Contents lists available at ScienceDirect

Decision Analytics Journal

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

A comparison performance analysis of eight meta-heuristic algorithms for optimal design of truss structures with static constraints

Nima Khodadadi [a,*], Aybike Özyüksel Çiftçioğlu [b], Seyedali Mirjalili [c,d,e], Antonio Nanni [a]

[a] *The Department of Civil and Architectural Engineering, University of Miami, Coral Gables, FL 33146-0630, USA*
[b] *Department of Civil Engineering, Faculty of Engineering, Manisa Celal Bayar University, Turkiye*
[c] *Centre for Artificial Intelligence Research and Optimization, Torrens University Australia, Australia*
[d] *Yonsei Frontier Lab, Yonsei University, Seoul, Republic of Korea*
[e] *University Research and Innovation Center, Obuda University, 1034 Budapest, Hungary*

ARTICLE INFO

Keywords:
Structural optimization
Meta-heuristic algorithms
Stochastic Paint Optimizer
Truss structures
Static constraints

ABSTRACT

Metaheuristics have been successfully used for solving complex structural optimization problems. Many algorithms are proposed for truss structure size and shape optimization under some constraints. This study considers eight population-based meta-heuristic methods: African Vultures Optimization Algorithm (AVOA), Flow Direction Algorithm (FDA), Arithmetic Optimization Algorithm (AOA), Generalized Normal Distribution Optimization (GNDO), Stochastic Paint Optimizer (SPO), Chaos Game Optimizer (CGO), Crystal Structure Algorithm (CRY) and Material Generation Algorithm (MGO). These meta-heuristics methods are used to optimize the size of three aluminum truss structures. Optimization aims to reduce the weight of the truss members while meeting a set of displacement and stress constraints. The performance of these methods is assessed by solving and optimizing three well-known truss structure benchmarks under some constraints. The results show that the Stochastic Paint Optimizer (SPO) outperforms the other algorithms in terms of accuracy and convergence rate.

1. Introduction

Optimization can be defined mathematically as minimizing or maximizing a problem-specific objective function while meeting certain constraints. Optimization algorithms are separated into two classes deterministic and stochastic. Deterministic methods are also classified as computational and gradient-based methods [1]. Computational methods do not need gradient function calculations but are slow and ineffective. Gradient-based methods use the slope or derivative of the objective function, yet they do not guarantee convergence to the global optima when the objective function is not flat and smooth. Solving high-dimensional, multimodal, or non-differentiable problems with deterministic methods is difficult or even impossible. In recent years, researchers have developed many nature-inspired metaheuristic algorithms for complex optimization problems that cannot be solved with conventional techniques in a reasonable time or with precision. Metaheuristic methods do not need gradient information and seek the global optimum by repeatedly calling the objective function. Such algorithms narrow the search space and try to find solutions effectively [2,3].

Many approximate techniques are global search methods called metaheuristics. These methods are developed to address the weaknesses of classical approaches. The metaheuristic algorithms can find near-global or global solutions by employing intelligence of natural phenomena. Various metaheuristic algorithms have been proposed in recent decades based on natural processes, collective behavior, art, physics, or mathematics rules. Genetic Algorithm (GA) [4], Differential Evolution (DE) [5], Ant Colony Optimization (ACO) [5], Particle Swarm Optimization (PSO) [6], Cuckoo Search (CS) [7], Golden Eagle Optimizer (GEO) [8], Stochastic Paint Optimizer (SPO) [9], Chaos Game Optimizer (CGO) [10], Mountain Gazelle Optimizer (MGO) [11] and Tunicate Swarm Algorithm (TSA) [12]. Many researchers utilized these algorithms for solving different structural optimization problems such as trusses [13,14], frames [15,16], and other real applications of engineering problems [17,18]. These studies demonstrate that metaheuristic algorithms can solve problems with good accuracy in a reasonable time when employed to deal with complex optimization problems. Ease of implementation, simple framework, good accuracy, and reasonable execution time are some advantages of metaheuristic algorithms compared with analytical techniques. Accordingly, many researchers improved these metaheuristic algorithms. Metaheuristic algorithms have been continually improved in order to solve a wider class of optimization

* Corresponding author.
E-mail addresses: Nima.khodadadi@miami.edu (N. Khodadadi), aybike.ozyuksel@cbu.edu.tr (A.Ö. Çiftçioğlu), ali.mirjalili@gmail.com (S. Mirjalili), nanni@miami.edu (A. Nanni).

https://doi.org/10.1016/j.dajour.2023.100266
Received 20 May 2023; Received in revised form 1 June 2023; Accepted 12 June 2023
Available online 15 June 2023

problems, especially for truss structures. Dynamic Arithmetic Optimization Algorithm (DAOA) [19], Improved Symbiotic Organisms Search (ISOS) [20], Improved Crow Search Algorithm (ICSA) [21], Advanced Charged System Search (ACSS) [16], Improved Whale Algorithm (IWA), Dynamic Water Strider Algorithm (DWSA) [22] are applied to truss structures. In addition, some metaheuristic algorithms are applied for multi-objective truss problems, such as Multi-objective Teaching–Learning-Based Optimization (MOTLBO) [23], Multi-Objective Stochastic Paint Optimizer (MOSPO) [24,25], Multi-objective Colliding Bodies Optimization (MOCBO) [25]

In recent years, the demand for raw materials has been increasing rapidly in the world, whose population is constantly growing and developing. Aluminum and steel, essential building materials, are used in many engineering applications [26]. Additionally, trusses are among the most widely used forms in structural engineering. However, as the number of members and the number of member groups of truss structures increases, the design and analysis of these structures become complex. Therefore, the number of possible solution combinations available for these structures also increases. Among these combinations of solutions, one of the main goals of the engineer is to design the lightest and, consequently, the most economical structure that meets the constraints under the given loading conditions. Consequently, unnecessary material waste will be avoided in response to the increasing need for raw materials by ensuring the optimum design of truss structures. Many researchers have used metaheuristic algorithms to overcome the challenge of designing and analyzing truss structures with many members.

Lingyun et al. [27] employed GA to address truss optimization on shape and sizing with frequency constraints. Gomes [28] utilized PSO to optimize planar and spatial truss structure layout. Kaveh et al. [29] optimized two- and three-dimensional truss structures using a hybrid of invasive weed optimization with a shuffled frog-Leaping algorithm (IWO-SFLA).

Tang et al. [30] employed an adaptive 3-stage hybrid teaching-based differential evolution to optimize truss structures for size and layout. Khodadadi and Mirjalili [31] employed the generalized normal distribution optimization (GNDO) for optimal designing truss structures where minimizing the overall weight is the objective function. Pierezan et al. [14] performed the trusses with discrete design variables and focused on minimizing the structure weight under the required constraints.

Jiang et al. [32] investigated the performance of an improved whale algorithm (IWA) method using two planar and space truss designs. Jawad et al. [33] employed an artificial bee colony (ABC) algorithm to apply the optimum design of members size and layout optimization of truss with displacement, stress, and buckling constraints. Bekdaş et al. [34] designed optimal truss structures and various benchmark design examples with six different metaheuristics and the modification of Lèvy flight for three algorithms.

Gholizadeh and Sojoudizadeh [35] performed the discrete design optimization of truss structures with a modified sine–cosine algorithm (SCA). Li et al. [36] developed the improved chicken swarm optimization (ICSO) algorithm for the minimal cost truss design. Jawad et al. [37] optimized five well-known planar and spatial steel trusses with the dragonfly algorithm (DA) with discrete sizing variables.

Kumar et al. [38] applied the optimal design of truss optimization problems subjected to multiple fundamental frequency constraints with shape and size variables. Mashayekhi and Yousefi [21] applied their proposed hybrid of the crow search algorithm and the cellular automata (CSA-CA) method to optimize the size and topology of truss structures. Serpik [39] employed the improved technique to achieve trusses' discrete size and shape optimization via a job search-inspired strategy together with genetic operators. Dede et al. [40] tested the efficiency of the Jaya algorithm (JA) for the design of the braced dome structures.

Altay et al. [41] proposed a novel approach termed the Modified Salp Swarm Algorithm (MSSA) for design optimization of truss structures. The study evaluated five truss structures that incorporated discrete and continuous variables. These structures had been previously optimized using metaheuristics and were assessed using both size and size-shape optimization methods. Initially, the SSA showed poor performance and convergence issues with random solutions. Nevertheless, it ultimately generated outcomes that were comparable to previously published findings, particularly in the context of continuous problems. Conversely, the MSSA exhibited superior performance in discrete problems and attained outcomes that were proximate to the optimal benchmarks in continuous problems. Furthermore, the convergence curves of the (MSSA) exhibited a moderate enhancement in convergence rates, particularly for discrete problems. The anticipated outcome of these discoveries was an improvement in the efficacy, rapidity of convergence, and dependability of forthcoming practical implementations.

The advance version of Neural Network Algorithm (ANNA) was developed by Khodadadi et al. [42]. The algorithm was developed utilizing the principles of biological nerve structures and artificial neural networks. The research investigated the efficacy of the suggested approach in the context of engineering design issues. Two methodologies were investigated to augment the conventional neural network algorithm. The initial approach entailed an enhanced initialization mechanism that employed opposite-based learning. The second methodology integrated a set of modifiable parameters to augment the algorithm's capacity to explore and exploit, leading to enhanced solutions while diminishing the requisite structural analyses. The efficacy of the novel algorithm was assessed through the utilization of five established constrained benchmarks, wherein its efficiency was compared to state-of-the-art optimization techniques. The findings of the conducted case studies indicated that the enhanced iteration of the algorithm displayed superior efficacy and resilience in comparison to the initial version and specific alternative approaches.

The novel and enhanced metaheuristic technique to augment the efficacy of the conventional Bat Algorithm (BA) was proposed by Vu-Huu et al. [43] in addressing optimization engineering problems and optimizing the design of truss structures. The BA algorithm, similar to several other metaheuristic algorithms, presented a straightforward implementation and the capacity to tackle a diverse array of problems with adaptability. Nevertheless, a limitation emerged when the utilization phase of the BA experiences swift fluctuations in volume and pulse discharge frequency, resulting in a state of inactivity after an initial stage. As a result, the algorithm's ability to accurately identified the optimal solution may be limited. In order to tackle this matter, a novel loudness function was suggested with the aim of proficiently adjusting the step size of the random walk. Furthermore, a push-process methodology was implemented to intervene in the BA algorithm and accelerate the identification of the true global optimum within a limited number of generations, as opposed to necessitating a multitude of computational iterations. The obtained outcomes were subjected to comparison and validation against established scholarly investigations. The BA algorithm was utilized to optimize the weight of truss structures in terms of their structural design.

The Bonobo Optimizer (BO) [44] algorithm was utilized to perform sizing optimization of truss structures, while considering both discrete and continuous variables. The BO algorithm simulates the social conduct and reproductive patterns exhibited by bonobos, serving as a source of inspiration. Similar to other primates, bonobos employ a fission–fusion group tactic, wherein they establish small groups of varying sizes and navigate autonomously throughout their designated territory. The primary aim of sizing optimization is to identify the truss configuration that exhibits the lowest possible weight, while simultaneously adhering to various loading conditions and limitations on member stresses and nodal displacements. In order to evaluate the dependability and resilience of the BO algorithm, it was implemented on five established truss illustrations with unchanging geometry, spreading from 1 to 160 elements. The findings explicitly indicate that the

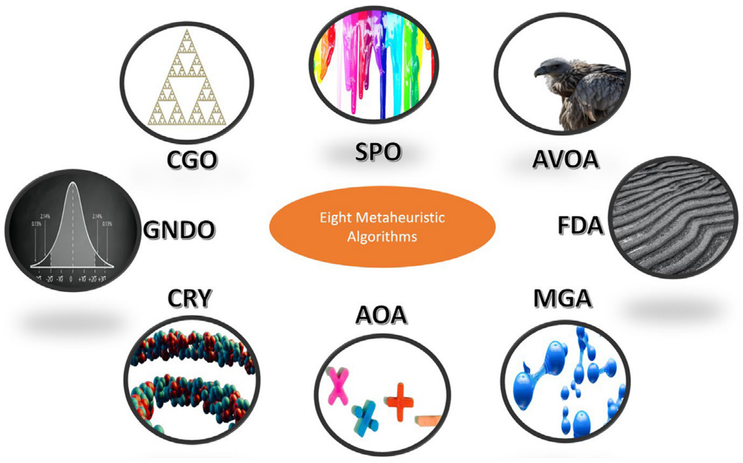

Fig. 1. Eight metaheuristic algorithms.

BO algorithm is a potent methodology for exploring and enhancing truss configurations. The method adeptly manages static constraints pertaining to discrete as well as continuous variables.

A bionic optimization algorithm called the Coronavirus Mask Protection Algorithm (CMPA), has been proposed in [45], which is inspired by COVID-19 prevention and based on the virus transmission of COVID-19 The process of infection and immunity in CMPA is comprised of three distinct phases, namely the infection stage, diffusion stage, and immune stage. It is noteworthy that the proper utilization of masks and adherence to safe social distancing protocols, both crucial elements for human self-preservation, bear a resemblance to the exploration and exploitation components observed in optimization algorithms. The present study employs mathematical simulation to model self-protection behavior and proposes an optimization algorithm. The evaluation and comparison of the performance of the CMPA are conducted in relation to other metaheuristic optimizers currently at the forefront of the field. This is achieved through three truss design problems. The statistical findings suggest that the CMPA exhibits greater competitiveness in comparison to the state-of-the-art algorithms. In summary, this paper makes the following contributions:

- The performance of eight metaheuristic algorithms was compared with each other on three design optimization problems.
- Three space truss structures are used for evaluating the mentioned algorithms.
- Statistical results and convergence curves compared different methods.
- The proposed SPO algorithm was found to be superior to other mentioned algorithms.

