A hybrid snake optimizer with crisscross learning strategy for constrained structural optimization

Pinghe Ni[a], Xiaoyu Su[a], Jinlong Fu[b,c,*], Yulei Bai[a]

[a] *State Key Laboratory of Bridge Engineering Safety and Resilience, Beijing University of Technology, Beijing 100124, China*
[b] *School of Engineering and Materials Science, Faculty of Science and Engineering, Queen Mary University of London, London, E1 4NS, UK*
[c] *Zienkiewicz Institute for Modelling, Data and AI, Faculty of Science and Engineering, Swansea University, Swansea, SA1 8EN, UK*

Abstract

Structural optimization aims to minimize material usage and total construction cost while ensuring comfort, safety and durability. Despite the wide application of metaheuristic algorithms in structural design, they encounter great challenges when solving non-convex optimization problems under multiple nonlinear constraint conditions, leading to low convergence rates and trapping in local optima as the solution space expands. In this study, a novel *hybrid Snake Optimizer with Crisscross Learning strategy* (SO-CL) is proposed to overcome these limitations by enriching population diversity, preventing prematurity, and enhancing solution-searching capability. The main innovations of SO-CL lie in: (i) integrating the lens-imaging learning strategy into snake optimizer to broaden solution range and mitigate search stagnation; (ii) applying the crossover learning strategy to access new possible solution regions and prevent premature convergence; and (iii) employing an adaptive weighting strategy to make the best balance between global exploration and local exploitation for computational efficiency improvement. The effectiveness of SO-CL has been validated by using it to solve five benchmark problems for constrained optimization of truss structures, along with a comprehensive comparison carried out between SO-CL and a series of metaheuristic algorithms in terms of optimization accuracy, convergence rate and computational stability. The results confirm that the proposed SO-CL algorithm is widely applicable for addressing diverse structural optimization problems with complex constraints while demonstrating competitive performance relative to well-known metaheuristic algorithms.

Keywords: Truss structures; Constrained optimization; Metaheuristic algorithm; Crisscross learning strategy; Lens-imaging learning strategy; Adaptive weighting strategy

1. Introduction

Structural optimization is motivated by the concept of designing reliable, sustainable, material-efficient and cost-effective structures, and it can be achieved by solving optimization problems to minimize the mass, volume or cost function satisfying multiple constraints. This study specially focuses on truss structures that are ubiquitous in industrial and civil construction, such as truss roof systems [1], truss arch bridges [2] and truss towers [3]. Truss optimization usually requires substantial resources to compute, because it often involves minimizing the structural weight while mandatorily satisfying multiple constraints that are non-convex and highly nonlinear. Over the recent decades, massive efforts have been made to advance the optimization techniques in this field [4, 5, 6]. Generally, truss optimization can be divided into three categories [4, 7]: size optimization, shape optimization and topology optimization. Size optimization aims to find the best cross-sectional areas of truss components while satisfying strength and stiffness requirements; shape optimization modifies the nodal coordinates of selected joints to improve structural performance and material efficiency; and topology optimization adjusts the layout of structural elements by eliminating unnecessary structural members.

Gradient-based algorithms [8, 9] are the classic approaches to solving truss optimization problems, but they rely heavily on the gradient information to search optimum solutions. Despite the success of gradient-based optimization in many cases, they can be failed if the gradient information of objective and constraint functions are unavailable. Besides, gradient-based algorithms have been reported that they lack consistent efficiency in addressing many larger-scale, non-continuous and non-differentiable engineering problems [10]. Recently, various

*Corresponding author: jinlong.fu@qmul.ac.uk

metaheuristic algorithms have been developed to solve different engineering optimization problems and design challenges. Different from gradient-based algorithms, metaheuristic algorithms are gradient-free, and they possess more powerful solution-searching capability because of their inherent adaptability and stochastic nature. Representative examples of metaheuristic algorithms include genetic algorithm (GA) [11], harmony search (HS) [12], firefly algorithm (FA) [13], teaching–learning-based optimization (TLBO) [14], particle swarm optimizer [15, 16], school-based optimization (SBO) [17], grey wolf optimizer (GWO)[18] butterfly optimization [19], Jaya algorithm [20], modified simulated annealing algorithm (MSAA) [21],, weighted superposition attraction [22] political optimizer (PO) [23], electromagnetism-like firefly algorithm (EFA) [24], Bonobo optimizer (BO) [25] and so on.

As the "No Free Lunch (NFL)" theorem [26] states, no single metaheuristic-type method can solve all optimization problems with the same efficiency. This statement emphasizes the necessity of aligning algorithmic strengths with problem-specific demands, which can yield much better solutions by effectively exploring the full design space. Following this idea, many improved variants and hybrid variants of metaheuristic algorithms have been proposed to solve structural optimization problems more accurately and efficiently. Rajasekaran et al. [27] hybridized genetic algorithm with an immune system mechanism for structural optimization. Kazemzadeh et al. [28] integrated the design oriented strategy to three different metaheuristic algorithms (namely, adaptive dimensional search, modified big bang-big crunch, and exponential big bang-big crunch algorithms) for optimal design of steel truss structures. Khatibinia et al. [29] combined gravitational search algorithm with simplex crossover and mutation operators to enhance local exploitation capability and speed up convergence rate. Jafari et al. [30] proposed a hybrid algorithm based on the elephant herding optimization (EHO) and cultural algorithm (CA), where the belief space defined by CA was used to enhance the solution-searching ability of EHO for truss design. Ozbasaran et al. [31] critically assessed the performances of the crow search algorithm and its variants, and highlighted the main challenges of parameter-controlled algorithms in structural optimization. Degertekin et al. [32] proposed three hybrid search algorithms including hybrid Simulated Annealing (SA), Harmony Search (HS) and Big Bang-Big Crunch (BBBC) for optimization design problems. Pierezan et al. [33] proposed a modified coyote optimization algorithm for structural optimization problems with the discrete design variables. Kaveh et al. [34] developed an improved slime mould algorithm for structural optimization with natural frequency constraints. The aim was to address the shortcomings of the original slime mould algorithm, specifically its convergence speed and premature convergence issues. Vu-Huu et al. [35] developed a new loudness function and push-process technique to improve the solution-searching ability of the bat algorithm, which can accelerate convergence rate and reduce computational effort in truss structure optimization.

The *Snake Optimizer* (SO) [36], an emerging metaheuristic algorithm, was inspired by the unique mating and fighting behaviors of snakes. It has unique advantages of no requirement for predefined parameters and excellent computational stability, which significantly contribute to its popularity. The SO and its variants have been successfully applied to various real-world optimization problems, such as cloud computing [37], optimal design [38], satellite communication [39] and capacity optimization of hybrid energy storage system [40]. To our best knowledge, the SO and its variants have not been used for structural optimization yet, in spite of their great success, which strongly motivates us to extend its application to optimal design of truss structures. Usually, the optimization objectives and constraints in truss design problems are non-convex and highly nonlinear functions, which can pose great challenges for the existing SO algorithms to solve, leading to slow convergence, low accuracy, poor stability and trapping in local optima.

In this study, a novel *hybrid Snake Optimizer with Crisscross Learning strategy* (SO-CL) is developed to solve complicated truss optimization problems under multiple nonlinear constraints. The lens-imaging learning strategy, the crossover learning strategy and an adaptive weighting strategy are integrated into the computational framework of SO, in order to enrich population diversity, prevent prematurity and enhance solution-searching capability. The primary innovations of this work are summarized as follows:

- The integration of the lens-imaging learning strategy into the basic SO greatly broadens the solution-searching range, thereby avoiding search stagnation;
- The application of the crossover learning strategy allows to visit new possible solution regions, which can significantly enrich population diversity and prevent premature convergence;
- The adoption of the adaptive weighting strategy can effectively balance exploration and exploitation to dynamically adjust the solution-searching state, ensuring high computational efficiency.

To verify the effectiveness of this new SO-CL algorithm, it has been applied to solve a series of benchmark problems, including constrained structural optimizations of the 10-bar, 25-bar, 72-bar, 120-bar and 200-bar trusses. And a comparative study has also been carried out between SO-CL and a number of standard metaheuristic algorithms, to assess its performance in aspects of optimization accuracy and convergence rate. The remainder of

this paper is organized as follows: Section 2 briefly describes the general mathematical formulation of constrained optimization problems for truss structure design; Section 3 detailedly explains the proposed SO-CL algorithms by providing the necessary clarification on the integration of enhancement strategies into the computational framework of SO; In Section 4 , the performance of SO-CL on truss optimization is demonstrated via five representative case studies, and its strengths and weaknesses are also discussed; Finally, the main contributions of this work are summarised in Section 5, and the potential directions for future research are pointed out as well.

2. Mathematical formulation of constrained structural optimization

Optimization design of truss structures is to find the minimum weight of components while satisfying the predefined constraints in terms of stress, displacement and buckling stability, in which the cross-sectional areas $\boldsymbol{A}$ of truss elements often serve as the continues design variables. Given a truss structure comprised of m bar members and n junction nodes, the corresponding optimal design problem under multiple constraints can be mathematically expressed by the following formulations:

$$\begin{aligned} &\text{Minimize}: \quad \mathcal{W}(\boldsymbol{A}) = \sum_{i=1}^{m} \rho_i A_i L_i \,, \\ &\text{Subjected to}: \begin{cases} \mathcal{S}_i(\boldsymbol{A}) = \frac{\sigma_i}{[\sigma_i]} - 1 \le 0, \quad i = 1, 2, ..., m, \\ \mathcal{D}_j(\boldsymbol{A}) = \frac{\delta_j}{[\delta_j]} - 1 \le 0, \quad j = 1, 2, ..., n, \\ \mathcal{B}_k(\boldsymbol{A}) = \frac{\sigma_k^{(\mathrm{com})}}{\left[\sigma_k^{(\mathrm{cr})}\right]} - 1 \le 0, \quad k = 1, 2, ..., m, \\ A_{\min,i} \le A_i \le A_{\max,i}, \quad i = 1, 2, ..., m, \end{cases} \end{aligned} \tag{1}$$

where $\mathcal{W}(\cdot)$ represents the total weight of the truss structure; $\boldsymbol{A} = [A_1, A_2, ..., A_m]$ denotes the design variable vector containing the cross-sectional areas of all m members; ρ_i and L_i are the density and length of the i-th member, respectively; $\mathcal{S}_i(\cdot)$, $\mathcal{D}_j(\cdot)$ and $\mathcal{B}_k(\cdot)$ denote the stress, displacement and buckling constrains for the i-th member, j-th node and k-th member, respectively; σ_i denotes the stress of the i-th member, and $[\sigma_i]$ denotes the allowable stress; δ_j is the displacement at the j-th node, and $[\delta_j]$ represents the maximum allowable displacement; $\sigma_k^{(\mathrm{com})}$ is the compressive stress of the k-th member, and $\left[\sigma_k^{(\mathrm{cr})}\right]$ denotes the critical value of allowable compressive stress to avoid member buckling; $A_{\min,i}$ and $A_{\max,i}$ are the lower and upper bounds for the cross-sectional area of the i-th member, respectively.

To solve the above constrained optimization problem, the penalty function technique [29, 41] is adopted here to convert it to an unconstrained optimization problem. Then, Eq. (1) can thus be equivalently transformed to the following formulations:

$$\begin{aligned} &\text{Minimize}: \quad \mathcal{L}(\boldsymbol{A}) = \mathcal{W}(\boldsymbol{A})\Big[1 + \alpha\,\mathcal{P}(\boldsymbol{A})\Big]^{\beta}, \\ &\mathcal{P}(\boldsymbol{A}) = \sum_{i=1}^{m} \max\Big[0,\, \mathcal{S}_i(\boldsymbol{A})\Big] + \sum_{j=1}^{n} \max\Big[0,\, \mathcal{D}_j(\boldsymbol{A})\Big] + \sum_{k=1}^{m} \max\Big[0,\, \mathcal{B}_k(\boldsymbol{A})\Big], \end{aligned} \tag{2}$$

where $\mathcal{P}(\cdot)$ represents the total penalty function; α is the penalty function scaling factor, and β is the penalty function index, both of which control the penalty degree applied to violated solutions. In this study, α is set as a constant equal to 1, while β starts at 1.5 and linearly increases to 6.0, as described by the following equation:

$$\beta = 1.5 + \frac{4.5t}{T}\,, \tag{3}$$

where t and T denote the current iteration step and the maximum number of iterations, respectively.

3. Methodology: Optimization framework

In this section, a *hybrid Snake Optimizer with Crisscross Learning strategy* (SO-CL) is developed to solve the complicated optimization problems with nonlinear constraints for truss structure design. The methodology of this new SO-CL is explained by briefly recapping the basic theory of snake optimizer (SO) and detailedly clarifying the integration of multiple enhancement strategies and operations into the standard SO to achieve the powerful optimization capability. In Fig. 1, the computational framework of SO-CL is graphically illustrated, where the lens-imaging learning strategy, the crossover learning strategy and an adaptive weighting strategy are integrated into SO, aiming to overcome the computational limitations of the standard SO by enriching population diversity, avoiding prevent prematurity and improving solution-searching capability.

