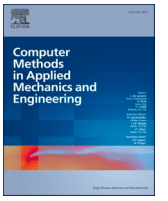
Computer Methods in Applied Mechanics and Engineering 446 (2025) 118256

Contents lists available at ScienceDirect

Computer Methods in Applied Mechanics and Engineering

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



Adaptive Strategy Management: A new framework for large-scale structural optimization design

Siamak Talatahari [a,b,c], Behnaz Nouhi [a], Amin Beheshti [b], Fang Chen [a], Amir H. Gandomi [a,d,*]

[a] Data Science Institute, Faculty of Engineering & Information Technology, University of Technology Sydney, Ultimo 2007, Australia
[b] School of Computing, Macquarie University, Sydney, Australia
[c] Department of Computer Science, Khazar University, Baku, Azerbaijan
[d] University Research and Innovation Center (EKIK), Óbuda University, Budapest 1034, Hungary

HIGHLIGHTS

- Adaptive Strategy Management developed as a new framework.
- Large-scale structural optimization design problems solved using developed method.
- Novel ASM variants yield superior performance in all tested problems.
- Close-based methods ensure feasibility and stability in large designs.

ARTICLE INFO

Keywords:
Structural optimization design
Adaptive Strategy Management
Chaos Game Optimization
Metaheuristic framework

ABSTRACT

This study introduces the Adaptive Strategy Management (ASM) framework designed to enhance the efficiency of computationally expensive optimization processes by dynamically switching between multiple solution-generation strategies. The ASM framework integrates three core steps: filtering, switching, and updating, which allow it to adaptively decide which solutions to evaluate based on real-time performance feedback. Several ASM-based variants are proposed, each implementing different filtering and switching mechanisms, such as generated-based selection, proximity-based filtering, and strategy switching guided by the current or global best solutions. Chaos Game Optimization (CGO) is selected as the core optimizer, with its updated equations modified to improve performance without incurring additional computational costs alongside strategy-level innovations. Extensive evaluations on medium-, large- and very large-scale structural problems demonstrate that the developed methods consistently outperform other approaches. Notably, the ASM-Close Global Best method, which combines proximity filtering with global best knowledge, achieved superior results across all performance intervals, showcasing robust convergence and high-quality solutions. These findings underscore the potential of Adaptive Strategy Management in improving large-scale optimization performance and open new directions for future research, including other strategy selections, broader applications across metaheuristics, and extensions to multi-objective and constrained optimization problems.

* Corresponding author at: Data Science Institute, Faculty of Engineering & Information Technology, University of Technology Sydney, Ultimo 2007, Australia.
E-mail address: Gandomi@uts.edu.au (A.H. Gandomi).

https://doi.org/10.1016/j.cma.2025.118256
Received 29 April 2025; Received in revised form 7 July 2025; Accepted 20 July 2025
Available online 29 July 2025

1. Introduction

For many years, engineers and designers have been striving to develop robust techniques that lead to reliable, stable, and optimal designs in various engineering fields [1–3]. Achieving an optimal structural design, in particular, has long been a critical challenge, given the complexity of real-world engineering problems and the conflicting objectives often involved. To meet this goal, a considerable amount of research has focused on the application and development of optimization methods aimed at finding the best possible designs. These efforts can generally be classified into three key categories, each contributing in different ways to the advancement of structural optimization:

- Application of General Optimization Methods: One of the most prevalent approaches involves applying general-purpose optimization techniques, originally developed for numerical, unconstrained optimization problems, for structural optimization problems [4–6]. The number of studies in this area has grown exponentially, largely driven by the continuous introduction of new optimization methods. These techniques, such as evolutionary algorithms [4], particle swarm optimization [7], and simulated annealing [8,9], have demonstrated great potential in structural design [10,11]. Recent studies, for example, [12–15], illustrate how the Symbiotic Organisms Search (SOS), Firefly Algorithm (FA), Chaos Game Optimization (CGO), Political Optimizer (PO), Gray Wolf Optimization (GWO) and Stochastic Paint Optimizer (SPO) methods have been successfully applied to the structural domain to achieve optimal designs, respectively. Many of these methods require reformulating the structural optimization problem to fit the framework of general optimization techniques. This is often achieved through constraint-handling mechanisms, with penalty functions being one of the most commonly employed strategies [16]. The penalty function method reformulates constrained optimization problems into unconstrained forms by incorporating penalty terms into the objective function, which quantifies how much the constraints are violated [17]. Nonetheless, other methods, such as barrier functions or Lagrangian multipliers [18,19], have also been explored to address this challenge. These general methods, while versatile and widely applicable across many optimization problems, often need to be carefully tuned for specific structural problems, and their effectiveness may fluctuate based on the complexity of the design space.

- Development of Specialized Techniques: A more focused area of research has been the development of optimization techniques specifically tailored for structural optimization problems. Unlike general optimization methods, these techniques are designed to exploit the unique characteristics of structural problems, leading to more efficient and effective solutions. Although these specialized methods may not be as broadly applicable across different domains, their power lies in their ability to address the intricacies of structural design, such as handling multi-material designs [20,21], topology optimization [22–25], or large-scale systems [24,26,27]. One example of such a specialized technique is the Guided Search Strategy proposed by [28], which has proven highly effective in navigating complex structural design spaces and identifying optimal solutions. The Guided Search Strategy incorporates problem-specific knowledge to direct the search process more efficiently toward promising regions of the design space, significantly improving both convergence speed and solution quality. Additionally, [16] introduced a fuzzy-based optimization technique that modifies the problem space by incorporating fuzzy logic to handle uncertainties and ambiguities in design variables and constraints. This approach enhances the optimization process by making it more adaptive to real-world scenarios where exact values may not always be known, and it offers a flexible and robust alternative to traditional methods. These specialized methods, though limited in scope, provide targeted solutions that can significantly outperform general optimization methods in structural applications, making them valuable tools for engineers.

- Adaptive and Hybrid Methods: Due to the unique challenges presented by structural optimization, such as the presence of highly constrained, nonlinear, and non-convex problems, standard optimization methods often require enhancements or adaptations to be effective. This necessity has led to the development of adaptive optimization methods, which refine traditional techniques to better address the demands of structural design. These methods incorporate specialized mechanisms or strategies [12] to tackle issues such as slow convergence, local minima, and constraint violations. A notable example is the work of Kaveh and Talatahari [29], who introduced an adaptive continuous search-space strategy to improve the performance of conventional optimization algorithms for structural optimization. Their approach dynamically evaluates new solutions immediately after generation, improving search efficiency and reducing the risk of premature convergence to suboptimal solutions. In addition, hybrid methods, which integrate elements from multiple optimization techniques, have gained increasing attention [30,5]. These approaches leverage the strengths of different algorithms to enhance performance. For instance, combining metaheuristic methods (global search strategies) with gradient-based techniques (local search refinements) has proven particularly effective in addressing the diverse requirements of structural design [31]. This synergy enables algorithms to navigate complex design spaces more effectively, ensuring both global exploration and fine-tuned local optimization.

In line with the adaptive method approach, this paper introduces an adaptive version of the recently developed Chaos Game Optimization (CGO) method [32,33]. CGO stands out for its simplicity and efficiency, being nearly parameter-free and employing multiple strategies for generating new solutions, resulting in high performance across a range of optimization problems [34]. Structural optimization problems, particularly in large-scale design scenarios, are characterized by high dimensionality, complex constraints, and computationally expensive evaluation processes. Metaheuristic algorithms such as CGO have shown promise in navigating these search spaces due to their stochastic and multi-strategy nature [32]. Nevertheless, they often require a high number of computations to find optimal solutions.