Metaheuristic algorithms, as many researchers have discovered, are helpful for designing and analyzing truss structures with many members. However, because the effectiveness of optimization algorithms varies depending on the problem type, not every algorithm performs efficiently in every case. The effectiveness of recently developed algorithms should be evaluated and compared by applying them to various real-world engineering problems. As a contribution to the literature, the most recently described optimization algorithms with state-of-the-art technology for truss structure optimization are considered in this study. Furthermore, this study contributes to the literature by comprehensively comparing all novel optimization methods utilized to solve challenging and real-world engineering problems.

The remaining sections of this paper are organized as follows: The eight meta-heuristic techniques used are briefly described in Section 2. Section 3 precisely addresses what the optimization problems consist of. Section 4 offers three examples of benchmark truss constructions. The paper ends with the final section.

2. Meta-heuristic algorithms

In this paper, eight population-based meta-heuristic optimization techniques are used to reduce the weight of truss structures. These algorithms consist of African Vultures Optimization Algorithm (AVOA) [46], Flow Direction Algorithm (FDA) [47], Arithmetic Optimization Algorithm (AOA) [48], Generalized Normal Distribution Optimization (GNDO) [49], Stochastic Paint Optimizer (SPO) [9], Chaos Game Optimizer (CGO) [10], Crystal Structure Algorithm (CRY) [50], Material Generation Algorithm (MGA) [51]. These algorithms are shown in Fig. 1. Three well-known benchmark truss structures were optimized using continuous size variables to test the effectiveness of the mentioned meta-heuristics.

Many algorithms are available for analyzing a specific problem, and determining which is best suited for the task at hand can be tricky. We may learn about the strengths and shortcomings of various algorithms by comparing their solution to a given problem. This information can guide us in making suitable selections. The motivation behind optimizing truss structures using new methods lies in the pursuit of finding more efficient and effective solutions. New optimization techniques may have limitations in terms of computational complexity, convergence speed, or solution quality. Therefore, we are motivated to explore new meta-heuristic algorithms, to overcome these limitations and improve the optimization process for future application.

Some of the mentioned algorithms are parameter-independent — this means that they can be applied to a wide range of problems without being limited by the specific parameters that might be assigned. It makes them easy to use and versatile. Additionally, they are the most recent optimization algorithms presented in the literature. To ensure a fair comparison, the experiments are conducted on the same device, and the initial parameters as the number of population and iterations, are the same for all methods. The algorithm's parameters were upgraded and modified based on each example in this work.

2.1. African Vultures Optimization Algorithm

Abdollahzadeh et al. [46] developed African Vultures Optimization Algorithm (AVOA) in 2021. Because vultures feed on carrion and prefer dead bodies, which many hunters do not prefer, they are called nature's scavengers. Many of their adaptations to life have developed in this direction. For example, the fact that vultures are hairless or short-haired serves to stabilize the temperature and an adaptation process that develops to avoid problems cleaning the feathers due to food residues on the head. The grasping claw seen in many birds of prey is differentiated in vultures to make it easier for them to walk on carrion on the ground. In addition, the low pH value of their stomachs, reaching up to 1, acts as a barrier that provides many bacteria that can be transmitted from carcasses to die [51]. There are clear differences between vultures, which are grouped under two groups old and new world vultures. For instance, old-world vultures spread in Asia, Africa, and Europe use their sense of sight to search for their food, while new-world vultures distributed in North and South America use the sense of smell, which is generally not significantly improved by birds. The AVOA algorithm, inspired by the feeding strategies of vultures living in Africa, computes the fitness of all individuals in a population of N vultures. The fitness values of vultures are denoted by F and express their saturation rate. The best and second results are considered the best vultures in the first and second groups, depending on the fitness values.

Consequently, the vultures are divided into two groups, and the best vulture (*R(i)*) is selected for each group. The worst solution belongs to the vulture, which is the most hungry and weak. Other vultures in the population try to get closest to the prey (optimal target) by aiming to get close to the two best vultures and away from the worst vultures. If the F value indicating the saturation rate of the vultures is less than 1, the vultures look for food near the solutions, and the algorithm starts the exploitation stage. If the F value is greater than 1, the vultures can investigate food in varied fields, and the algorithm starts the exploration stage. During the iterations, the F value and position of vultures are updated according to various strategies to find food. This iterates until the stopping standard is met. The pseudo-code of the AVOA is introduced below:

```
Initialization
Set the method parameters N and T Maximum number of iterations.
Initialize the random population $P_i$ (i = 1,2,...,N)
Compute the fitness values of vultures
While t (iteration number) $\leq$ T
Step 1: Set $P_{BestVulture1}$ as the position of the first best vulture and $P_{BestVulture2}$ as the position of the second best vulture.
Step 2: for each vulture ($P_i$) select R(i).
Step 3: for each vulture, calculate F.
Step 4: If $F \geq 1$, start the exploration phase, or if $F < 1$, start the exploitation phase.
Step 5: Update the position of the vultures.
Step 6: Update t
Step 7: Return $P_{BestVulture1}$
End While
```

2.2. Flow Direction Algorithm

Flow Direction Algorithm (FDA) is an optimization method created by Karami et al. [47] by imitating water flow direction in a basin. According to the algorithm, each stream has a position and a height. The flow moves towards one of its neighbors with a specific elevation or slope around it. The D8 method is regarded as one of the common methods for determining the runoff direction. The D8 technique is used to determine the flow direction in the algorithm. This technique supposes that each flow has eight neighbors around it, and the neighbors' fitness functions (heights) are measured. If the best neighbor (with the lowest height) has a better objective function than the current flow, the flow moves to that neighbor with velocity V. The velocity V increases or decreases in direct proportion to the slope. The fitness functions of the current flows are calculated, and if they are better than the earlier flows, the fitness function and the positions of the flows are updated. If the best neighbor's objective function is not better than the current flow, the flow moves in the direction of the general slope. The flow location is updated. It proceeds until the termination condition is met. The pseudo-code of the FDA is introduced below.

```
Initialization:
Set the method parameters: $\alpha$ (number of population), $\beta$ (number of neighbors), $\Delta$ (neighborhood radius)
Initialize the random population
The objective function values of the population are computed, and the best objective function value is assumed as the outlet point.
While iteration number $\leq$ max. iteration
Step 1: Create neighbors.
Step 2: Compute the fitness values of each neighbor and determine the best neighbor.
Step 3: If the objective function value of the best neighbor is less than the current flow, step 4; otherwise, step 5 is performed.
Step 4: Flow velocity is calculated and moves towards the best neighbor.
Step 5: Flow moves in the direction of the dominant slope.
Step 6: The new flow objective function value is calculated. The flow objective function and position are updated if it is better than before.
Step 7: Update iteration number
Step 8: Return the position of the best flow
End While
```

2.3. Arithmetic Optimization Algorithm

Abualigah et al. [48] devised the Arithmetic Optimization Algorithm (AOA), a math-based strategy that emulates the sequence of operations of mathematical operators. The algorithm expresses it by division: D, multiplication: M, subtraction: S, and addition: A. The arithmetic operators are considered from outside to inside (from general to specific), D, M, S, and A come in order. The optimization process begins with the creation of the initial population. The fitness values of the candidate solutions are computed, and the best result is determined. Each mathematical operator tries to get closer to the optimum solution throughout the iterations. The optimum searching solution consists of two stages: exploration and exploitation. A random number is compared with the Math Optimizer accelerated function to decide which phase to choose during iterations in the algorithm. D and M operators are used in the exploration phase. Since the distributions of these operators are high, they cannot easily approach the optimum result. S and A operators are used in the exploitation phase. Since these operators have low dispersion, they can approach the optimum result more efficiently. Finally, the algorithm is stopped when the termination condition is met. The pseudo-code of the AOA is introduced below.

```
Initialization:
Set the method parameters: $\alpha$ (a parameter that determines the accuracy of exploitation), $\mu$ (control parameter used in the search process)
Initialize the candidate solutions
While iteration number $\leq$ max. iteration
Step 1: Calculate the fitness values
Step 2: Determine the best solution
Step 3: Decide which phase to select based on the value of the Math Optimizer accelerated function.
Step 4: Approach the optimum result using the D, M, S, or A operators.
Step 5: Update the locations of the solutions.
Step 6: Update iteration number
Step 7: Return the best solution
End While
```

2.4. Generalized Normal Distribution Optimization

Zhang et al. [49] developed the Generalized Normal Distribution Optimization (GNDO), inspired by the normal distribution (Gaussian distribution) theory. A normal distribution has two parameters, the position parameter, which expresses the mean value of the random variables, and the scale parameter, which defines the standard variance. In

GNDO, the locations of all individuals are assumed as random variables to fit the normal distribution criteria. In the beginning iterations, the average positions of the individuals and the best position are further apart. In the following iterations, it aims to bring the average position and best position closer to each other and gradually reduce the standard variance of the locations of all solutions to the lowest. GNDO consists of local exploitation and global exploration phases. The possibility of being selected for both phases is equal. In local exploitation, the best position in the population (best solution) and the average position of the population are calculated.

Regarding these, the generalized average positions of each individual are computed. Once the generalized standard variance and penalty factor are found, the individuals' positions are updated depending on the local exploitation strategy. In local exploitation, the best and average positions of the population are calculated. The aim is to move individuals in the population in the direction between these two positions. Therefore, once calculating the generalized average position, generalized standard variance, and penalty factor for each individual, the positions of the individuals are updated according to the local exploitation strategy. Depending on the standard normal distribution strategy, solution spaces are investigated globally in the global exploration phase. The positions of individuals are updated to find a better *solution*. The pseudo-code of the GNDO is introduced below:

```
Initialization:
Initialize population (N = number of the population)
Calculate the fitness values of individuals and determine the best individual.
While iteration number ≤ max. iteration
Step 1: Generate a random value and decide which phase to select based on the value
Step 2: Update the position of individuals based on the strategy of the selected stage
Step 3: Update iteration number
Step 4: Return the best solution
End While
```

2.5. Stochastic Paint Optimizer

Kaveh et al. [9] proposed the Stochastic Paint Optimizer (SPO) algorithm based on the science of using color. The clustering of colors in the algorithm is performed according to the color wheel based on color theory as follows. Primary Colors: these are considered the best colors and from which all other colors are derived (red, yellow, and blue). Secondary Colors: the second-best colors (orange, purple, etc.) are obtained by mixing the primary colors. Tertiary Colors: the worst colors (yellow–violet, blue–orange, etc.) are formed when primary and secondary colors are mixed. All paints' initial colors are randomly determine in the algorithm's first step. Here, the objective function optimizes the beauty index corresponding to the dyes. As mentioned above, the colors are organized into groups based on their objective function values. New colors are created using analog, complementary, triple, or quadruple combination techniques. Among the new colors created the ones with better objective function values than the previous colors are selected, and the others are left. This cycle continues until the termination condition is met. The pseudo-code of the SPO is introduced below

```
Initialization:
Initialize colors of all paints
While iteration number ≤ max. iteration
Step 1: Find objective function values of colors
Step 2: Sort paints
Step 3: Create groups
Step 4: Create new colors of paints
Step 5: Evaluate and update new paints
Step 6: Update iteration number
Step 7: Return the best solution
End While
```

2.6. Chaos Game Optimization

Talatahari and Azizi [10] developed Chaos Game Optimizer (CGO) inspired by chaos game theory. Chaos, as a word, means complexity or disorder. Chaos theory is the science of predicting the behavior of "naturally unpredictable" systems. Chaotic systems are a close mixture of order and chaos, exhibiting unpredictable and chaotic behavior from the outside but similar in their inner workings. In mathematics, the chaos game generates fractals using a polygon form and a random starting point. Fractal is the common name for complex geometric shapes that often show similarity within themselves. To create a fractal, the polygon's vertices, which will be the principal form of the fractal, are located, and a random starting point is chosen. The next point to be created is assigned as a fraction of the length between the starting point and the randomly selected vertices of the polygon. This process is repeated in each iteration, creating new points and, thus a fractal. Suppose this process is performed so that the polygon has three vertices and a factor of 1/2, a Sierpinski triangle is formed. In the algorithm, the Sierpinski triangle is determined as the search area, and some suitable points (initial solution) are randomly located in the search area. The self-similarity of these solutions can be expressed as fitness. The points with the best and worst fitness values are the best and worst solutions, respectively. Once evaluating the suitability of the initial points in the algorithm, different suitable points are searched within the search area. For this purpose, temporary triangles are drawn for each appropriate point. The position of the relevant point forms the vertices of the triangles, the position of the best point, and the mean position of some chosen random point that has a probability of containing the point of interest. Four new points are created for each of the temporary triangles. It is checked whether the newly created points are outside the borders, and the fitness values of the new points within the borders are calculated. New points replace existing points with the worst fitness values with better fitness values. The pseudo-code of the CGO is introduced below

```
Initialization:
Create random positions of initial points
Evaluate the fitness values for each point
While iteration number ≤ max. iteration
Step 1: Create temporary triangles for each point
Step 2: Create four new points from temporary triangles
Step 3: Control the location restraints for new points and change it
Step 4: Evaluate the fitness values of new points
Step 5: If the fitness value of the new points is better than the fitness value of the previous points, replace the previous points with new ones.
Step 6: Update iteration number
Step 7: Return the best point (solution)
End While
```

2.7. Crystal Structure Algorithm

Talatahari et al. [50] introduced the Crystal Structure Algorithm (CRY), inspired by the formation of crystal structures. Depending on the way the atoms are arranged, the properties of materials change. The structure formed by the three-dimensional arrangement of the atoms of solid materials according to a certain geometric order is called crystal structure. The smallest repeated volumetric unit of the crystal structure is called the basis. A crystal lattice is formed by arranging several bases side by side. In the algorithm, each solution is considered a crystal in space. Primarily, a set of crystals whose initial positions are randomly determined in the search space is generated. All the crystals at the corners are considered main crystals in the randomly generated candidate solution, $\mathrm{Cr}_{\mathrm{main}}$. The fitness value of each of the crystals is determined. The crystal with the best configuration and the mean value of the crystals are considered as Cr_{b} and F_{c}, respectively.