3.1. Standard snake optimizer

Snake Optimizer (SO) [36, 37] is an emerging metaheuristic algorithm inspired by the unique mating behavior of snakes. The mating behavior of snakes is closely linked to their survival habits and is primarily governed by two critical factors: food supply and environmental temperature. When food quantity is scarce, snakes tend to engage in extensive exploration. After acquiring an adequate amount of food, these snakes enter the exploitation phase, and temperature starts to be the critical influencing factor. When the temperature is high, snakes find it difficult to mate and focus solely on consuming the available food. When the temperature is low, snakes exhibit two unique behaviors associated with mating: fighting and mating. In the fighting mode, snakes compete for the best mate, and after that they enter the mating mode and mate with nearby snakes. The workflow of the standard SO algorithm is briefly introduced in the following contents, and its pseudo-code is also provided in Algorithm 1.

Algorithm 1: The workflow of standard snake optimizer (SO)

Input: Population size N, the maximum number of function evaluations NFE, the dimension D of optimization problem, and the boundaries of design valuables $[b_{\min}, b_{\max}]$.
Output: The optimal solution.
Initialization: Randomly initialize N snakes using Eq (4), and then divide them into two groups: male and female;
The maximum number of iterations $T = NFE / N$, and the current iteration step $t = 0$.
while $t \leq T$ **do**
  Evaluate the fitness of each group, including male and female;
  Find the best male $X_{\mathrm{b}}^{(\mathrm{m})}$ and female $X_{\mathrm{b}}^{(\mathrm{f})}$ snakes;
  Calculate the environmental temperature $Temp$ and food supply Q using Eqs. (5) and (6), respectively;
  if $Q < 0.25$ **then**
    Perform *exploration phase* using Eqs. (7) and (8);
  else
    if $Temp \geq 0.6$ **then**
      Perform *exploitation phase* using Eq. (11);
    else
      if $r \geq 0.6$ **then**
        The snakes are in the *fighting mode*, and their positions are updated using Eqs. (12) and (13);
      else
        The snakes are in the *mating mode*, and their positions are updated using Eqs. (16) and (17);
        Replace the worst snakes using Eqs. (20) and (21);
      end
    end
  end
end
return: The optimal solution.

3.1.1. Initialization and parameter definition

In the initial stage of optimization, SO randomly generates N individuals that are uniformly distributed in the solution space. Each individual represents a potentially feasible solution to the considered optimization problem. The initial population can be generated by:

$$X_i = b_{\min} + r_i \times (b_{\max} - b_{\min}), \tag{4}$$

where X_i represents the position of the i-th individual; r_i is a random value uniformly distributed within the interval [0, 1]; $b_{\max}$ and $b_{\min}$ are the predefined maximum and minimum bounds, respectively. The entire population should be equally divided into two sub-populations: male and female. Their behaviors are mainly controlled by two critical parameters: environmental temperature ($Temp$) and food quantity (Q), which are described by the following equations:

$$Temp = \exp\left(-\frac{t}{T}\right), \tag{5}$$

$$Q = c_1 \times \exp\left(\frac{t}{T} - 1\right), \tag{6}$$

where t denotes the current iteration step, T denotes the maximum number of iterations, and c_1 is the food supply coefficient set to be 0.50 in this work.

3.1.2. Exploration phase

During the exploration phase, the food supply is insufficient ($Q < 0.25$). The snakes move randomly to search food, and their positions are thus updated, which can be described as follows:

$$X_i^{(\mathrm{m})}(t+1) = X_{\mathrm{r}}^{(\mathrm{m})}(t) \pm c_2 \times B_{\mathrm{m}} \times \left[(b_{\max} - b_{\min}) \times r_i + b_{\min}\right], \tag{7}$$

$$X_i^{(\mathrm{f})}(t+1) = X_{\mathrm{r}}^{(\mathrm{f})}(t) \pm c_2 \times B_{\mathrm{f}} \times \left[(b_{\max} - b_{\min}) \times r_i + b_{\min}\right], \tag{8}$$

where $X_i^{(\mathrm{m})}(t+1)$ and $X_j^{(\mathrm{m})}(t+1)$ are the positions of the i-th male and the j-th female at the next time step, respectively; $X_{\mathrm{r}}^{(\mathrm{m})}(t)$ and $X_{\mathrm{r}}^{(\mathrm{f})}(t)$ are the positions of male and female randomly selected from the sub-populations at current time step, respectively; c_2 is the searching coefficient set to be 0.05 in this work; B_{m} and B_{f} represent the ability of a male and female individuals to find food, respectively, which can be calculated as follows:

$$B_{\mathrm{m}} = \exp\left(-\frac{F_{\mathrm{r}}^{(\mathrm{m})}}{F_i^{(\mathrm{m})}}\right), \tag{9}$$

$$B_{\mathrm{f}} = \exp\left(-\frac{F_{\mathrm{r}}^{(\mathrm{f})}}{F_i^{(\mathrm{f})}}\right), \tag{10}$$

where $F_{\mathrm{r}}^{(\mathrm{m})}$ and $F_{\mathrm{r}}^{(\mathrm{f})}$ denote the fitness values of the randomly selected male $X_{\mathrm{r}}^{(\mathrm{m})}$ and female $X_{\mathrm{r}}^{(\mathrm{f})}$, respectively; $F_i^{(\mathrm{m})}$ and $F_i^{(\mathrm{f})}$ denote the fitness values of the i-th male $X_i^{(\mathrm{m})}$ and female $X_i^{(\mathrm{f})}$, respectively.

3.1.3. Exploitation phase

During the exploitation phase, the food supply is sufficient ($Q \geq 0.25$). If the environmental temperature $Temp \geq 0.6$, all snakes (both males and females) will move to the positions where food is available, which can be formulated as follows:

$$X_i(t+1) = X_{\mathrm{food}} \pm c_3 \times Temp \times r_i \times \left(X_{\mathrm{food}} - X_i(t)\right), \tag{11}$$

where X_{food} denotes the position of the best individual, and c_3 is the mating coefficient set to be 2.00 in this study.

If the environmental temperature $Temp < 0.6$, all snakes enter two modes, which are fighting ($r \geq 0.6$) and mating ($r < 0.6$) modes. The fighting mode is represented as follows:

$$X_i^{(\mathrm{m})}(t+1) = X_i^{(\mathrm{m})}(t) + c_3 \times \phi_{\mathrm{m}} \times r_i \times \left(Q \times X_{\mathrm{b}}^{(\mathrm{m})} - X_i^{(\mathrm{m})}(t)\right), \tag{12}$$

$$X_i^{(\mathrm{f})}(t+1) = X_i^{(\mathrm{f})}(t) + c_3 \times \phi_{\mathrm{f}} \times r_i \times \left(Q \times X_{\mathrm{b}}^{(\mathrm{f})} - X_i^{(\mathrm{f})}(t)\right), \tag{13}$$

where $X_{\mathrm{b}}^{(\mathrm{m})}$ and $X_{\mathrm{b}}^{(\mathrm{f})}$ denote the positions of the best male and female, respectively; ϕ_{m} and ϕ_{f} represent the fighting ability of male and female snakes, respectively, which can be formulated as follows:

$$\phi_{\mathrm{m}} = \exp\left(-\frac{F_{\mathrm{b}}^{(\mathrm{m})}}{F_i^{(\mathrm{m})}}\right), \tag{14}$$

$$\phi_{\mathrm{f}} = \exp\left(-\frac{F_{\mathrm{b}}^{(\mathrm{f})}}{F_i^{(\mathrm{f})}}\right), \tag{15}$$

where $F_{\mathrm{b}}^{(\mathrm{m})}$ and $F_{\mathrm{b}}^{(\mathrm{f})}$ denote the fitness values of the best male $X_{\mathrm{b}}^{(\mathrm{m})}$ and female $X_{\mathrm{b}}^{(\mathrm{f})}$, respectively. The mating mode is represented as follows:

$$X_i^{(\mathrm{m})}(t+1) = X_i^{(\mathrm{m})}(t) + c_3 \times M_{\mathrm{m}} \times r_i \times \left(Q \times X_i^{(\mathrm{f})}(t) - X_i^{(\mathrm{m})}(t)\right), \tag{16}$$

$$X_i^{(\mathrm{f})}(t+1) = X_i^{(\mathrm{f})}(t) + c_3 \times M_{\mathrm{f}} \times r_i \times \left(Q \times X_i^{(\mathrm{m})}(t) - X_i^{(\mathrm{f})}(t)\right), \tag{17}$$

where M_{m} and M_{f} represent the mating ability of male and female snakes, respectively. They can be formulated as follows:

$$M_{\mathrm{m}} = \exp\left(-\frac{F_i^{(\mathrm{f})}}{F_i^{(\mathrm{m})}}\right), \tag{18}$$

$$M_{\mathrm{f}} = \exp\left(-\frac{F_i^{(\mathrm{m})}}{F_i^{(\mathrm{f})}}\right), \tag{19}$$

If the eggs hatch, the worst male or female will be selected and replaced as follows:

$$X_{\mathrm{w}}^{(\mathrm{m})} = r \times (b_{\max} - b_{\min}) + b_{\min}\,, \tag{20}$$

$$X_{\mathrm{w}}^{(\mathrm{f})} = r \times (b_{\max} - b_{\min}) + b_{\min}\,, \tag{21}$$

where $X_{\mathrm{w}}^{(\mathrm{m})}$ and $X_{\mathrm{w}}^{(\mathrm{f})}$ are the worst individuals in the male and female groups, respectively.

3.2. Hybrid snake optimizer with multiple learning and adaptive strategies

Structural optimization design often contains non-convex objective functions with numerous nonlinear constraints, which can pose a huge challenge for the standard Snake Optimizer (SO) to deal with, leading to a low convergence rate and trapping in local optima as the solution space expands. To overcome these limitations, multiple learning strategies and adaptive operators are integrated into the basic framework of SO, including the lens-imaging learning strategy, the crossover learning strategy and an adaptive weighting factor. The developed *hybrid Snake Optimizer with Crisscross Learning strategy* (SO-CL) is far superior to the standard SO in aspects of population diversity, prematurity prevention and solution-searching ability. The computational framework of the innovative SO-CL algorithm is graphically illustrated in Fig. 1, and its pseudo-code is also given in Algorithm 2.

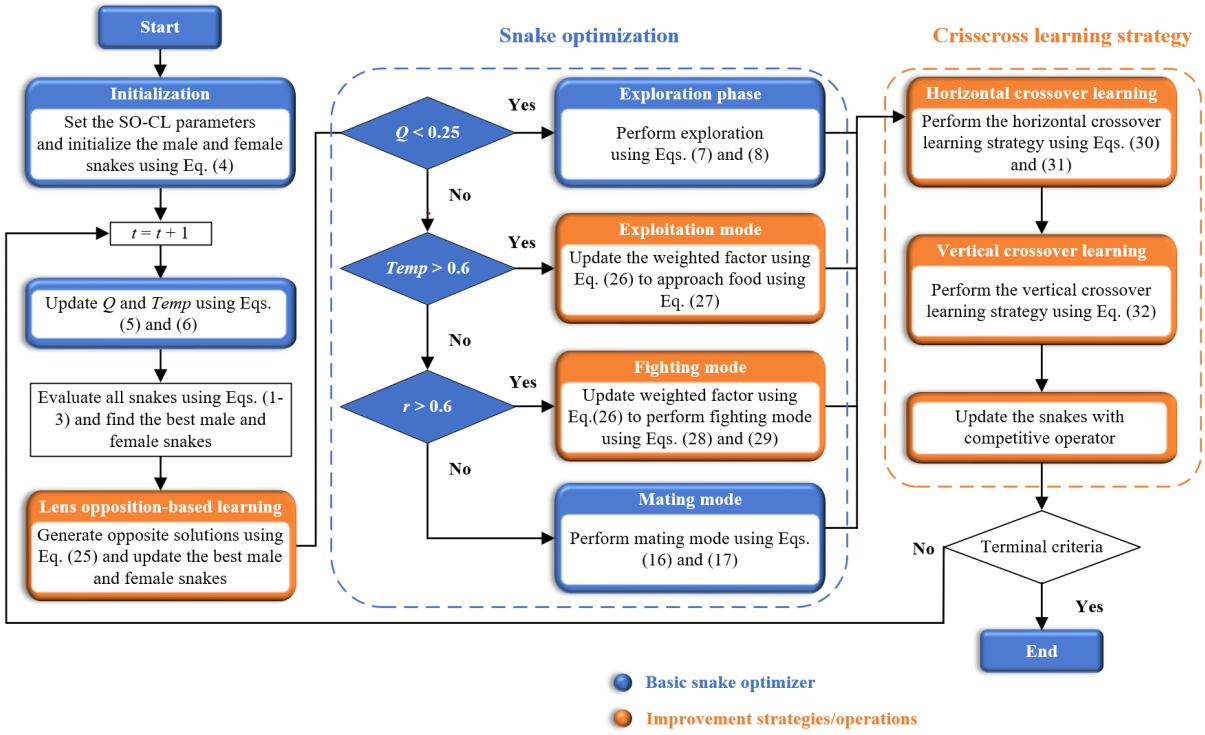


Fig. 1. The computational framework of the proposed *hybrid Snake Optimizer with Crisscross Learning strategy* (SO-CL) for nonlinear constrained structural optimization.

