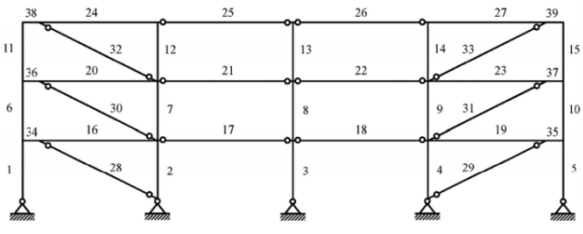


Optimal layout found for the 37-bar bridge.

9) Optimum performance-based design of eccentrically braced frames [44]

This article introduces a weight minimization of an eccentrically braced frame (EBF) while following the performance-based design (PBD) method. As an objective function, both structural weight and structural damage are considered to be minimized. Four metaheuristic optimization algorithms are employed in this investigation work: accelerated water evaporation optimization (AWEO) [45], particle swarm optimization PSO [46], colliding bodies optimization (CBO) [47] and enhanced colliding bodies optimization ECBO [48]. All are population-based algorithms that are swarm-intelligence based. Two benchmark structures are optimized in order to assess the above-mentioned four algorithms in the EBF weight optimization using the PBD method, these two structures are 2D steel EBFs: four-span three-story frame and five-span six-story frame. Both problems have a discrete domain for the design variables. The Penalty Function method is used for the constraint handling. The structural analysis is performed using the nonlinear static pushover analysis.

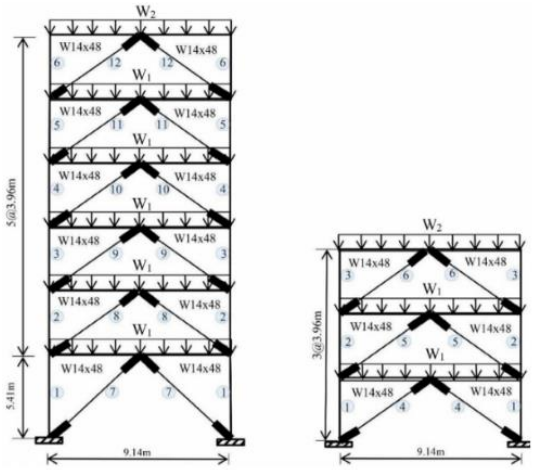
The results shown that among the four mentioned metaheuristics, the ECBO offered the best optimal, followed by the CBO's solutions, for both benchmark structures.



10) Optimum design of buckling-restrained braced frames [49]

This article is presented in order to introduce an approach of minimizing the weight and maximizing the dissipated energy (that results in greater ductility) of the buckling-restrained braced frames (BRBF) under seismic excitation, through performing each of SSA and ECBO, separately. The optimization is implemented using two different metaheuristic algorithms: Enhanced Colliding Body Optimization (ECBO) [48] and Salp Swarm Algorithm (SSA) [50]. Both algorithms are swarm-based. Two benchmark structures are optimized to investigate the proposed method: 3-stories and 6-stories BRBFs. Both have discrete and continuous spaces for the design variables. The Penalty Function method is used for the constraint handling.

The results shown that the ECBO outperformed the SSA for the optimization of the BRBF structures. Also, it is observed that the Buckling-Restrained Braces (BRBs) effectively reduce the weight of the structure by minimizing the base shear.



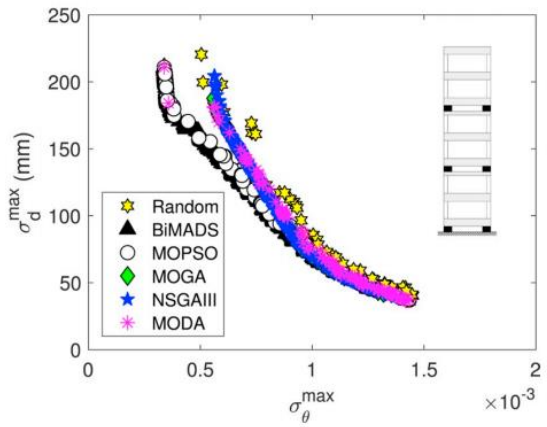
sample frames

11) A Hybrid Particle Swarm Optimization and Genetic Algorithm for Truss Structures with Discrete Variables [51]

In this article a new method is introduced where the GA algorithm is merged with PSO in one algorithm called Hybrid Particle Swarm Optimization and Genetic Algorithm (PSOGA), not only to enhance the global exploration by overriding the local optimal traps which is a shortcoming of the standard PSO, but also to help in achieving better optimal design. This proposed algorithm is investigated on a discrete size optimization problems of truss structures. Using some benchmarks truss problems found in the literature: 25-bar spatial truss, 10-bar truss, 52-bar truss, 72-bar spatial truss. The result from the proposed method PSOGA are compared to those of several popular metaheuristic algorithms in the literature. The results came to outperform the proposed algorithm in this article (PSOGA) on the rest algorithms in the comparison, where the obtained weight by the PSOGA is the minimum among the weights by the rest algorithms.

12) Multi-objective optimization of inter-story isolated buildings using metaheuristic and derivative-free algorithms [52]

This article assesses the performance of six multi-objective optimization (MOO) algorithms in selecting the optimum configuration of which specific stories should host the seismic isolation devices, and in finding the effective stiffness and damping properties of the isolation layers, in the Inter-Story Isolation (ISI) technique. The considered objective functions are the maximum standard deviations of: the inter-story drift ratio and the isolation drift. The above-mentioned six MOO algorithms are: four metaheuristic algorithms (Multi-objective genetic algorithm (MOGA), non-dominated sorting genetic Algorithm-III (NSGA-III), multi-objective particle swarm optimization (MOPSO), multi-objective dragonfly algorithm (MODA)), one derivative-free deterministic algorithm (bi-objective Mesh Adaptive Direct Search (BiMADS)) and a random search algorithm. In this investigation work, a six-story building is considered as a benchmark structure. Where a single, 2 or 3 isolation layers are considered as different configurations of the ISI. It is observed that by increasing the number of the isolation layers, the results from the different algorithms become less consistent. So that, the 3 layers configuration is followed in the comparison between the algorithms, where the isolation layers applied at the base, the 2nd story and the 4th story of the building. The comparison result referred to the superiority of the deterministic derivative-free BiMADS in such kind of MOO over the rest algorithms, followed by MODA and MOPSO.



Pareto fronts for the different MOO algorithms

C. Other Nature-Inspired Metaheuristics

1) Interactive autodidactic school: A new metaheuristic optimization algorithm for solving mathematical and structural design optimization problems [53]

This article introduces to the literature a new algorithm called Interactive autodidactic School (IAS), which based on similar concepts of the well-known Teaching-Learning Based Optimization (TLBO) [54] that holds two main stages of improving each student's capability: once through direct guide by the best student in the class (the candidate with the best fitness), and another through interacting with the other students, but with an additional third stage that helps in overriding the optimal traps, which is called "the challenge of the new student" (CNS). Instead of the "teacher phase" and "learner phase" in the TLBO, these two stages are called in this proposed algorithm "individual training session" (ITS) and "collective training session" (CTS), respectively. Hence, the proposed algorithm process follows three consecutive stages: ITS, CTS, then CNS. First of all, the best-fitness candidate is determined and positioned as the leading student. Then the ITS stage starts through randomly splits the population into sub-groups, each group holds two candidates. Each candidate goes into a learning-directed interaction session with the leading student, then through the CTS stage these two members meet and review the information they had received in their peer-to-peer sessions with the leading student during the ITS stage. Then they gather with the leading student in order to discuss any misunderstandings. Through such behaviors, it is obvious that not only the knowledge status of each candidate determines his capability, but also the candidate's social interaction does so, e.g. communication skills, team work skills, and cooperation. Which definitely affects the learning process efficiency of the group. Eventually, the CNS stages comes to play in order to emphasize the exploration capability of the algorithm. As when the best-found candidate so far (which may be a local optimal in fact) is highly affecting and pulling the rest candidates, that may lead the entire population to exploit just the area around this best-found candidate, while missing exploring other promising spots within the search space. So that, the CNS mechanism is equipped into the IAS algorithm in order for a new randomly-positioned student to encourage the population to revolt against the temporary leader from time to time, pulling the population to explore more spots within the search space. As if that new candidate shows higher capability (fitness function value) than the current-state leader, it takes over the lead onwards. It worthy here to mention that the IAS outperforms the other well-known metaheuristic optimization algorithms for being a parameter-free algorithm.

This proposed IAS algorithm is investigated in this article through twenty mathematical and seven structural benchmark optimization problems. Comparing the results with some of the most-known metaheuristic optimization algorithms. The comparison is presented with respect to the optimal solution (e.g., structural weight) and the computational cost (number of the objective function evaluations). The results of the twenty mathematical benchmarks shown that among seven used optimizers, the IAS was the only optimizer that succeeded to get the exact optimal for all the twenty functions.

The seven structural benchmark problems are: 25-bar spatial truss, 72-bar spatial truss, 200-bar planar truss, stepped cantilever beam, reinforced concrete beam, welded beam and cylindrical pressure vessel. The first-mentioned three problems are discrete (specific cross section areas) size

optimization, the fourth is continuous (cross section height and width design variables) size optimization, the fifth is discrete (cross section height, steel bars sectional area) and continuous (cross section width) size optimization, the sixth is continuous (weld's thickness and height, member width and height) size optimization, and the seventh problem is continuous (thicknesses of the shell and the head plate, inner diameter and length of the cylinder) size optimization. The results shown that for the first-mentioned three truss optimization, the proposed IAS outperformed six metaheuristics: GA, HS, HPSO, HPASCO, DE and DAJA in the obtained optimal weight, however the IAS was not the best in terms of the computational cost (number of objective function evaluations). For the stepped cantilever beam, the IAS got the same optimal weight of further five optimization algorithms: MMA, GCA(I), GCA(II), CS and SOS. For the reinforced concrete beam, the IAS obtained the best solution among other six optimization algorithms: SDRC, GHNALM, GHNEP, GA, FLCAHGA, CS. For the welded beam, the IAS got the same optimal weight of further eight optimization algorithms: GA4, CPSO, CAEP, HPSO, NMPSO, TLBO, MBA, CSA, however in mush less computational cost (number of objective function evaluations) than the other eight algorithms. And for the cylindrical pressure vessel, the IAS obtained the best optimal solution among other ten algorithms: GA3, GA4, CPSO, CAEP, HPSO, NMPSO, TLBO, MBA, CSA, GQPSO and CDE, in the least computation cost (number of objective function evaluations) as well.

Ch 3 State-of-the-Art MOAs in Structural Optimization

In this section, some of the most promising and trending MOAs in structural optimization are introduced in short but descriptive paragraphs, each is presented in the form of: inspiration, mathematical modelling and how-it-works. These algorithms are classified into three categories: Evolutionary Computation, Swarm Intelligence and other Nature-Inspired algorithms.