New candidate solutions with four types of update operations regarding these values are created. It is checked whether the positions of new candidate solutions are within limits. The fitness values of the new candidate solutions within the bounds are calculated. Global Best is updated if a better result than previous solutions is found. The pseudo-code of the CGO is introduced below

```
Initialization:
Create random positions for initial crystals
Evaluate the fitness values for each crystal
While iteration number ≤ max. iteration
Step 1: Create $Cr_{main}$ and, based on it, create new crystals
Step 2: Create $Cr_b$ and, based on it, create new crystals
Step 3: Create $F_c$ and, based on it, create new crystals
Step 4: Create new crystals based on $Cr_b$ and $F_c$ values
Step 5: Check the location restraints for new crystals and change it
Step 6: Assess the fitness values for new crystals
Step 7: Replace Global Best if a better fitness value is found
Step 8: Update iteration number
Step 9: Return Global Best
End While
```

2.8. Material Generation Algorithm

A physical change occurs only in the external appearance without changing the structure of the products. No new products are formed as a result of physical changes. Chemical changes are changes that occur in the internal structure of materials. As a result of chemical changes, the identity of the material changes, and new materials are formed. Talatahari et al. [51] developed the Material Generation Algorithm (MGA) inspired by new material generation. The algorithm determines initial materials consisting of elements whose positions are randomly determined in the search space. Fitness values showing the chemical stability of these materials are calculated. Chemical stability is the resistance of a material to changes (rusting, deterioration, etc.) due to internal and external effects. New materials are produced using the assumptions of chemical compounds and reactions. Once checking the boundary conditions of new materials are, the fitness values are calculated. The ones with the worst fitness value are replaced with new materials with better fitness values from the previous materials. Global best value is updated accordingly. The pseudo-code of the CGO is introduced below

```
Initialization:
Create random positions for initial materials
Evaluate the fitness values for each material
While iteration number ≤ max. iteration
Step 1: Generate new material using the concept of chemical components
Step 2: Generate new material using the concept of chemical reaction
Step 3: Check the boundary conditions of new materials
Step 4: Evaluate the fitness values for new materials
Step 5: Replace materials with the worst fitness values with new materials with better fitness values
Step 6: Replace Global Best if a better fitness value is found
Step 7: Update iteration number
Step 8: Return Global Best
End While
```

3. Truss structures definition

Mathematical formulas utilized in the size optimization of this study are found here. It is necessary to achieve ideal member cross-section (A_i) values in order to reduce the structural weight W resulting in the optimized truss structure size. In addition, this minimal design must meet the constraints on the size of design variables and structural reactions. The following is the optimum design problem:

$$\begin{aligned} &Minimize && W(\{A_i\}) = \sum_{i=1}^{n} \gamma_i . A_i . L_i && \\ &Subject && \delta_{\min} \le \delta_i \le \delta_{\max} && i = 1,2,\dots,m \\ & && \sigma_{\min} \le \sigma_i \le \sigma_{\max} && i = 1,2,\dots,n \\ & && \sigma_i^b \le \sigma_i^i \le 0 && i = 1,2,\dots,ns \\ & && A_{\min} \le A_i \le A_{\max} && i = 1,2,\dots,ng \end{aligned} \tag{1}$$

where the structure's weight is defined by $W(\{A_i\})$, n denotes the number of members, m represents the number of nodes, the number of compression elements is determined by ns, ng represents the number of design variables or the number of member groups. Furthermore, the material density of the member i is defined by γ_i, L_i denotes to the length of the member i, A_i is the cross-sectional area of the member i chosen between $A_{\min}$ and $A_{\max}$, σ_i and δ_i are the stress and nodal deflection, respectively, σ_i^b denotes the allowable buckling stress in the member i when it is in compression. The following equation is described as a penalty function:

$$f_{penalty}(X) = (1 + \varepsilon_1 . v)^{\varepsilon_2} \qquad v = \sum_{i=1}^{n} \max[0, v_i] \tag{2}$$

where v is the total amount of the constraints violated, and constants ε_1 and ε_2 are selected considering the exploration and the exploitation rate of the search space. In this case, ε_1 is set to 1, ε_2 is chosen to minimize penalties and reduce cross-sections. At the beginning of the search process, ε_2 is set to 1.5 and is subsequently increased to 3.

4. Results and discussion of structural examples

Three truss structural problems with continuous design variables are optimized in this Section utilizing AVOA [51], FDA [47], AOA [48], GNDO [49], SPO [9], CGO [10], CRY [50], and MGA [51]. Structure optimization aims to decrease the weight of the structure by identifying the optimal element cross-section areas to fulfill the constraints. All runs were done on a MacBook Pro for truss problems with a CPU 2.3 GHz (8-Core an Intel Core i9 computer platform) and 16 GB RAM on a Macintosh computer, and the code was written in MATLAB (macOS Monterey). The truss structures are a 25-member truss structure, a 72-member truss structure, and a 120-member dome structure. This research aims to compare the algorithms' susceptibility and robustness regarding convergence and stability. All trusses are subjected to linear static analysis. Algorithms for each truss structure perform 30 independent runs, and 10,000 analyses are considered a stopping condition for the algorithms. The beginning population is generated randomly for each run, and designs are created. Optimized results, convergence curves, displacement ratios, element stresses, average weight, and standard deviations are provided to compare the performance of meta-heuristic methods.

4.1. The 25-member truss structure

The 25-member truss structure is the first example of this paper. The 25-member space truss under consideration is depicted in Fig. 2. There are eight distinct groups of structural components, each with its unique set of material and cross-sectional properties (See Table 1). Table 2 illustrates the truss's two different load scenarios.

Groups are based on the number of participants in each group, as shown in Table 2. Each node was regulated by a maximum displacement restriction of ± 0.35 in in each direction, and compressive and tensile stress limitations for each group are shown in Table 3. The range of cross-sectional areas is from 0.01 to 3.4 in^2. The material (Aluminum) has a modulus of elasticity of 10 000 ksi and a mass density of 0.1 lb/in^3.

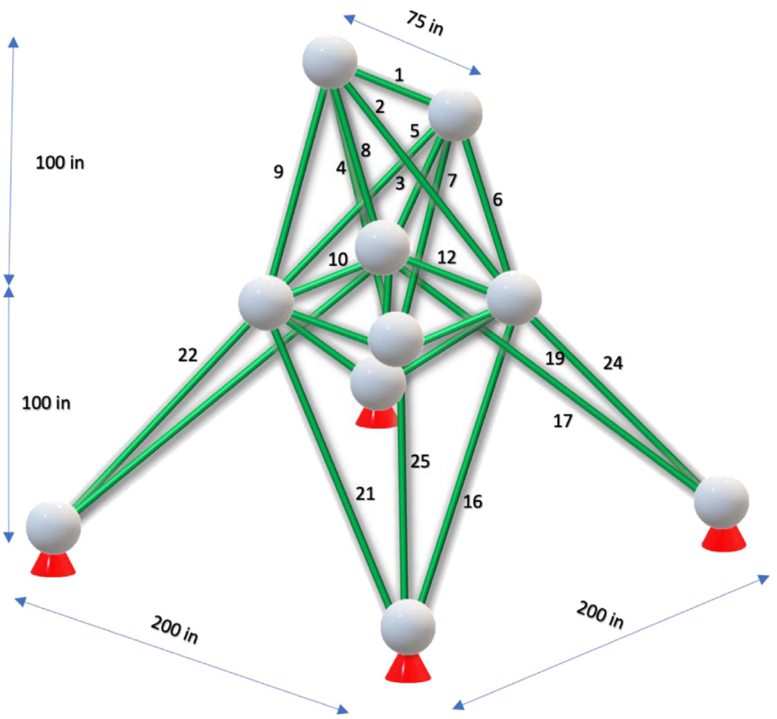

Fig. 2. The 25-member truss design structures.

Table 1
The 25-member truss group number.

Element group number							
1	2	3	4	5	6	7	8
1:(1,2)	2:(1,4)	6:(2,4)	10:(6,3)	12:(3,4)	14:(3,10)	18:(4,7)	22:(10,6)
	3:(2,3)	7:(2,5)	11:(5,4)	13:(6,5)	15:(6,7)	19:(3,8)	23:(3,7)
	4:(1,5)	8:(1,3)			16:(4,9)	20:(5,10)	24:(4,8)
	5:(2,6)	9:(1,6)			17:(5,8)	21:(6,9)	25:(5,9)

Table 2
The loading scenario for 25-member truss.

Node number	Load (kips)					
	Case1			Case 2		
	P_x	P_y	P_z	P_x	P_y	P_z
1	0	20	−5	1	10	−5
2	0	−20	−5	0	10	−5
3	0	0	0	0.5	0	0
6	0	0	0	0.5	0	0

Table 3
Compressive and tensile stress limitation for 25-member truss.

Element group		Compressive stress limitations Ksi (MPa)	Tensile stress limitations Ksi (MPa)
1	A1	35.092 (241.96)	40.0 (275.80)
2	A_2~A_5	11.590 (79.0913)	40.0 (275.80)
3	A_6~A_9	17.305 (119.31)	40.0 (275.80)
4	A_{10}~A_{11}	35.092 (241.96)	40.0 (275.80)
5	A_{12}~A_{13}	35.092 (241.96)	40.0 (275.80)
6	A_{14}~A_{17}	6.759 (46.603)	40.0 (275.80)
7	A_{18}~A_{21}	6.959 (47.982)	40.0 (275.80)
8	A_{22}~A_{25}	11.082 (76.410)	40.0 (275.80)

Table 4 shows the optimal designs for the 25-member truss structures generated by the algorithms. The results indicate that the optimum designs of SPO, GNDO, and FDA are incredibly close to each other, with SPO being the most successful algorithm. AVOA is also among the highest-performing algorithms, behind them by a slight margin in terms of best and average weights. AOA, MGA, CRY, and CGO have yielded design results that are, on average, 0.9%, 5.5%, 6%, and 7.3% heavier than the optimum design, respectively. In general, the best performing algorithms are SPO, GNDO, and FDA. Based on their standard deviations, the best algorithms for stability are SPO and GNDO.

As illustrated in Fig. 3, where the convergence graphs are compared for the best run of each algorithm, SPO is the algorithm that can get the optimal outcome (545.0375 lb); it has designed with the fewest number of analyzes. In terms of convergence rates, GNDO and FDA rank second and third, respectively, after SPO. CGO is the algorithm with the worst convergence performance, needing the most analyses to identify the best solution among the algorithms.

Fig. 4 illustrates convergence curves that plot the average fitness of 30 runs versus the number of iterations for all algorithms. Consequently, SPO converges to the optimal outcome first, followed by GNDO, FDA, and AVOA, in the same order as the best-run curves. While AOA shows average performance in terms of convergence speed compared to other algorithms, CRY, MGA, and CGO are the algorithms that need the most analysis to reach the optimum result.

Fig. 5 illustrates the displacement ratio of the 25-member truss evaluated at the best design optimized using the SPO algorithm. The displacement ratio for both load cases is lower than the maximum ratio. Fig. 6 depicts the element stress of a 25-member truss designed with the best run of SPO. According to this figure, the stresses of two elements for both load cases are at the lower limit, while the stresses of the remaining elements are between the upper and lower limits. This means that displacement ratio and element stress constraints control the design of this example.

Fig. 7 depicts the weights, average weights, and best weights obtained by all algorithms in all runs ranging from 1 to 30. According to the figure, the average and best weights discovered by the SPO algorithm for the 25-member truss design problem are very close compared to other algorithms. The standard deviation is the lowest, and the stability is the highest. In terms of stability, the algorithm that follows SPO is GNDO, and the best weight it finds after SPO, and its average weight are the closest to each other.

Table 4
The results of different methods for 25-member truss structure.