3.2.1. Lens-imaging learning strategy

In the exploitation phase of the basic SO, all searching agents are attracted by the current best male $X_{\mathrm{b}}^{(\mathrm{m})}$ and female $X_{\mathrm{b}}^{(\mathrm{f})}$ snakes, as explained by Eqs. (11-13). This means that the evolution of the entire population is guided by the current best individuals to search the promising areas towards the optimal solution. However, the optimization functions in structural design are often multimodal, with multiple local optima, which can make it challenging for the standard SO to find the global optimum. If the current best individuals are trapped in local optima, it can cause premature convergence of the entire population and a loss of population diversity. To address these issues, opposition-based learning and its variants [42] can be used to explore the solution space more

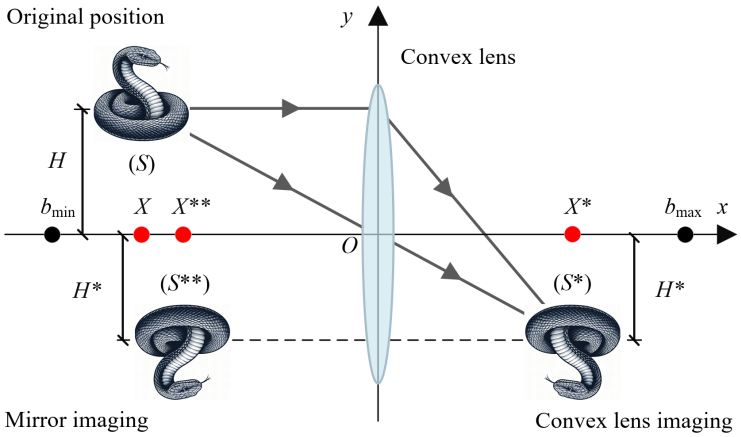


Fig. 2. Graphical illustration of the lens-imaging learning strategy.

effectively. Opposition-based learning is a machine learning technique that incorporates the concept of opposition to enhance the optimization process, thereby increasing the likelihood of finding the global optimum.

Here, the lens-imaging learning strategy [43] is adopted to generate the opposite of the current solution, which is the latest variant of opposition-based learning. This strategy first obtains opposition solutions through convex lens imaging. Then, it processes these solutions symmetrically within the search range to generate new solutions that are symmetric to the original opposition solutions. By using this strategy, the range of opposition solutions of convex lens imaging is extended, making it easier to escape local optima during the optimization process. As graphically illustrated in Fig. 2, the search range of the solution X is within $[b_{\min}, b_{\max}]$, and a convex lens is placed at the midpoint of this search range. Suppose one of the best snakes S at the original position has a projection X beyond the convex focal length on the x-axis, with a height of H in the y-axis direction. Through convex lens imaging, an inverted real image S^* is formed at position X^* on the right side of the x-axis, with the height of H^*. The final opposite snake S^{**} can be generated via mirror imaging, which is located on the left side of the x-axis with the same height H^*, where the projection satisfies $X^{**} = -X^*$ on the x-axis. After the above process, the opposition individuals S^* and S^{**} are obtained from the original snake S. According to the principles of convex lens imaging, the relationship between the original position X and the opposite site X^* can be derived as follows:

$$\frac{(b_{\max} + b_{\min})/2 - X}{X^* - (b_{\max} + b_{\min})/2} = \frac{H}{H^*}, \tag{22}$$

Let $H/H^* = sf$, where sf is called the scaling factor. The opposite location X^* can then be calculated as follows:

$$X^* = (b_{\max} + b_{\min})/2 + (b_{\max} + b_{\min})/2sf - X/sf. \tag{23}$$

Obviously, the projection points on the x-axis during the convex lens imaging process vary with the thickness of the convex lens, and the lens thickness can be defined as an adjustable factor, known as the scaling factor (sf). Adjusting this scaling factor aims to obtain more convex lens imaging solutions. In this work, a novel nonlinear dynamic method is adopted to adjust the scaling factor, which has two stages in its iterative loop. In the early stage, a large sf is used, which enables the algorithm to search more broadly across different dimensional areas, thereby improving the population diversity. In the later stage, the sf value becomes smaller, which allows the algorithm to perform a refined search near the optimal individual, thus enhancing the local search capability [43, 44]. This nonlinear dynamic adjustment method can be mathematically expressed as follows:

$$sf = sf_{\max} \times \left[1 - 2 \times (t/T)^2\right], \tag{24}$$

where $sf_{\max}$ denotes the maximum value of sf.

Generally, the lens-imaging learning strategy described by Eq. (23) can be extended to multi-dimensional space, yielding the following expression:

$$X^*_{i,j} = \left(b_{\max,j} + b_{\min,j}\right)\Big/2 + \left(b_{\max,j} + b_{\min,j}\right)\Big/2sf - X_{i,j}\Big/sf, \tag{25}$$

where $X^*_{i,j}$ and $X_{i,j}$ are the j-th dimensional components of X^*_i and X_i, respectively; and $b_{\max,j}$ and $b_{\min,j}$ are the j-th dimensional components of $b_{\max}$ and $b_{\min}$, respectively. From Eq. (25), it is evident that the formula becomes the traditional opposition-based learning strategy [42], when the scaling factor $sf = 1$. Therefore, the traditional opposition-based learning strategy is merely a special case of the lens-imaging learning strategy.

3.2.2. Adaptive weighting strategy

For various swarm intelligence optimization algorithms, it is still a great challenge to balance the global and local searching, and this balance is crucial for effective exploration and exploitation. Inertia weights have been proven effective in balancing global exploration and local exploitation in many studies [40, 45]. The weight factor controls an individual's dependence on its previous state during the updating process, significantly impacting convergence performance. Generally, a larger inertia weight can enhance global search capability and help explore new solution spaces in the early iterations. As the iterative computation progresses [46], it gradually approaches the optimal solution, and a smaller inertia weight can improve local search ability and facilitate fine searches in local areas.

In this work, an adaptive weighting strategy is developed to gradually reduce the inertia weight as the optimization progresses, aiming to achieve a smooth transition from global search to local search. The adaptive weights $\omega(t)$ can be calculated as follows:

$$\omega\left(t\right) = \left(\omega_{\max} - \omega_{\min}\right) \times \left(1 - \frac{t}{T}\right) + \omega_{\min}\,, \tag{26}$$

where $\omega_{\max}$ and $\omega_{\min}$ are the maximum and minimum values of the dynamic inertia weight, respectively; and they are set to 1 and 0 respectively in the case studies of this work.

By incorporating the adaptive weights into the standard SO, the equations (11), (12) and (13) can be rewritten, and the updated formulas with the integrated improvement strategies can be rewritten as follows:
The hot mode ($Q \geq 0.25$ and $T \geq 0.6$):

$$X_i(t+1) = \omega(t) \times X_{\text{food}} \pm c_3 \times Temp \times r_i \times \left(X_{\text{food}} - X_i(t)\right), \tag{27}$$

The fighting mode ($Q \geq 0.25$ and $T < 0.6$):

$$X_i^{(\text{m})}(t+1) = X_i^{(\text{m})}(t) + c_3 \times \phi_{\text{m}} \times r_i \times \left(\omega(t) \times Q \times X_{\text{b}}^{(\text{m})} - X_i^{(\text{m})}(t)\right), \tag{28}$$

$$X_i^{(\text{f})}(t+1) = X_i^{(\text{f})}(t) + c_3 \times \phi_{\text{f}} \times r_i \times \left(\omega(t) \times Q \times X_{\text{b}}^{(\text{f})} - X_i^{(\text{f})}(t)\right). \tag{29}$$

3.2.3. Crossover learning strategy

To prevent the optimization process from falling into local optima and the issues related to poor accuracy, a crossover learning strategy is integrated into the computational framework of the basic snake optimizer. This learning strategy was inspired by the excellent performance of crossover operators, as demonstrated by multiple studies [47, 48]. Crossover operators can simultaneously update snake positions at both the population and dimensional levels. The incorporation of crossover operators significantly enhances the optimizer's exploration and exploitation capabilities. Meng et al. [49] first proposed crossover operators, including the horizontal crossover (HC) operator and the vertical crossover (VC) operator. Crossover operators utilize information from previous samples to improve the learning of future search information [47]. Both operators develop moderated solutions called MS_{hc} and MS_{vc}. A competitive operator is then applied after each crossover search to update the population according to the evolutionary rule of "survival of the fittest".

The HC operator enhances the ability of social learning among individuals and facilitates global exploration by integrating information from different individuals. Specifically, it executes crossover operations across all dimensions between two different snakes. Assuming there are two last iteration parent snakes ($X(i,:)$ and $X(k,:)$) used to execute HC operator at the d-th dimension. The moderated solution MS_{hc} can be obtained as follows:

$$MS_{\text{hc}}(i,d) = r_1 \times X(i,d) + (1 - r_1) \times X(k,d) + c_1 \times \left(X(i,d) - X(k,d)\right), \tag{30}$$

$$MS_{\text{hc}}(k,d) = r_2 \times X(k,d) + (1 - r_2) \times X(i,d) + c_2 \times \left(X(k,d) - X(i,d)\right), \tag{31}$$

where $MS_{\text{hc}}(i,:)$ is the moderated solution who is the offspring of $X(i,:)$, and $MS_{\text{hc}}(k,d)$ is the moderated solution who is the offspring of $X(j,:)$; r_1 and r_2 are two random numbers uniformly distributed in the interval $[0,1]$; c_1 and c_2 are random values uniformly distributed in the interval $[-1,1]$. According to Eqs. (30) and (31), the HC operator is to sample new solutions within a hypercube formed by the parent solutions. By introducing the expansion factor c_1 and c_2, the operator is a higher probability of sampling near the center and a lower probability at the edges. This strategy reduces blind spots and ensures a more comprehensive exploration of the search space, significantly enhancing the capability of SO-CL to find the global optimal solutions.

Algorithm 2: The workflow of hybrid Snake Optimizer with Crisscross Learning strategy (SO-CL)

Input: Population size N, maximum number of function evaluations NFE, the dimension D of optimization problem, and the boundaries of design variables $[b_{\min}, b_{\max}]$.

Output: The optimal solution.

Initialization: Randomly initialize N snakes using Eq. (4), and then divide them into two groups: male and female; The maximum number of iterations $T = NFE/N$, and the current iteration step $t = 0$.

while $t \leq T$ **do**

- Evaluate the fitness of each group, including male and female;
- Find the best male $X_b^{(m)}$ and female $X_b^{(f)}$ snakes;
- Get the opposite positions of $X_b^{(m)}$ and $X_b^{(f)}$ using Eq. (25), which are denoted by $X_b^{(m)*}$ and $X_b^{(f)*}$, respectively;
- Compare the fitness values between $X_b^{(m)}$ and $X_b^{(m)*}$, and then set the better one as $X_b^{(m)}$;
- Compare the fitness values between $X_b^{(f)}$ and $X_b^{(f)*}$, and then set the better one as $X_b^{(f)}$;
- Update the environmental temperature $Temp$ and food supply Q using Eqs. (5) and (6), respectively;
- **if** $Q < 0.25$ **then**
  - Perform *exploration phase* using Eqs. (7) and (8);
- **else**
  - **if** $Temp \geq 0.6$ **then**
    - Update the adaptive weights using Eq. (26);
    - Perform *exploitation phase* using Eq. (27);
  - **else**
    - **if** $r \geq 0.6$ **then**
      - Update the adaptive weights using Eq. (26);
      - Perform *fighting mode* using Eqs. (28) and (29);
    - **else**
      - Perform *mating mode* using Eqs. (16) and (17);
    - **end**
  - **end**
- **end**
- B = permutate(N);
- **for** $i = 1$ to $N/2$ **do**
  - Let $i = B(2 \times i - 1)$ and $k = B(2 \times i)$;
  - **for** $d = 1$ to D **do**
    - Update four random values $r_1, r_2 \in [0, 1]$, $c_1, c_2 \in [-1, 1]$;
    - Update new snake positions based on horizontal operator using Eqs. (30) and (31);
  - **end**
- **end**
- B = permutate(D)
- **for** $d = 1$ to $D/2$ **do**
  - Generate a uniformly random value $p \in [0, 1]$;
  - Let $d = B(2 \times i - 1)$, and $d_1 = B(2 \times i)$;
  - **if** $p < P_V$ **then**
    - **for** $i = 1$ to N **do**
      - Update random values $r \in [0, 1]$, $c \in [-1, 1]$;
      - Update new snake positions based on vertical operator using Eq. (33);
    - **end**
  - **end**
- **end**
- **for** $i = 1$ to N **do**
  - Calculate the fitness value of $MS(i, :)$;
  - **if** $MS(i, :)$ is better than its parent X_i **then**
    - $X(i, :) = MS(i, :)$
  - **end**
- **end**

end

return: The optimal solution.

After the crossover operation, the presence of the new moderated solution brings competition with its parent solutions. The competitive operator employs a greedy selection mechanism, where the solution with superior fitness value is retained as the new generation, to update the parent solutions for the next iterations. A competitive operator is defined as follows:

$$X = \begin{cases} MS_{\text{hc}}, & \text{if } F(MS_{\text{hc}}) \leq F(X); \\ X, & \text{otherwise.} \end{cases} \tag{32}$$

where $F(\cdot)$ denotes the function to calculate fitness value.