A. Evolutionary Computation Metaheuristics

1) Differential Evolution [55]

In 1995, Storn and Price [13] proposed a new floating-point evolutionary algorithm for global optimization and named it differential evolution (DE), by implementing a special kind operator which sought to create new offsprings from parent chromosomes. DE generates new vectors by adding the weighted difference vector between two population members to a third member. If the resulting vector corresponds to a better objective function value than a population member, the newly generated vector replaces this member. The comparison is performed between the newly generated vector and all the members of the population excluding the three ones used for its generation. Furthermore, the best parameter vector is evaluated in every generation in order to keep track of the progress achieved during the optimization process. Several variants of DE have been proposed so far, but the two most widely used are the following.

According to the variant implemented, a donor vector $v_{i,g+1}$ is generated first according to:

$$v_{i,g+1} = s_{r_1,g} + F \cdot (s_{r_2,g} - s_{r_3,g})$$

before the computation of the i[th] parameter vector $s_{i,g+1}$. This step is equivalent to the mutation operator step of genetic algorithms or evolution strategies. Integers r1, r2 and r3 are chosen randomly from the interval [1,NP] while i≠ r_1, r_2 and r_3. F is a real constant value, called mutation factor, which controls the amplification of the differential variation $(s_{r_2,g} - s_{r_3,g})$ and is defined in the range [0,2]. In the next step the crossover operator is applied by generating the trial vector $u_{i,g+1}$= $[u_{1,i,g+1}, u_{2,i,g+1},\ldots, u_{D,i,g+1}]$T which is defined from the elements of the vector $s_{i,g}$ and the elements of the donor vector $v_{i,g+1}$whose elements enter the trial vector with probability CR as follows:

$$u_{j,i,g+1} = \begin{cases} v_{j,i,g+1}\ if\ \mathrm{rand}_{j,i} \le CR \text{ or } j = I_{rand} \\ s_{j,i,g}\ if\ \mathrm{rand}_{j,i} > CR \text{ or } j \ne I_{rand} \end{cases}$$
$$i = 1,2,\ldots,NP \text{ and } j = 1,2,\ldots,n$$

where $rand_{j,i} \sim U[0,1], I_{rand}$ is a random integer from [1,2,…,n] that ensures that $v_{i,g+1} \ne s_{i,g}$. The last step of the generation procedure is the implementation of the selection operator where the vector $s_{i,g}$, is compared to the trial vector $u_{i,g+1}$:

$$s_{i,g+1} = \begin{cases} u_{i,g+1} \text{ if } f(\mathrm{u}_{i,g+1}) \le f(s_{i,g}) \\ s_{i,g} \text{ otherwise} \end{cases} \qquad i = 1,2,\ldots,NP$$

2) Multi-Trial vector-based Differential Evolution MTDE [56]

Differential Evolution algorithms' performance is highly affected by the employed search strategy and the parameter settings. As per the no-free-lunch theorem, a combination of search strategies can be an effective way to cover a variety of problems using multiple search strategies instead of only one search strategy. Hence, this algorithm (MTDE) is proposed, enriched by an efficient combination of three different search strategies, producing a new approach called Multi-Trial Vector MTV. These three different search strategies are Trial Vector Procedures (TVP): Representative based (R-TVP) which maintains the diversity, Local Random based (L-TVP) that ensures a proper balance between exploration and exploitation, and Global Best History based (G-TVP) that enhances the exploitation ability of the algorithm. In contrast with the previous DEs works which distribute the main population into smaller subpopulations with same sizes, the MTV employs a winner-based distribution policy, that distributes the subpopulation between the TVPs not in equal manner, but the better search strategy will be, the larger subpopulation it will handle. The MTV approach introduces adaptive movement steps thanks to using a life-time archive that preserves and shares the information of the restored promising solutions. This life-time archive also maintains the population diversities in the MTV approach. In another words, the MTV approach consists of four phases: winner-based distributing, multi-trial vector producing, evaluating and population updating, and life-time archiving.

B. Swarm Intelligence Metaheuristics

1) Grey Wolf Optimizer [57]

GWO is a swarm-based Metaheuristic which is inspired by the grey wolves (Canis lupus). It mimics the leadership hierarchy and hunting mechanism of grey wolves in nature. To model this social behavior mathematically, a random population of grey wolves (candidate solutions) is firstly generated, then over course of iterations, the population candidates are defined within 4 classes: alpha, beta, delta and omega -the 4 types of grey wolves-. The fittest solution is alpha, the second and third best solutions are beta and delta, respectively. The rest of the candidates are omega. The hunting (optimization) is guided by alpha, beta, and delta, mega candidates follow them. The three steps of getting the prey (optimization) are encircling prey, hunting and attacking prey.

To mathematically model these three behaviors, the following equations are employed:

- The distance of the ith wolf (search agent) to the prey:
  $D = \left|C \cdot X_p(t) - X(t)\right|$

  where t indicates the current iteration, C is a coefficient vector ($C = 2 \cdot r_2$). X is the current position vector of the ith search agent. X_p is the position vector of the prey.

- The next position vector:
  $X(t+1) = X_p(t) - A \cdot D$

  where A is a coefficient vector ($A = 2a \cdot r_1 - a$, r_1 & r_2 are two random vectors in [0,1], a is linearly decreased from 2 to 0 to switch the value of A outside & inside the range of [-1,1]), when |A| > 1 (exploration), while if |A| < 1 (exploitation). (·) is an element-by-element multiplication.

- Since the alpha, beta, and delta are the agents who lead the search, omega is a follower. This hunting is mathematically described through the following equations:
  $D_\alpha = |C_1 \cdot X_\alpha - X|, D_\beta = \left|C_2 \cdot X_\beta - X\right|, D_\delta = |C_3 \cdot X_\delta - X|$
  $X_1 = X_\alpha - A_1 \cdot D_\alpha, X_2 = X_\beta - A_2 \cdot D_\beta, X_3 = X_\delta - A_3 \cdot D_\delta$
  $X(t+1) = (X_1 + X_2 + X_3)/3$

The GWO algorithm starts with a set of random solutions. Then, iteratively, the search agents update their positions with respect to the prey, the prey may be either a randomly chosen solution or the best obtained solution so far, in case of exploration or exploitation, respectively. The switch between the exploration and exploitation components is controlled by linear reduction of the parameter "a" from 2 to 0. As, when 0 < a < 1, the value of |A| < 1 which activates the exploitation mechanism, and the opposite occurs when 1 < a <2. The WOA has only two adaptive parameters: "a" and "C", "a" guarantees the exploration-exploitation balance, while "C" emphasizes the random behavior during the exploration. Although, this GWO is considered as an efficient metaheuristic for the optimization problems in some fields such as engineering, machine learning, medical, and bioinformatics, it suffers from insufficient diversity of the population, inefficient -exploration-exploitation balance, and premature convergence.

2) Improved Grey Wolf Optimizer [58]

I-GWO holds an important improvement of the GWO [57], That enhances the exploration-exploitation balance and also maintains the population diversity. That is done through developing a new search strategy called "dimension learning-based hunting" (DLH), through which, neighboring information can be shared between the candidates. DLH search strategy is inspired by the individual hunting behavior of wolves in nature, and it increases the domain of global search by multi neighbors learning. Then, in each iteration, the I-GWO selects the candidate either from the GWO or the DLH search strategies based on the quality of their new positions.

The mathematical description of this selection either from GWO or DLH candidates, is as follow:

$$X_{i-DLH,d}(t+1) = X_{i,d}(t) + rand \times (X_{n,d}(t) - X_{r,d}(t))$$
$$X_i(t+1) = X_{i-GWO}(t+1), if\ f(X_{i-GWO}) < f(X_{i-DLH}), or$$
$$X_i(t+1) = X_{i-DLH}(t+1), if\ f(X_{i-GWO}) > f(X_{i-DLH})$$

The I-GWO is a single-objective algorithm for optimization problems with continuous search space.

3) Whale Optimization Algorithm [59]

WOA is a swarm-based metaheuristic optimization approach. It is inspired by the spiral bubble-net hunting strategy of the humpback whales. This hunting (optimization) technique consists of three phases; Search for Prey (exploration phase), Encircling Prey, and the Bubble-Net Attacking method (exploitation phase). The Bubble-Net attacking is when the whales swim around the prey in two simultaneous movements: in a shrinking circle as well to in a spiral-shaped path, towards the sea surface. The combination between these two movements is done with a probability of 50% for each.

The mathematical modelling of these phases is as follow:

- The distance of the i[th] whale (search agent) to the prey:
  $D = |C \cdot X^*(t) - X(t)|$
  where t indicates the current iteration, C is a coefficient vector ($C = 2 \cdot \mathrm{r}$, r is a random vector in [0,1]). X is the current position vector of the i[th] search agent. X^* is the position vector of the prey.

- The next position vector:
  $[X(t+1) = X^*(t) - A \cdot D,\ in\ case\ of\ p < 0.5]$, or $[X(t+1) = D' \cdot e^{bl} \cdot \cos(2\pi l) + X(t),\ in\ case\ of\ p \geq 0.5]$, for the simultaneous encircling and spiral upward movements, respectively. where A is a coefficient vector ($A = 2a \cdot \mathrm{r} - a$, a is linearly decreased from 2 to 0). $(\cdot)$ is an element-by-element multiplication. D' is the same as D ($D' = |X^*(t) - X(t)|$), which is the distance of the i[th] whale to the prey, but without considering the random-behavior component "C" (D' is used in computing the spiral movement -exploitation-). b is a constant for defining the shape of the logarithmic spiral, l is a random number in [−1,1].

The WOA algorithm starts with a set of random solutions. Then, iteratively, the search agents update their positions with respect to the prey, the prey may be either a randomly chosen solution or the best obtained solution so far, in case of exploration or exploitation, respectively. The switch between the exploration and exploitation components is controlled by linear reduction of the parameter "a"

from 2 to 0. As, when 0 < a < 1, the value of |A| < 1 which activates the exploitation mechanism, and the opposite occurs when 1 < a <2. The WOA has only two adaptive parameters: “a” and “C”, “a” guarantees the exploration-exploitation balance, while “C” emphasizes the random behavior during the exploration.

4) Dragonfly Algorithm [60]

This is a swarm-intelligence metaheuristic technique. Which mimics the surviving swarm behavior of the dragonflies. The algorithm is equipped by five parameters to control the five behaviors of any swarm movement; separation, alignment, cohesion, attraction towards food, and distraction outwards enemies. The dragonflies follow this scheme either in the static swarming (hunting), or in the dynamic swarming (migration). The former 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. While the dynamic swarming is simulated during the exploitation phase, where the dragonflies fly in bigger swarms and along one direction.
To mathematically model this scheme, the following equations are used:

- The step vector: $\Delta X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta X_t$
  where S_i, A_i, C_i, F_i, E_i are the five parameters of the swarm behavior:
  - Separation: $S_i = -\sum_{j=1}^{N} X - X_j$ (maintains avoidance of individuals collision in a neighborhood)
  - Alignment: $A_i = \frac{\sum_{1=j}^{N} V_j}{N}$ (indicates the velocity matching between the neighborhood individuals)
  - Cohesion: $C_i = \frac{\sum_{1=j}^{N} X_j}{N} - X$ (refers to the individuals’ tendency to the neighborhood center of mass)
  - Attraction to food: $F_i = X^+ - X$
  - Distraction from enemies: $E_i = X^- + X$

  s, a, c, f and e, are weighting factors, each for each corresponding parameter. w is an inertia weight.