A key innovation in our approach is the introduction of an “Adaptive Strategy Management” (ASM), which enhances CGO’s flexibility and adaptability. ASM dynamically adjusts the algorithm’s behaviour based on the characteristics of the problem space, allowing CGO to better manage the highly constrained and complex nature of the structural design. This adaptive mechanism allows the algorithm to concentrate efforts on the most promising search directions, improving convergence speed, solution quality, and computational efficiency. Based on these points, the rationale for selecting CGO as the baseline method in this paper includes the

following points:

1. **Simple and Modular Structure:** CGO has a lightweight and transparent structure, with no reliance on complex evolutionary operators or numerous hyperparameters. This simplicity makes CGO particularly suitable for prototyping ASM, as it allows for clear assessment of ASM's effect without interference from confounding algorithmic components.
2. **Natural Compatibility with ASM:** CGO generates multiple candidate solutions per seed using a set of distinct strategy modules, unlike many metaheuristics that apply a single update rule per candidate. This is structurally aligned with ASM, which adaptively filters, switches, and updates strategies based on their performance. This intrinsic multi-strategy design enables seamless integration with ASM's decision-making pipeline.
3. **Algorithm-Agnostic Nature of ASM:** ASM is developed as a general, algorithm-independent framework, capable of being integrated into any multi-strategy optimization algorithm. CGO was chosen for this study due to its structural suitability and clarity, but the framework can be embedded into a wide range of metaheuristics. Future research will explore ASM's integration into other optimization paradigms to confirm its portability and generality.
4. **Efficiency Gains through ASM Integration:** The results comparing other algorithms with CGO indicate that these methods generally achieve similar final solution quality, i.e. some slightly better, some slightly worse, but all within a close range to CGO. However, when ASM is integrated into CGO, the total computational cost is significantly reduced, often by several times, while improving solution quality. This is achieved without changing the core structure of CGO. Given ASM's algorithm-independent and modular design, it is reasonable to expect that similar efficiency improvements could be realized by embedding ASM into other metaheuristics as well.

In addition, we introduce several refinements to the solution generation process, enhancing how a new candidate design is produced and evaluated. These improvements significantly boost the algorithm's robustness and efficiency in identifying optimal solutions. Notably, we also develop a boundary constraint handling method, which, together with the improved solution generation strategy, is built upon a memory-based mechanism. This integrated approach enables the algorithm to generate high-quality solutions more consistently, especially in structural optimum design problems.

When discussing performance improvements in computationally expensive domains, surrogate-assisted techniques cannot be overlooked. As an instance, for neural architecture search (NAS), methods using surrogate models with multiple comparisons and semi-online learning [35], score predictor-assisted evolution [12], smart-block discovery [13], and surrogate-assisted ranking [17] have shown success in reducing the number of expensive function evaluations by learning predictive models of the fitness landscape. Also, a recently developed framework [36] has explored machine learning techniques for structural analysis. While surrogate models can be valuable in the analysis phase required for optimization, their application in structural design requires caution. The current study, however, focuses on improving the design process, rather than the analysis component.

Moreover, in highly sensitive and precision-critical applications such as structural optimization, surrogate models may struggle to maintain accuracy across all regions of the design space. The cost of inaccurate predictions in such contexts can be significant, potentially leading to infeasible or suboptimal solutions. Therefore, although surrogate models provide value in various engineering applications, their use in structural design must be carefully assessed. This reinforces the importance of developing an adaptive, lightweight strategy management framework that targets optimization performance directly without adding significant complexity or reducing reliability.

The key contributions of this research are as follows:

1. *Introducing Adaptive Strategy Management Framework:* We introduce a novel Adaptive Strategy Management Framework that enables CGO by enabling dynamic adaption to problem search spaces, which effectively balances computational cost and optimization performance, making it well-suited for complex structural design applications.
2. *Refined Solution Generation Process:* The solution generation strategy within CGO is enhanced to accelerate convergence, improve solution quality, and strengthen robustness in solving structural optimization problems.
3. *Tailored CGO for Structural Optimization:* different variants of algorithms are developed, integrating the above innovations to tackle the specific challenges of structural design with improved performance and reliability.

These contributions aim to equip the structural design community with a powerful and adaptable optimization tool capable of meeting the increasing complexity of modern engineering challenges.

The rest of the paper is organized as follows: Section 2 reviews the fundamental principles of CGO and outlines the enhancements incorporated in the improved version. Section 3 introduces the Adaptive Strategy Management framework, detailing its core concepts and functionality. Section 4 provides different variants of the framework. Section 5 presents numerical experiments and results, including performance evaluations and comparative analyses. Section 6 discusses some conceptual and theoretical aspects of the developed framework. Finally, Section 6 wraps up the paper by highlighting the main findings and promising avenues for future research.

2. Chaos Game Optimization

2.1. Standard Chaos Game Optimization

The core idea of CGO is inspired by the Chaos Game, a mathematical process where seemingly random movements and directions result in an ordered structure that exhibits fractal properties [33]. Fractals, known for their self-similar nature, are mathematical sets where a pattern repeats at different scales [37]. One of the most well-known chaos games involves three points forming the vertices of a triangle. In this game, random movements towards these vertices, over time, generate a chaotic yet structured fractal pattern [38].

Building on this concept, CGO represents three solutions as the "vertices" of a triangle:

1. The **mean seed** represents the average of a selected group of the current population.
2. The **best seed** is the solution with the best fitness in the current population.
3. A **random seed** is selected from the population to introduce stochastic behaviour.

The movement process in CGO mimics the dynamics of the chaos game, where each seed (solution) moves randomly toward one of three vertices: the mean solution, the best solution, or a randomly selected seed. Additionally, random perturbations in certain dimensions can be introduced to further enhance diversity. The extent of each movement is determined randomly, ensuring an effective balance between exploration and exploitation. This stochastic nature allows CGO to efficiently navigate the solution space, much like how the chaos game progressively forms an ordered fractal. The process for generating new seeds follows these equations [33]:

$$Seed_i^1 = X_i + \alpha_i \times (\beta_i \times GB - \gamma_i \times MG_i),\ i = 1, 2, \ldots, n. \tag{1}$$

$$Seed_i^2 = GB + \alpha_i \times (\beta_i \times X_i - \gamma_i \times MG_i),\ i = 1, 2, \ldots, n. \tag{2}$$

$$Seed_i^3 = MG_i + \alpha_i \times (\beta_i \times X_i - \gamma_i \times GB),\ i = 1, 2, \ldots, n. \tag{3}$$

$$Seed_i^4 = X_i\left(x_i^k = x_i^k + R\right),\ k = [1, 2, \ldots, d]. \tag{4}$$

In these equations, X_i denotes the *ith* seed (solution candidate), GB represents the current global best, and MG_i refers to the mean seed. The parameter α_i is a randomly generated scalar factor, while β_i and γ_i are random integers taking values of either 0 or 1. Additionally, k is a randomly selected integer within the range $[1, d]$ and R is a uniformly distributed random variable. Algorithm 1 provides the pseudocode for the original CGO algorithm.

It is important to note that the challenges encountered when using CGO for large-scale structural optimization are not unique to CGO itself; rather, they are common among many metaheuristic algorithms when applied to high-dimensional, constrained engineering problems. Nevertheless, the main issues include:

1. **Lack of Built-in Constraint Handling Mechanisms:** CGO, like many standard optimization methods, was originally designed for unconstrained optimisation. It lacks dedicated mechanisms for managing feasibility in complex structural designs, where constraint

Algorithm 1
. Original Chaos Game Optimization.

Step 1: Initialization
1.1. Define the problem condition: Search space; Objective, constraints, and penalty functions
1.2. Determine the number of population (N_{pop}) and maximum number of iterations ($Iter_{max}$)
Step 2: Generate initial seeds
2.1. Generate N_{pop} random seeds in the determined search space
2.2. Find objective and constraints values for generated initial seeds
2.3. Determine penalized function value for each seed
2.4. Find the best one and set it as the Global best
Step 3: Searching process
While iter $\leq Iter_{max}$:
For N_{pop} repeat:
For i =1:4
3.1. Generate new seed i
3.2. Return violated seed to the search space
3.3. Find objective and constraints values for the new seeds
3.4. Determine penalized function value for the new seeds
3.5. If the new seed i is better than the current seed, replace it
End for
End for
3.6. sort all new seeds and find the best one.
3.7. if the best seed outperforms the global best, update the global best with the new seed.
End while
Step 4: Reporting results
4.1. Report the global best as the final optimum solution

violations (e.g., stress, displacement, stability) are often the main challenge. Without external handling strategies, CGO may generate high-quality but infeasible solutions or require extra post-processing to correct them.
2. **Uniform Strategy Usage Without Adaptation:** In its basic form, CGO applies all strategy modules uniformly without assessing their effectiveness in real-time. This non-adaptive behavior can be inefficient in high-dimensional spaces, where certain strategies may become less effective over time or for specific problem structures.
3. **Scalability Challenges:** As the number of design variables increases (e.g., in high-rise or complex braced structures), CGO's uniform solution generation may lead to slow convergence and increased computational cost, especially when combined with expensive FEM-based evaluations.