The VC operator improves each individual's self-learning capability, thereby addressing the problem of premature stagnation by exchanging information within an individual's dimension. Specifically, it instigates the dimensional information exchange by performing a crossover operation within a snake individual, with this process occurring with a probability P_V. Assume that d and d_1 are different dimensions of a snake $X(i,:)$, and the moderated solution MS_{vc} can then be generated as follows:

$$MS_{\text{vc}}(i,d) = r \times X(i,d) + (1-r) \times X(i,d_1), \quad i \in (1,N), \quad d, d_1 \in (1,D), \tag{33}$$

where r is a random number uniformly distributed in the interval $[0,1]$. Before the VC operation, the parent solutions are normalized according to the top and low bounds of each design variable. According to Eq. (33), each VC operator updates only a single dimension of the snake. This strategy avoids disrupting other dimensions that can be the global optimum by focusing on escaping the local optimum of the stagnating dimension.

4. Numerical study

In this section, five well-known benchmark problems of constrained truss optimization are used to validate the effectiveness of the proposed SO-CL algorithm. These benchmark problems include the 10-bar planar, 25-bar spatial, 72-bar spatial, 120-bar dome, and 200-bar planar truss structures. The optimization results obtained from SO-CL are systematically compared with that obtained from a series of optimization algorithms (as listed in Table 1), in terms of the best solutions, average values, standard deviations, and the number of structural analyses performed. To guarantee the accuracy of optimization results, the maximum numbers of structural analyses are larger than 20,000 for all case studies. Due to the computational complexity of the 200-bar truss case, the maximum number of structural analyses is set to 75,000 after multiple manual adjustments.

Table 1: The list of metaheuristic optimization algorithms involved in the comparative study

Optimization algorithm	**Parameter settings**	**Reference**
TLBO	$N = 30$ and $nd = 4$	[14]
CA	$N = 10$ and $bs = 4$	[50]
EHOC	$N = 40$, $\alpha = 2.5$ and $\beta = 0.05$	[30]
ECSA	$N = 70$, $fl = 3$ and $AP = 0.1$	[51]
WSA	$N = 10$, $\tau = 0.8$, $\lambda = 0.78$ and $\phi = 0.001$	[22]
ACCS	$N = 3 \times D$	[52]
PO	$N = 11$, $\lambda = 0.5$, $N = 90/60$ and $P_0 = 0.1$	[23]
BO	$N = 90$, $ts = 0.04$, $\alpha = 1.35$, $\beta = 1.40$ and $rcp = 0.0036$	[25]
MCOA	$N_p = 10$ and $N_c = 5$	[33]
SO	$N = 50$	[36]
CSO	$N = 50$ and $P_V = 0.8$	[49]
SO-CL	$N = 50$ and $P_V = 0.8$	(Proposed)

As listed in Table 1, the metaheuristic optimization algorithms used for comparison in this work are summarized. These algorithms can be roughly categorized into the following three groups: (i) The original algorithms including the Snake optimizer (SO) [36], and the Crisscross optimizer (CSO) [49]; (ii) The algorithms inspired by physics and human's behaviors such as the Teaching learning-based optimization (TLBO) [14], the Weighted superposition attraction algorithm (WSA) [22], the Artificial coronary circulation system algorithm (ACCS) [52], the Cultural algorithm (CA) [50], and the Political optimizer (PO) [23]; and (iii) The Nature-inspired algorithms including the Elephant herding optimization cultural algorithm (EHOC) [30], the Enhanced crow search algorithm (ECSA) [51], the Bonobo optimizer (BO) [25], and the Chaotic coyote algorithm (MCOA) [33]. The parameter settings of these algorithms follow the suggestions in the corresponding original literature, including population size (N), problem dimension (D), the number of designs generated in the learning phase (nd), belief space (bs),

flight length (fl), awareness probability (AP), party switching rate (λ), the initial probability of the extra group mating (P_0), the sharing coefficients for the alpha bonobo (α) and selected bonobo (β), the rate of the change in the phase probability (rcp), the maximum value for the temporary sub-group size factor (ts), the number of coyotes (N_c), and the number of packs (N_p). As to the proposed SO-CL algorithm, vertical crossover probability (P_V) is one of the key parameters, and it is set to 0.8 according to the suggestion in Ref. [49].

Considering the stochastic nature of metaheuristic optimization algorithms, each algorithm is independently run for 30 times, and the obtained results are then compared for performance evaluation in terms of mean values and standard deviations. To ensure a fair comparison, all optimization algorithms are implemented in the MATLAB software environment. These benchmark truss structures are analyzed using the direct stiffness method [53], and the computational results are obtained from the same computer with a Core(TM) i5-12400 CPU at 2.5 GHz.

Table 2: Material properties and cross-sectional area bounds for numerical case studies

Case study	**Material density** ρ (lb/in^3)	**Cross-sectional area limits** (in^2)	**Young's modulus** E (psi)
10-bar planar truss	0.100	$0.1 \le A \le 35.0$	1×10^7
25-bar spatial truss	0.100	$0.01 \le A \le 3.40$	1×10^7
72-bar spatial truss	0.100	$0.1 \le A \le 4.0$	1×10^7
120-bar dome truss	0.288	$0.775 \le A \le 20.000$	3.045×10^7
200-bar planar truss	0.283	$0.1 \le A \le 20.0$	3×10^7

4.1. Case 1: A 10-bar planar truss

As shown in Fig. 3, the first benchmark problem is the optimal design of a 10-bar truss structure, and it has been investigated in many previous studies [12, 41, 35]. In this constrained optimization problem, cross-sectional areas of all ten bars are treated as the continuous design variables. A load with $P = 100$ kips is applied to the 2nd and 4th nodes of the truss structure. The material properties and cross-sectional area limits are summarized in Table 2. Stress constraints are set for each bar, and displacement constraints are set for each node in both the x and y directions. The range of allowable stress is from -25 ksi to 25 ksi, and the allowable displacement is $\pm$ 2 in (inch). Totally, there are 32 nonlinear design constraints in this constrained optimization design problem, including 10 tensile constraints, 10 compression constraints and 12 displacement constraints.

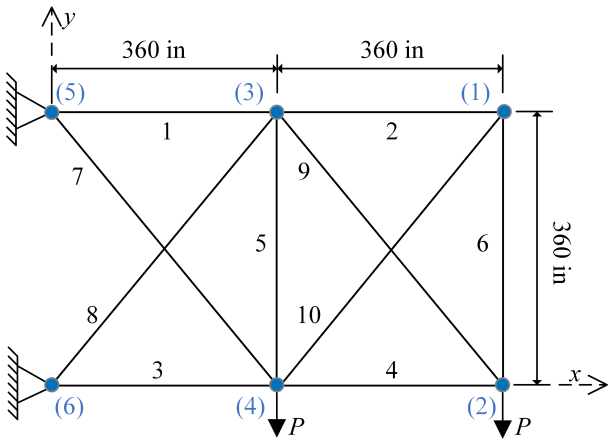


Fig. 3. The schematic diagram of a 10-bar planar truss (in denotes inch)

Table 3: The design results for the 10-bar planar truss using different optimization algorithms

Design results	TLBO	ECSA	ACCS	PO	BO	SO	CSO	**SO-CL**
A_1 (in^2)	30.4286	30.5096	30.6460	30.5010	30.5455	30.6155	30.5587	30.5218
A_2 (in^2)	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000
A_3 (in^2)	23.2436	23.2253	23.1030	23.1980	23.2106	23.1216	23.1825	23.1999
A_4 (in^2)	15.3677	15.2315	15.0630	15.2470	15.2091	15.2042	15.2711	15.2229
A_5 (in^2)	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000
A_6 (in^2)	0.5751	0.5517	0.5730	0.5551	0.5489	0.5502	0.5425	0.5514
A_7 (in^2)	7.4404	7.4561	7.4780	7.4562	7.4574	7.4680	7.4636	7.4572
A_8 (in^2)	20.9665	21.0276	21.0940	21.0350	21.0359	21.0422	21.0625	21.0364
A_9 (in^2)	21.5330	21.5239	21.5320	21.5260	21.5159	21.0422	20.14527	21.5284
A_{10} (in^2)	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1001	0.1000
Best weight (lb)	5060.960	5060.910	5061.030	5060.850	5060.856	5060.876	5060.907	5060.853
Mean weight (lb)	5064.81	5063.41	5061.07	5061.23	5060.88	5061.69	5078.28	5061.18
Standard deviation (lb)	6.371	5.430	0.090	0.530	0.036	0.800	9.485	0.910
Number of structural analyses	13767	16401	12000	7920	62280	19560	17280	7140

As listed in Table 3, the design results obtained from the proposed SO-CL and other seven optimization algorithms are recorded. The total weight of the bars optimized by SO-CL is 5060.853 lb, which is very close to the best design result (5060.850 lb) obtained from PO. And SO-CL exhibits better performance than the remaining six algorithms in terms of the bar weight, whose design results are 5060.96 lb (TLBO), 5060.91 lb (ECSA), 5061.03 lb (ACCS), 5060.855 lb (BO), 5060.876 lb (SO) and 5060.907 lb (CSO), respectively. The developed SO-CL algorithm reaches its optima only after 7140 structural analyses, which is much fewer than the other algorithms (TLBO: 13767 analyses, ECSA: 16401 analyses, ACCS: 12000 analyses, BO: 62280 analyses, SO: 19560 analyses, and CSO: 17280 analyses). Besides, SO-CL is much more stable than TLBO, ECSA and CSO, as demonstrated by the mean weight and standard deviation results.

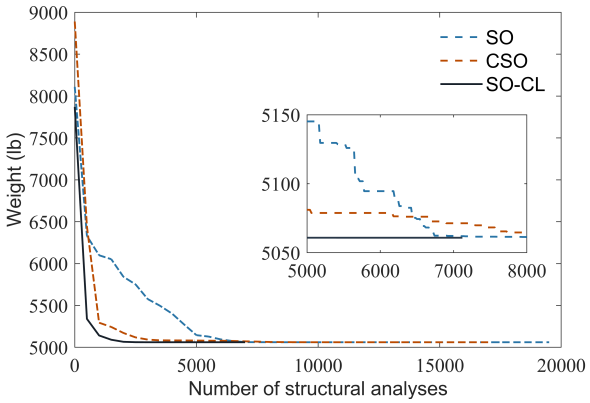
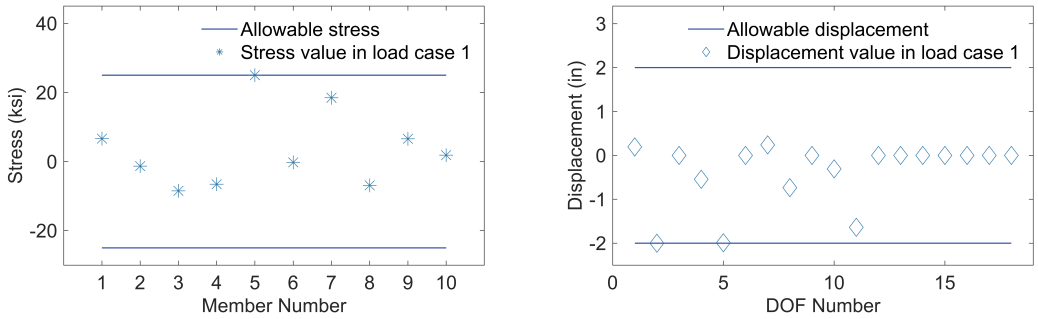


Fig. 4. Comparison of convergence rates between SO, CSO and the proposed SO-CL algorithms (case 1)

As shown in Fig. 4, the convergence rate of SO-CL is graphically compared with that of SO and CSO. Obviously, SO-CL converges faster than both CSO and SO-CL, and it reaches its stable state only after 7140 analyses, which demonstrates that SO-CL is a cost-effective approach. In Fig. 5, the stress and displacement states corresponding to the optimal design results obtained from SO-CL are also plotted. It is clear that all of the predefined constraints are strictly satisfied, which confirms the effectiveness of SO-CL. In summary, the proposed SO-CL algorithm not just exhibits a stronger solution-searching capability than the standard SO, and it also retains the computational stability of SO.

4.2. Case 2: A 25-bar spatial truss tower

The second case study is a 25-bar spatial truss tower (as shown in Fig. 6), which is composed of 25 bar members and 10 junction nodes. The cross-sectional areas (design variables) of all bars are divided into 8 groups for structural optimization design. The material properties and cross-sectional area limits have been summarized in Table 2. The maximum allowable tensile stress is 40 ksi for all bars; while the maximum allowable compressive



(a) Stress constraints **(b)** Displacement constraints

Fig. 5. The constraint conditions are strictly satisfied by using SO-CL for structural optimization (case 1)

Table 4: Loading conditions for the 25-bar spatial truss tower

Node index	Loading condition 1			Loading condition 2		
	x	y	z	x	y	z
1	1	10	-5	0	20	-5
2	0	10	-5	0	-20	-5
3	0.5	0	0	0	0	0
6	0.5	0	0	0	0	0

stress varies with the buckling strengths of bars in different groups, with the values of 35.092 ksi for the A_1 group (bar 1), 11.590 ksi for the A_2 group (bar 2-5), 17.305 ksi for the A_3 group (bar 6-9), 35.092 ksi for the A_4 group (bar 10-11), 35.092 ksi for the A_5 group (bar 12-13), 6.759 ksi for the A_6 group (bar 14-17), 6.9590 ksi for the A_7 group (bar 18-21), and 11.082 ksi for the A_8 group (bar 22-25). The displacement of all nodes is limited to ± 0.35 inches. Here, structural optimization is carried out for the truss tower under two different loading conditions, as listed in Table 4. In summary, structural optimization of this truss tower has totally 124 nonlinear design constraints under each loading condition, including 50 tensile stress constraints, 50 compressive stress constraints, 12 vertical displacement constraints and 12 horizontal displacement constraints.