- The position vector: $X_{t+1} = X_t + \Delta X_{t+1}$

- When there is no neighbors around, a random walk (Lèvy flight) equation is followed: $X_{t+1} = X_t + \text{Lèvy(d)} * X_t$
  where $\text{Lèvy(x)} = 0.01 * \frac{r_1 \times \sigma}{\frac{1}{|r_2|^{\beta}}}$ (r1, r2 are random numbers [0:1], β is a constant, σ is a f(β))

In the DA, to ensure the switch and balance between exploration and exploitation modes, there are two approaches: Either by tuning the neighborhood’s radius around each artificial dragonfly. As a reasonably large radius refers to low swarm’s alignment but high swarm’s cohesion (exploitation), while a small radius refers to high alignment but low cohesion (exploration). Or by tuning the swarming factors: s, a, c, f and e.

5) Grasshopper Optimization Algorithm [61]

This algorithm mathematically modelled and mimicked the swarming behavior of grasshoppers in nature for solving optimization problems. Proposed to solve single-objective problems with continuous variables. The grasshoppers have swarming behavior in both nymph and adulthood phases of their life cycle. The nymph grasshopper's behavior is slow and small movements (exploitation), while the adult's behavior is long and abrupt steps (exploration). The grasshopper movement is based mainly on the social interaction with its neighbors. Which is described in terms of attraction, repulsion, or comfort zone state. In order to lead the entire swarm to converge at one point (global optimal), two mathematical terms are added to let the best-found solution at each iteration affect the swarm direction, and also to shrink the 3 zones the controls each grasshopper movement; attraction, repulsion or comfort zone. As without this shrinking term, the grasshopper may be stuck at the comfort zone so early with no more movements (trapped in local optima).
To model such behavior mathematically, these equations are equipped into the algorithm:

- The position of i_{th} grasshopper: $X_i^d = c\left(\sum_{j=1}^{N} c\frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j - x_i}{d_{ij}}\right) + \hat{T}_d$
  where c is a shrinking coefficient (c≤1), ub_d and lb_d are the upper and lower bounds, $\hat{T}_d$ is the target (best found solution), and the expression " $s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j^d - x_i^d}{d_{ij}}$ " defines the social interaction of the grasshopper with the neighbors, where s represents the social force, either attraction (exploitation) or repulsion (exploration), and calculated as follow: $s(r) = f\mathrm{e}^{\frac{-r}{l}} - \mathrm{e}^{-r}$. where f is the intensity of attraction and l is an attractive length scale. by f and l, the size of the attraction, repulsion and comfort zones could be set.

- The shrinking coefficient is calculated as follow: $c = c_{Max} - l\frac{c_{max} - c_{Min}}{L}$, L is the max number of iterations, l is the number of the current iteration, and c_{max} & c_{min} are two bounds lower than 1.

The gradual convergence of grasshoppers towards the target over the course of iteration, is actually due to decreasing the factor c, and the target effect of pulling the swarm.
The next position of a grasshopper is defined based on the factors; the current position, the position of the target, and the position of all other grasshoppers. In contrast to the well-known PSO, in which the swarm particles positions don't play any role in defining the next movement of a particle. For a balance between exploration and exploitation, the parameter c is required to be decreased proportional to the number of the iteration.

6) Salp Swarm Algorithm [62]

This simple and easy to implement metaheuristic optimization algorithm mimics the swarming behavior of the ocean salps travelling in form of a salp chain. This salp chain consists of a leader and followers. The leader goes towards an artificial source of food, and the followers just enjoy the ride behind the leader. In the optimization process, a set of salps (solutions) are initialized with random positions and stored in a two-dimensional matrix, then these solutions are evaluated through a fitness function, assigning the best-found solution as a target to be chased by the salps afterwards iteratively. The algorithm is equipped by two movement equations for both leader and followers, separately. The leader walk is a random movement actually, 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. SSA has only one controlling parameter, which is updated adaptively as the number of the iteration goes higher, in order to maintain the balance between the exploration and exploitation phases.

7) Particle Swarm Optimization [55]

In particle swarm optimization, multiple candidate solutions coexist and collaborate simultaneously. Each solution is called "a particle" having a position and a velocity in the multidimensional design space while a population of particles is called a swarm. A particle "flies" in the problem search space looking for the optimal position. As "time" passes through its quest, a particle adjusts its velocity and position according to its own "experience" as well as the experience of other neighboring particles. A particle's experience is built by tracking and memorizing the best position encountered. A PSO system combines local search (through self-experience) with global search (through neighboring experience), attempting to balance exploration and exploitation. Each particle maintains its two basic characteristics, velocity and position, in the multi-dimensional search space that are updated as follows:

$$\boldsymbol{v}^j(t+1) = w\boldsymbol{v}^j(t) + c_1 r_1 \circ \left(s^{\mathrm{Pb,j}} - s^j(t)\right) + c_2 r_2 \circ \left(s^{\mathrm{Gb}} - s^j(t)\right)$$

$$s^j(t+1) = s^j(t) + \boldsymbol{v}^j(t+1)$$

where $\boldsymbol{v}^j(t)$ denotes the velocity vector of j$^{\mathrm{th}}$ particle at time t, $s^j(t)$ represents the position vector of j$^{\mathrm{th}}$ particle at time t, vector $s^{\mathrm{Pb,j}}$ is the personal best ever position of the j$^{\mathrm{th}}$ particle, and vector s^{Gb} is the global best location found by the entire swarm. The acceleration coefficients c_1 and c_2 indicate the degree of confidence in the best solution found by the individual particle (c_1 - cognitive parameter) and by the whole swarm (c_2 - social parameter), respectively, while r_1 and r_2 are two random vectors uniformly distributed in the interval [0,1].

8) Krill Herd Algorithm [63]

KH is a biologically-inspired metaheuristic algorithm, that handles optimization problems in a stochastic way. The mechanism of the algorithm is inspired by the krill herding, 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 function in the algorithm is attained simulating the objective of the krill herd in nature, which is the minimum distances of both the center of the herd density and from the food location as well. In seek of higher efficiency, two genetic reproduction mechanism of the well-known GA are equipped into the KH algorithm: crossover and mutation.

For modelling the motion of the individuals mathematically, this motion formula is defined into the algorithm: $\frac{\mathrm{d}X_i}{\mathrm{d}t} = N_i + F_i + D_i$

where N_i is the motion induced by the swarm movement, F_i is the motion in seek of food, and D_i is the random spread motion.

- $N_i^{new} = N^{max}\alpha_i + w_n N_i^{old}$
  where N^{max} is the max induced speed, which is experimentally measured and taken as 0.01 m/s. α_i is the effect of the swarm motion, it combines the effect of both neighbors and the best krill. w_n is an inertia weight, in the range [0,1]. N_i^{old} is the previous motion of that individual.
- $F_i = V_f \beta_i + w_f F_i^{old}$
  where V_f is the speed towards the food, β_i is the attraction effect on the i_{th} individual, it combines the effect of the best fitness of that i_{th} individual so far, besides the effect of the food.
- $D_i = D^{max}\,\delta$
  where D^{max} is the maximum speed of the random motion. δ is a random directional vector.

The interesting fact about KH algorithm is that it gets the values it its coefficients from real world empirical studies, then only one parameter is to be tuned. However, the coefficients could be determined through an outsourcing metaheuristic algorithm instead of these mentioned real-world empirical studies.

9) Pity Beetle Algorithm [64]

PBA is a nature-inspired metaheuristic optimization algorithm based on the swarm intelligence. PBA is a single-objective optimization algorithm for the unconstrained problems, inspired by the searching for food behavior of the six-toothed bark beetle (*pityogenes chalcographus* beetle). This beetle feeds on the bark of the trees. The PBA simulates the searching for food behavior of this bark beetle, in a way consists of three main stages: *initialization* of a population consists of males and females, *regeneration* of new populations, and *location update* stage. In the first stage, an initial population consists 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 the premature convergences. To ensure the diversification, the initial population in PBA is generated by means of a random sampling technique (RST). In the second stage, after the initial populations are formed, each single particle will look for better position in the search space to create his own population. This is done in PBA through five types of hypervolume selection pattern; 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 kept in the memory for the memory consideration search volume.

C. Other Nature-Inspired Metaheuristics

1) Ant Lion Optimizer [65]

ALO is a nature-inspired metaheuristic that mimics the smart hunting mechanism of antlions in nature, in which the antlion digs a cone-shaped pit in sand by moving along a circular path and throwing out sands, then it hides underneath the bottom of the cone waiting an ant to fall down into the cone to catch it. The ALO algorithm mimics the interaction between antlions and ants during the hunting process, that is done in three phases (A, B, and C).

At (A), the ALO algorithm builds four matrices: two position matrices of the ants and antlions separately, and two fitness function matrices also one for ants and another for antlions.

At (B), the position of the ants and antlions are updated through implementing five operators: random walk of ants (which maintains the exploration capability), building traps by antlions (a roulette wheel operator is utilized to select antlions based on the fitness function of each, that also maintains the exploration capability), entrapment of ants in traps (this occurs through an adaptive shrinking mechanism of the radius of an ant's next random walks, which guarantees the exploitation), catching preys (if an ant is fitter than its corresponding antlion, then the antlion updates its position to the latest position of this hunted ant, that leads to investigation of the most promising areas in the search space, which maintains also the exploration), and re-building traps for another prey.

At (C), the algorithm returns the elite antlion when the end criterion is satisfied, the best antlion obtained so far in each iteration is saved and considered as an elite.

For the mathematical modelling of phase (B) operators, the following equations are employed:

- For the ants' random walk:
  $X(\mathrm{t}) = [0, \mathrm{cumsum}(2\mathrm{r}(t_1) - 1);\ \mathrm{cumsum}(2\mathrm{r}(t_2) - 1), \ldots . \mathrm{cumsum}(2\mathrm{r}(t_n) - 1)]$
  where n is the maximum number of iterations, r(t) is a stochastic function= (1 if rand>0.5, else if, equals zero. "rand" is a random number generated with uniform distribution in the interval of [0,1]).
  In order to keep the random walks within the search space's boundaries, the following min–max normalization function is employed:
  $$X_i^t = \left[\left(X_i^t - a_i \right) * \frac{d_i^t - c_i^t}{b_i - a_i} \right] + c_i^t$$
  where a_i and b_i are the minimum and maximum of the random walk of ith variable, respectively. c_i^t and d_i^t are the minimum and maximum of all variables of the ith ant at tth iteration, respectively. Since the ants' random walk should be affected by the antlions' trap, the c_i^t and d_i^t are computed as follow:
  $c_i^t = Antlion_j^t + c^t, d_j^t = Antlion_i^t + d^t,$
  where $c^t$and $d^t$is the minimum and maximum of all the variables among all the ants at tth iteration, respectively.