These challenges motivated the development of ASM that not just to enhance CGO, but to propose a general-purpose mechanism for better convergence and feasibility across structural design problems.

2.1. Improved Chaos Game Optimization

As an improved version of CGO, our approach aims to enhance at least one of the equations used for generating new solutions. Typically, such modifications may increase computational cost or introduce additional algorithmic steps. However, our objective is to leverage the core principles of the original CGO while maintaining computational efficiency. One crucial concept that can be effectively utilized is memory-based solution generation. In the original CGO, memory plays a crucial role in the updating step by storing the best-so-far position for each seed. This information is used to evaluate the acceptability of newly generated solutions. Furthermore, strategically integrating memory into the solution generation process can enhance solution quality without significantly increasing computational cost.

One of the most well-known algorithms that effectively incorporates memory for solution generation is the Harmony Search (HS) algorithm [39,40]. HS utilizes a harmony memory mechanism, where previously generated solutions influence the creation of new candidate solutions. Inspired by this approach, we explore ways to adapt memory utilization within CGO, ensuring that past high-quality solutions guide the search process while preserving the stochastic nature of the algorithm. Therefore, we provide the following formulation to generate a new solution [39]:

$$Seed_{i,j}^{4} = \begin{cases} Eq.\ (6)\ if\ rand_1 \le MCR \\ x_{j,min} + rand.\left(x_{j,max} - x_{j,min}\right)\ otherwise \end{cases} \tag{5}$$

Use memory:

$$Seed_{i,j}^{4} = \begin{cases} x_{m,j}\ if\ rand_2 \le PAR \\ x_{m,j} + \delta\ otherwise \end{cases} \tag{6}$$

where MCR is the memory consideration rate and PAR is the pitch adjustment rate, as in the standard HS (Harmony Search) concept. $x_{j,min}$ and $x_{j,max}$ are maximum and minimum values for jth variable. $x_{m,j}$ is the value of the jth variable of the mth memory (selected randomly). δ is a small random perturbation near $x_{m,j}$, used to explore the neighborhood.

Algorithm 2 presents the process of generating new solutions using this concept. To keep the number of strategies the same as in the original CGO, we replace the last original strategy (Eq. (4)) with this new strategy.

In addition to providing the new formulation for $Seed^4$, we apply the same concept to handle seeds that move out of boundaries. In other words, violated dimensions are corrected using the developed technique, rather than being replaced with purely random values, a common approach that often disrupts convergence and reduces solution quality. By leveraging the structured memory-based correction mechanism, the algorithm ensures that boundary-violating solutions are adjusted in a way that maintains continuity with previous high-quality solutions. This method preserves valuable search direction information and enhances the stability and

Algorithm 2
. HS-based solution generation strategy.

Step 1: Define Memory:
1.1. Set current best solutions as the historical memory (HM)
1.2. Determine the parameters: MCR and PAR
Step 2: Generate a new seed
For i= *Number of variables*:
If $rand_1 \le$ MCR
If $rand_2 \le$ PAR
2.1: Assign the i-th component of the new seed from a randomly selected solution in HM.
Else
2.2: Randomly select a solution from HM.
2.3: Set dimension i of the new seed to a value close to the corresponding value of dimension i in the selected solution.
End if
Else
2.4: Set dimension i of the new seed to a random value within the feasible search space.
End if
End for

searching process.

3. Introduction of the strategy management framework

One of the key distinctions between recent optimization methods, such as CGO, and traditional algorithms like Particle Swarm Optimization (PSO) lies in their solution generation mechanisms, as illustrated in Fig. 1. PSO typically employs a "single solution-generating strategy", where each particle is updated using a unified equation. In contrast, CGO adopts "multiple solution-generating strategies", utilizing multiple techniques to generate new solutions for each seed. This diversified approach has demonstrated strong performance in solving general optimization problems. However, when applied to domain-specific problems such as structural design, where the computation of objective-constraint functions incurs high costs , this multi-strategy process may introduce significant computational overhead.

To address this issue, we propose the "Strategy Management" (SM) framework. The main objective of this framework is to dynamically switch between multiple solution-generating strategies while minimizing computational costs through selective evaluation. Rather than applying and evaluating all strategies at every iteration, SM "filters" the most appropriate strategy, and "switches" it to the "active solution". Only the active solution is evaluated for its fitness, while the remaining strategies, referred to as "passive solutions", are excluded from further updates, as shown in Fig. 2. This selective evaluation reduces the number of function evaluations (FEs), thereby improving computational efficiency without compromising the quality of the solutions.

Beyond reducing computational overhead, SM also helps optimize resource allocation during the search process. By avoiding unnecessary evaluations, it enables the algorithm to focus computational power on promising candidate solutions, accelerating convergence while maintaining accuracy. This efficiency is particularly beneficial for large-scale and computationally expensive problems, where function evaluations contribute significantly to the total computational burden. Furthermore, the mechanism ensures that the CGO algorithm remains scalable, making it applicable to high-dimensional and real-world engineering optimization tasks.

4. Random and adaptive strategy management

As outlined in the previous section, SM provides a framework for categorizing solution generation techniques into active and passive strategies. By focusing solely on the active strategy, it enables a significant reduction in computational cost. This flexible framework allows for the development of various switching strategies tailored to different problem contexts. In this study, we propose two specific strategies built upon SM: Random Strategy Management (RSM) and Adaptive Strategy Management (ASM).

RSM follows a straightforward process. After generating a set of potential solutions, typically four seeds in CGO, RSM randomly selects one as the active solution. This active solution is then evaluated and used for subsequent updates, while the remaining (passive) solutions are ignored at this step. The randomness in RSM ensures diverse exploration of the solution space, as each seed has an equal probability of being selected. Although this approach offers computational savings and may maintain broad coverage of the search space, it lacks a directed mechanism for guiding the switching process. Consequently, RSM may lead to inefficient use of resources during the exploitation phase, where a more informed selection would be beneficial. As a result, for more complex or sensitive problems, such as structural optimization, a more refined, Adaptive Strategy Management becomes necessary.

Adaptive Strategy Management (ASM) adopts a problem-specific approach by dynamically selecting the active solution based on key characteristics of the optimization process. In contrast to RSM, which relies on purely stochastic selection, ASM aims to make informed decisions by considering problem-specific features such as constraint satisfaction and objective function behaviour.

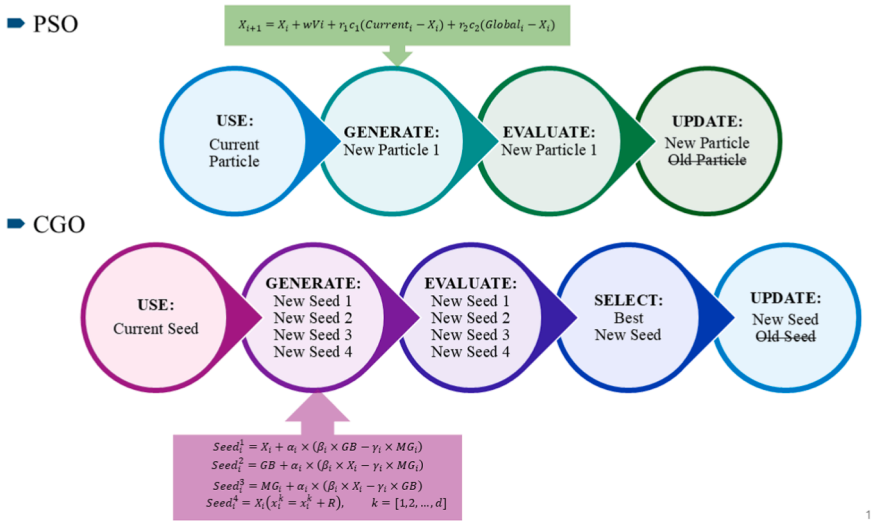


Fig. 1. General procedure for PSO (Single solution-generating strategy) and CGO (Multiple solution-generating strategies).