Member group	AVOA	FDA	AOA	GNDO	SPO	CGO	CRY	MGA
1 (A_1)	0.0750	0.0101	0.0438	0.0101	0.0101	0.9446	1.1237	1.1384
2 (A_2~A_5)	2.1802	2.0522	2.0874	2.0494	2.0461	2.8865	2.0030	2.3953
3 (A_6~A_9)	2.8366	3.0044	3.1081	2.9964	2.9959	2.4519	3.0834	2.7725
4 (A_{10}~A_{11})	0.0100	0.0100	0.0136	0.0100	0.0100	0.0100	0.0293	0.0460
5 (A_{12}~A_{13})	0.0102	0.0100	0.0100	0.0101	0.0100	1.1425	0.2488	0.1176
6 (A_{14}~A_{17})	0.6640	0.6810	0.7799	0.6837	0.6836	0.6278	0.7281	0.8442
7 (A_{18}~A_{21})	1.5940	1.6144	1.5679	1.6204	1.6234	1.4653	1.8112	1.7904
8 (A_{22}~A_{25})	2.7442	2.6763	2.5909	2.6735	2.6732	2.8828	2.4131	2.4748
Best weight (lb.)	545.9610	545.0414	550.8519	545.0388	**545.0375**	585.3567	578.7504	577.4840
Average weight (lb.)	549.3792	545.2373	575.1403	545.0650	**545.0625**	614.3526	615.8074	610.7876
Standard deviation	4.4424	0.1746	13.3803	0.0281	**0.0181**	14.3608	20.6859	15.3682
Number of analyses	10 000	10 000	10 000	10 000	10 000	10 000	10 000	10 000

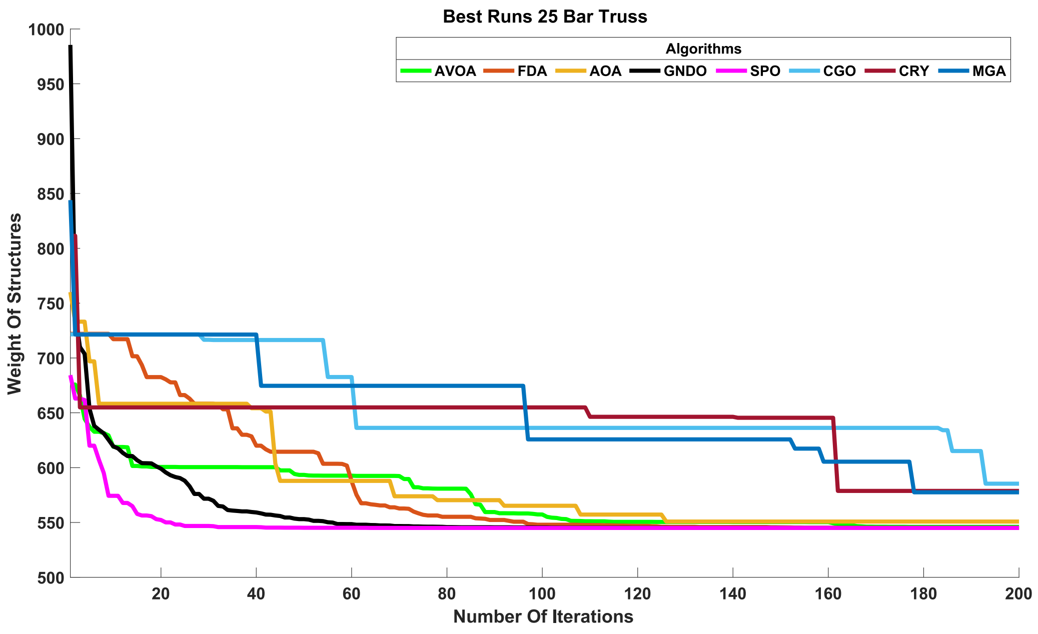

Fig. 3. The best runs convergence curves for 25-member truss design structures.

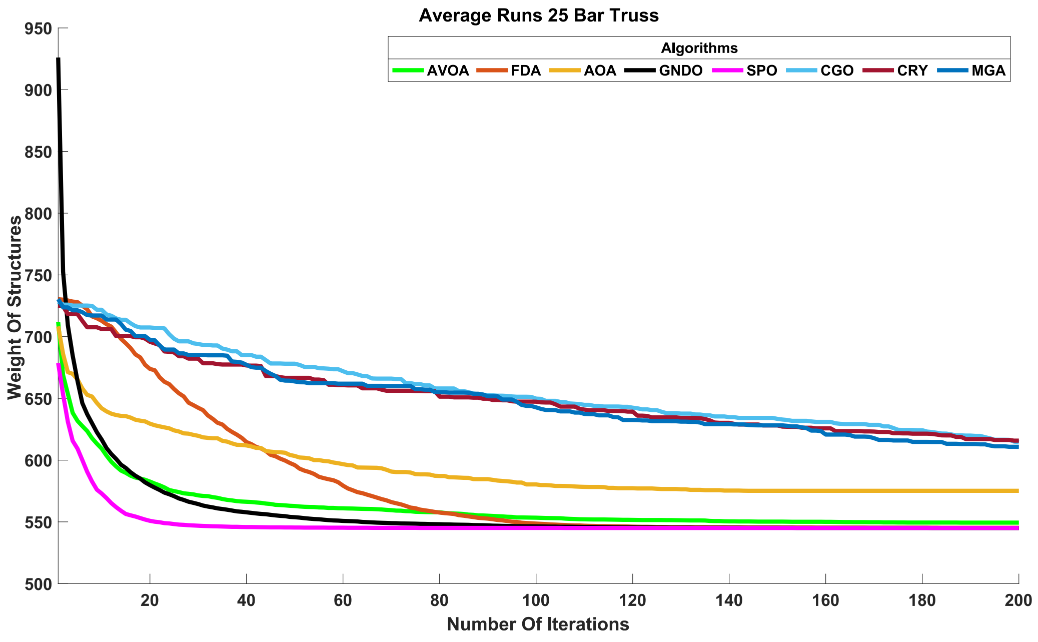

Fig. 4. The mean runs convergence curves for 25-member truss design structures.

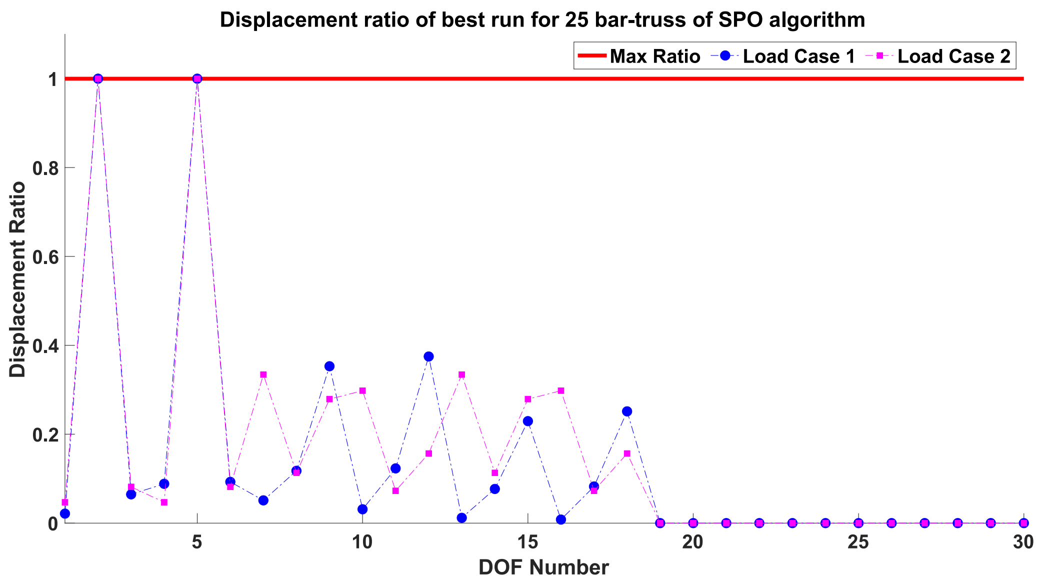

Fig. 5. The displacement ratio of SPO for 25-member truss design structures.

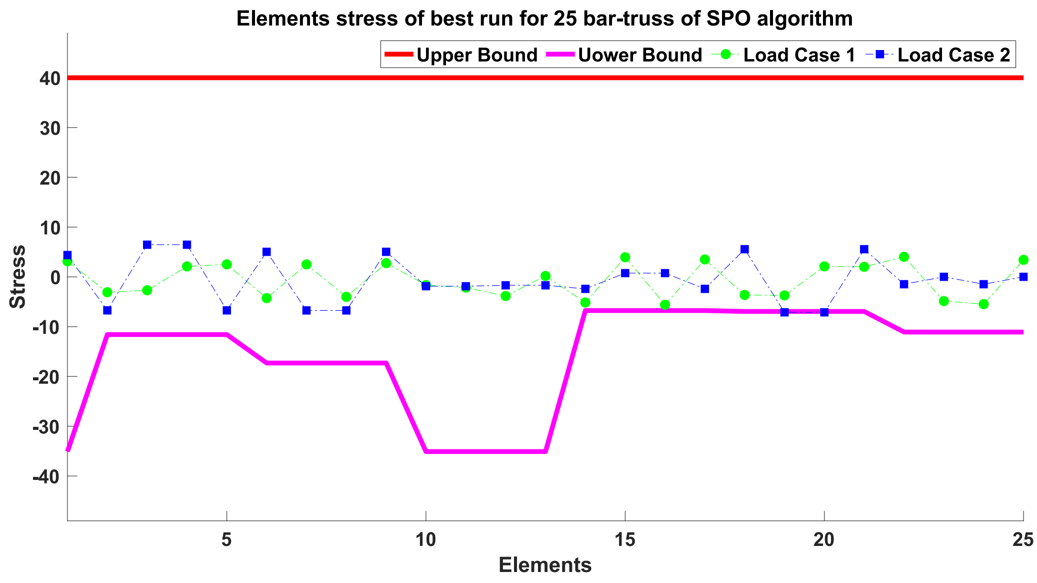

Fig. 6. The elements stress of SPO for 25-member truss design structures.

4.2. The 72-member truss structure

An example of a 72-member space truss as the second example is depicted in Fig. 8. The material has a density of 0.1 lb/in^3 and a modulus of elasticity of 10 000 ksi. This example's components are organized into 16 distinct groups. The maximum allowable stress is 25 ksi for both the tensile and compressive stress. The top node displacement is less than 0.25 in in both x and y directions. Each member can have a cross-sectional area between 0.10 in^2 and 4.00 in^2 at the most. For the two alternative space truss load scenarios, the values and directions are shown in Table 5.

The optimum design results for the 72-member truss structures provided by the algorithms are shown in Table 6. SPO is the algorithm from the results that calculates the lowest weight (379.6338 lb), while the FDA is incredibly near to it. AVOA and GNDO are two other algorithms that closely follow FDA. AOA, MGA, CGON, and CRY design structures 34.3%, 80.8%, 81.4%, and 101.58% heavier, respectively, than the optimum weight. SPO is the algorithm that has exhibited the best general performance in terms of both best and average weight.

Table 5
Loading condition for 72-member truss.

Node number	Load (kips)					
	Case1			Case 2		
	P_x	P_y	P_z	P_x	P_y	P_z
17	5	5	−5	0	0	−5
18	0	0	0	0	0	−5
19	0	0	0	0	0	−5
20	0	0	0	0	0	−5

Furthermore, it also has the slightest standard deviation. In contrast, CRY has the worst performance overall, with both best and average weight below the rest of the algorithms used. Additionally, CGO and MGA have the highest standard deviation.

SPO is the most effective at reaching an optimal solution with the fewest number of analyzes (4150), as shown in Fig. 9 AVOA, GNDO, and FDA all follow SPO in terms of best runs' convergence rates. CGO, CRY, and MGA need more analysis than other algorithms to determine

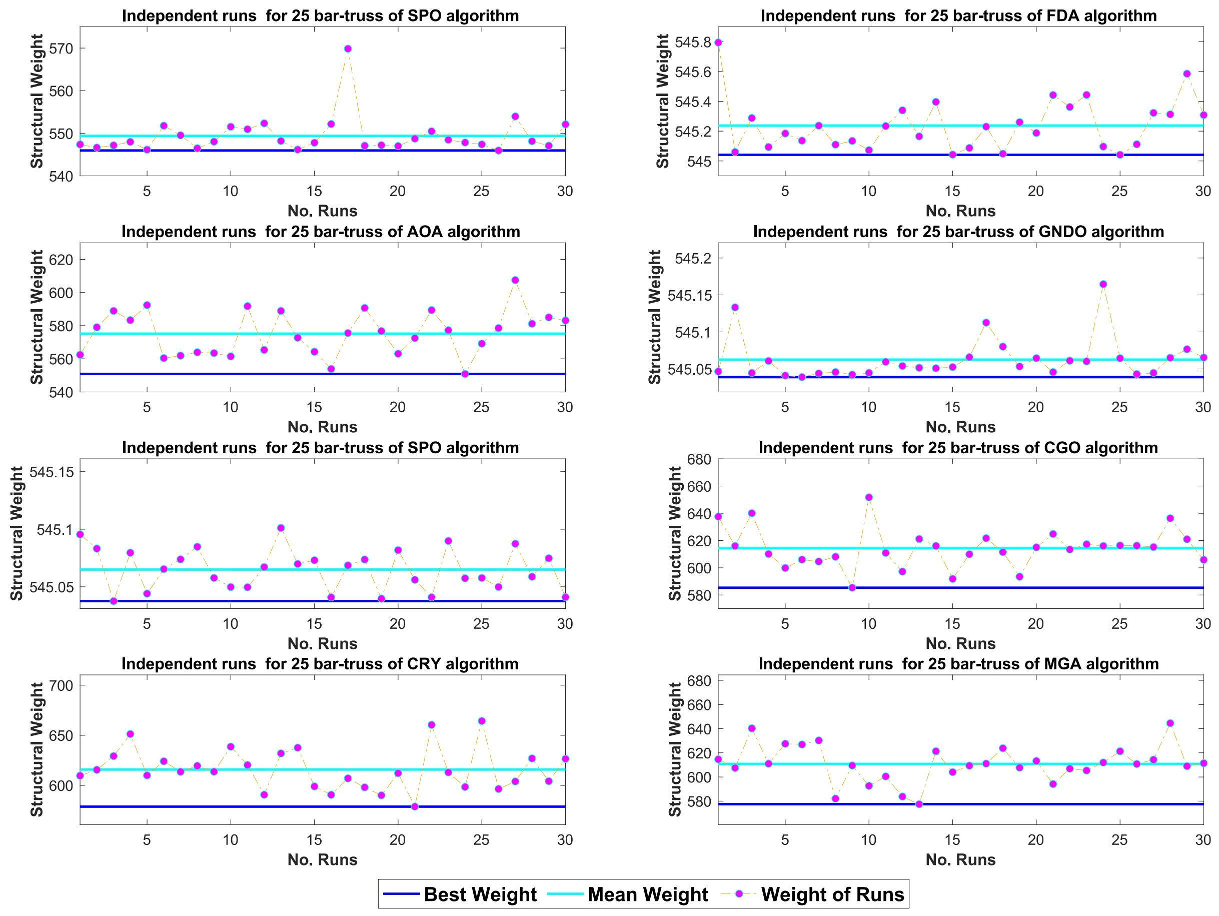

Fig. 7. The 30 independent runs for 25-member truss design structures of all algorithms.