- Every random walk of each ant is affected simultaneously by two antlions: one selected by the roulette wheel and another which is saved in the memory as the elite antlion. In this regard, the following equation is employed:
  $Ant_i^t = (R_A^t + R_E^t)/2$
  where Ant_i^t indicates the position of i[th] ant at t[th] iteration. R^t is the ant's random walk, equals X_i^t, R_A^t is the random walk around the antlion selected by the roulette wheel at t[th] iteration, R_E^t is the random walk around the elite antlion at t[th] iteration.

- For the entrapment of ants: to mimic the sliding of the ants down towards the antlion, by shrinking the radius of the ants' random walk, these equations are used:
  $c^t = c^t/I$, $d^t = d^t/I$
  where I is a ratio: $I = 10^w * t/T$, t is the current iteration number, T is the total number of iterations, w is a constant ranges from 2 to 6 depending on the current iteration number (the higher t, the higher w). Basically, w can adjust the accuracy level of the exploitation.

- By catching a fitter prey, the antlion reposition itself to the latest position of that prey through the following equation:
  $Antlion_j^t = Ant_i^t$, if $f(Ant_i^t) > f(Antlion_j^t)$

The exploitation phase of the ALO is guaranteed thanks to the adaptive boundary shrinking mechanism and the elitism. While the exploration is guaranteed thanks to the random walk and roulette wheel selection operators.

2) Moth-Flame Optimization [66]

This another recent population-based metaheuristic, which is inspired by the Transverse Orientation navigation method of moths. Accurately, 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 method, a moth flies by fixing a certain angle with respect to the light source, forming a spiral fly path, which ensures a convergence. To maintain investigating the most promising areas of the search space, moths (search agents) takes flames (best-found solutions) as the source of light and fly spirally around them. And to maintain a well exploration and local optimum avoidance, each moth is allowed to update its position using only one specific flame. This behavior is equipped is a three-tuple algorithm; MFO = (I,P,T). In MFO, two matrices are formulated to define moths and flames; M and F. Then consequently, two fitness function matrices are formulated; OM and OF. I is the initial population generator, I:ø → {M,OM}. T is the termination criterion, T: M→ {true, false}. P is the position updating function, P: $M_i \rightarrow M_{i+1}$, by which a moth updates its position around a flame through a spiral motion, M_{i+1}= S(M_i, F_j). S is the spiral function, which is chosen as a logarithmic spiral function in this investigation work. $S(M_i, F_j) = D_i \cdot e^{bt} \cdot cos(2\pi t) + F_j$, D_i indicates the distance between M_i and F_j, b is a constant for defining the shape of the logarithmic spiral, and t is a random number in [r, 1], 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 t, the closer 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, till it ends that all the moths at the final step update their positions with respect to only one flame.

3) Multi-Verse Optimizer [67]

This population-based metaheuristic is inspired by the multi-verse theory in physics. In designing this algorithm, every universe (solution) owns objects (variables). Three concepts of multi-universe are simulated; white, black and wormholes, by which how the multiple universes interact. The white holes in nature are where the so called big-bang occurs. The blackholes are where things are attracted inside. The wormholes are considered as tunnels through which the objects of a universe are able to travel between any corners of the same or different universes. Objects (variables) travel from white holes (solution with high fitness function value) to black holes (solutions with low fitness function value), in seek of better fitness values, the white holes are selected using a roulette wheel mechanism, this exchange of variables through white and black holes maintains the exploration of the search space. While the 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 withing its space randomly. The optimization process starts with creating a set of random solutions. At each iteration, variables in the solution agents with high fitness values tend to move to others with low fitness values via white and black holes (exploration). Meanwhile, all the universes variables are moved towards the best solution randomly regardless its solution fitness value, which maintains the diversity.

4) Sine Cosine Algorithm [68]

The SCA is a metaheuristic optimization algorithm that initialize random solution agents, then push them to fluctuate either towards or outwards the best-found solution, in a sin-cosine behavior. This positioning of the agents iteratively is guided by random-walk function:
$X_i^{t+1} = x_i^t + r_1 \times sin/cos(r_2) \times \left| r_3 p_i^t - x_i^t \right|$, sin: if $r_4 < 0.5$, cos: if $r_4 \geq 0.5$
where r_1, r_2, r_3, r_4 are random numbers, r_1 indicates the next position's region, which could be 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. r_4 switches between the sine and cosine.
The cyclic pattern of sine and cosine function guarantee a well exploitation of the space. On the other side, the exploration is achieved by changing the range r_1 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, an equation for adaptively decreasing the range r_1 is employed: $r_1 = a - t\frac{a}{T}$, where t and T are the number of the current iteration and the maximum number of iterations, respectively, a is a constant.

5) Harmony Search [55]

The harmony search algorithm was originally inspired by the improvisation process of Jazz musicians. According to the analogy between improvisation and optimization, each musician (saxophonist, bassist, guitarist etc.) corresponds to each decision variable; each musical instrument's pitch range corresponds to a decision variable's value range. Musical harmony at certain times corresponds to a solution vector at certain iterations, and audience's aesthetics corresponds to the objective function. According to the above algorithmic concept, the HS algorithm consists of the following five steps: parameter initialization; harmony memory initialization; new harmony improvisation; harmony memory update; and termination criterion check. Different from those population-based approaches, it only utilizes a single search memory to evolve. Therefore, the HS method has the distinguishing feature of computational simplicity.

In the first step, the optimization problem is specified where n is the number of decision variables (equivalent to the number of music instruments), while $s_i^L \leq s_i \leq s_i^U, i = 1,2,\ldots,n$ determines the range of the ith decision variable's value. The HS algorithm parameters are also specified in this step: HMS is the harmony memory size that corresponds to the number of simultaneous solution vectors stored in harmony memory, HMCR defines the harmony memory considering rate, while PAR is the pitch adjusting rate. In the second step, the harmony memory (HM) is initialized with HMS randomly generated solution vectors defining the musician's harmony memory matrix:

$$HM = \begin{bmatrix} s_1^1 & s_2^1 & s_3^1 \ldots & s_n^1 \\ s_1^2 & s_2^2 & s_3^2 \ldots & s_n^2 \\ \ldots & \ldots & \ldots & \ldots \\ s_1^{HMS} & s_2^{HMS} & s_3^{HMS} \ldots & s_n^{HMS} \end{bmatrix}$$

In the third step, a new harmony vector is improvised following three rules: random selection, memory consideration and pitch adjustment. According to the random selection, the value of the decision variable s_i is chosen randomly from the pitches stored in $HM = [s_i^1, s_i^2, \cdots, s_i^{HMS}]$ with probability of HMCR (0≤ HMCR≤1) or according to the memory consideration it is randomly chosen with a probability of (1-HMCR) within its value range, as a musician plays any pitch within the instrument's pitch range:

$$s_i = \begin{cases} s_i \in [s_i^1, s_i^2, \cdots, s_i^{HMS}] \text{ with probability HMCR} \\ s_i^L \leq s_i \leq s_i^U \text{with probability (1-HMCR)} \end{cases}$$

After the value s_i is randomly picked according to the above memory consideration process, it can be further adjusted into neighboring values by adding certain amount to the value, with probability of HMCR × PAR (0≤ PAR ≤1) while the original pitch obtained in HM consideration is just kept with a probability of HMCR × (1-PAR):

$$s_i = \begin{cases} s_i(k+m) \text{ with probability } HMCR \times PAR \\ s_i \text{ with probability } HMCR \times (1 - PAR) \end{cases}$$

If the new generated harmony vector is better than the worst harmony vector of the HM, with reference to the objective function value, the worst harmony is replaced by the new harmony vector.

6) Imperialist Competitive Algorithm [69]

ICA is an evolutionary sociopolitical inspired metaheuristic optimization algorithm. Introduced to the literature back at 2007, to deal with continuous optimization problems. Then, many versions have been developed later to handle discrete problems as well. The idea behind this algorithm is considering all the possible solutions as countries, the best set of them are considered as imperialists, and the rest are colonies. Each imperialist is supposed to possess portion of the colonies, forming an empire. The evolutionary improvement of the candidate solutions during the optimization process is implemented through six main operators: *initialization, assimilation, revolution, title exchange, empires survival/collapse and convergence*.

Initial population is generated at first (N_{pop}). Then, some promising candidates based on their objective function values (OFs) are selected as imperialists (N_{imp}), the rest candidates are colonies ($N_{col} = N_{pop} - N_{imp}$). Each imperialist takes over a portion of the colonies according to its power, forming an empire. An imperialist's power is defined based on its normalized OF, as follow:

Imperialist's normalized OF: $C_n = max\{c_i\} - c_n$

Imperialist's power: $P_n = \left| \frac{C_n}{\sum_{N_{imp}}^{i=1} C_i} \right|$

Therefore, the number of colonies that taken over by each imperialist: $NC_n = P_n \times N_{col}$.

Assimilation process is then performed, in which the power of each colony approaches gradually that of its respective imperialist. For real-life example of the assimilation process, both the British and French Empires attempted assimilation by constructing New England and New France within their respective colonies, imposing their own characteristics (social, cultural, economic, political, ..etc) within their colonies. These characteristics are considered as the problem variables. In the assimilation process within the ICA, the colonies' move in random distances (x), along with (d) direction towards their respective imperialist, but with random variation in the direction, maintaining both exploration and exploitation phases in cases of high and low variations, respectively. At some point consequently, a colony's power is upgraded to reach out that of its respective imperialist, or to become even higher.

Revolution operator is another aspect of ICA that maintains better exploration, in which some colonies resist to be realized by the imperialists. So, they suddenly jump outside the empire. Which emphasizes exploring new promising areas within the search space but don't host any imperialist. That consequently may lead to a promotion of a colony's power even higher than an existing imperialist's.

Title Exchange operator is performed to promote a colony to be an imperialist in the following iterations, and vice versa. The promoted colony is a colony that got higher power than an existing imperialist's, either through the assimilation or the revolution operators.

Empires survival/collapse occurs after performing assimilation, revolution and title exchanging processes, when the empires get either weaker or more powerful. The weak empires collapse leaving behind its colonies that will be taken over by survived ones.

Convergence in ICA occurs when just one empire is survived (called grand empire) and all the rest are collapsed. Or when any other set criterion is met, like specific number of iteration or specific processing time.