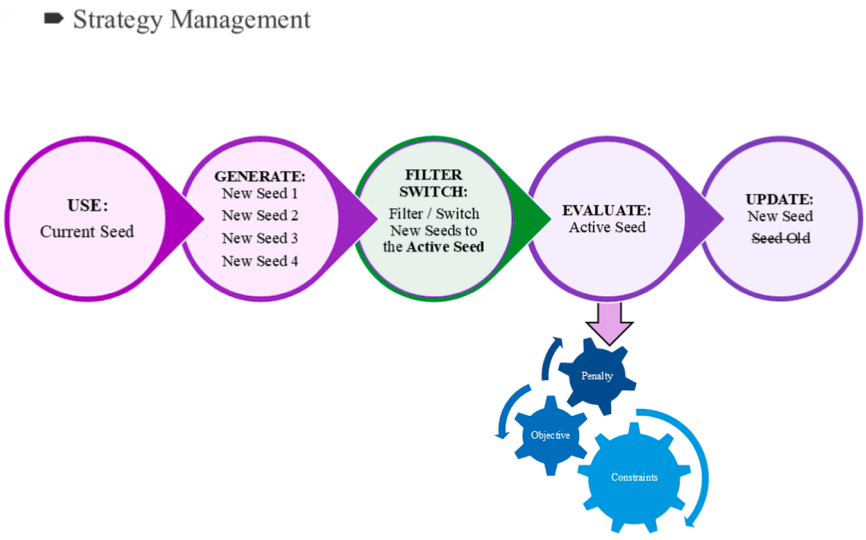


Fig. 2. General procedure for strategy management framework.

Therefore, to formalize the implementation of ASM, we begin by reviewing the mathematical formulation of structural optimization problems, which serves as the foundation for adaptive decision-making within the framework:

$$\min f(X)$$

$$s.t.\ g_i(X) \le 0 \tag{7}$$

where, $f(X)$ represents the objective function (e.g., minimizing weight or cost), and $g_i(X)$ is the ith constraint (such as material strength or displacement limits). Note that the objective function evaluation in structural optimization is simple and can be expressed as:

$$f(X) = \sum_{el=1}^{Nel} \rho l_i x_i = \ \rho\ L.X \tag{8}$$

where Nel is the total number of elements, ρ is the material density, l_i is the length of the element i, and X is the design variable vector. This function is computationally inexpensive compared to constraint evaluations.

Structural constraints are significantly more complex to evaluate. These constraints include serviceability and strength constraints, following the AISC-LRFD (2001) provisions:

Serviceability Constraints

1. **Maximum lateral displacement constraint**

$$C_D^t = \Delta_{Max} - \Delta_{Max}^a \le 0 \tag{9}$$

2. **Inter-story drift constraint**

$$C_F^d = [\delta_J]_F - [\delta^a]_F \le 0 \tag{10}$$

Strength Constraints

1. **Axial and flexural strength interaction constraints**

$$C_{el}^i = \left[\frac{P_{uJ}}{\phi P_n}\right]_{el} + \frac{8}{9}\left(\frac{M_{uxJ}}{\phi_b M_{nx}} + \frac{M_{uyJ}}{\phi_b M_{ny}}\right)_{el} - 1 \le 0 \ for\ \left[\frac{P_{uJ}}{\phi P_n}\right]_{el} \ge 0.2 \tag{11}$$

$$C_{el}^i = \left[\frac{P_{uJ}}{2\phi P_n}\right]_{el} + \left(\frac{M_{uxJ}}{\phi_b M_{nx}} + \frac{M_{uyJ}}{\phi_b M_{ny}}\right)_{el} - 1 \le 0 \ for\ \left[\frac{P_{uJ}}{\phi P_n}\right]_{el} < 0.2 \tag{12}$$

2. **Shear strength constraint**

$$C_{el}^{i} = \left(V_{uj}\right)_{el} - \left(\phi_v V_n\right) \leq 0 \tag{13}$$

where:

1. Δ_{MaxJ} and $[\delta_J]_F$ are the maximum lateral displacement and inter-story drift, respectively, while Δ_{Max}^{a} and $[\delta^a]_F$ denote their corresponding maximum allowable values.
2. P_{uJ}, M_{uxJ} (or M_{uyJ}) and V_{uJ} are the required axial, flexural, and shear strengths for element J, respectively.
3. P_n, M_{nx} , M_{ny} and V_n are the nominal strengths.
4. ϕ, ϕ_b and ϕ_v are resistance factors.

Finally, to handle constraints, we typically combine the objective function and constraint violations using a penalty function, defined as [16]:

$$f_{penalty}(X) = f(X) \times \left(1 + \alpha \sum_{Constraints} max(C, 0)\right) \tag{14}$$

where C represents all the constraints defined in the problem, and α is the penalty parameter, which is set to one. Later in the study, we discuss how the proposed framework addresses one of the main challenges in constraint handling, tuning the penalty parameter, by reducing the need for manual adjustment.

Unlike the objective function, evaluating constraints in structural optimization is computationally expensive due to:

1. Generating a large stiffness matrix (K), which has a high computational cost.
2. Solving structural equilibrium equations.
3. Computing serviceability constraints, including drifts and many strength constraints.

On average, the finite element analysis (FEA) step, which underpins constraint evaluations, accounts for the vast majority of computational costs in large-scale structural optimization. This insight motivated the development of ASM, which leverages solution history and structural behavior to reduce redundant analyses and improve computational efficiency. To fulfill this aim, the ASM

Algorithm 3
. ASM-based Chaos Game Optimization.

Step 1: Initialization
1.1. Define the problem condition: Search space; Objective, constraints, and penalty functions
1.2. Determine the number of population (N_{pop}) and maximum number of iterations ($Iter_{max}$)
Step 2: Generate initial Seeds
2.1. Generate N_{pop} random seeds in the determined search space
2.2. Find objective and constraint values for generated initial seeds (Eqs. (8)–(13))
2.3. Determine penalized function value for each seed (Eq. (14))
2.4. Establish memory: Set initial seeds as the current best ones
2.5. Find the best one and set it as the Global best
Step 3: Searching process
While iter ≤ $Iter_{max}$:
For N_{pop} repeat:
For i =1:4
3.1. Generate new seed i (Eqs. (1)–(3), (5), (6))
3.2. Apply the memory-based correction mechanism to the violated seed.
3.3. Find just objective value for the new seeds (Eq. (8))
End for
3.4. Apply filtering step on 4 new seeds and determine the filtered seed
3.5. Apply switching step and determine the active seed
3.5. Apply Updating step:
If any Active seed?
3.5.1. Find constraint values for the active seed (Eqs. (9)–(13))
3.5.2. Find penalized function value for the active seed (Eq. (14))
End if
3.6. If the new seed i is better than the current seed, replace it and update memory
End for
3.7. Sort all new seeds and find the best one.
3.8. If the best seed outperforms the global best, update the global best with the new seed.
End while
Step 4: Reporting results
4.1. Report the global best as the final optimum solution

framework employs a stepwise filtering and switching process, guided by objective function evaluations and adaptive update criteria. The mechanism is based on the principles outlined in Table 1. In this framework, the filtering step serves as a preliminary step that identifies candidate solutions with potential for further consideration but does not immediately select any as active. Filtered solutions that meet the criteria are passed to the next stage without additional computation. In contrast, the switching step is responsible for designating each filtered solution as either passive or active, based on its performance and suitability for guiding the search. At these steps, only the objective function of generated solutions is used for evaluation. We utilized the current best and global best values as thresholds in certain filtering and switching techniques. These values are evaluated using a penalized objective function to account for constraint violations observed during previous optimization iterations. After determining the active solution, it undergoes constraint evaluation, and its fitness is updated using a penalty-based approach. If the active solution outperforms the current one, it replaces the previous solution in the population.

Algorithm 3 presents the ASM-based CGO, where the concepts of filtering, switching, and updating are incorporated into the original algorithm. According to this algorithm, after generating new candidate solutions, only their objective function values are initially evaluated. The filtering and switching mechanisms then determine whether a constraint evaluation is necessary. Specifically, Steps 3.3, 3.4, and 3.5 implement this process by selectively evaluating and updating solutions based on their relevance and potential, thereby reducing unnecessary computational overhead.

Depending on the selected filtering or switching process, different ASM frameworks can be developed. In this study, we consider four distinct ASM models, as illustrated in Table 2.