Table 6
The results of different methods for 72-member truss structure.

Member group	AVOA	FDA	AOA	GNDO	SPO	CGO	CRY	MGA
1 ($A_{1\sim}A_4$)	1.9592	1.9353	1.5500	1.8632	1.8957	1.2740	2.4438	2.4866
2 ($A_{5\sim}A_{12}$)	0.4993	0.5102	0.8425	0.4826	0.5131	1.2747	0.7430	1.0832
3 ($A_{13\sim}A_{16}$)	0.1001	0.1000	0.1000	0.1001	0.1000	0.8114	2.0054	0.1000
4 ($A_{17\sim}A_{18}$)	0.1000	0.1003	0.1500	0.1004	0.1000	1.3423	1.6212	0.1000
5 ($A_{19\sim}A_{22}$)	1.2984	1.2646	3.1108	1.1619	1.2659	0.8551	1.6537	1.3842
6 ($A_{23\sim}A_{30}$)	0.5067	0.4932	0.7325	0.5135	0.5094	0.6696	0.4577	0.8587
7 ($A_{31\sim}A_{34}$)	0.1000	0.1017	0.1000	0.1000	0.1001	0.7345	0.1331	0.2995
8 ($A_{35\sim}A_{36}$)	0.1000	0.1000	0.4192	0.1010	0.1000	0.6961	1.0755	1.2563
9 ($A_{37\sim}A_{40}$)	0.5601	0.5163	0.3188	0.5322	0.5245	2.2399	0.7833	0.4926
10 ($A_{41\sim}A_{48}$)	0.5208	0.5059	0.6582	0.5263	0.5162	0.3182	0.5435	0.3130
11 ($A_{49\sim}A_{52}$)	0.1001	0.1003	0.2142	0.1003	0.1000	0.1000	0.1000	0.2062
12 ($A_{53\sim}A_{54}$)	0.1025	0.1061	0.2332	0.1000	0.1000	1.1225	0.2479	2.8108
13 ($A_{55\sim}A_{58}$)	0.1556	0.1557	0.6607	0.1546	0.1564	0.5152	2.3697	1.6969
14 ($A_{59\sim}A_{66}$)	0.5674	0.5512	0.6663	0.5848	0.5486	0.7660	0.6659	0.5794
15 ($A_{67\sim}A_{70}$)	0.3909	0.4273	0.1000	0.4105	0.4094	0.7315	1.0100	1.2147
16 ($A_{71\sim}A_{72}$)	0.4922	0.6038	0.3353	0.6168	0.5637	0.7379	2.2596	0.2328
Best weight (lb.)	380.2781	380.0084	509.8685	380.5450	**379.6338**	688.7320	765.2620	686.3712
Average weight (lb.)	383.5203	381.9343	626.4494	388.9339	**379.6749**	856.2211	873.9106	858.4968
Standard deviation	2.9270	1.2127	57.8632	8.6151	**0.0298**	78.8107	50.3802	76.0597
Number of analyses	10 000	10 000	10 000	10 000	10 000	10 000	10 000	10 000

the optimal weight and thus have a bad history of convergence. Fig. 10, which shows convergence plots of the best run of each algorithm, yields the same findings as Fig. 11, which shows convergence plots of the average run. SPO is the most robust algorithm in terms of convergence performance, while CGO, CRY, and MGA are the worst.

Fig. 11 depicts the displacement ratio of the 72-member truss at the best design optimized with the SPO method. The displacement ratio reaches a maximum of 99.96% in the final DOF numbers. The element stress of a 72-member truss designed with the best run of SPO is visualized in Fig. 12. The four-element stresses in load case 2 and two element stress in load case 1 are at the lowest limit, while all other element stresses are between the boundaries, as shown in this figure. It is clear that both displacement ratio and element stress constraints control the design of this example.

Fig. 13 shows the weights, average weights, and best weights achieved by all algorithms in all runs from 1 to 30. The average

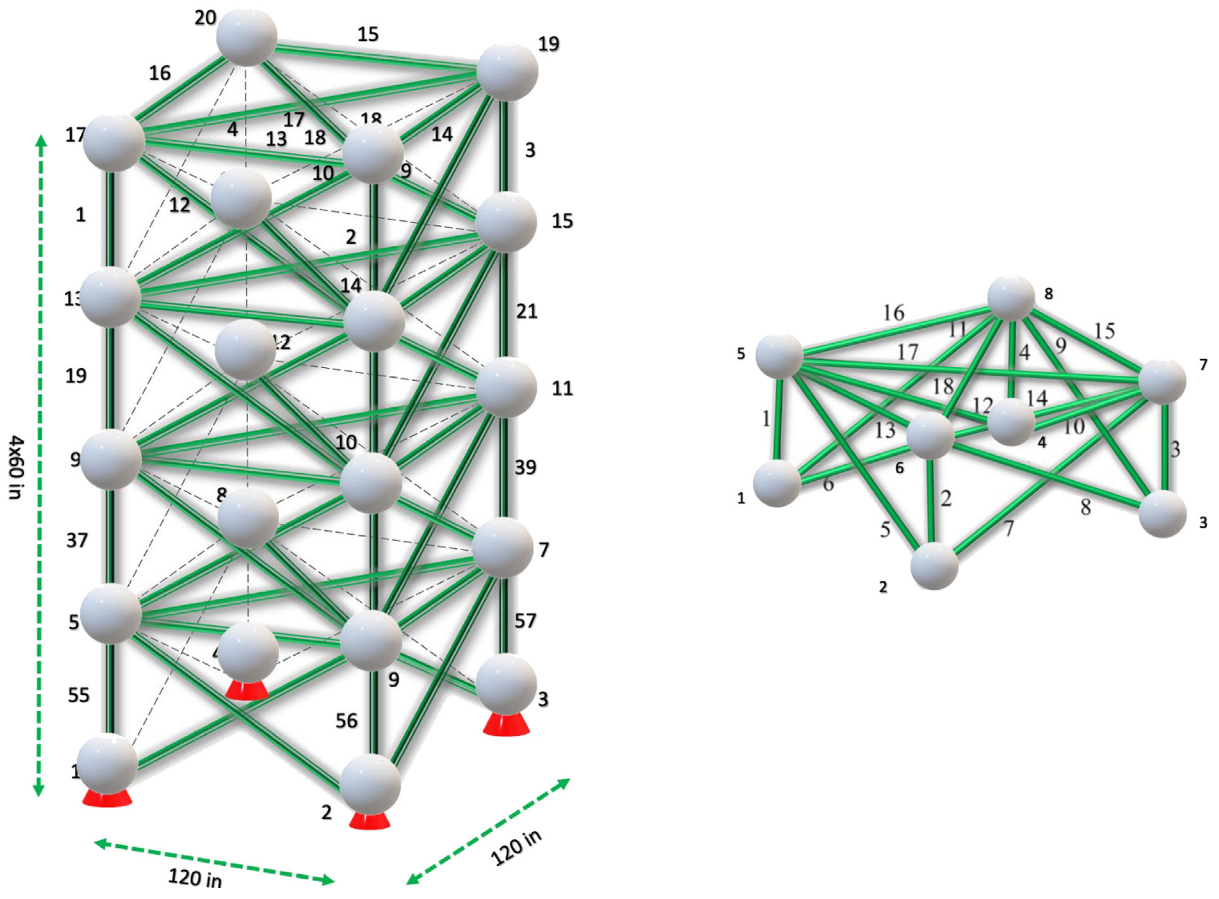

Fig. 8. The 72-member truss design structures.

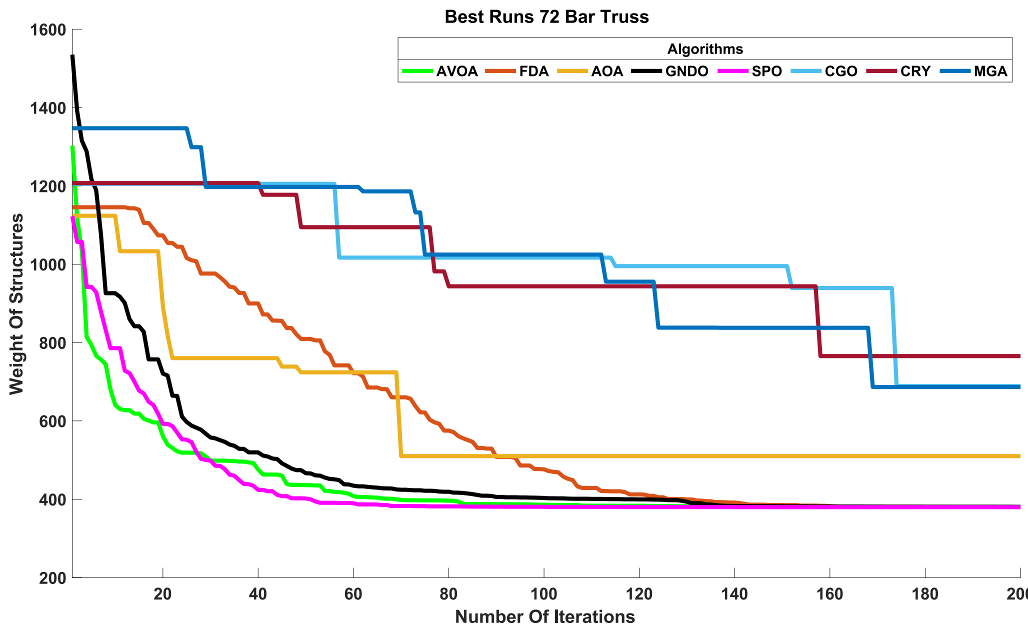

Fig. 9. The best runs convergence curves for 72-member truss design structures.

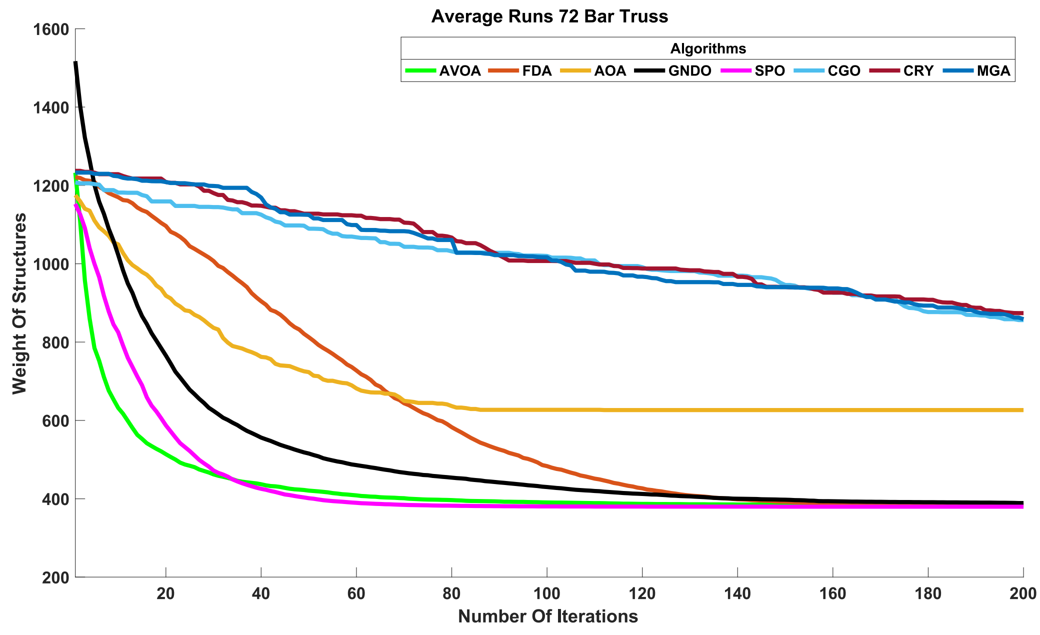

Fig. 10. The mean runs convergence curves for 72-member truss design structures.