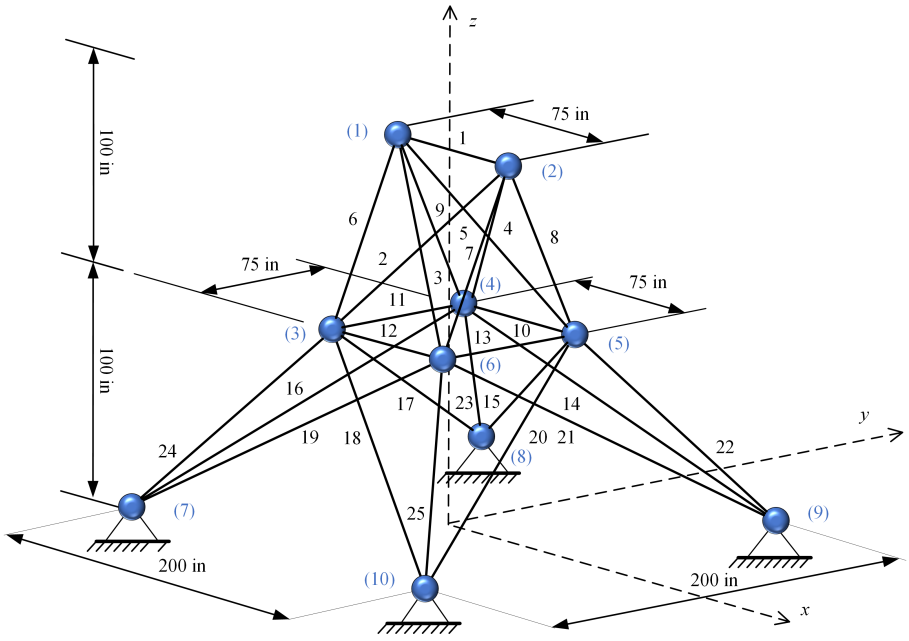


Fig. 6. The schematic diagram of a 25-bar spatial truss tower (in denotes inch)

The proposed SO-CL together with other seven metaheuristic algorithms are applied to this case, and the optimization design results are summarized in Table 5 for comparison. The total weight of bars optimized by SO-CL (545.162 lb) is slightly lighter than that obtained from ACCS (545.163 lb), PO (545.165 lb), SO (545.216

lb) and CSO (545.255 lb). Although BO (541.645 lb) provides the lightest designed bars, but its design result violates the displacement constraints at node 1 and 2. Besides, the design results obtained from TLBO and EHOC also violate the stress constraints. Furthermore, the presented SO-CL algorithm has a higher converge rate and it reaches the optima only after 6600 analyses, compared to TLBO, ACCS, PO, SO and CSO. By independently performing each optimization algorithm 30 times, the mean and standard deviation of the truss weight results can be calculated, as listed in Table 5. The standard deviation of the truss weight results computed by SO-CL is 0.0034, demonstrating its computational stability.

Table 5: The design results for the 25-bar spatial truss tower using different optimization algorithms

Design results	TLBO	EHOC	ACCS	PO	BO	S0	CSO	**SO-CL**
A_1 (in^2)	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100
A_2 (in^2)	2.0712	1.9945	1.9825	1.9797	2.1912	2.0307	1.9715	1.9873
A_3 (in^2)	2.9570	2.9816	3.0004	3.0052	3.0905	2.9289	2.9941	2.9932
A_4 (in^2)	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100
A_5 (in^2)	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	0.0102	0.0100
A_6 (in^2)	0.6891	0.6884	0.6833	0.6817	0.6135	0.6861	0.6961	0.6837
A_7 (in^2)	1.6209	1.6756	1.6776	1.6779	1.7198	1.6704	1.6902	1.6768
A_8 (in^2)	2.6768	2.6598	2.6610	2.6617	2.3564	2.6780	2.6439	2.6626
Best weight (lb)	545.090	545.148	545.163	545.165	541.645	545.216	545.255	545.162
Mean weight (lb)	545.410	545.660	545.166	545.450	541.671	545.281	546.234	545.166
Standard deviation (lb)	0.4200	0.5600	0.00162	0.2100	0.0184	0.0837	0.9950	0.0034
Number of structural analyses	15318	6640	10000	8316	5700	10140	19680	6660

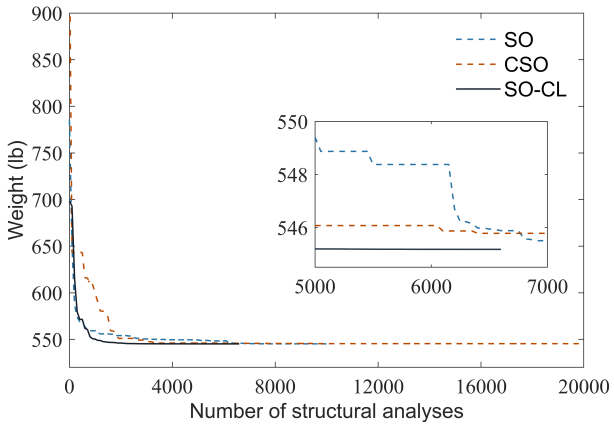
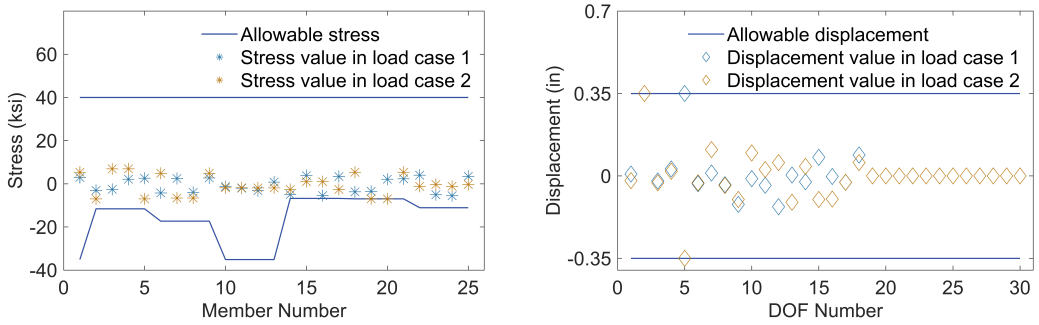


Fig. 7. Comparison of convergence rates between SO, CSO and the proposed SO-CL algorithms (case 2)



(a) Stress constraints **(b)** Displacement constraints

Fig. 8. The constraint conditions are strictly satisfied by using SO-CL for structural optimization (case 2)

In Fig. 7, the proposed SO-CL algorithm is directly compared with SO and CSO in terms of convergence rate. Apparently, SO-CL converges faster than SO and CSO, and it can reach a stable value only after 2000 structural analyses, which greatly saves computational cost. As illustrated in Fig. 8, all predefined stress and displacement

constraints have been strictly satisfied for structural optimization using SO-CL, even under two distinct working conditions.

Table 6: Loading conditions for the 72-bar spatial truss tower

Node index	Loading condition 1			Loading condition 2		
	x	y	z	x	y	z
17	5.0	5.0	-5.0	0	0	-5.0
18	0	0	0	0	0	-5.0
19	0	0	0	0	0	-5.0
20	0	0	0	0	0	-5.0

4.3. Case 3: A 72-bar spatial truss tower

The third benchmark problem is the structural optimization of a 72-bar spatial truss tower, as can be seen in Fig. 9. This truss tower consist of 72 bar members and 24 junction nodes, and the cross-sectional areas of all bars are the design variables. To solve this optimization problem, these 72 bar members are divided into 16 groups, given by: group A_1 (bars 1–4), group A_2 (bars 5–12), group A_3 (bars 13–16), group A_4 (bars 17–18), group A_5 (bars 19–22), group A_6 (bars 23–30), group A_7 (bars 31–34), group A_8 (bars 35–36), group A_9 (bars 37–40), group A_{10} (bars 41–48), group A_{11} (bars 49–52), group A_{12} (bars 53–54), group A_{13} (bars 55–58), group A_{14} (bars 59–66), group A_{15} (bars 67–70) and group A_{16} (bars 71–72). Table 2 provides the material properties and cross-sectional area limits. As to the constraints of structural optimization, the allowable maximum tensile and compressive stresses are 25 ksi, and the allowable displacements in all directions at all nodes are ±0.25 inches. Two different loading conditions are considered for the optimal design of this truss tower, and more details are given in Table 6. In summary, there are 264 nonlinear design constraints in this structural optimization problem, including 72 tensile stress constraints, 72 compressive stress constraints and 120 displacement constraints.

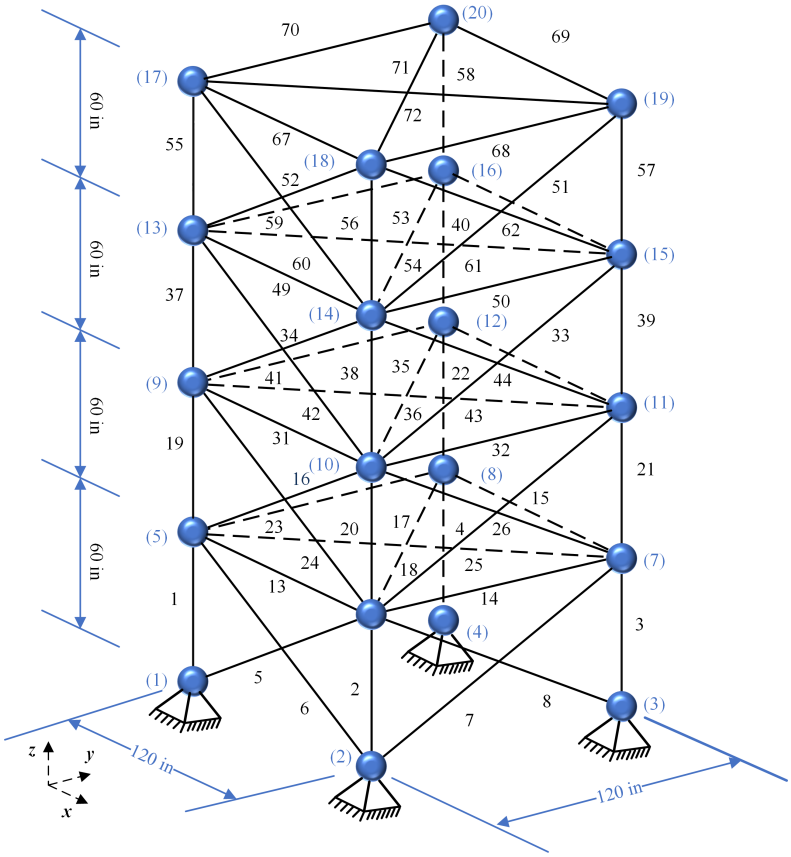


Fig. 9. The schematic diagram of a 72-bar spatial truss tower (in denotes inch)

Table 7: The design results for the 72-bar spatial truss tower using different optimization algorithms

Design results	TLBO	EHOC	ACCS	PO	BO	SO	CSO	**SO-CL**
A_1 (in^2)	1.9064	1.8911	1.8665	1.8607	1.8904	1.8826	1.8704	1.8875
A_2 (in^2)	0.5061	0.5051	0.5062	0.5063	0.5138	0.5101	0.5107	0.5118
A_3 (in^2)	0.1000	0.1000	0.0100	0.1000	0.1000	0.1000	0.1000	0.1000
A_4 (in^2)	0.1000	0.1000	0.1002	0.1000	0.1000	0.1000	0.1000	0.1000
A_5 (in^2)	1.2617	1.2778	1.2909	1.2749	1.2652	1.986	1.3016	1.2656
A_6 (in^2)	0.5111	0.5082	0.5177	0.5095	0.5112	0.5267	0.5263	0.5102
A_7 (in^2)	0.1000	0.1004	0.1001	0.1000	0.1000	0.1000	0.1013	0.1000
A_8 (in^2)	0.1000	0.1000	0.1008	0.1001	0.1000	0.1000	0.1017	0.1000
A_9 (in^2)	0.5317	0.5297	0.5148	0.5287	0.5228	0.5041	0.5067	0.5237
A_{10} (in^2)	0.5159	0.5343	0.5196	0.5236	0.5160	0.5305	0.5172	0.5178
A_{11} (in^2)	0.1000	0.1000	0.1001	0.1001	0.1000	0.1000	0.1000	0.1000
A_{12} (in^2)	0.1000	0.1000	0.1014	0.1000	0.1000	0.1000	0.1003	0.1000
A_{13} (in^2)	0.1562	0.1566	0.1567	0.1563	0.1565	0.1598	0.1575	0.1564
A_{14} (in^2)	0.5493	0.5431	0.5447	0.5500	0.5460	0.5330	0.5291	0.5476
A_{15} (in^2)	0.4097	0.4069	0.4176	0.4013	0.4105	0.4261	0.4116	0.4091
A_{16} (in^2)	0.5698	0.5500	0.5601	0.5858	0.5681	0.5860	0.5822	0.5703
Best weight (lb)	379.630	379.720	379.751	379.680	379.617	379.647	379.899	379.619
Mean weight (lb)	380.200	380.390	379.810	379.790	379.626	380.914	383.422	379.732
Standard deviation (lb)	0.410	0.540	0.148	0.139	0.008	1.139	6.341	0.241
Number of structural analyses	19709	8400	12000	7788	33570	10200	18120	8960

The proposed SO-CL and other seven optimization algorithms are applied to this truss tower to compute the design results, as recorded in Table 7. By comparison, it is found that SO-CL provides one of the best design results. The bar members optimally designed by SO-CL have a total weight of 379.619 lb, which is lighter than that obtained from other optimization algorithms: TLBO (379.630 lb), EHOC (379.720 lb), ACCS (379.751 lb), PO (379.680 lb), SO (380.054 lb) and CSO (379.899 lb). As to computational efficiency, SO-CL requires fewer structural analyses (8960 analyses) than TLBO, ACCS, BO, SO and CSO. All these optimization algorithms are independently run 30 times in this case study, and the low value of standard deviation (0.241) demonstrates the computational stability of SO-CL, especially compared with the standard SO.