In the ASM-Generated method (Fig. 3(a)), both the filtering and switching processes are based on the generated mechanism. This means among the four generated candidates, the one with the best structural weight is selected as the active seed for further evaluation. This approach ensures that, in each iteration, there is always a promising solution identified and passed through the evaluation stage.

In the ASM-Current Best method (Fig. 3(b)), the filtering step is still based on the generated solutions, but switching depends on a comparison with the current best. Among the four generated candidates, the best one is selected, but it only becomes the active seed if it outperforms the so-far best existing solution. This mechanism ensures that evaluations are performed only when a better solution is identified, avoiding unnecessary computations. If no improvement is found, the iteration proceeds without evaluation.

In the ASM-Global Best method (Fig. 3(c)), the filtering is still based on the generated candidates, selecting the best among them. However, switching is governed by a comparison with the global best solution found so far. The selected seed is only evaluated if it shows an improvement over the global best. This ensures that evaluations are performed strictly when a potential global improvement is identified. If no generated solution surpasses the global best, the algorithm skips the evaluation and update phases for that iteration.

In the ASM-Close Current Best method (Fig. 3(d)), both filtering and switching are influenced by the current best solution. Unlike the other models, the filtering stage prioritizes solutions that are not necessarily the absolute best among the newly generated ones, but those closest to the current best. If this candidate also improves upon the current best, it becomes the active seed and proceeds to evaluation. Otherwise, no evaluation is performed.

Finally, in the ASM-Close Global Best method (Fig. 3(e)), both the filtering and switching mechanisms are guided by the global best solution. During the filtering stage, the newly generated seeds are assessed, and the one closest to the global best is selected. If this filtered candidate further improves upon the global best, it is designated as the active solution for evaluation and updating. Otherwise, the iteration proceeds without evaluation.

The ASM evaluates candidate solutions based on their objective function performance in the initial step. Seeds that demonstrate better performance are selected as active solutions. Since this switching process does not involve constraint evaluations at this stage, the computational overhead is significantly reduced compared to conventional approaches. This adaptive selection enables ASM to concentrate on the most promising regions of the solution space, improving the efficiency of function evaluations and overall optimization performance. By leveraging insights from the problem landscape, ASM minimizes redundant computations, making it particularly effective for large-scale structural optimization tasks where constraint evaluations are computationally expensive.

Although many adaptive or hybrid strategies in metaheuristics introduce additional parameters that require careful tuning, the proposed ASM framework is designed to remain parameter-independent. All adaptations and decisions within ASM are governed by

Table 1
Principles of filtering, switching and updating.

Step	Technique	*Description*
1. **Filtering**		
	Random-based	A randomly selected solution among the generated ones is chosen.
	Generated-based	The solution with the best (lowest) objective value is selected.
	Current Best	The solution closest to the current best is selected.
	Global Best	The solution closest to the global best is selected.
2. Switching		
	Generated-based	The filtered solution is directly considered as the active solution.
	Current Best	The filtered solution becomes active if it outperforms the current best seed.
	Global Best	The filtered solution becomes active if it outperforms the global best seed.
	—	Otherwise, it is treated as passive; no active solution proceeds.
3. Updating		
	Active Solution	Undergoes constraint evaluation; objective value is updated.
	Passive Solution	No constraint evaluation; excluded from future consideration.

Table 2
Developed RSM and ASM frameworks.

Method	Filtering Step	Switching Step	Updating Step
RSM	Random-based	Generated-based	Always
ASM-Generated	Generated-based	Generated-based	Always
ASM-Current Best	Generated-based	Current Best	If the active solution is better than the current best
ASM-Global Best	Generated-based	Global Best	If the active solution is better than the global best
ASM-Close Current Best	Current Best	Current Best	If the active solution is better than the current best
ASM-Close Global Best	Global Best	Global Best	If the active solution is better than the global best

dynamic, performance-based rules derived from the internal state of the algorithm (e.g., current and global best solutions), without the need for manually tuned thresholds or external control parameters. This ensures that the nearly parameter-free nature of CGO is preserved, enhancing both the usability and reproducibility of the overall optimization framework. Fig. 4 presents the flowchart of the developed ASM-based CGO framework, summarizing the core steps of the ASM framework within the optimization process.

5. Numerical study

5.1. Comparison indicators

In this study, we considered three well-established medium, large, and very large-scale steel structures. These examples are used to evaluate the performance of developed optimization techniques to achieve better optimal solutions or reduce computational costs while maintaining solution quality [16,41]. Given the vast search space and the considerable computational effort required to handle numerous design constraints, developing optimization techniques that can either improve solution quality or reduce the number of required structural evaluations is of great importance. This paper addresses these challenges by enhancing the CGO method through two key innovations: the development of the ASM framework and a refined solution generation formulation.

In this section, we evaluate the impact and effectiveness of the improved CGO approaches. To do so, we introduce and apply a comprehensive set of performance indicators to assess the quality, reliability, and efficiency of the optimization results.

Traditionally, the two primary indicators for evaluating optimization algorithms are the quality of the obtained solution and the number of optimization iterations [42]. These two indicators should be considered together for a comprehensive evaluation. The number of iterations directly determines the number of function evaluations (N_{FE}), which is a fundamental metric for estimating the computational cost of an optimization method:

$$N_{FE} = N_{Iter} \times N_{Pop} \tag{15}$$

where N_{Iter} is the number of iterations and N_{Pop} is the population size. This formulation assumes that each generated solution is also evaluated, making the total number of new solutions equal to the total number of evaluations. However, this assumption does not hold for the proposed SM. In this technique, particularly when applied within the CGO framework, up to $4\times N_{Pop}$ solutions may be generated per iteration; however, not all of them are necessarily evaluated. Instead, based on the rules of the mechanism, only a subset of these solutions (the "active" ones) are evaluated, while the "passive" ones are discarded. To ensure a fair and consistent comparison, a fixed number of structural analyses is assumed for all algorithms, rather than relying on the number of iterations. This fixed value is deliberately chosen to be lower than those commonly used in related studies, thereby imposing a stricter computational constraint. Under this setting, an additional set of performance indicators is introduced to comprehensively evaluate the effectiveness of the developed methods, as detailed in the rest of this section.

The second key indicator concerns the quality of the solutions. The results must be evaluated not only based on their optimality but also in terms of reliability and statistical certainty. To this end, we conducted 30 independent runs of each algorithm, generating a robust and statistically meaningful dataset that allows for a comprehensive assessment of performance consistency. From this dataset, three important performance metrics were extracted:

1. *Global Best Solution (GBS)*: reflects the maximum potential of the algorithm in achieving optimal performance. To report the Global Best Solution (GBS), we present the best objective values obtained by each algorithm within a predefined range of evaluations, with particular emphasis on the final best result achieved at the end of the optimization process. This indicator highlights the peak performance capability of each method.
2. *Average Performance (AveP)*: indicates the dispersion of results across multiple runs and reflects the consistency and robustness of the algorithm. For the AveP, we illustrate the mean convergence history along with the standard deviation at each iteration. By analyzing the mean trajectory and the associated variation range, one can assess the consistency of algorithm performance across multiple runs and understand how variability influences the final outcomes.
3. *Confidence Interval (CI)*: provides a statistical range within which the true performance of the algorithm is expected to lie with a certain probability. The CI complements the standard deviation by offering a formal measure of uncertainty, thus enhancing the reliability of performance comparisons. To report CI, we adopt a 95% confidence level. The CI provides a statistical range that likely contains the true mean performance of the algorithm. This measure is visualized during the optimization process to reflect the uncertainty and reliability of the achieved results, offering a more robust basis for comparison among the competing methods.