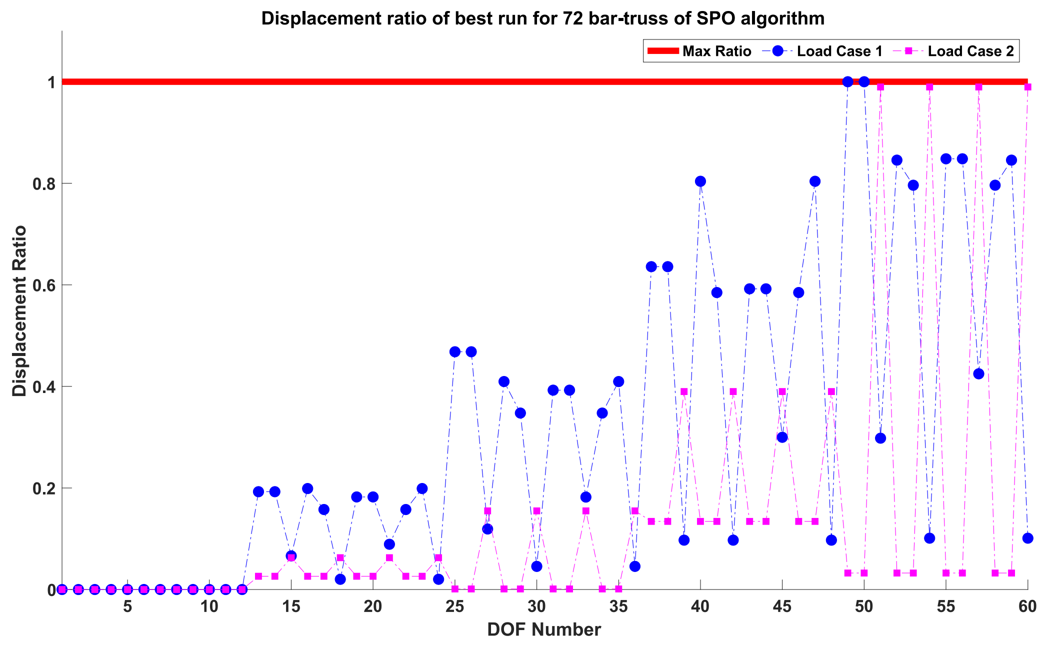

Fig. 11. The displacement ratio of SPO for 72-member truss design structures.

weight and best weight discovered by the SPO algorithm for the 72-member truss design problem are quite close when compared to other algorithms, as seen in this figure, and hence the stability is the highest.

4.3. *The 120-member dome structure*

The 120-member dome structure was depicted as the third example of size optimization in this study as Fig. 14. The material has a density of 0.288 $\mathrm{lb/in^3}$ and a modulus of elasticity of 30 450 ksi. The yield stress of steel is taken as 58 ksi. The dome is considered to be subjected to vertical loading at all the unsupported joints as follows.

I. The load of 13.49 kips was employed at node 1.
II. The load of 6.744 kips was employed at nodes 2 to 14.
III. The load of 2.248 kips was employed at the rest of the nodes.

Assuming symmetry about the z-axis, the components are divided into seven groups. 20 $\mathrm{in^2}$ is the greatest cross-sectional area, with a minimum of 0.775 $\mathrm{in^2}$. The allowable tensile and compressive stresses are set according to the AISC ASD code.

Table 7 provides the optimum design results for the 120-member dome structures. SPO algorithm produces the lightest weight solution, and GNDO, FDA, and AVOA are the most successful in finding a lighter structure after SPO. From Table 7, SPO is the best-performing algorithm in terms of minimum weight (best) or maximum variation between different designs' weights (average), meaning it has found at least one structure with either the smallest best or slightest average difference in weight compared to other algorithms considered here. MGA has the worst performance with a design that is 16.99% heavier than SPO in terms of best weight, and CGO has the worst performance with a

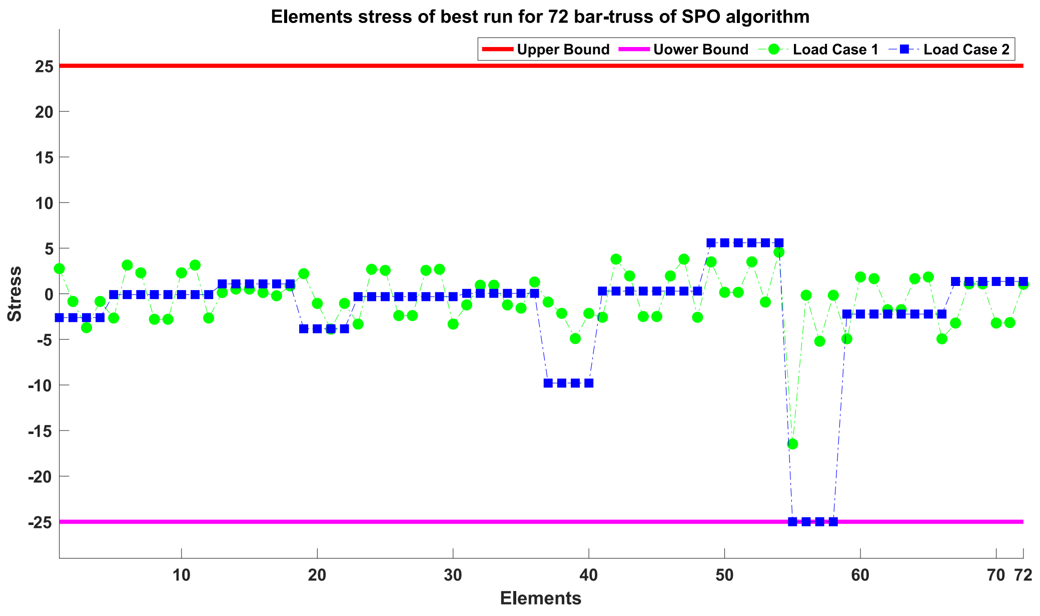

Fig. 12. The element stress of SPO for 72-member truss design structures.

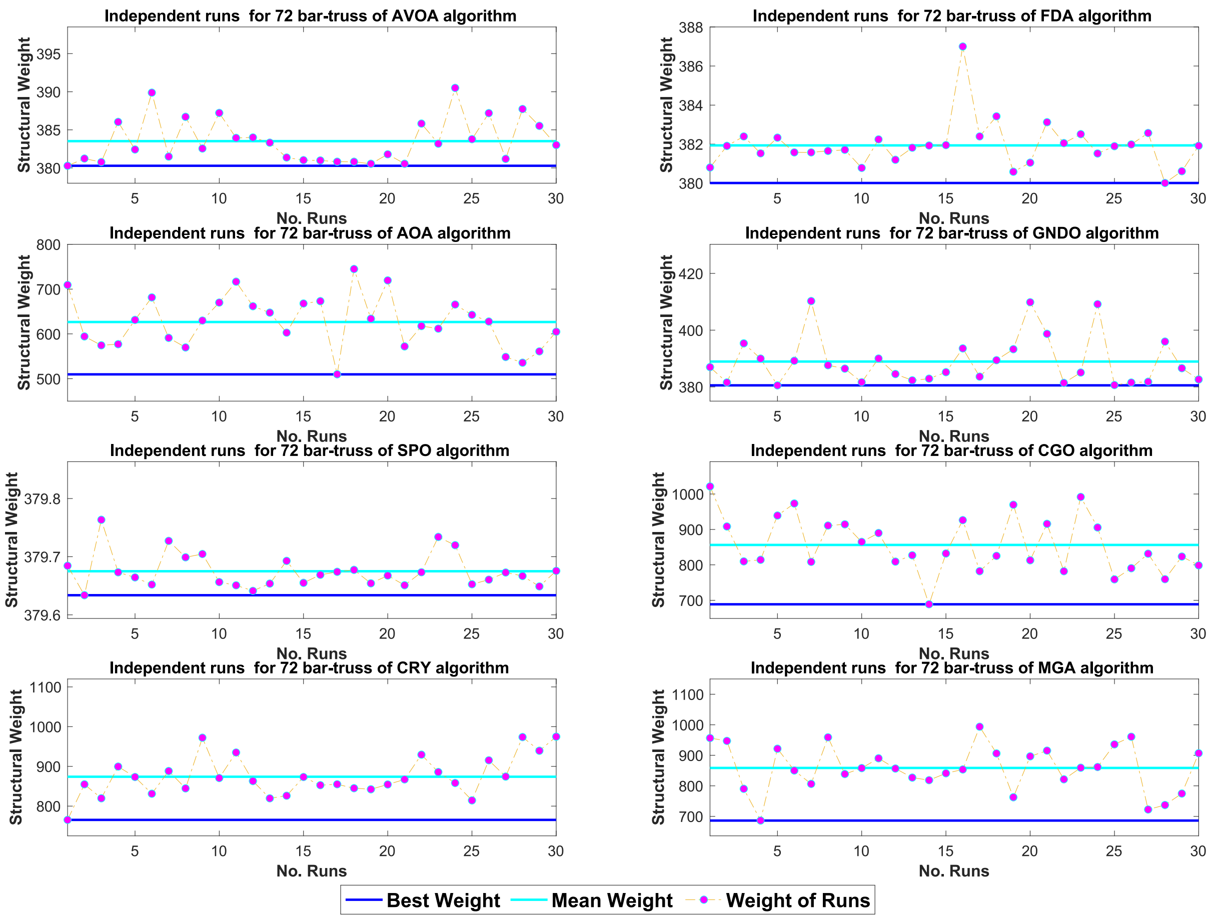

Fig. 13. The 30 independent runs for 72-member truss design structures of all algorithms.

design that is 34.12% heavier than SPO in terms of average weight. Furthermore, AOA, CRY, CGO, and MGA have the highest standard deviation.

Fig. 15 presents the convergence curves of the best algorithm runs, whereas Fig. 16 displays the average convergence curves of 30 algorithm runs. As shown by the congruent findings in both figures, SPO is the algorithm that achieves the best outcome with the least amount of analysis. AVOA, FDA, and GNDO are algorithms that follow SPO and perform similarly to SPO in convergence speed. AOA has an average convergence rate; however, CGO, MGA, and CRY are algorithms that require the greatest number of analyses to achieve the best result and so have relatively low convergence performance.

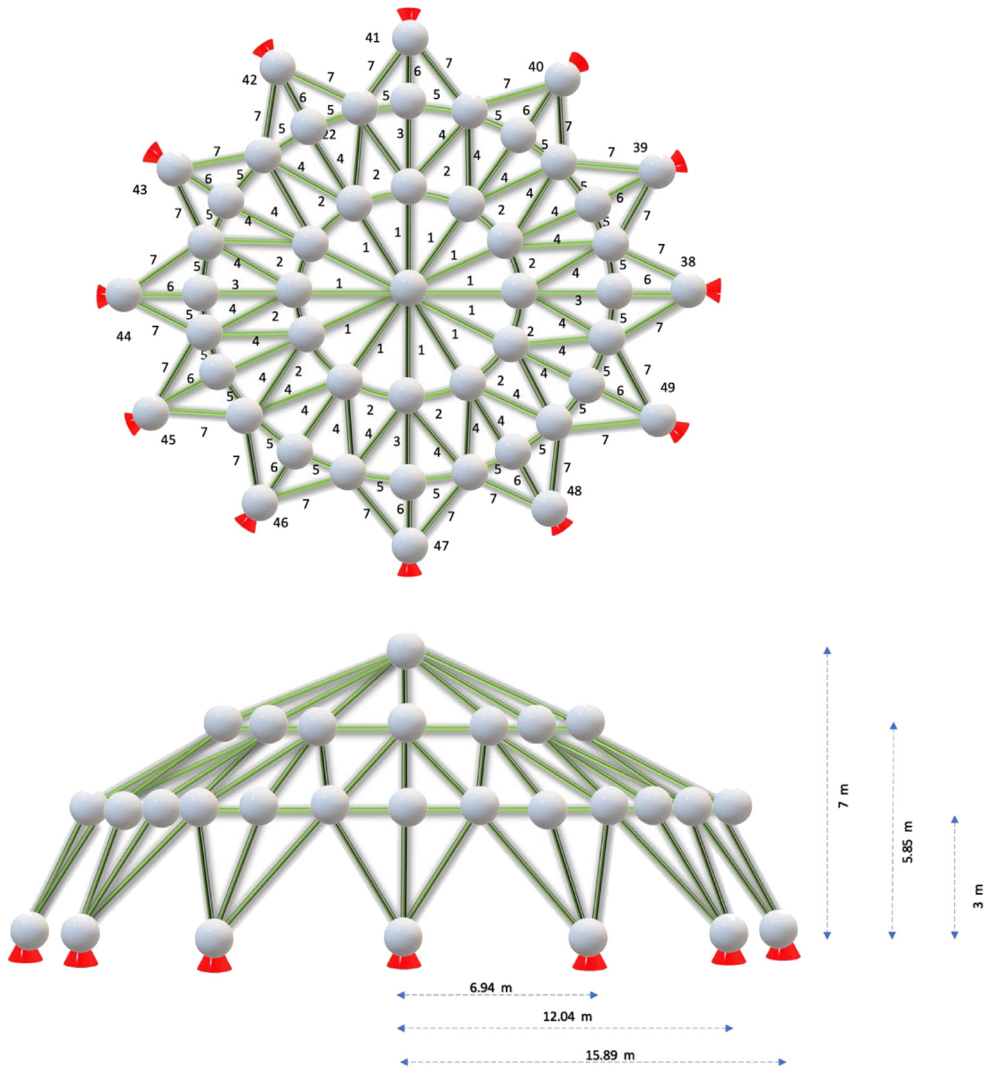

Fig. 14. The 120-member dome design structures.

Table 7
The results of different methods for 120-member dome structures.