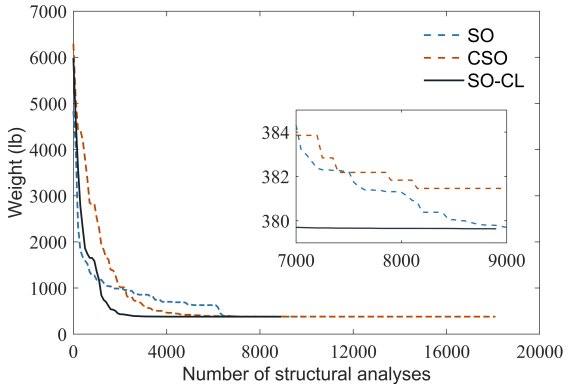
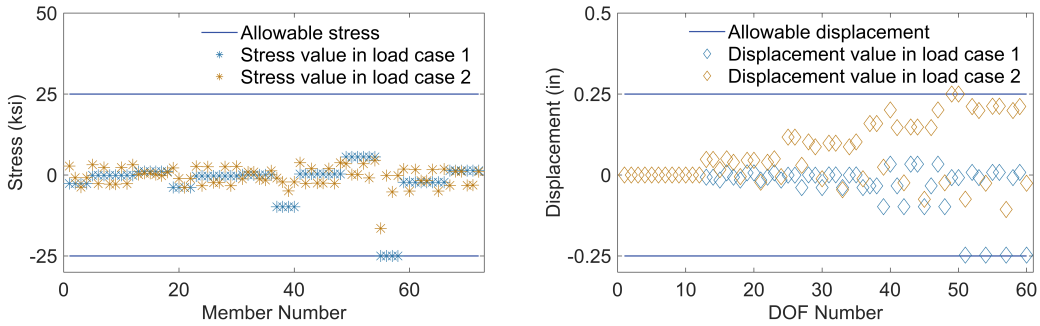


Fig. 10. Comparison of convergence rates between SO, CSO and the proposed SO-CL algorithms (case 3)

In Fig. 10, the proposed SO-CL algorithm is graphically compared with SO and CSO in terms of convergence rate. It can be clear seen that SO-CL can quickly reach the optima via much fewer structural analyses, demonstrating its high computational efficiency. In addition, the constraints corresponding to the optimization design results of SO-CL are also graphically illustrated in Fig. 11. Both the predefined stress and displacement constraints are strictly met during the optimization process, which further confirms the effectiveness of the proposed SO-CL algorithm in solving constrained optimization problems.



(a) Stress constraint **(b)** Displacement constraint

Fig. 11. The constraint conditions are strictly satisfied by using the SO-CL algorithm (case 3)

4.4. Case 4: A 120-bar spatial dome truss

A 120-bar spatial dome truss is used as the fourth benchmark case to test the effectiveness of the proposed SO-CL algorithm, as shown in Fig. 12. This truss structure consists of 120 bar members that are connected by 49 junction nodes. The cross-sectional areas of these bars are treated as the continuous design variables, and they are divided into seven groups according to the bar length. The material properties and cross-sectional area limits have been summarized in Table 2. The stress constraints for optimization design is defined based on the AISC-ASD (1989) code [54], given by:

$$\begin{cases} \sigma_i^+ = 0.6\sigma_{\mathrm{y}}, & \text{if } \sigma_i \geq 0 \\ \sigma_i^-, & \text{if } \sigma_i < 0 \end{cases} \tag{34}$$

where σ_{y} is the yield stress; σ_i^+ is the stress of the i-th bar when it is in tension; σ_i is the actual stress of the i-th bar; and σ_i^- can be calculated based on elastic and inelastic buckling:

$$\sigma_i^- = \begin{cases} \dfrac{\left(1-\lambda_i^2/2C_{\mathrm{c}}^2\right)\sigma_{\mathrm{y}}}{\left(5/3+3\lambda_i/8C_{\mathrm{c}}-\lambda_i^3/8C_{\mathrm{c}}^3\right)}, & \text{if } \lambda_i < C_{\mathrm{c}} \text{ (inelastic buckling)} \\ \dfrac{120\pi^2 E}{23\lambda_i^2}, & \text{if } \lambda_i \geq C_{\mathrm{c}} \text{ (elastic buckling)} \end{cases} \tag{35}$$

where E represents the elastic modulus; C_{c} is critical slenderness ratio used to determine whether the bar is in elastic or inelastic buckling $\left(C_{\mathrm{c}} = \sqrt{2\pi^2 E/f_{\mathrm{y}}}\right)$; λ_i is the slenderness ratio ($\lambda_i = k_i L_i/r_i$); k_i represents the effective length factor ($k_i = 1$ is assumed for all truss bars in a pin-pin scenario); r_i represents the radius of gyration, which can be defined as $r_i = aA_i^b$, where a and b are constants related to the types of selected sections; in this study, $a = 0.4993$ and $b = 0.4993$ are considered, referring to the previous studies [30, 51].

As to structural displacement constraints, the maximum allowable displacement of all nodes is limited to ±0.1969 inches in all directions. In summary, there are totally 387 nonlinear constraints for the structural optimization design, including 120 tensile stress constraints, 120 compressive stress constraints and 147 displacement constraints. The constrained structural optimization is carried out under the following loading condition: a vertical load of -13.49 kips applied to the node 1, a vertical loads of -6.744 kips applied to the nodes 2 through 14, and vertical loads of 2.248 kips applied to the remaining nodes.

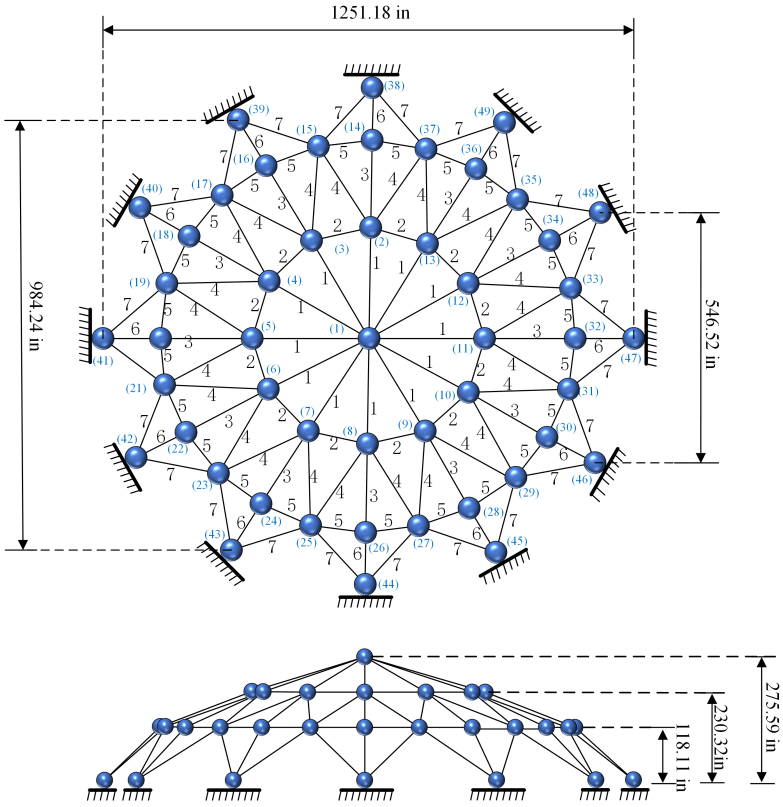


Fig. 12. The schematic diagram of a 120-bar spatial dome truss structure (in denotes inch)

Table 8: The design results for the 120-bar spatial dome truss using different optimization algorithms

Design results	CA	EHOC	ECSA	ACCS	WSA	SO	CSO	**SO-CL**
A_1 (in^2)	3.02591	3.0245	3.0241	3.0243	3.02409	3.0241	3.0260	3.0242
A_2 (in^2)	14.7652	14.6932	14.7998	14.7924	14.78262	14.7398	14.5411	14.7863
A_3 (in^2)	5.08463	5.1066	5.0779	5.0594	5.05121	5.0576	5.2736	5.0777
A_4 (in^2)	3.13569	3.1252	3.1358	3.1361	3.13703	3.1383	3.1344	3.1366
A_5 (in^2)	8.43852	8.5087	8.4737	8.4828	8.49807	8.5147	8.4123	8.4793
A_6 (in^2)	3.35678	3.3016	3.2819	3.2984	3.29158	3.2900	3.3434	3.2824
A_7 (in^2)	2.49627	2.4995	2.4964	2.4967	2.49676	2.4967	2.4966	2.4965
Best weight (lb)	33253.95	33250.37	33249.5	33250.28	33249.75	33249.91	33263.48	33249.43
Mean weight (lb)	N/A	33267.4	33251.02	33255.27	33249.79	33252.27	33376.62	33258.49
Standard deviation (lb)	N/A	62.850	3.220	3.113	0.041	2.323	170.341	7.819
Number of structural analyses	5800	5920	13582	5700	10000	10240	14760	9400

The proposed SO-CL together with other seven metaheuristic algorithms are used to solve the constrained optimization problem. The optimization design results obtained from different algorithms are summarized in Table 8. By comparison, the best design results is derived from SO-CL, and the corresponding designed bars (33,249.43 lb) are lighter than that computed from the other seven algorithms. Besides, SO-CL only requires 9400 structural analyses to search the optima, which is fewer than ECSA (13582 analyses), WSA (10000 analyses), SO (10240 analyses) and CSO (14740 analyses), demonstrating its higher convergence rate. Furthermore, after independently performing SO-CL 30 times, we find its optimization design results are quite stable with a low value of standard deviation, as recorded in Table 8.

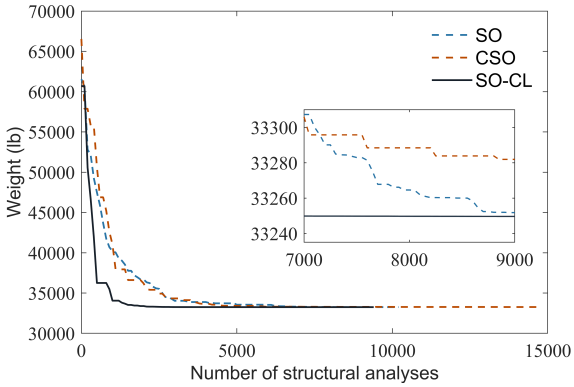
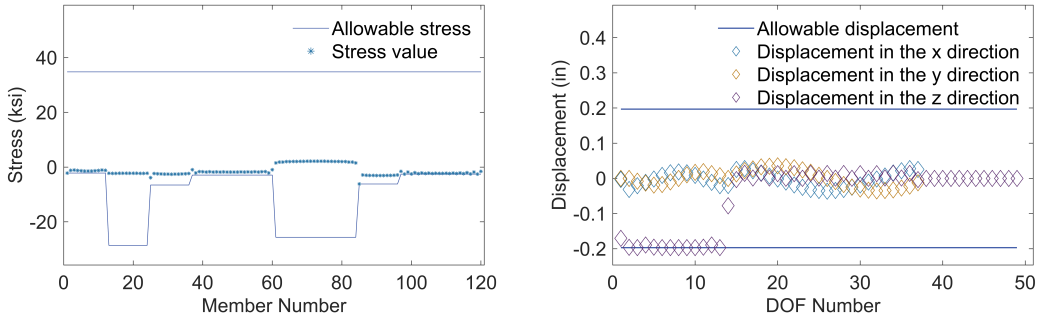


Fig. 13. Comparison of convergence rates between SO, CSO and the proposed SO-CL algorithms (case 4)

As shown in Fig. 13, the convergence curves of SO, CSO and the proposed SO-CL algorithms are graphically compared. Obviously, SO-CL converged much faster than SO and CSO, which indicates the present algorithm is a computationally efficient optimizer. In Fig. 14, the constraint results corresponding to the optimization design derived from SO-CL are graphically illustrated. It can be seen that the buckling stress constraints and the displacement constraints at the top of the roof play the major controlling role and all constraints are strictly satisfied. The high accuracy of optimization results and high rate of convergence demonstrate the effectiveness of the proposed SO-CL algorithm in solving constrained optimization problems.