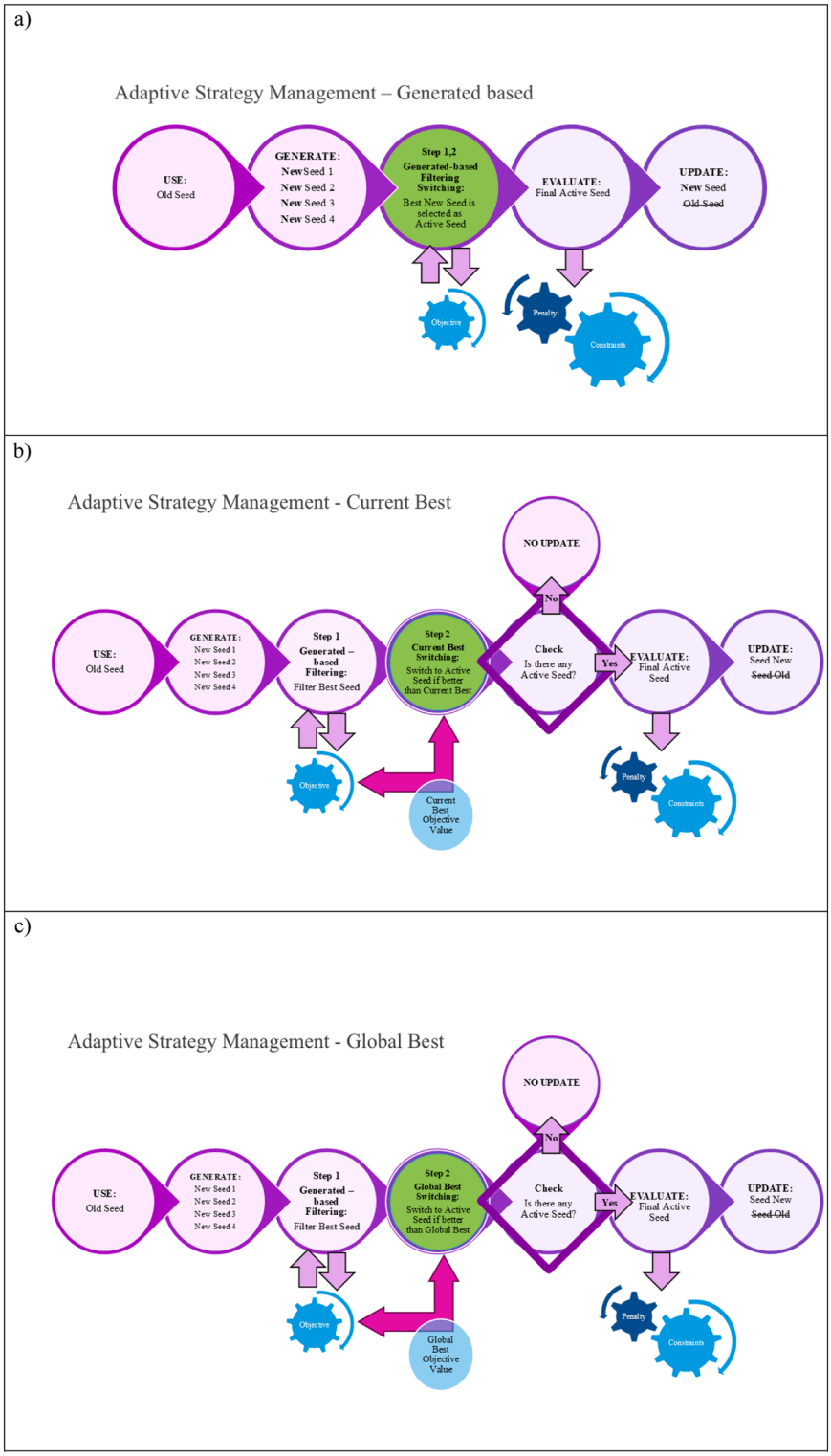


Fig. 3. Adaptive Strategy Management: a) Generated-based, b) Current Best, c) Global Best, d) Close Current Best, e) Close Global Best.

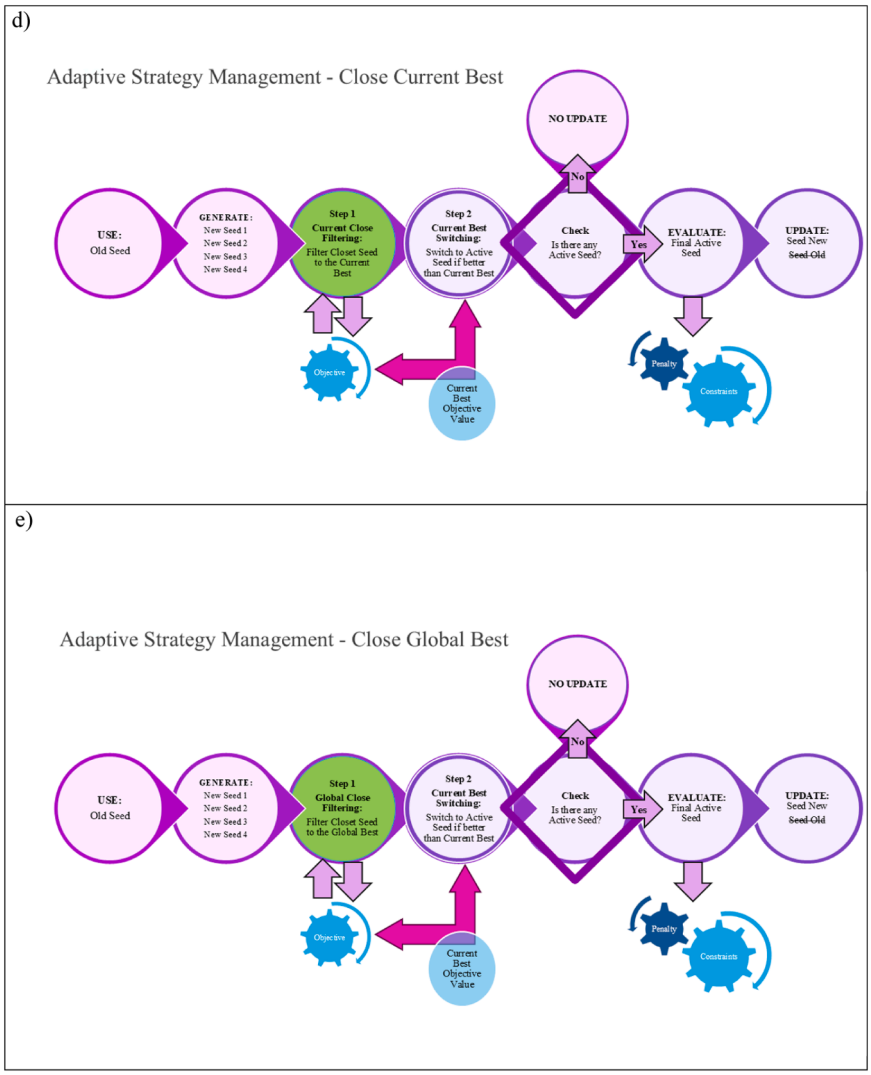


Fig. 3. (*continued*).

The third category of indicators introduced in this study focuses on evaluating the improvement dynamics of algorithm performance throughout iterations. These indicators provide insights into how effectively an algorithm refines solutions during the optimization process.

1. *Convergence History Plot (CHP):* This plot tracks the rate of improvement in the objective function values throughout iterations, using the convergence trajectory of the best run from each algorithm. CHP is particularly useful for evaluating how quickly an algorithm approaches optimal or near-optimal solutions.
2. *Best Local Improvement (BLI):* This metric quantifies the improvement observed in the best-performing run of each algorithm within a predefined range of iterations. To fulfill this aim, the difference between the initial and final objective function values in a selected segment of iterations is computed. To ensure a fair evaluation, the initial value at each step is set to be the same and corresponds to the worst value among all algorithms. BLI reflects the algorithm's ability to sustain progress and improve the solution quality over time. It is particularly useful for understanding the strength of an algorithm's starting performance and its consistency in achieving improvements throughout the search process.
3. *Global Mean Improvement (GMI):* GMI measures the average improvement across all independent runs of an algorithm, starting from a common initial reference point. For fairness in comparison, the same starting value is assumed for all methods. The improvement is calculated as the difference between the initial value and the mean objective function value over all runs at given iterations. This indicator reflects the general effectiveness of an algorithm in improving solutions across a wide range of trials.
4. *Strategy-Switch Dynamic (SSD) Graph:* The SSD graph visualizes how often, and at which points in the optimization process, different embedded strategies are activated during algorithm execution. The graph tracks strategy usage over iterations, providing insights