7) Teaching–Learning-Based Optimization [54]

This population-based metaheuristic optimization algorithm is inspired by the human teaching & learning behavior. The idea behind it is considering the possible solutions as a set of students in a classroom, the teacher is a candidate as well but holds the most knowledge (best fitness). Throughout two main operators: *Teacher Phase* and *Learner Phase*, the students (solutions) are improved in terms of their grades (fitness function value). The taught subjects in the class are represented into the algorithm as the design variables. The algorithm is also following the fact of the higher capability of the teacher is, the more promising the students will be. The indication of the students' level of knowledge in a specific subject, is the mean value of their grades in this subject.

The algorithm starts by initializing a set of random solutions:
$P = \begin{bmatrix} X(1,1) & \cdots & X(1,D) \\ \vdots & \ddots & \vdots \\ X(P,1) & \cdots & X(P,D) \end{bmatrix}$, in which each candidate is represented by the design variables from 1 to D.

The best candidate (best fitness) is set as a teacher, then for all the rest candidates, the mean grade for each subject is calculated, producing the mean solution: $M_{,D} = [m_1, m_2, \cdots m_D]$.

Then the *Teacher Phase* 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 ($X_{teacher,D}$). This is done in the algorithm through considering the teacher solution as a new artificial mean: $M\_new_{,D} = X_{teacher,D} = X_{f(x)=min}$. Then the difference between both the original and the artificial means is calculated: $Difference_{,D} = r\,(M\_new_{,D} - T_F \cdot M_{,D})$, where r is a random number in the range [0,1], T_F is randomly set 1 or 2 with equal probability. Then after, each candidate (solution) moves by this difference, producing new generation: $X_{new,D} = X_{old,D} + Difference_{,D}$, $X_{new,D}$ is accepted if it has better fitness.

The *Learner Phase* also provides an improvement for the solutions through the interaction between the candidates themselves, based on the fact of "A learner learns something new if the partner has more knowledge (better fitness)". So, if $f(X_i) < f(X_j)$, $X_{new,i} = X_{old,i} + r_i\,(X_i - X_j)$, else, $X_{new,i} = X_{old,i} + r_i\,(X_j - X_i)$, $X_{new,i}$ is accepted if it has better fitness.

8) Interior Search Algorithm [70]

This ISA is inspired from the architectural process of the interior design and decoration. The interior design and decoration process, there are two main concepts in order to find the best view and decoration; Composition and Mirror concepts. The Composition concept refers to the process of replacing the items position till it gives the best view, while the Mirror concept refers to placing mirrors near to the most beautiful items in order to emphasize their beauty. These concepts are followed in the ISA, where the candidates (except the fittest candidate) are randomly grouped into two parts: Composition group in which the candidates change their position only when it gives fitter values, and Mirror group in which some mirrors are placed near to the fittest candidates giving them higher weights among the swarm. For that, α parameter is defined and tuned, where the candidates are grouped based on the following role: if $r_1 < \alpha$, the corresponding candidate goes to Mirror group, else, it goes to Composition group. where r_1 is a random value between 0 and 1. In the Composition group, the position of each candidate follows this formula: $x_i^j = LB^j + \left(UB^j - LB^j\right) r_2$. While in the Mirror group, a mirror is placed randomly somewhere between each candidate and the fittest candidate so far. The location of these mirrors follows this formula: $x_{m,i}^j = r_3\, x_i^{j-1} + (1 - r_3\,) x_{gb}^j$ where r_2, r_3 are random value between 0 and 1.

The ISA has only one tuned parameter (α).

9) Slime Mould Algorithm [71]

In nature, the slime mold of the Physarum polycephalum forages in a way that leads to the food through optimal paths, this occurs using a bio-oscillator that produces a propagating wave, where the cytoplasmic flow indicates the thickness of the veins in a slime mold, The faster the flow, the thicker the vein. Hence, the optimal path is determined. SMA is stochastic optimizer that uses adaptive weights as well in order to simulate this process of producing the positive and negative indications out of the propagation wave that is resulted from the bio-oscillator. The slim mold grows in a venous shape, that allows it to use multi food sources in the same time. The main mechanisms of such behavior of slime mold foraging that are mathematically modelled in the SMA: approaching food, wrap food and oscillation. The balance between the exploration and exploitation phases in SMA is maintained mainly through the adaptive weight, vibration parameter, utilization of the fitness values and the decision parameter of the location updating.

10) The Arithmetic Optimization Algorithm [72]

AOA is a very recent however promising population-based metaheuristic optimization algorithm. Inspired by the main arithmetic operators in mathematics: addition (A), subtraction (S), multiplication (M) and division (D). Like any other metaheuristic algorithm, the optimization process consists of two main phases: exploration and exploitation. The exploration phase in AOA is implemented employing the D and M operators, due to their high dispersion, which indicates exploration in the optimization dictionary. On the other side, A and S are employed when the exploitation phase is activated during the optimization process, due to their low dispersion. In order to switch between exploration and exploitation phases during the optimization process: if $r_1 > MOA$, activate the exploration phase, otherwise, activate the exploitation.

where MOA (Math Optimizer Accelerated function) linearly increases from 0.2 to 0.9:

$$MOA(C_{-Iter}) = Min + C_{-Iter}\ \frac{Max - Min}{M_{-Iter}}$$

For the exploration phase, the algorithm is equipped by this random walk formula:

$$x_{i,j}(C_{-Iter}+1) = \begin{cases} best(x_j) \div (MOP+\epsilon) \times \left((UB_j - LB_j) \times \mu + LB_j \right), & r_2 < 0.5 \\ best(x_j) \times (MOP) \times \left((UB_j - LB_j) \times \mu + LB_j \right), & otherwise \end{cases}$$

While for exploitation:

$$x_{i,j}(C_{-Iter}+1) = \begin{cases} best(x_j) - (MOP) \times \left((UB_j - LB_j) \times \mu + LB_j \right), & r_3 < 0.5 \\ best(x_j) + (MOP) \times \left((UB_j - LB_j) \times \mu + LB_j \right), & otherwise \end{cases}$$

where $MOP(C_{-Iter}) = 1 - \frac{{C_{-Iter}}^{1/\alpha}}{{M_{-Iter}}^{1/\alpha}}$

where C_{-Iter} abbreviates the current iteration, M_{-Iter} abbreviates the maximum number of iterations, Max and Min denote maximum and minimum of MOA, (r_1, r_2, r_3) are random values in [0,1], ϵ is a small integer number, MOP (Match Optimizer Probability) is a coefficient, μ is a control parameter that emphasize exploration not only at the first iterations but also during the last iterations, α is second parameter that emphasize exploitation accuracy during the optimization. The AOA has only two tunning parameters: μ and α, alongside with the standard parameters: the population size and the stopping criterion.

Ch 4 Numerical Tests

4.1. Introduction

Six well-known benchmark problems: 10-bar truss, 25-bar truss, 72-bar truss, welded beam, pressure vessel and tension-compression string, were used in order to test the capability of four Metaheuristic Optimization Algorithms, selected from the introduced state-of-the-art algorithms in chapter (3): Grey Wolf Optimizer (GWO), Grasshopper Optimization Algorithm (GOA), Particle Swarm Optimization (PSO) and Differential Evolution (DE).

The scope of this chapter is to investigate the capability of these four algorithms, by being assessed on the same benchmark problems. The algorithms are implemented through MATLAB codes. In addition, the same stopping criterion corresponding to specific number of function evaluations (n*10,000 function evaluations) is used for all the algorithms. The population size parameter is set to 100 agents for each of GWO, GOA and DE, and 30 particles for the PSO. Furthermore, a common technique for dealing with the problem constraints were used. Lastly, implementing a common procedure for the discrete and integer design variables.

Regarding the constraints handling different approaches, in penalty methods, users usually try different values of the penalty factor to reach the optimum solution. To overcome this problem, Deb (2000) [1] originally proposed the Feasibility Criteria method, where the employed selection operator that compares and chooses between two solutions at a time, is based on the following rules:

- Any feasible individual is chosen over any infeasible one.
- For two feasible individuals, the individual with the better objective function value is chosen.
- For two infeasible individuals, the individual with smaller constraint violation is chosen.

In the implemented tests, a similar approach is employed, but with small modifications. The following fitness function is implemented:

$$F(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & \text{if } g_j(\boldsymbol{x}) \geq 0 \forall j = 1,2,\dots,m \\ f_{\max} + \sum_{j=1}^{m} \langle g_j(\boldsymbol{x}) \rangle & \text{otherwise} \end{cases}$$

Where $f_{\max}$ is the worst objective function value among the feasible solutions. In case of no feasible solution exists within the entire population, $f_{\max}$ is set to zero. In this method, Deb sums all the constraint violations and compares a single value, while the infeasible solutions are compared based only on their constraint violation.

In the followed approach in this thesis, some modifications are added to the mentioned-above selection operator's rules is order to emphasize the global exploration, as well to better catching of more promising feasible solutions.

In order to compute the fitness function value of the infeasible individuals, the $p_{\text{violation}}$ factor is introduced. $p_{\text{violation}}$ is the individual's normalized maximum constraint violation multiplied by a term that considers the number of the violated constraints by that individual.

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

Where n_{const} is the total number of constraints and $n_{\text{constviol}}$ is the number of the violated constraints.

To compute the individual's fitness function, the $p_{\text{violation}}$ factor is multiplied with the maximum objective function value of both the best-found feasible individual so far and that individual itself.

$$F(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & \text{if } g_j(\boldsymbol{x}) \leq 0 \forall j = 1,2,\dots,m \\ \max(f_{\text{best feasible}}, f(\boldsymbol{x})) \times p_{\text{violation}} & \text{otherwise} \end{cases}$$

According to the above, the selection operator's rules, in order choose between two solutions at a time, are modified as follow:

- Among a feasible individual and another infeasible, the selection of the feasible one depends on the objective function value, the constraint violation value and the number of violated constraints of the infeasible individual.
- For two feasible individuals, the individual with the better objective function value is chosen.
- For two infeasible individuals, the individual with smaller objective function value, constraint violation value and number of violated constraints is chosen.

The considered constraints are indeed two types: functions and bounds. The former holds the inequality and equality constraints, which hold more complexity. The latter holds the variables' upper and lower limits.

4.2. Benchmark Structural Optimization Problems

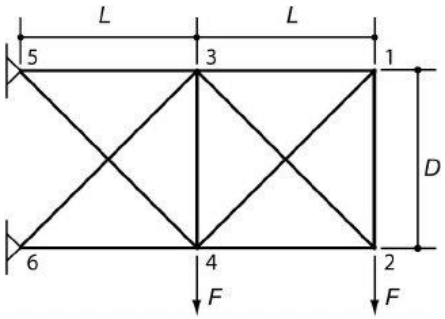
The four optimization algorithms are applied at first on three well-known truss structures' problems. Then they are applied on three other famous structural optimization problems. Each optimization problem was solved through 20 independent runs by each algorithm, in order to obtain the probabilistic characteristics of the results obtained by every algorithm. The mathematical formulations of these problems are further elaborated.