Member group	AVOA	FDA	AOA	GNDO	SPO	CGO	CRY	MGA
1	3.0261	3.0252	3.1850	3.0246	3.0243	3.4806	3.4483	3.0742
2	14.7834	14.7664	18.9905	14.8500	14.7626	13.1129	11.3884	10.6301
3	4.9595	5.1440	5.0804	5.0929	5.0876	6.4437	7.8715	7.1543
4	3.0926	3.1307	2.7501	3.1336	3.1369	4.3015	3.3907	3.4238
5	8.4451	8.4129	6.7730	8.4260	8.4795	9.4229	8.5720	10.7356
6	3.6784	3.3314	7.9835	3.2989	3.2875	4.0022	7.6157	6.6660
7	2.5015	2.4952	2.7326	2.4961	2.4963	3.1130	3.0617	3.3623
Best weight (lb.)	33 299.7891	33 253.0103	36 548.2763	33 250.6747	**33 249.5608**	38 263.1107	38 341.4169	38 897.5332
Average weight (lb.)	33 541.5996	33 287.1615	41 609.2175	33 256.3555	**33 251.0038**	44 598.0629	43 883.3807	44 162.8185
Standard deviation	188.5024	30.4286	3117.4733	7.1625	**1.9190**	2717.4724	3035.5354	2429.5537
Number of analyses	10 000	10 000	10 000	10 000	10 000	10 000	10 000	10 000

Fig. 17 displays the element stress of a 120-member dome designed with the best run of SPO. Components are generally stressed between limitations, as seen in this figure. Fig. 18 displays the displacement ratio of the 120-member dome at the best SPO-optimized design. When the DOF number is up to 40, the displacement ratio is near the maximum limit of 1, and the displacement ratio is within the limitations when the DOF number is between 40 and 110. This means that both displacement ratio and element stress constraints control the design of this example.

Fig. 19 depicts the weights, average weights, and best weights obtained by all algorithms in all runs ranging from 1 to 30. This figure shows that the average weight and best weight discovered by the SPO

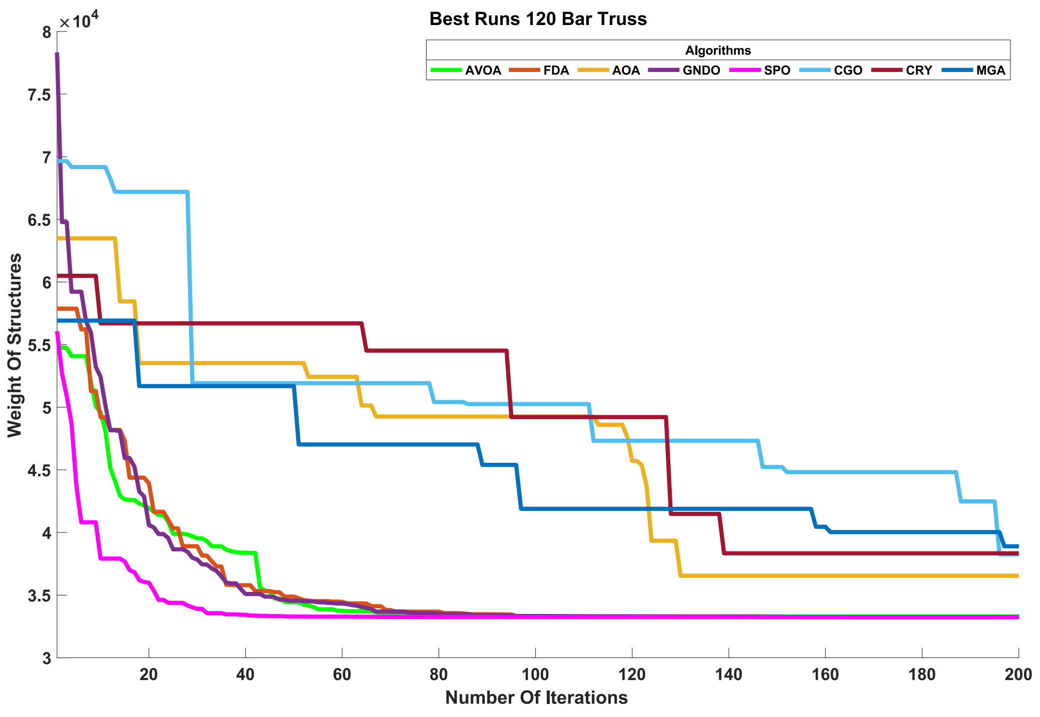

Fig. 15. The best runs convergence curves for 120-member dome design structures.

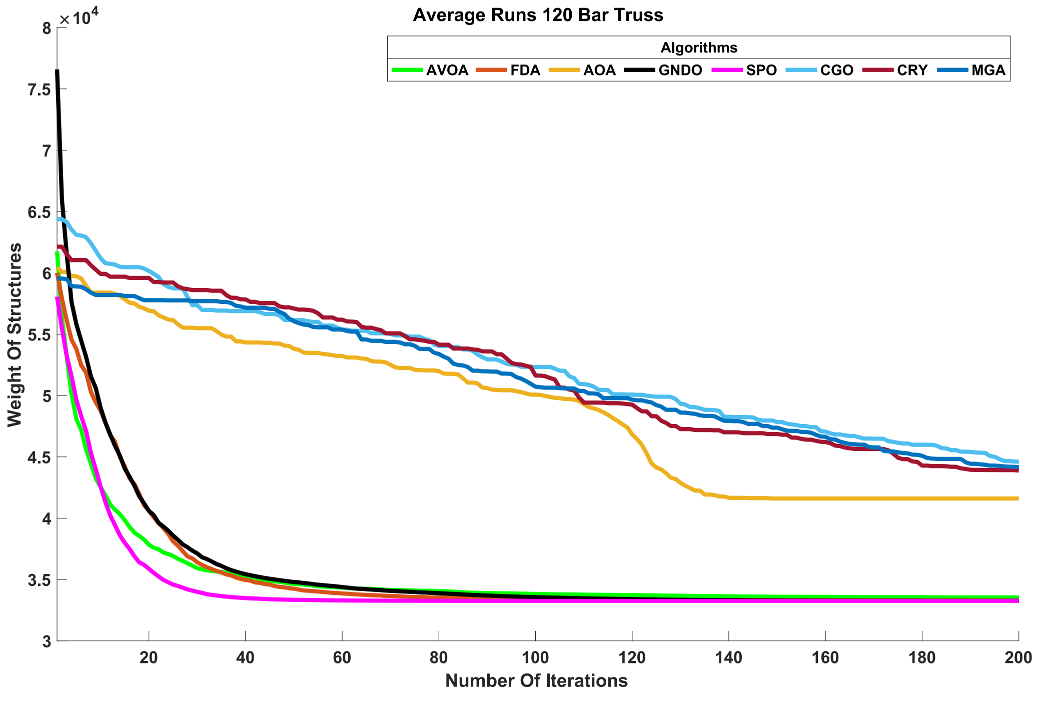

Fig. 16. The mean runs convergence curves for 120-member dome design structures.

algorithm for the 120-member dome design problem are relatively close to other algorithms. Hence, stability is the highest.

The robustness of SPO is demonstrated by the statistical results collected from 720 separate runs for all eight mentioned metaheuristic algorithms. A good exploration and exploitation for the SPO are achieved by four simple color combination rules, eliminating the requirement for any internal parameter. One notable characteristic of SPO is that it removes the need for preliminary fine-tuning of parameters, distinguishing it from many existing meta-heuristic algorithms. The efficiency of SPO is assessed using three different truss structures, and the results indicate its superior performance in terms of both efficiency and precision compared to other algorithms.

5. Conclusion and future work

This study investigated the performance of eight different meta-heuristic algorithms for the continuous size optimization of space truss structures. The examined algorithms included the African Vultures Optimization Algorithm, Flow direction algorithm, Arithmetic Optimization Algorithm, Generalized Normal Distribution Optimization,

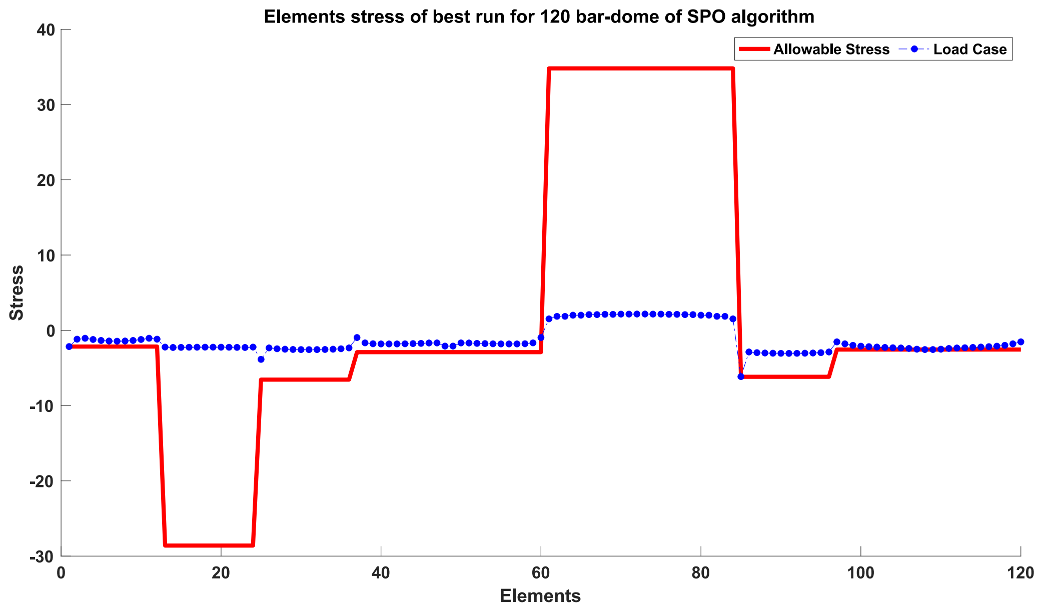

Fig. 17. The element stress of SPO for 120-member design structures.

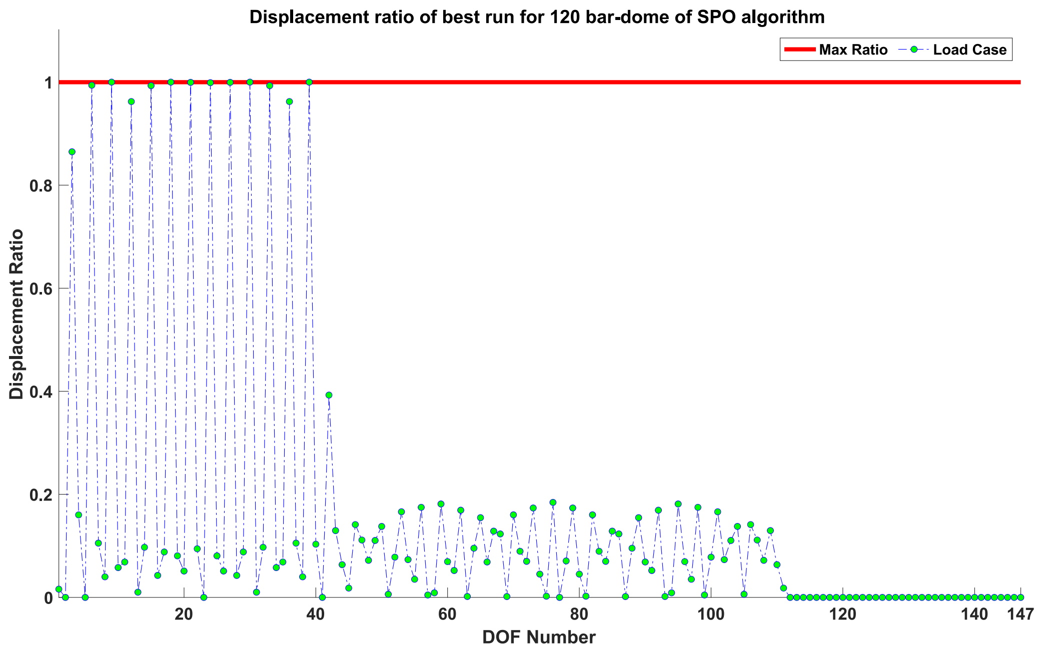

Fig. 18. The displacement ratio of SPO for 120-member dome design structures.

Stochastic Paint Optimizer, Chaos Game Optimizer, Crystal Structure Algorithm, and Material Generation Algorithm. The objective was to assess the capabilities and efficiencies of these meta-heuristics by applying them to the size optimization of three benchmark truss structures. The optimization process results revealed that the Stochastic Paint Optimizer (SPO) outperformed the other algorithms in size optimization for space truss structures. A thorough analysis of the optimization results demonstrated that the SPO achieved the best outcomes in terms of weight, average weight, and standard deviation.

Additionally, the convergence curves indicated that the SPO had higher convergence speeds than the other investigated algorithms. Further evaluation studies are required to explore the applicability of the SPO algorithm in solving optimization problems related to steel frame structures and other areas of structural engineering. Furthermore, future works will discuss improved and multi-objective versions of these methods to enhance their performance in frames and larger truss statures and optimization tasks.

Declaration of competing interest

The authors have declared no conflict of interest.

Data availability

No data was used for the research described in the article.

Acknowledgment

The authors gratefully acknowledge the financial support from the National Science Foundation I/U-CRC Center for Integration of Composites into Infrastructure (CICI) under grant #1916342.