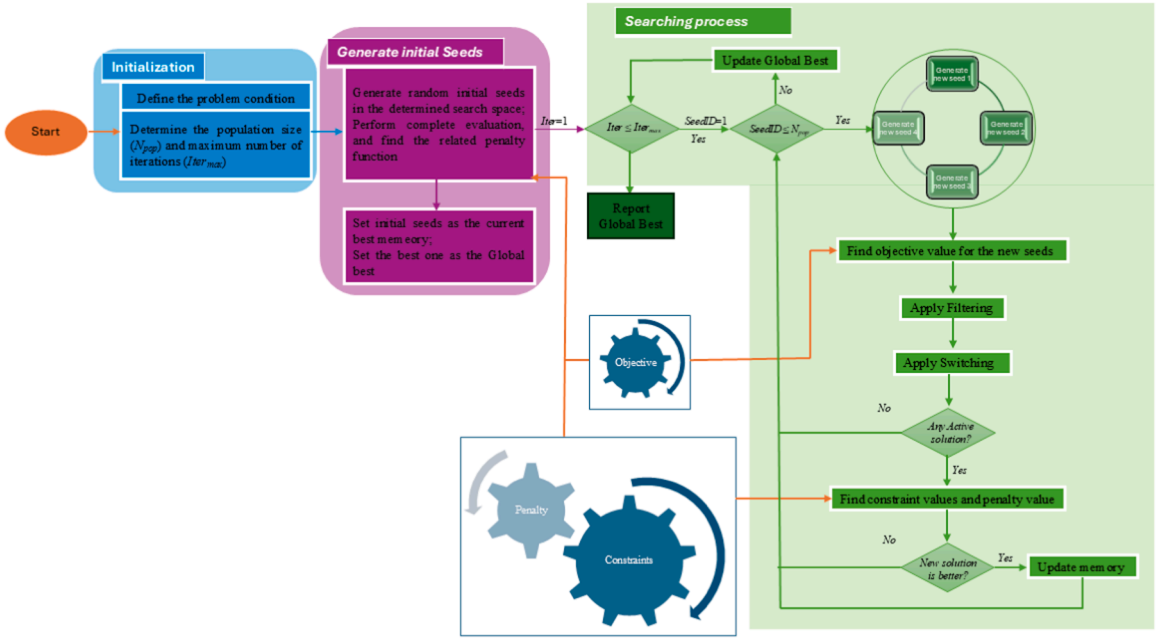


Fig. 4. Flowchart of the ASM-enhanced CGO framework.

into the adaptability and responsiveness of the framework. SSD helps evaluate the internal dynamics of ASM and highlights how the algorithm navigates through different regions of the search space.

5. *Solution Distribution Plot (CDP):* The Solution Distribution Plot displays the spread of solutions obtained by the algorithm across multiple independent runs. This plot helps assess the diversity, robustness, and stability of the algorithm. A tight clustering of solutions near the optimal region suggests consistent convergence, while a wider spread may indicate exploratory behavior.
6. *Exploration–Exploitation Ratio (EER)*: The Exploration–Exploitation Ratio (EER) quantifies the balance between two key search behaviors during the optimization process: Exploration refers to the algorithm's ability to investigate diverse regions of the search space. Exploitation refers to intensifying the search around promising regions already identified. EER is computed by tracking the diversity of generated solutions (spread in design space) relative to improvements in solution quality. A high ratio indicates dominant exploration, while a low ratio suggests dominant exploitation. Monitoring EER over time allows for analyzing how the algorithm transitions between the exploration and exploitation phases.

5.2. *Description of structural examples*

This section briefly describes the structural benchmark examples used in this study. These examples have been widely investigated in the structural optimization literature and are regarded for their effectiveness in evaluating both the theoretical performance and practical applicability of optimization algorithms. Each example is designed not only to meet academic research standards but also to reflect realistic engineering conditions, such as detailed design codes, strength constraints evaluated at multiple locations per element, and practical section assignments. Unlike many previous studies that rely on simplified assumptions, the examples presented here incorporate full-scale design conditions. Load combinations follow established design codes rather than reduced or idealized formats. Moreover, strength constraints are evaluated at five critical stations along each structural element, increasing the total number of constraints by a factor of five compared to simplified approaches. This ensures comprehensive verification of all safety and performance criteria and makes the optimization outcomes more realistic and applicable in practice.

The first example involves a mid-rise 10-story building (Fig. 5(a)), significantly larger in scale, consisting of 1,026 structural elements. This model presents a more complex design scenario, including lateral and gravity load-resisting elements. The elements are organized into 32 design groups to reduce the complexity of the optimization process while maintaining realistic design flexibility.

The second structural example considered in this study is an X-braced 20-story steel building (Fig. 5(b)), comprising 3860 frame elements. These elements are grouped into 73 distinct design groups to ensure practical implementation of the optimization outcomes. All members are selected from standard W-shaped steel sections, complying with design code provisions.

The final example involves a very large-scale mega-braced tubed high-rise building (Fig. 5(c)) with 8,272 structural elements and 103 design groups, representing an extensive and computationally intensive design problem. This example is intended to evaluate the scalability and efficiency of the proposed algorithm when applied to very large, real-world structural systems.

All examples are modelled in accordance with applicable building design codes and incorporate realistic loading scenarios, including gravity and lateral loads, and relevant strength and serviceability constraints. This ensures that the optimized designs are not only theoretically sound but also viable for practical implementation. Further technical details regarding geometry, loading, boundary

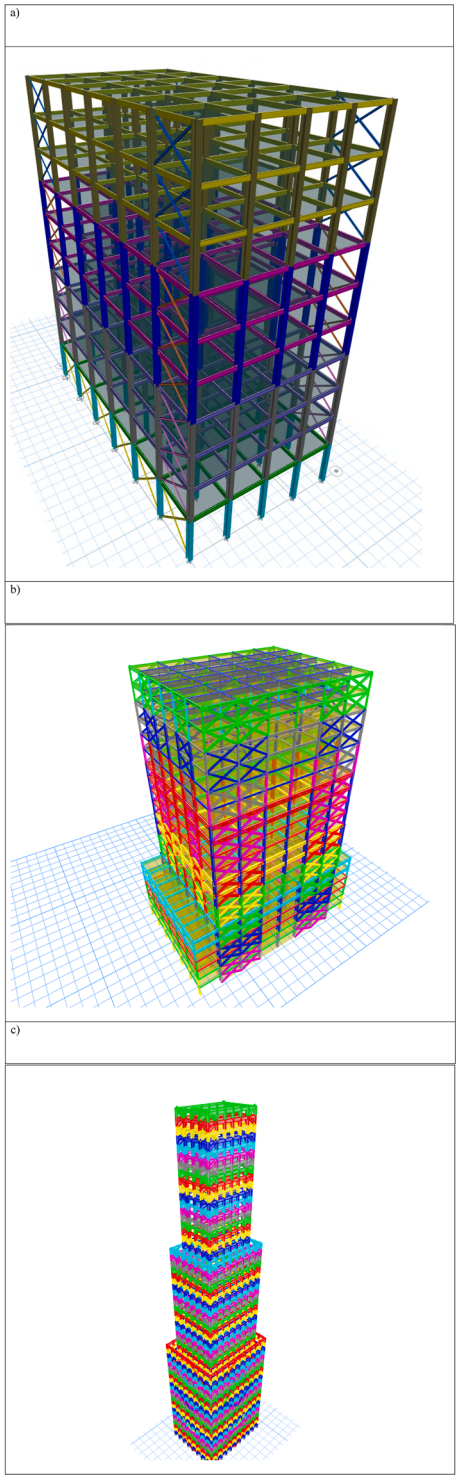


Fig. 5. The 3D representations of the structures: a) 10-story structure, b) X-braced 20-story structure, c) High-rise building.

conditions, and code-specific requirements for each example can be found in [16,17]. The parameters of other methods are presented in Table 3 [16]. All tests were conducted on a server equipped with an Intel Core i9-9900 K CPU @ 3.60 GHz, 32 GB of RAM, and an NVIDIA GeForce RTX 4090 GPU.

Table 3
Parameter summary of the alternative metaheuristic algorithms, [16].