The three employed trusses for this investigation work are: 10-bar, 25-bar and 72-bar truss structures. They are all steel structures, employed for a size optimization, where the design variables range from 8 to 16 variables. The design variables spaces are continuous for each element's cross-section area. The objective function for the three structures is considered as the weight to be minimized.

4.1.1 10-bar truss

An independent design variable considered for each bar, resulting in a total of 10 design variables. The material density is 0.1 lb/in^3, and Young's modulus is 10000 ksi. The stress limitations of the members are considered as ±25 ksi, and the displacement limitation of all nodes is ±2 in. The applied load is F= 100 kips. The length: L = D = 360 in.

The minimum-found value of weight in the literature is equal to 5057.88 lb.

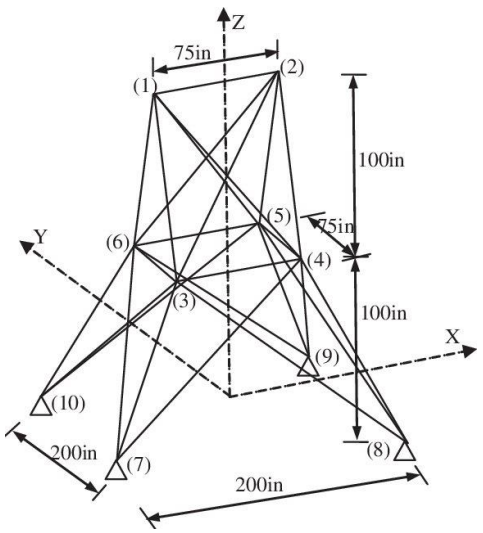


4.1.2 25-bar truss

The structural elements are grouped, resulting in a total of 8 design variables. The material density is 0.1 lb/in^3, and Young's modulus is 10000 ksi. The stress limitations of the members are considered as 35 ksi for tension while in case of compression, the stress is limited according to the AISC code. The displacement limitation of all nodes is ±0.35 in. The applied loads (kips) and dimensions of the structure are shown in the following figures.

The minimum-found value of weight in the literature is equal to 545.175 lb.

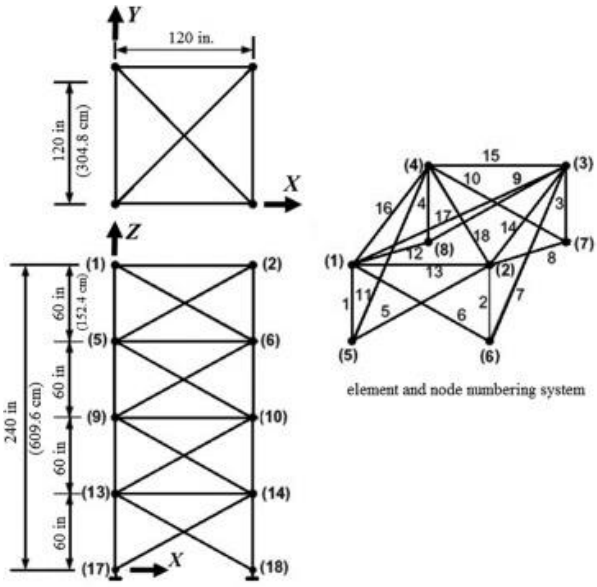
Nodes	P_x	P_y	P_z
9	1	-10	-10
10	0	-10	-10
7	0.5	0	0
5	0.6	0	0



4.1.3 72-bar truss

The structural elements are grouped, resulting in a total of 16 design variables. The material density is 0.1 lb/in^3, and Young's modulus is 10000 ksi. The stress limitations of the members are considered as 25 ksi for tension while in case of compression, the stress is limited according to the AISC code. The displacement limitation of all nodes is ±0.25 in. The applied loads are at node (17): 5, 5 and -5 in directions x, y and z, respectively. The dimensions of the structure are shown in the following figures.

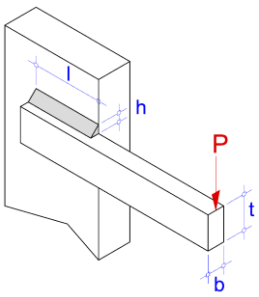
The minimum-found value of weight in the literature is equal to 379.66 lb.



4.1.4 Welded Beam

The well-known welded beam design problem holds 4 design variables: height and length of the weld and height and width of the beam section. Where $P = 6000\ \mathrm{lb}, L = 14\ in, E = 30 \times 10^6\ psi, G = 12 \times 10^6\ psi, \tau_{\max} = 13600\ psi, \sigma_{\max} = 30000\ psi, \delta_{\max} = 0.25\ in.$

The minimum-found value of weight in the literature is equal to 1.72485084 lb.



The mathematical expressions for this optimization problem are as follow:

$$f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)$$

Subject to:

$$\begin{aligned}
g_1(x) &= \tau(x) - \tau_{\max} \le 0 \\
g_2(x) &= \sigma(x) - \sigma_{\max} \le 0 \\
g_3(x) &= x_1 - x_4 \le 0 \\
g_4(x) = 1.10471x_1^2x_2 &+ 0.04811x_3x_4(14 + x_2) - 5.0 \le 0 \\
g_5(x) &= 0.125 - x_1 \le 0 \\
g_6(x) &= \delta(x) - \delta_{\max} \le 0 \\
g_7(x) &= P - P_c(x) \le 0 \\
0.1 &\le x_1 \le 2 \\
0.1 &\le x_2 \le 10 \\
0.1 &\le x_3 \le 10 \\
0.1 &\le x_4 \le 2
\end{aligned}$$

Where:

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau')^2}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}$$

$$\tau'' = \frac{MR}{J}$$

$$M = P\left(L + \frac{x_2}{2}\right)$$

$$R = \frac{x_2^2}{4} + \left(\frac{x_1 + x_2}{2}\right)^2$$

$$J = 2\left[\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right]$$

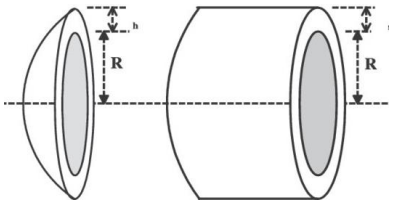
$$\sigma(x) = \frac{6PL}{x_4x_3^2}$$

$$\delta(x) = \frac{4PL^3}{Ex_4{x_3}^3}$$

$$P_c(x) = \frac{4{,}013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$

4.1.5 Pressure Vessel

The well-known pressure vessel design problem holds 4 design variables: $x_1,\ x_2,\ x_3\ and\ x_4$. The thickness of the vessel, thickness of the head, inner radius of the vessel and the length of the vessel, respectively.
The minimum-found value of weight in the literature is equal to 5885.3328 lb.



The mathematical expressions are as follow:

$$minimize\ f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

$$g_1(x) = -x_1 + 0.0193x_3 \leq 0$$

$$g_2(x) = -x_2 + 0.00954x_3 \leq 0$$

$$g_3(x) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0$$

$$g_4(x) = x_4 - 240 \leq 0$$

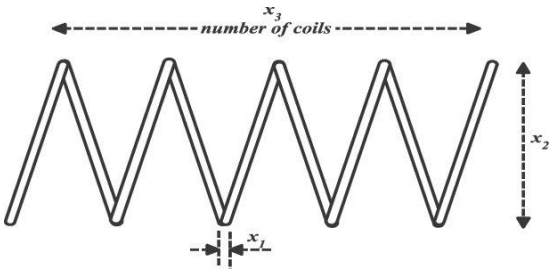
$$x_1 \geq 0.0625$$

$$x_2 \leq 99 \times 0.0625$$

$$10 \leq x_3, x_4 \leq 200$$

4.1.6 Tension-Compression String

The well-known tension-compression string design problem holds 3 design variables: $x_1,\ x_2\ and\ x_3$. The wire diameter (d), coil diameter (D) and the number of the active coils (N), respectively.
The minimum-found value of weight in the literature is equal to 0.012665 lb.



The mathematical expressions for this optimization problem are as follow:

$$minimize\ f(\vec{X}) = (x_3 + 2)x_2x_1^2$$

$$\vec{X} = [x_1, x_2, x_3] = [d, D, N]$$

$$g_1(\vec{x}) = 1 - \frac{x_2^3}{71785x_1^4} \leq 0$$

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0$$

$$g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0$$

$$g_4(\vec{x}) = \frac{x_2 + x_1}{1.5} - 1 \leq 0$$

$$0.05 \leq x_1 \leq 2$$

$$0.25 \leq x_2 \leq 1.30$$

$$2 \leq x_3 \leq 15$$

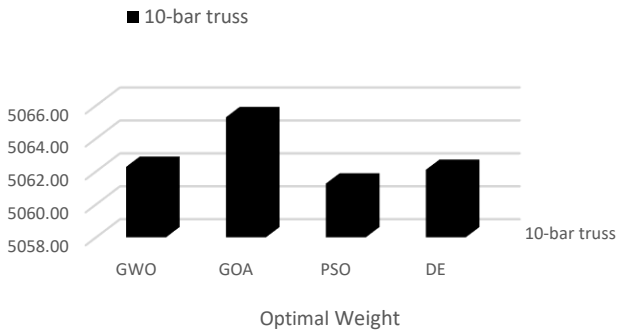
4.3. Results

Following is presented the obtained result for each of the used benchmarks, in form of comparisons between the investigated algorithm with respect to the optimal weight, standard deviation and the coefficient of variation.

4.3.1 Optimal Weight Comparison

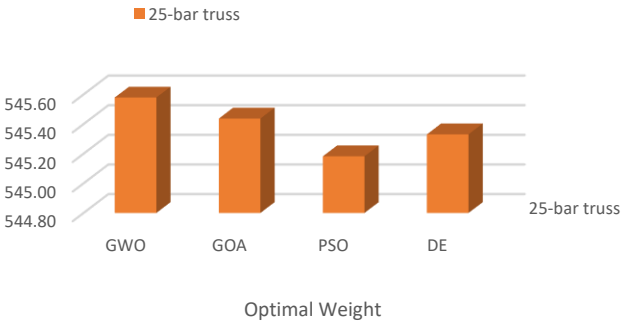
4.3.1.1 10-bar truss

As shown, the obtained optimal weights by the four algorithms are approximately the same (with differences less than **4** lb), PSO= **5061.27**, DE= 5062.10, GWO= 5062.28, GOA= 5065.29. While the least-found weight in the literature was **5057.88** lb.