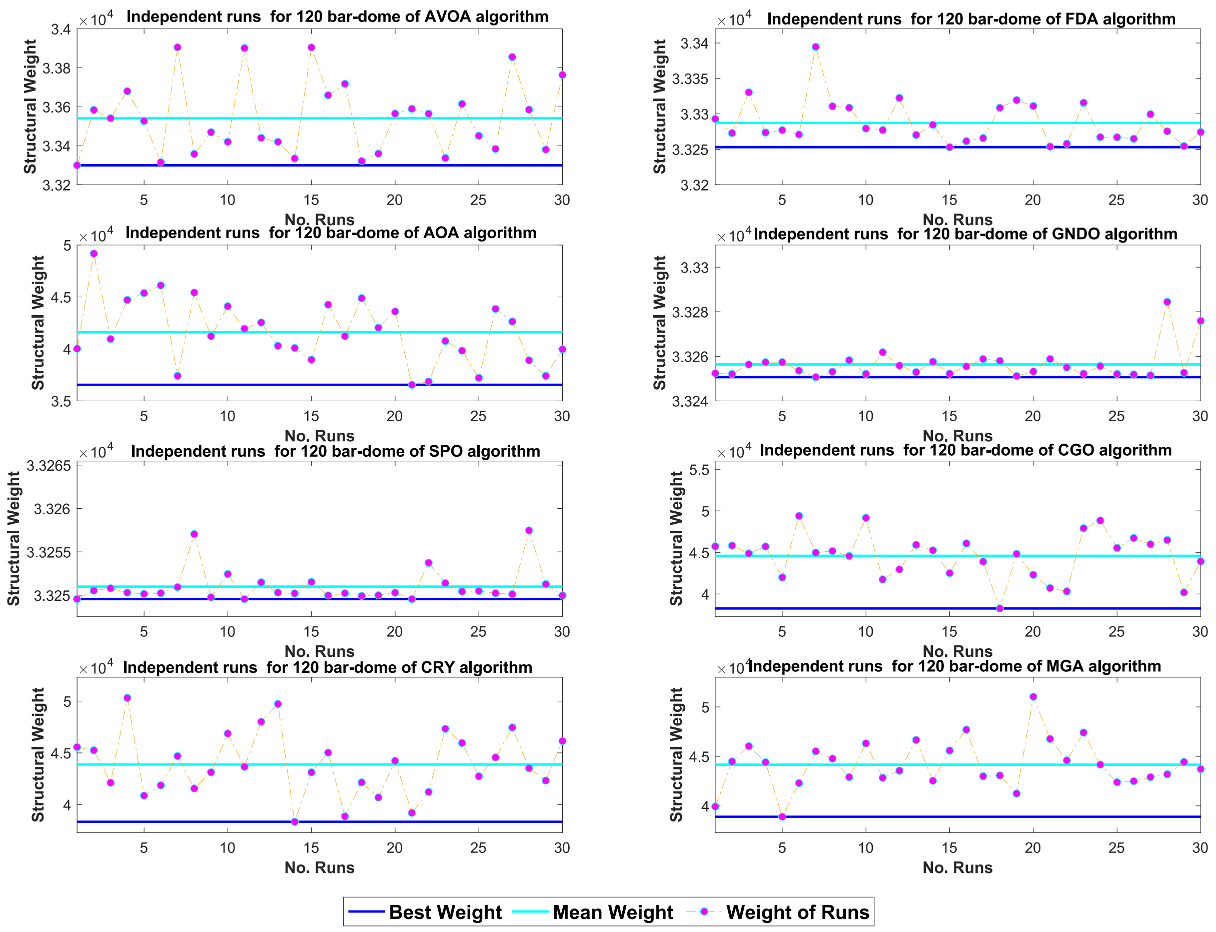

Fig. 19. The 30 independent runs for 120-member dome design structures of all algorithms.

References

[1] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.

[2] S. Khalilpourazari, S. Khalilpourazary, An efficient hybrid algorithm based on water cycle and Moth–Flame optimization algorithms for solving numerical and constrained engineering optimization problems, Soft Comput. 23 (5) (2019) 1699–1722, http://dx.doi.org/10.1007/s00500-017-2894-y.

[3] H. Faris, S. Mirjalili, I. Aljarah, M. Mafarja, A.A. Heidari, Salp swarm algorithm: Theory, literature review, and application in extreme learning machines, Stud. Comput. Intell. 811 (January) (2020) 185–199, http://dx.doi.org/10.1007/978-3-030-12127-3_11.

[4] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–73.

[5] K. v Price, Differential evolution, in: Handbook of Optimization, Springer, 2013, pp. 187–214.

[6] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, IEEE, 1995, pp. 1942–1948.

[7] A.H. Gandomi, X.-S. Yang, A.H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Eng. Comput. 29 (1) (2013) 17–35.

[8] A. Mohammadi-Balani, M.D. Nayeri, A. Azar, M. Taghizadeh-Yazdi, Golden eagle optimizer: A nature-inspired metaheuristic algorithm, Comput. Ind. Eng. 152 (2021) 107050.

[9] A. Kaveh, S. Talatahari, N. Khodadadi, Stochastic paint optimizer: theory and application in civil engineering, Eng. Comput. (2020) 1–32.

[10] M.-Y. Cheng, D. Prayogo, Symbiotic organisms search: a new metaheuristic optimization algorithm, Comput. Struct. 139 (2014) 98–112.

[11] J.O. Agushaka, A.E. Ezugwu, L. Abualigah, Dwarf mongoose optimization algorithm, Comput. Methods Appl. Mech. Engrg. 391 (2022) 114570.

[12] S. Kaur, L.K. Awasthi, A.L. Sangal, G. Dhiman, Tunicate swarm algorithm: A new bio-inspired based metaheuristic paradigm for global optimization, Eng. Appl. Artif. Intell. 90 (2020) 103541.

[13] S.O. Degertekin, G.Y. Bayar, L. Lamberti, Parameter free jaya algorithm for truss sizing-layout optimization under natural frequency constraints, Comput. Struct. 245 (2021) 106461.

[14] J. Pierezan, L. dos Santos Coelho, V.C. Mariani, E.H. de Vasconcelos Segundo, D. Prayogo, Chaotic coyote algorithm applied to truss optimization problems, Comput. Struct. 242 (2021) 106353.

[15] A. Kaveh, N. Khodadadi, S. Talatahari, A comparative study for the optimal design of steel structures using Css and Acss algorithms, Iran Univ. Sci. Technol. 11 (1) (2021) 31–54.

[16] A. Kaveh, N. Khodadadi, B.F. Azar, S. Talatahari, Optimal design of large-scale frames with an advanced charged system search algorithm using box-shaped sections, Eng. Comput. (2020) 1–21.

[17] N. Khodadadi, M. Azizi, S. Talatahari, P. Sareh, Multi-objective crystal structure algorithm (MOCryStAl): Introduction and performance evaluation, IEEE Access 9 (2021) 117795–117812.

[18] D. Ustun, An enhanced adaptive butterfly optimization algorithm rigorously verified on engineering problems and implemented to ISAR image motion compensation, Eng. Comput. (Swansea) (2020).

[19] N. Khodadadi, S. Vaclav, S. Mirjalili, Dynamic arithmetic optimization algorithm for truss optimization under natural frequency constraints, IEEE Access 10 (2022) 16188–16208.

[20] G.G. Tejani, V.J. Savsani, V.K. Patel, S. Mirjalili, Truss optimization with natural frequency bounds using improved symbiotic organisms search, Knowl. Based Syst. 143 (2018) 162–178.

[21] M. Mashayekhi, R. Yousefi, Topology and size optimization of truss structures using an improved crow search algorithm, Struct. Eng. Mech. Int. J. 77 (6) (2021) 779–795.

[22] A. Kaveh, A.D. Eslamlou, N. Khodadadi, Dynamic water strider algorithm for optimal design of skeletal structures, Period. Polytech. Civ. Eng. 64 (3) (2020) 904–916.

[23] S. Kumar, G.G. Tejani, N. Pholdee, S. Bureerat, P. Jangir, Multi-objective teaching-learning-based optimization for structure optimization, Smart Sci. 10 (1) (2022) 56–67.

[24] N. Khodadadi, S. Talatahari, A. Dadras Eslamlou, MOTEO: a novel multi-objective thermal exchange optimization algorithm for engineering problems, Soft Comput. (2022) 1–26.
[25] A. Kaveh, V.R. Mahdavi, Multi-objective colliding bodies optimization algorithm for design of trusses, J. Comput. Des. Eng. 6 (1) (2019) 49–59.
[26] A. Kaveh, S. Talatahari, N. Khodadadi, The hybrid invasive weed optimization-shuffled frog-leaping algorithm applied to optimal design of frame structures, Period. Polytech. Civ. Eng. 63 (3) (2019) 882–897.
[27] W. Lingyun, Z. Mei, W. Guangming, M. Guang, Truss optimization on shape and sizing with frequency constraints based on genetic algorithm, Comput. Mech. 35 (5) (2005) 361–368.
[28] H.M. Gomes, Truss optimization with dynamic constraints using a particle swarm algorithm, Expert Syst. Appl. 38 (1) (2011) 957–968.
[29] A. Kaveh, S. Talatahari, N. Khodadadi, Hybrid invasive weed optimization-shuffled frog-leaping algorithm for optimal design of truss structures, Iran. J. Sci. Technol. Trans. Civ. Eng. 44 (2) (2019) 405–420.
[30] H. Tang, T.N. Huynh, J. Lee, A novel adaptive 3-stage hybrid teaching-based differential evolution algorithm for frequency-constrained truss designs, in: Structures, Elsevier, 2022, pp. 934–948.
[31] N. Khodadadi, S. Mirjalili, Truss optimization with natural frequency constraints using generalized normal distribution optimization, Appl. Intell. (2022) 1–14.
[32] F. Jiang, L. Wang, L. Bai, An improved whale algorithm and its application in truss optimization, J. Bionic Eng. 18 (3) (2021) 721–732.
[33] F.K.J. Jawad, C. Ozturk, W. Dansheng, M. Mahmood, O. Al-Azzawi, A. Al-Jemely, Sizing and layout optimization of truss structures with artificial bee colony algorithm, in: Structures, Elsevier, 2021, pp. 546–559.
[34] G. Bekdaş, M. Yucel, S.M. Nigdeli, Evaluation of metaheuristic-based methods for optimization of truss structures via various algorithms and lèvy flight modification, Buildings 11 (2) (2021) 49.
[35] S. Gholizadeh, R. Sojoudizadeh, Modified sine-cosine algorithm for sizing optimization of truss structures with discrete design variables, Iran Univ. Sci. Technol. 9 (2) (2019) 195–212.
[36] Y. Li, S. Wang, M. Han, Truss structure optimization based on improved chicken swarm optimization algorithm, Adv. Civ. Eng. 2019 (2019).
[37] F.K.J. Jawad, M. Mahmood, D. Wang, A.-A. Osama, A.-J. Anas, Heuristic dragonfly algorithm for optimal design of truss structures with discrete variables, in: Structures, Elsevier, 2021, pp. 843–862.
[38] S. Kumar, G.G. Tejani, S. Mirjalili, Modified symbiotic organisms search for structural optimization, Eng. Comput. 35 (4) (2019) 1269–1296.
[39] I. Serpik, Discrete size and shape optimization of truss structures based on job search inspired strategy and genetic operations, Period. Polytech. Civ. Eng. 64 (3) (2020) 801–814.
[40] T. Dede, M. Grzywiński, R. Venkata Rao, Jaya: A new meta-heuristic algorithm for the optimization of braced dome structures, in: Advanced Engineering Optimization Through Intelligent Techniques, Springer, 2020, pp. 13–20.
[41] O. Altay, O. Cetindemir, I. Aydogdu, Size optimization of planar truss systems using the modified salp swarm algorithm, Eng. Optim. (2023) 1–17.
[42] N. Khodadadi, S. Talatahari, A.H. Gandomi, ANNA: Advanced neural network algorithm for optimization of structures, Proc. Inst. Civ. Eng. Struct. Build. (2023) 1–59.
[43] T. Vu-Huu, S. Pham-Van, Q.-H. Pham, T. Cuong-Le, An improved bat algorithms for optimization design of truss structures, in: Structures, Elsevier, 2023, pp. 2240–2258.
[44] V. Goodarzimehr, U. Topal, A.K. Das, T. Vo-Duy, Bonobo optimizer algorithm for optimum design of truss structures with static constraints, in: Structures, Elsevier, 2023, pp. 400–417.
[45] Y. Yuan, et al., Coronavirus mask protection algorithm: A new bio-inspired optimization algorithm and its applications, J. Bionic Eng. (2023) 1–19.
[46] B. Abdollahzadeh, F.S. Gharehchopogh, S. Mirjalili, African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems, Comput. Ind. Eng. 158 (2021) 107408.
[47] H. Karami, M.V. Anaraki, S. Farzin, S. Mirjalili, Flow direction algorithm (FDA): A novel optimization approach for solving optimization problems, Comput. Ind. Eng. 156 (2021) 107224.
[48] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, A.H. Gandomi, The arithmetic optimization algorithm, Comput. Methods Appl. Mech. Engrg. 376 (2021) 113609.
[49] Y. Zhang, Z. Jin, S. Mirjalili, Generalized normal distribution optimization and its applications in parameter extraction of photovoltaic models, Energy Convers. Manag. 224 (2020) 113301.
[50] S. Talatahari, M. Azizi, M. Tolouei, B. Talatahari, P. Sareh, Crystal structure algorithm (CryStAl): A metaheuristic optimization method, IEEE Access 9 (2021) 71244–71261.
[51] S. Talatahari, M. Azizi, A.H. Gandomi, Material generation algorithm: A novel metaheuristic algorithm for optimization of engineering problems, Processes 9 (5) (2021) 859.