(a) Stress constraints **(b)** Displacement constraints

Fig. 14. The constraint conditions are strictly satisfied by using the SO-CL algorithm (case 4)

4.5. Case 5: A 200-bar planar truss

The proposed SO-CL algorithm is also applied to a complicated case with more design variables and optimization constraints. As shown in Fig. 15, a planar truss structure consisting of 200 bar members and 77 junction nodes is investigated here. The elementary bars are classified into 29 groups according to their length, as listed in Table 9, and it means that there will be 29 design variables (cross-sectional areas) in the relevant optimization problem. Table 2 lists the material properties and cross-sectional area limits. As to constraint conditions, both the tensile and compressive stress constraints are set to 10 ksi, and no displacement constraint is considered in this case study. Totally, this optimization design problem considers 400 nonlinear constraints, including 200 tensile stress and 200 compressive stress constraints. Three different loading conditions are independently considered for the structural optimization of this 200-bar truss, and the details are given as followings: (1) Horizontal loads (positive x-direction) with the magnitude of 1.0 kip are applied to nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, and 71; (2) Vertical loads (negative y-direction) with the magnitude of 10.0 kip are applied to nodes 1-6, 8, 10, 12, 14-20, 22, 24, 26, 28-34, 36, 38, 40, 42-48, 50, 52, 54, 56-62, 64, 66, 68, and 70-75; (3) Combination of the above two loading conditions is applied to this truss structure.

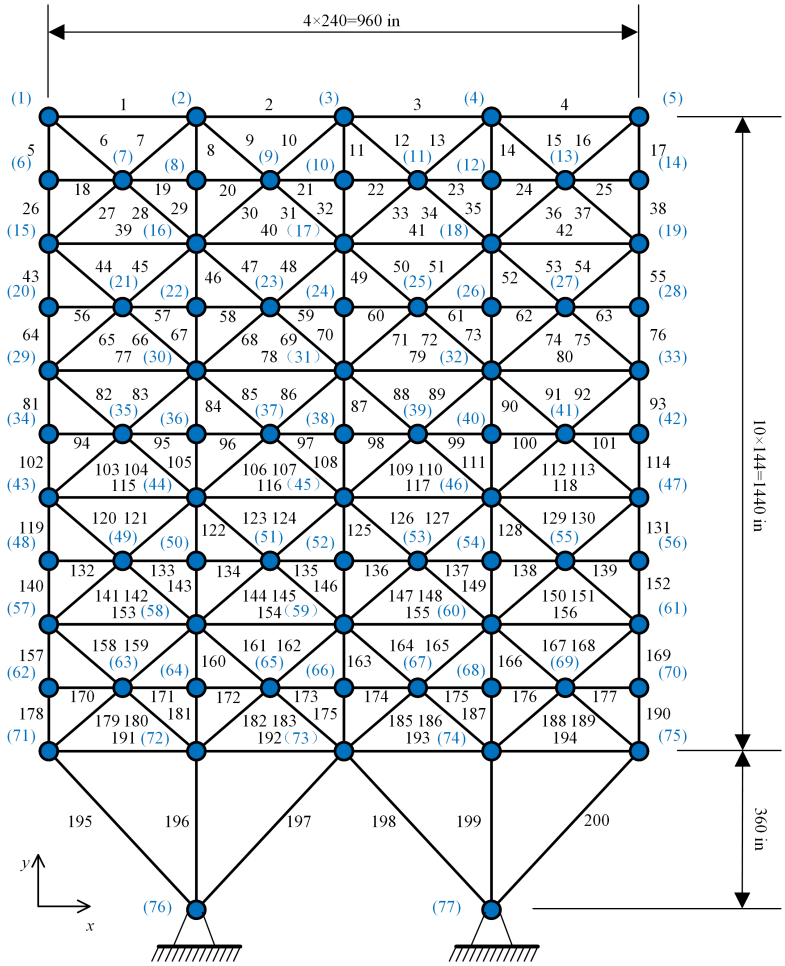


Fig. 15. The schematic diagram of a 200-bar planar truss (in denotes inch)

Table 9: Design variables for structural optimization of the 200-bar planar truss

Design variables	Bar member group	Design variables	Bar member group
A_1	1, 2, 3, 4	A_{16}	82, 83, 85, 86, 88, 89, 91, 92, 103, 104, 106, 107, 109, 110, 112, 113
A_2	5, 8, 11, 14, 17	A_{17}	115, 116, 117, 118
A_3	19, 20, 21, 23, 24	A_{18}	119, 122, 125, 128, 131
A_4	18, 25, 56, 63, 94, 101, 132, 139, 170, 177	A_{19}	133, 134, 135, 136, 137, 138
A_5	26, 29, 32, 35, 38	A_{20}	140, 143, 146, 149, 152
A_6	6, 7, 9, 10, 12, 13, 15, 16, 27, 28, 30, 31, 33, 34, 36, 37	A_{21}	120, 121, 123, 124, 126, 127, 129, 130, 141, 142, 144, 145
A_7	39, 40, 41, 42	A_{22}	147, 148, 150, 151
A_8	43, 46, 49, 52, 55	A_{23}	153, 154, 155, 156
A_9	57, 58, 59, 60, 61, 62	A_{24}	157, 160, 163, 166, 169, 171, 172, 173, 174, 175, 176
A_{10}	64, 67, 70, 73, 76	A_{25}	178, 181, 184, 187, 190
A_{11}	44, 45, 47, 48, 50, 51, 53, 54, 65, 66, 68, 69, 71, 72, 74, 75	A_{26}	158, 159, 161, 162, 164, 165, 167, 168, 179, 180, 182, 183, 185, 186, 188, 189
A_{12}	77, 78, 79, 80	A_{27}	191, 192, 193, 194
A_{13}	81, 84, 87, 90, 93	A_{28}	195, 197, 198, 200
A_{14}	95, 96, 97, 98, 99, 100	A_{29}	196, 199
A_{15}	102, 105, 108, 111, 114		

Table 10: The design results for the 200-bar planar truss using different optimization algorithms

Design results	TLBO	ACCS	WSA	PO	MCOA	SO	CSO	**SO-CL**
A_1 (in^2)	0.1460	0.1270	0.1448	0.1391	0.139	0.1494	0.1374	0.1459
A_2 (in^2)	0.9410	0.9540	0.9431	0.9628	0.9355	0.9077	0.9738	0.9436
A_3 (in^2)	0.1000	0.1220	0.1012	0.1100	0.1000	0.1033	0.1059	0.1002
A_4 (in^2)	0.1010	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000
A_5 (in^2)	1.9410	1.9680	1.9431	1.9430	1.9355	2.0740	1.9404	1.9436
A_6 (in^2)	0.2960	0.2930	0.2963	0.2953	0.2909	0.2764	0.2946	0.2960
A_7 (in^2)	0.1000	0.1150	0.1033	0.1001	0.1000	0.4838	0.1098	0.1000
A_8 (in^2)	3.1210	3.0950	3.1144	3.0987	3.0816	3.0220	3.1090	3.1157
A_9 (in^2)	0.1000	0.1030	0.1025	0.1376	0.1000	0.1001	0.1052	0.1000
A_{10} (in^2)	4.1730	4.0940	4.1144	4.0992	4.0816	4.0183	4.1090	4.1158
A_{11} (in^2)	0.4010	0.4100	0.4004	0.4193	0.3967	0.5144	0.4119	0.3977
A_{12} (in^2)	0.1810	0.1520	0.1140	0.1559	0.2959	0.2444	0.1719	0.1135
A_{13} (in^2)	5.4230	5.3930	5.3886	5.4344	5.3854	5.4398	5.4140	5.3874
A_{14} (in^2)	0.1000	0.1980	0.1000	0.1044	0.1000	0.2891	0.1034	0.1000
A_{15} (in^2)	6.4220	6.3950	6.3886	6.4355	6.3853	6.4413	6.4140	6.3874
A_{16} (in^2)	0.5710	0.5970	0.5332	0.5651	0.6332	0.6744	0.5821	0.5312
A_{17} (in^2)	0.1560	0.1060	0.3945	0.1575	0.1842	0.1042	0.4390	0.3775
A_{18} (in^2)	7.9580	7.9900	7.9419	7.9672	8.0396	8.1146	8.0654	7.9283
A_{19} (in^2)	0.1000	0.1030	0.1009	0.1001	0.1000	0.1000	0.1532	0.1000
A_{20} (in^2)	8.9580	8.9870	8.9419	8.9675	9.0395	9.1148	9.0654	8.9287
A_{21} (in^2)	0.7200	0.6920	0.8348	0.7222	0.7460	0.7319	0.9590	0.8231
A_{22} (in^2)	0.4780	0.1420	0.1511	0.4843	0.1306	0.6647	0.3042	0.1701
A_{23} (in^2)	10.8970	10.7470	10.9401	10.9130	10.9114	11.1814	11.4189	10.9060
A_{24} (in^2)	0.1000	0.1010	0.1000	0.1047	0.1000	0.1043	0.3130	0.1002
A_{25} (in^2)	11.8970	11.7470	11.9401	11.9140	11.9114	12.1821	12.4189	11.9060
A_{26} (in^2)	1.0800	0.8340	0.8973	1.0842	0.8627	1.2109	1.2577	0.9017
A_{27} (in^2)	6.4620	7.4520	6.8488	6.5093	6.9169	5.9222	5.2407	6.9387
A_{28} (in^2)	10.7990	11.1690	10.8848	10.7000	10.9674	10.2457	9.9771	10.9154
A_{29} (in^2)	13.9220	13.4820	13.7495	13.9520	13.6742	14.4006	14.6681	13.7226
Best weight (lb)	25488.15	25,481.45	25,453.77	25479.67	25450.18	25811.34	25605.47	25457.24
Mean weight (lb)	25533.14	25,527.66	25,455.67	25614.23	25522.07	26025.37	26425.37	25603.61
Standard deviation (lb)	27.440	25.490	2.337	376.110	47.620	288.375	431.005	79.638
Number of structural analyses	28059	45000	10000	27984	27720	38320	74760	59940

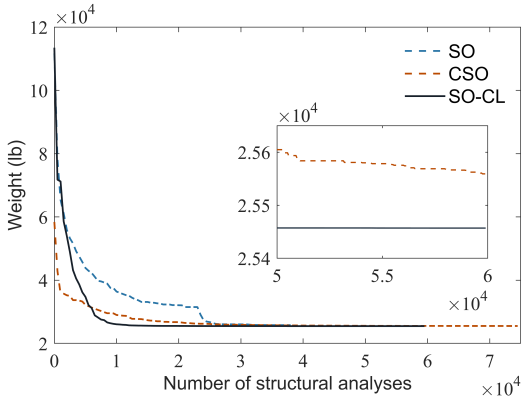


Fig. 16. Comparison of convergence rates between SO, CSO and the proposed SO-CL algorithms (case 5)

The developed SO-CL and other 7 metaheuristic algorithms are used to optimize this 200-bar truss structure, and the obtained design results are recorded in Table 10. By comparison, it can be found that the optimum design result derived from SO-CL is superior to that of TLBO, ACCS, PO, MRFO, SO and CSO, because SO-CL can provide designed bars with lighter weight. Besides, the standard deviation of the design results obtained from 30

independent running of SO-CL is much smaller than that of SO, CSO and PO, demonstrating the computational stability of the proposed algorithm.

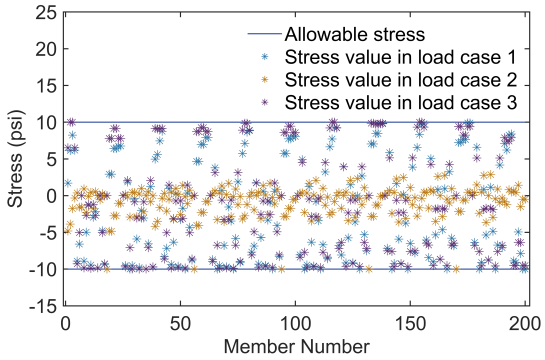


Fig. 17. The constraint conditions are strictly satisfied by using the SO-CL algorithm (case 5)

In Fig. 16, the proposed SO-CL is directly compared with SO and CSO in terms of convergence rate. Although CSO initially converges fast, its convergence rate significantly declines over time. Similarly, SO stagnates at the 35,000th structural analyses, resulting in premature convergence to non-optimal solutions. By contrast, SO-CL reaches the global optima much faster than SO and CSO, which demonstrates its strong solution-searching capability and high computational efficiency. In addition, the constraint results corresponding to the optimal solution derived from SO-CL are graphically illustrated in Fig. 17. Clearly, all of the predefined stress constraints have been strictly satisfied under three distinct loading conditions, which further confirms the effectiveness of SO-CL in solving constrained structural optimization problems.