4.3.1.2 25-bar truss

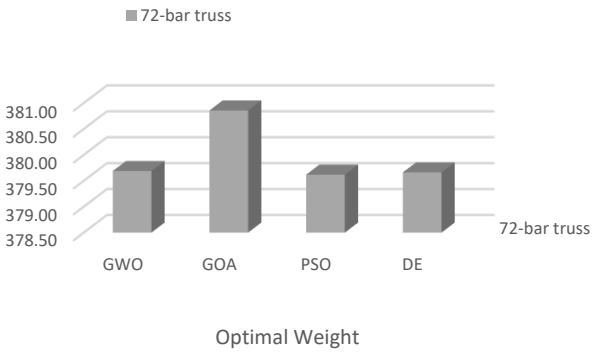
For the 25-bar truss, the obtained optimal weights are also almost identical (with differences less than **0.5** lb), PSO= **545.18**, DE= 545.33, GWO= 545.58, GOA= 545.44. While being the least-found weight in the literature: **545.175** lb.



4.3.1.3 72-bar truss

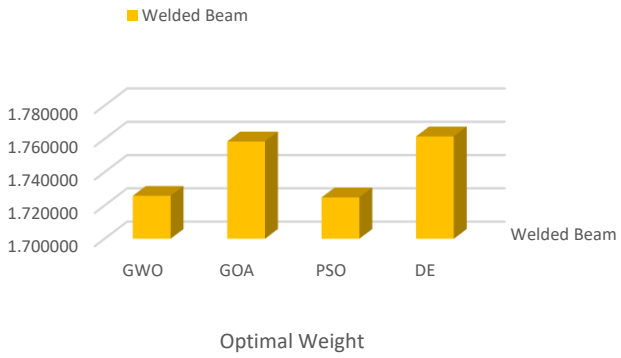
The obtained optimal weights of the 72-bar truss by the four algorithms are as well approximately the same (with differences less than **1.5** lb), PSO= **379.62**, DE= 379.66, GWO= 379.69, GOA= 380.86. While the least-found weight in the literature: **379.66** lb.

Although with small difference, but it is worthy to mention that the implemented test throughout this thesis brought optimal weight by the PSO even lower than the least-found in the literature weight.



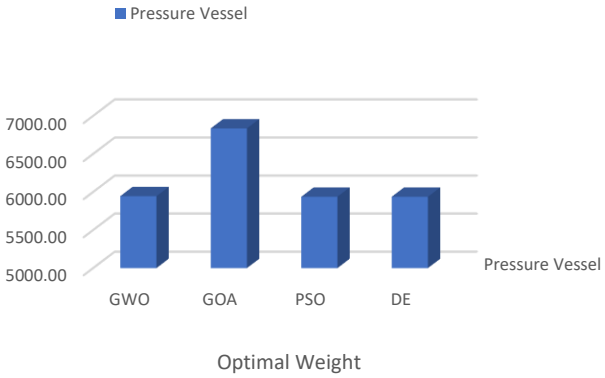
4.3.1.4 Welded Beam

For the welded beam, the optimal weights by the four algorithms are similar (with differences less than **4** lb), PSO= **1.7249**, DE= 1.7615, GWO= 1.7258, GOA= 1.7585. While the least-found weight in the literature: **1.7249** lb



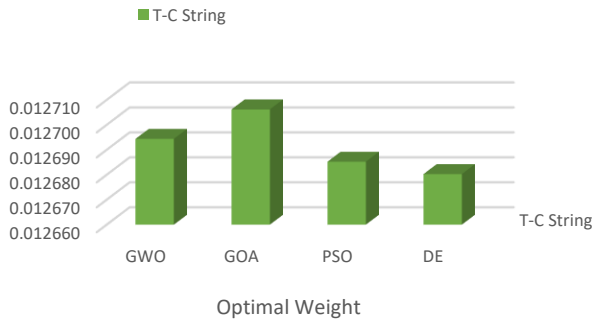
4.3.1.5 Pressure Vessel

The obtained optimal weights for the pressure vessel by the four algorithms are similar (with differences less than **4** lb), PSO= **5937.14**, DE= 5937.14, GWO= 5944.53, GOA= 6835.21. While the least-found weight in the literature: **5885.33** lb.



4.3.1.6 Tension-Compression String

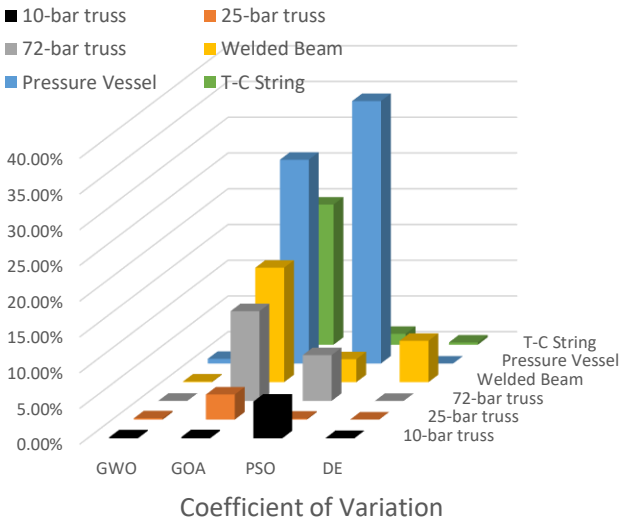
For the tension-compression string, the optimal weights by the four algorithms are almost identical, PSO= 0.012685, DE= **0.012680**, GWO= 0.012694, GOA= 0.012706. While the least-found weight in the literature: **0.012665** lb.



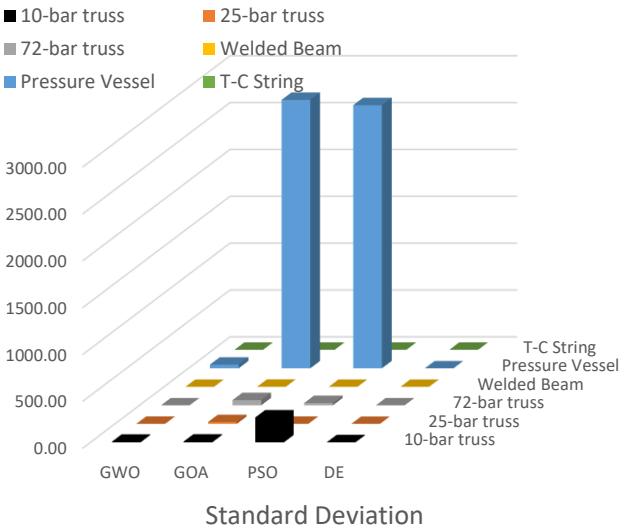
4.3.2 Other Comparisons

Following are presented infographic comparisons between the four algorithms, among the 6 benchmark optimization problems, with respect to: the coefficient of variation, standard deviation and the ratio between the optimum obtained weight by each algorithm to the target weight (least-found in the literature).

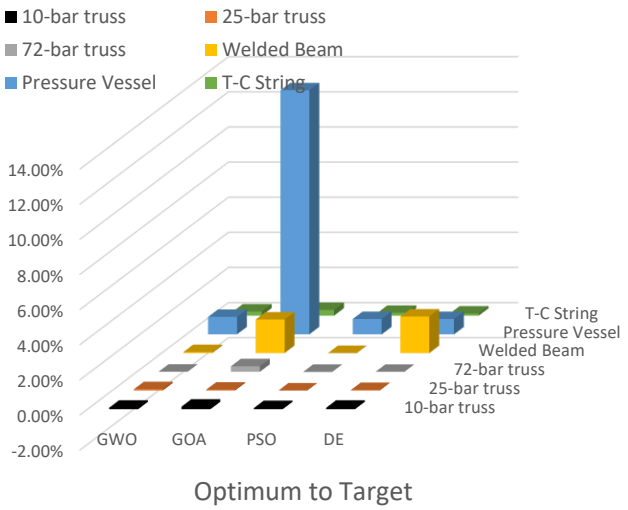
The GWO has shown the best coefficient of variation among the six benchmark problems.



The DE has shown the best standard deviation among the six benchmark problems.



The four MOAs have shown good capability through obtaining optimal weights that are almost the same as the least-found weight in the literature for the three truss structures. However, it is obvious that the GOA has performed not in good manner in the pressure vessel optimization problem.



Ch 5 Conclusion

- In the current era of structural optimization, neural networks are broadly utilized instead of the accurate, but complex, FE methods in evaluating and predicting different aspects (e.g., structural time history responses).
- The combination between MOAs and NN is a trending direction in the modern literature that shows high capability in reducing the computational time and efforts.
- Equipping AI features (e.g., Fuzzy Logic) within an optimization technique, is observed through the literature review as a trending approach in the field, that gets the process done faster and more open to handle general families of optimization problems.
- Penalty function is the most used approach for handling the constraints.
- It is trending recently to use a random-search optimization algorithm during the exploration phase to spot just the promising areas within the search space, then to employ a deterministic or derivative-based optimization algorithm in order to exploit that promising areas, in seek of finding the exact optimal solution.

References

[1] Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Methods Appl. Mech. Eng. 186(2–4), 311–338 (2000).

[2] Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Trans. Evol. Comput. 4(3), 284–294 (2000).

[3] A.G.M. Michell. LVIII. The limits of economy of material in frame structures. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1904.

[4] Guoyong Cuia, Changyu Cuia,b,Shape Optimization Based on ANSYS, Journal of Information & Computational Science–4297 July 20, 2015.

[5] Prendes Gero, M., García, A., & del Coz Díaz, J. (2006, 12). Design optimization of 3D steel structures: Genetic algorithms vs. classical techniques. Journal of Constructional Steel Research, 62(12), 1303-1309.

[6] Gholizadeh, S., Salajegheh, E., & Torkzadeh, P. (2008, 4). Structural optimization with frequency constraints by genetic algorithm using wavelet radial basis function neural network. Journal of Sound and Vibration, 312(1-2), 316-331.

[7] Kashani, A., Gandomi, M., Camp, C., & Gandomi, A. (2020, 5). Optimum design of shallow foundation using evolutionary algorithms. Soft Computing, 24(9), 6809-6833.

[8] Liu, M. (2011, 3). Progressive collapse design of seismic steel frames using structural optimization. Journal of Constructional Steel Research, 67(3), 322-332.

[9] & Adeli, H. (2013). Two-phase genetic algorithm for size optimization of free-form steel space-frame roof structures. Journal of Constructional Steel Research, 90, 283-296.

[10] Kociecki, M., & Adeli, H. (2014). Two-phase genetic algorithm for topology optimization of free-form steel space-frame roof structures with complex curvatures. Engineering Applications of Artificial Intelligence, 32, 218-227.

[11] Kociecki, M., & Adeli, H. (2015, 2). Shape optimization of free-form steel space-frame roof structures with complex geometries using evolutionary computing. Engineering Applications of Artificial Intelligence, 38, 168-182.

[12] Lavaei, A., & Lohrasbi, A. (2015). Dynamic optimization of structures subjected to earthquakes. Civil-Comp Proceedings, 109.

[13] S. Gholizadeh and E. Salajegheh, “An intelligent neural system for predicting structural response subject to earthquakes,” in Proc. the Fifth International Conference on Engineering Computational Technology, 2006, p. 63.