5. Conclusion

The primary contribution of this work is the development of a novel *hybrid Snake Optimizer with Crisscross Learning strategy* (SO-CL) for solving non-convex structural optimization problems with multiple nonlinear constraints. The key innovation of SO-CL lies in seamlessly integrating multiple enhancement strategies into the computational framework of Snake Optimizer (SO), aiming to enrich population diversity, prevent premature convergence, and improve solution-searching capability. Specifically, the lens-imaging learning strategy is adopted to expand the solution-searching range and avoid search stagnation; the crossover learning strategy is employed to enhance global search capability and prevent premature convergence; and an adaptive weighting strategy is applied to balance exploration and exploitation by dynamically adjusting the search state, thereby improving computational efficiency. The proposed SO-CL algorithm has been successfully applied to solve five challenging benchmark problems of constrained truss structure optimization, demonstrating its excellent performance through comprehensive comparisons with seven well-known metaheuristic algorithms. All results confirm that SO-CL exhibits significantly stronger solution-searching ability and higher convergence rates than the standard SO and other algorithms, making it an effective and robust tool for addressing complex optimization issues. In terms of future research directions, one potential avenue is extending the proposed SO-CL algorithm to tackle large-scale structural optimization problems involving discrete variables, multi-objectives and multi-constraints.

Data availability statement

All data, codes and numerical models that support the findings of this study are available from GitHub through the following link: https://github.com/ihorizon2018/SO-CL.

Acknowledgment

The authors would like to acknowledge the support from the National Key Technologies Research & Development Program of China (2023YFB2604401; 2023YFB2604402) and the research grant from the Natural Science Foundation of Beijing (No. JQ23036).

Declarations

The authors declare that they have no conflict of interest in this paper.

References

[1] Cüneyt Vatansever. Investigation of buckled truss bars of a space truss roof system. *Engineering Failure Analysis*, 106:104156, 2019.

[2] Jin Cheng. Optimum design of steel truss arch bridges using a hybrid genetic algorithm. *Journal of Constructional Steel Research*, 66(8-9):1011–1017, 2010.

[3] Norapat Noilublao and Sujin Bureerat. Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms. *Computers & Structures*, 89(23-24):2531–2538, 2011.

[4] Mathias Stolpe. Truss optimization with discrete design variables: a critical review. *Structural and Multidisciplinary Optimization*, 53:349–374, 2016.

[5] Linfeng Mei and Qian Wang. Structural optimization in civil engineering: A literature review. *Buildings*, 11(2), FEB 2021.

[6] Christopher Renkavieski and Rafael Stubs Parpinelli. Meta-heuristic algorithms to truss optimization: Literature mapping and application. *Expert Systems with Applications*, 182:115197, 2021.

[7] Chao Wang, Zhi Zhao, Ming Zhou, Ole Sigmund, and Xiaojia Shelly Zhang. A comprehensive review of educational articles on structural and multidisciplinary optimization. *Structural and Multidisciplinary Optimization*, pages 1–54, 2021.

[8] Raphael T Haftka and Zafer Gürdal. *Elements of structural Optimization*, volume 11. Springer Science & Business Media, 2012.

[9] Klaus Schittkowski, Christian Zillober, and Rainer Zotemantel. Numerical comparison of nonlinear programming algorithms for structural optimization. *Structural Optimization*, 7:1–19, 1994.

[10] Guohua Wu, Witold Pedrycz, P.N. Suganthan, and Rammohan Mallipeddi. A variable reduction strategy for evolutionary algorithms handling equality constraints. *Applied Soft Computing*, 37:774–786, 2015.

[11] Mark J Jakiela, Colin Chapman, James Duda, Adenike Adewuya, and Kazuhiro Saitou. Continuum structural topology design with genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):339–356, 2000.

[12] Kang Seok Lee and Zong Woo Geem. A new structural optimization method based on the harmony search algorithm. *Computers & Structures*, 82(9-10):781–798, 2004.

[13] Letícia Fleck Fadel Miguel and Leandro Fleck Fadel Miguel. Shape and size optimization of truss structures considering dynamic constraints through modern metaheuristic algorithms. *Expert Systems with Applications*, 39(10):9458–9467, 2012.

[14] SO Degertekin and MS Hayalioglu. Sizing truss structures using teaching-learning-based optimization. *Computers & Structures*, 119:177–188, 2013.

[15] Herbert Martins Gomes. Truss optimization with dynamic constraints using a particle swarm algorithm. *Expert Systems with Applications*, 38(1):957–968, 2011.

[16] Ali Mortazavi and Vedat Toğan. Simultaneous size, shape, and topology optimization of truss structures using integrated particle swarm optimizer. *Structural and Multidisciplinary Optimization*, 54:715–736, 2016.

[17] Mohammad Farshchin, Charles V Camp, and Mohsen Maniat. Optimal design of truss structures for size and shape with frequency constraints using a collaborative optimization strategy. *Expert Systems with Applications*, 66:203–218, 2016.

[18] A. Kaveh and P. Zakian. Improved gwo algorithm for optimal design of truss structures. *Engineering with Computers*, 34(4):685–707, 2018.

[19] Hongyuan Zhou, Guangcai Zhang, Xiaojuan Wang, Pinghe Ni, and Jian Zhang. A hybrid identification method on butterfly optimization and differential evolution algorithm. *Smart Structures and Systems, An International Journal*, 26(3):345–360, 2020.

[20] SO Degertekin, G Yalcin Bayar, and L Lamberti. Parameter free jaya algorithm for truss sizing-layout optimization under natural frequency constraints. *Computers & Structures*, 245:106461, 2021.

[21] Carlos Millan-Paramo and João Elias Abdalla Filho. Exporting water wave optimization concepts to modified simulated annealing algorithm for size optimization of truss structures with natural frequency constraints. *Engineering with Computers*, 37(1):763–777, 2021.

[22] Adil Baykasoglu and Cengiz Baykasoglu. Optimal design of truss structures using weighted superposition attraction algorithm. *Engineering with Computers*, 36(3):965–979, JUL 2020.

[23] Rafiq Awad. Sizing optimization of truss structures using the political optimizer (po) algorithm. *Structures*, 33:4871–4894, 2021.

[24] Duc Thang Le, Dac-Khuong Bui, Tuan Duc Ngo, Quoc-Hung Nguyen, and H. Nguyen-Xuan. A novel hybrid method combining electromagnetism-like mechanism and firefly algorithms for constrained design optimization of discrete truss structures. *Computers & Structures*, 212:20–42, 2019.

[25] Vahid Goodarzimehr, Umut Topal, Amit Kumar Das, and Trung Vo-Duy. Bonobo optimizer algorithm for optimum design of truss structures with static constraints. *Structures*, 50:400–417, 2023.

[26] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[27] S Rajasekaran and S Lavanya. Hybridization of genetic algorithm with immune system for optimization problems in structural engineering. *Structural and Multidisciplinary Optimization*, 34:415–429, 2007.

[28] Saeid Kazemzadeh Azad. Enhanced hybrid metaheuristic algorithms for optimal sizing of steel truss structures with numerous discrete variables. *Structural and Multidisciplinary Optimization*, 55:2159–2180, 2017.

[29] Mohsen Khatibinia and Hessam Yazdani. Accelerated multi-gravitational search algorithm for size optimization of truss structures. *Swarm and Evolutionary Computation*, 38:109–119, 2018.

[30] Malihe Jafari, Eysa Salajegheh, and Javad Salajegheh. An efficient hybrid of elephant herding optimization and cultural algorithm for optimal design of trusses. *Engineering with Computers*, 35:781–801, 2019.

[31] Hakan Ozbasaran and Meltem Eryilmaz Yildirim. Truss-sizing optimization attempts with csa: a detailed evaluation. *Soft Computing*, 24(22):16775–16801, 2020.

[32] E. Ficarella, L. Lamberti, and S.O. Degertekin. Comparison of three novel hybrid metaheuristic algorithms for structural optimization problems. *Computers & Structures*, 244:106395, 2021.

[33] Juliano Pierezan, Leandro dos Santos Coelho, Viviana Cocco Mariani, Emerson Hochsteiner de Vasconcelos Segundo, and Doddy Prayogo. Chaotic coyote algorithm applied to truss optimization problems. *Computers & Structures*, 242:106353, 2021.

[34] Ali Kaveh, Kiarash Biabani Hamedani, and Mohammad Kamalinejad. Improved slime mould algorithm with elitist strategy and its application to structural optimization with natural frequency constraints. *Computers & Structures*, 264:106760, 2022.

[35] T Vu-Huu, Sy Pham-Van, Q-Hoan Pham, and Thanh Cuong-Le. An improved bat algorithms for optimization design of truss structures. In *Structures*, volume 47, pages 2240–2258. Elsevier, 2023.

[36] Fatma A Hashim and Abdelazim G Hussien. Snake optimizer: A novel meta-heuristic optimization algorithm. *Knowledge-Based Systems*, 242:108320, 2022.

[37] Huifang Li, Guanghao Xu, Boyuan Chen, Shuangxi Huang, Yuanqing Xia, and Senchun Chai. Dual-mutation mechanism-driven snake optimizer for scheduling multiple budget constrained workflows in the cloud. *Applied Soft Computing*, 149:110966, 2023.

[38] Lakshminarayana Janjanam, Suman Kumar Saha, and Rajib Kar. Optimal design of hammerstein cubic spline filter for nonlinear system modeling based on snake optimizer. *IEEE Transactions on Industrial Electronics*, 70(8):8457–8467, 2022.

[39] Zhonglin Li, Zijing Cheng, and Yan Wang. Parameter tuning of active disturbance rejection control based on improved snake optimization algorithm. In *2022 International Conference on Artificial Intelligence, Information Processing and Cloud Computing (AIIPCC)*, pages 316–322. IEEE, 2022.

[40] Chen Wang, Shangbin Jiao, Yujun Li, and Qing Zhang. Capacity optimization of a hybrid energy storage system considering wind-solar reliability evaluation based on a novel multi-strategy snake optimization algorithm. *Expert Systems with Applications*, 231:120602, 2023.

[41] Charles V Camp and Mohammad Farshchin. Design of space trusses using modified teaching–learning based optimization. *Engineering Structures*, 62:87–97, 2014.

[42] Wen Long, Jianjun Jiao, Ming Xu, Mingzhu Tang, Tiebin Wu, and Shaohong Cai. Lens-imaging learning harris hawks optimizer for global optimization and its application to feature selection. *Expert Systems with Applications*, 202:117255, 2022.

[43] Liguo Yao, Panliang Yuan, Chieh-Yuan Tsai, Taihua Zhang, Yao Lu, and Shilin Ding. Eso: An enhanced snake optimizer for real-world engineering problems. *Expert Systems with Applications*, 230:120594, 2023.

[44] Yaning Xiao, Hao Cui, Abdelazim G. Hussien, and Fatma A. Hashim. Msao: A multi-strategy boosted snow ablation optimizer for global optimization and real-world engineering applications. *Advanced Engineering Informatics*, 61:102464, 2024.

[45] Guoyuan Ma and Xiaofeng Yue. An improved whale optimization algorithm based on multilevel threshold image segmentation using the otsu method. *Engineering Applications of Artificial Intelligence*, 113:104960, 2022.

[46] Chunliang Mai, Lixin Zhang, and Xue Hu. Combining dynamic adaptive snake algorithm with perturbation and observation for mppt in pv systems under shading conditions. *Applied Soft Computing*, 162:111822, 2024.

[47] Gang Hu, Jingyu Zhong, Congyao Zhao, Guo Wei, and Ching-Ter Chang. Lcaha: A hybrid artificial hummingbird algorithm with multi-strategy for engineering applications. *Computer Methods in Applied Mechanics and Engineering*, 415:116238, 2023.

[48] Anbo Meng, Cong Zeng, Peng Wang, De Chen, Tianmin Zhou, Xiaoying Zheng, and Hao Yin. A high-performance crisscross search based grey wolf optimizer for solving optimal power flow problem. *Energy*, 225:120211, 2021.

[49] An bo Meng, Yu cheng Chen, Hao Yin, and Si zhe Chen. Crisscross optimization algorithm and its application. *Knowledge-Based Systems*, 67:218–229, 2014.

[50] Shahin Jalili and Yousef Hosseinzadeh. A cultural algorithm for optimal design of truss structures. *Latin American Journal of Solids and Structures*, 12(9):1721–1747, 2015.

[51] Armin Javidi, Eysa Salajegheh, and Javad Salajegheh. Enhanced crow search algorithm for optimum design of structures. *Applied Soft Computing*, 77:274–289, 2019.

[52] M. Kooshkbaghi and A. Kaveh. Sizing optimization of truss structures with continuous variables by artificial coronary circulation system algorithm. *Iranian Journal of Science and Technology - Transactions of Civil Engineering*, 44(1):1–20, MAR 2020.

[53] Hasan Tahsin Öztürk and Hamdi Tolga Kahraman. Meta-heuristic search algorithms in truss optimization: Research on stability and complexity analyses. *Applied Soft Computing*, 145:110573, 2023.

[54] American Institute of Steel Construction. *Manual of Steel Construction: Allowable Stress Design*. American Institute of Steel Construction, Chicago, Ill., 1989.