[14] T. Kohonen, Self-Organization and Associative Memory, 2nd edition, Springer-Verlag, Berlin, 1987.

[15] J. S. Arora, Optimization of Structures Subjected to Dynamic Loads, Structural dynamic systems computational techniques and optimization, Gordon and Breach Science Publishers, 1999.

[16] Assimi, H., Jamali, A., & Nariman-zadeh, N. (2017, 12). Sizing and topology optimization of truss structures using genetic programming. Swarm and Evolutionary Computation, 37, 90-103.

[17] P. Hajela, E. Lee, C.-Y. Lin, Genetic algorithms in structural topology optimization, in: Martin Philip Bendsøe, Carlos A. Mota Soares (Eds.), Topology Design of Structures, Springer, 1993, pp. 117–133.

[18] H. Adeli, S. Kumar, Distributed genetic algorithm for structural optimization, J. Aerosp. Eng. 8 (3) (1995) 156–163.

[19] J. Yang, C.K. Soh, An integrated shape optimization approach using genetic algorithms and fuzzy rule-based system, in: Proceedings of the Developments in Neural Networks and Evolutionary Computing for Civil and Structural Engineering, 1995, pp. 171–177.

[20] K. Deb, Design of truss-structures for minimum weight using genetic algorithms, Finite Elem. Anal. Des. 37 (2001) 447–465.

[21] K.S. Lee, Z.W. Geem, A new structural optimization method based on the harmony search algorithm, Comput. Struct. 82 (2004) 781–798.

[22] L.J. Li, Z.B. Huang, F. Liu, Q.H. Wu, A heuristic particle swarm optimizer for optimization of pin connected structures, Comput. Struct. 85 (2007) 340–349.

[23] G.-C. Luh, C.-Y. Lin, Optimal design of truss structures using ant algorithm, Struct. Multidiscip. Optim. 36 (2007) 365–379.

[24] C.-Y. Wu, K.-Y. Tseng, Truss structure optimization using adaptive multi-population differential evolution, Struct. Multidiscip. Optim. 42 (2010) 575–590.

[25] M. Fenton, C. McNally, J. Byrne, E. Hemberg, J. McDermott, M. O'Neill, Automatic innovative truss design using grammatical evolution, Autom. Constr. 39 (2014) 59–69.

[26] A. Kaveh, T. Bakhshpoori, E. Afshari, An efficient hybrid particle swarm and swallow swarm optimization algorithm, Comput. Struct. 143 (2014) 40–59.

[27] A. Kaveh, T. Bakhshpoori, A new metaheuristic for continuous structural optimization: water evaporation optimization, Struct. Multidiscip. Optim. 54 (2016) 23–43.

[28] Ye, J., & Lu, M. (2020, 6). Guided genetic algorithm for dome optimization against instability with discrete variables. Journal of Constructional Steel Research, 139.

[29] Ye, J., & Lu, M. (2020, 6). Design optimization of domes against instability considering joint stiffness. Journal of Constructional Steel Research, 169.

[30] Parastesh, H., Hajirasouliha, I., Taji, H., & Bagheri Sabbagh, A. (2019, 4). Shape optimization of cold-formed steel beam-columns with practical and manufacturing constraints. Journal of Constructional Steel Research, 155, 249-259.

[31] Mojtabaei, S., Hajirasouliha, I., Ye, J., & Lim, J. (2020, 5). Coupled element and structural level optimisation framework for cold-formed steel frames. Journal of Constructional Steel Research, 168.

[32] Nan, B., Bai, Y., & Wu, Y. (2020, 3). Multi-objective optimization of spatially truss structures based on node movement. Applied Sciences (Switzerland), 10(6).

[33] Deb, K. Multi-Objective Optimization Using Evolutionary Algorithms; Wiley: Chichester, UK, 2001.

[34] Skoglund, O., Leander, J., & Karoumi, R. (2020, 10). Optimizing the steel girders in a high strength steel composite bridge. Engineering Structures, 221.

[35] Salajegheh, E., Gholizadeh, S., & Khatibinia, M. (2008, 3). Optimal design of structures for earthquake loads by a hybrid RBF-BPSO method. Earthquake Engineering and Engineering Vibration, 7(1), 13-24.

[36] Yang, B., Zhang, Q., & Li, H. (2015, 4). Solving truss topological optimization with discrete design variables via swarm intelligence. KSCE Journal of Civil Engineering, 19(4), 952-963.

[37] Achtziger, W. and Stolpe, M. (2007). "Truss topology optimization with discrete design variables-guaranteed global optimality and benchmark examples." Structural and Multidisciplinary Optimization, Vol. 34, No. 1, pp. 1-20.

[38] Mortazavi, A., Toğan, V., & Nuhoğlu, A. (2017, 2). An integrated particle swarm optimizer for optimization of truss structures with discrete variables. Structural Engineering and Mechanics, 61(3), 359-370.

[39] Mortazavi, A., Toğan, V., & Nuhoğlu, A. (2017, 11). Weight minimization of truss structures with sizing and layout variables using integrated particle swarm optimizer. Journal of Civil Engineering and Management, 23(8), 985-1001.

[40] Mortazavi, A., Toğan, V., Daloğlu, A., & Nuhoğlu, A. (2018). Comparison of Two Metaheuristic Algorithms on Sizing and Topology Optimization of Trusses and Mathematical Functions.

[41] Mortazavi, A. (2019, 5). Interactive fuzzy search algorithm: A new self-adaptive hybrid optimization algorithm. Engineering Applications of Artificial Intelligence, 81, 270-282.

[42] Mortazavi, A. (2020, 4). Large-scale structural optimization using a fuzzy reinforced swarm intelligence algorithm. Advances in Engineering Software, 142.

[43] Mortazavi, A. (2021). Size and layout optimization of truss structures with dynamic constraints using the interactive fuzzy search algorithm. Engineering Optimization, 53(3), 369-391.

[44] Fathali, M., & Hoseini Vaez, S. (2020, 1). Optimum performance-based design of eccentrically braced

[45] Kaveh A, Bakhshpoori T. An accelerated water evaporation optimization formulation for discrete optimization of skeletal structures. Comput Struct 2016, 177:218–28.

[46] Kennedy J, Eberhart R. Particle swarm optimization. Neural Networks, 1995 Proceedings, IEEE International Conference on1995. p. 1942-8 vol.4.

[47] Kaveh A, Mahdavi VR. Colliding bodies optimization: a novel meta-heuristic method. Comput Struct 2014, 139:18–27.

[48] Kaveh A, Ilchi Ghazaan M. Enhanced colliding bodies optimization for design problems with continuous and discrete variables. Adv Eng Softw 2014, 77:66–75.

[49] Abedini, H., Hoseini Vaez, S., & Zarrineghbal, A. (2020, 6). Optimum design of buckling-restrained braced frames. Structures, 25, 99-112.

[50] Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM. Salp Swarm Algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 2017, 114:163–91.

[51] Omidinasab, F., & Goodarzimehr, V. (2020). A hybrid particle swarm optimization and genetic algorithm for truss structures with discrete variables. Journal of Applied and Computational Mechanics, 6(3), 593-604.

[52] Skandalos, K., Afshari, H., Hare, W., & Tesfamariam, S. (2020, 5). Multi-objective optimization of inter-story isolated buildings using metaheuristic and derivative-free algorithms. Soil Dynamics and Earthquake Engineering, 132.

[53] Jahangiri, M., Hadianfard, M., Najafgholipour, M., Jahangiri, M., & Gerami, M. (2020, 7). Interactive autodidactic school: A new metaheuristic optimization algorithm for solving mathematical and structural design optimization problems. Computers and Structures, 235.

[54] Rao, R., Savsani, V., & Vakharia, D. (2011, 3). Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. CAD Computer Aided Design, 43(3), 303-315.

[55] Piliounis, G., & Lagaros, N. (2014). Reliability analysis of geostructures based on metaheuristic optimization. Applied Soft Computing Journal, 22, 544-565.

[56] Nadimi-Shahraki, M., Taghian, S., Mirjalili, S., & Faris, H. (2020, 12). MTDE: An effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems. Applied Soft Computing Journal, 97.

[57] Mirjalili, S., Mirjalili, S., & Lewis, A. (2014). Grey Wolf Optimizer. Advances in Engineering Software, 69, 46-61.

[58] Nadimi-Shahraki, M., Taghian, S., & Mirjalili, S. (2021, 3). An improved grey wolf optimizer for solving engineering problems. Expert Systems with Applications, 166.

[59] Mirjalili, S., & Lewis, A. (2016, 5). The Whale Optimization Algorithm. Advances in Engineering Software, 95, 51-67.

[60] Mirjalili, S. (2016, 5). Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Computing and Applications, 27(4), 1053-1073.

[61] Saremi, S., Mirjalili, S., & Lewis, A. (2017, 3). Grasshopper Optimisation Algorithm: Theory and application. Advances in Engineering Software, 105, 30-47.

[62] Mirjalili, S., Gandomi, A., Mirjalili, S., Saremi, S., Faris, H., & Mirjalili, S. (2017, 12). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software, 114, 163-191.

[63] Gandomi, A., & Alavi, A. (2012, 12). Krill herd: A new bio-inspired optimization algorithm. Communications in Nonlinear Science and Numerical Simulation, 17(12), 4831-4845.

[64] Kallioras, N., Lagaros, N., & Avtzis, D. (2018, 7). Pity beetle algorithm – A new metaheuristic inspired by the behavior of bark beetles. Advances in Engineering Software, 121, 147-166.

[65] Mirjalili, S. (2015). The ant lion optimizer. Advances in Engineering Software, 83, 80-98.

[66] Mirjalili, S. (2015, 11). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems, 89, 228-249.

[67] Mirjalili, S., Mirjalili, S., & Hatamlou, A. (2016, 2). Multi-Verse Optimizer: a nature-inspired algorithm for global optimization. Neural Computing and Applications, 27(2), 495-513.

[68] Mirjalili, S. (2016, 3). SCA: A Sine Cosine Algorithm for solving optimization problems. Knowledge-Based Systems, 96, 120-133.

[69] Hosseini, S., & Al Khaled, A. (2014). A survey on the Imperialist Competitive Algorithm metaheuristic: Implementation in engineering domain and directions for future research. Applied Soft Computing, 24, 1078-1094.

[70] Gandomi, A. (2014). Interior search algorithm (ISA): A novel approach for global optimization. ISA Transactions, 53(4), 1168-1183.

[71] Li, S., Chen, H., Wang, M., Heidari, A., & Mirjalili, S. (2020, 10). Slime mould algorithm: A new method for stochastic optimization. Future Generation Computer Systems, 111, 300-323.

[72] Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. (2021, 4). The Arithmetic Optimization Algorithm. Computer Methods in Applied Mechanics and Engineering, 376.