Metaheuristic	*Parameter*	*Description*	*Value*
ACO	N_{ant}	*Colony Size*	*50*
	α	*Intensity of pheromone control parameter*	*1.0*
	β	*Visibility parameter*	*0.4*
	ρ	*Pheromone evaporation*	*0.2*
	ξ	*pheromone trail*	*0.1*
ICA	N_{cont}	*Countries Size*	*50*
	N_{emp}	*Number of Empires/Imperialists*	*10*
	α	*Selection Pressure*	*1*
	β	*Assimilation Coefficient*	*1.5*
	p_r	*Revolution Probability*	*0.05*
	μ	*Revolution Rate*	*0.1*
	ζ	*Colonies Mean Cost Coefficient*	*0.2*
CSS	N_{pop}	*Charged Particle Size*	*50*
	a	*Radius of Charged Sphere*	*0.1*
	HMCR	*Harmony Memory Consideration Rate*	*0.85*
	PAR	*Pitch Adjustment Rate*	*0.15*
	kt	*Attract-Repel Coefficient*	*0.9*
	N_{cm}	*Charged Memory Size*	*12*
	k_a	*Acceleration Coefficient*	*0.5*
	k_v	*Velocity Coefficient*	*0.5*
PSO	N_{pop}	*Particle Size*	*50*
	w	*Intertia Coefficient*	*1*
	w_{damp}	*Damping Ratio of Inertia Coefficient*	*0.99*
	C_1	*Personal Acceleration Coefficient*	*2*
	C_2	*Social Acceleration Coefficient*	*2*
FA	N_{pop}	*Fireflies Size*	*50*
	β_0	*Attractiveness parameter*	*1.0*
SOS	N_{pop}	*Population Size*	*50*
WOA			
ISA			
CGO	N_{pop}	*Population Size*	*25*

5.3. Results and discussion

Each example is solved using seven methods developed in this study, including the base CGO, CGO with RSM, and CGO with different ASM-based strategies: ASM-Generated, ASM-Current Best, ASM-Global Best, ASM-Close Current Best, and ASM-Close Global Best. Each method is executed over 30 independent runs with different random initializations to support a comprehensive statistical analysis and ensure a robust evaluation of performance.

5.3.1. Numerical results for the 10-story structure

This structural benchmark problem has been widely used in recent studies to assess the performance of optimization algorithms.

Table 4
Final optimum results reported by other researchers compared to the developed methods for the 10-story structure.

				[16]					
Methods	GA1	PSO1	ACO1	SOS1	ICA1	WOL1	ISA1	FA1	CSS1
Best Result (tons)	591.34	591.36	591.35	549.51	571.03	558.05	563.19	595.71	549.24
Mean Result	832.09	866.11	862.81	654.03	808.24	766.50	733.15	765.13	645.81
No. Analyses	13,500	13,500	13,500	13,500	13,500	13,500	13,500	13,500	13,500
				[16]					
	GA2	PSO2	ACO2	SOS2	ICA2	WOL2	ISA2	FA2	CSS2
Best Result (tons)	598.27	581.05	578.96	550.61	568.48	560.26	559.32	576.45	543.02
Mean Result	739.09	740.758	755.55	627.94	732.23	771.40	655.76	711.74	618.11
No. Analyses	13,500	13,500	13,500	13,500	13,500	13,500	13,500	13,500	13,500
		[17]				Current Work (CGO)			
	MBB-BC	MCC-MB	EBB-BC	MCC-EB	ASM-Generated Based	ASM-CurrentBest	ASM- Global Best	ASM-Close Current Best	ASM-Close Global Best
Best Result (tons)	512.00	504.34	517.71	510.71	500.81	498.07	**491.42**	509.52	494.41
Mean Result	582.67	537.33	551.75	523.31	530.92	528.07	525.09	525.78	**511.27**
No. Analyses	50,000	50,000	50,000	50,000	**7,000**	**7,000**	**7,000**	**7,000**	**7,000**

Table 4 presents the final results reported in earlier works alongside those obtained using the CGO-based methods developed in this study. Previous studies typically involved a high number of structural analyses (13,500 or 50,000) and reported GBS ranging from 549.24 tons to 504.34 tons, with mean values often exceeding 550 tons. In contrast, the methods introduced in this work were evaluated using only 7,000 structural analyses per run, yet consistently outperformed previous approaches in both best and average performance metrics. Notably, the best GBS values were achieved by the ASM-Global Best (491.42 tons) and ASM-Close Global Best (494.41 tons), representing the lowest values among all compared methods and indicating a significant advancement in solution quality. The ASM-Generated and ASM-Current Best variants also performed well, achieving GBS values of around 500 tons, making them the second-best group of performing variants. Meanwhile, the ASM- Close Current Best approach, with a best result of 509.52 tons, did not surpass the performance of other ASM variants. It is important to highlight that both the base CGO and CGO-RSM methods resulted in relatively poor final solutions, with GBS values exceeding 600 tons. While these two methods can potentially reach results of around 530 tons when allowed to continue up to 13,500 iterations, the focus of this study was to achieve high-quality solutions with a significantly reduced number of evaluations. This reinforces the efficiency and practicality of the proposed ASM frameworks for solving real-world structural optimization problems.

The best mean values are also attributed to the developed methods, reflecting not only their ability to reach superior optima but also their consistency across multiple runs. Among these methods, the ASM-Close Global Best framework achieves the lowest mean result (511.27 tons), establishing it as the most robust and stable approach overall. It is closely followed by the ASM-Global Best (525.09 tons) and ASM-Close Current Best (525.78 tons), both of which also demonstrate strong performance with relatively low variance. These findings underscore the effectiveness of incorporating global information into both the filtering and switching steps, especially in the ASM Close-Global Best variant. The ASM-Current Best variant yields a mean result of approximately 528.07 tons, positioning it as the third best-performing method among the proposed frameworks. This demonstrates that incorporating the current best solution into the switching logic remains a powerful strategy for driving solution quality. In contrast, the ASM-Generated method, with a higher mean of 530.92 tons, shows a drop in performance. Notably, the ASM-based methods either outperform or are at least comparable to previous works, with the best mean result reported in earlier studies being approximately 523 tons. This highlights the overall superiority and robustness of the ASM framework in addressing the considered optimization problem. Meanwhile, the base CGO and CGO-RSM report much higher mean values, approximately 710 tons for both, highlighting their comparatively weaker performance in terms of convergence accuracy and consistency. This suggests that relying solely on the random generation of seeds without guidance from the best-known solutions limits the method's capacity for effective convergence.

Fig. 6 illustrates the *CHP* of the best-performing run for each method. The plots confirm that the proposed frameworks not only achieve superior final solutions but also exhibit faster and more stable convergence patterns, particularly during the initial stages of the optimization process. This behaviour highlights the effectiveness of the ASM methods. Additionally, these methods reduce computational effort by accelerating convergence without compromising solution quality. This figure shows that all ASM-based methods perform very closely to one another, both in terms of convergence speed and final objective values. In contrast, the original CGO and the RSM-augmented CGO exhibit noticeably less competitive performance. These two methods follow almost identical convergence trajectories and fail to reach the best results found by ASM variants.

In Fig. 7, we present the *AveP* indicators for each optimization method to demonstrate the central tendency and variability of their convergence behaviours. The mean values reflect the average structural weight achieved across all runs, while the standard deviation captures the consistency of each method's performance. The shaded regions represent one standard deviation above and below the mean, offering insight into the dispersion around the average solution quality. From the results, it is evident that CGO-RSM shows a notable improvement in stability compared to the original CGO. Specifically, the standard deviation decreases from 52.5 tons to 33.2 tons by the end of the optimization, a reduction of approximately 36.7%, indicating that the RSM technique enhances convergence stability. Notably, this improvement in consistency occurs while the mean performance remains nearly unchanged, suggesting that RSM contributes to robustness without compromising solution quality. Both these CGO variants still underperform relative to the ASM methods.

In contrast, all ASM-based frameworks demonstrate significant improvements, not only in mean performance but also in result

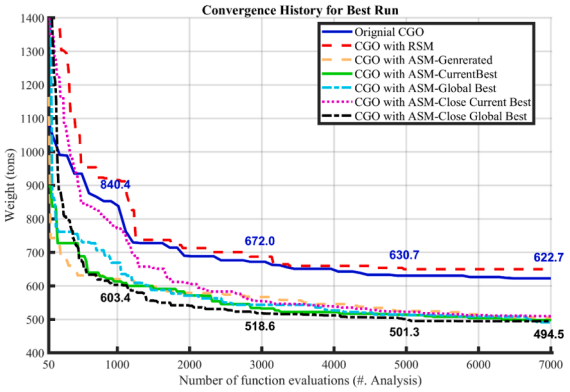


Fig. 6. Convergence history Plot (*CHP*) for the 10-story structure.

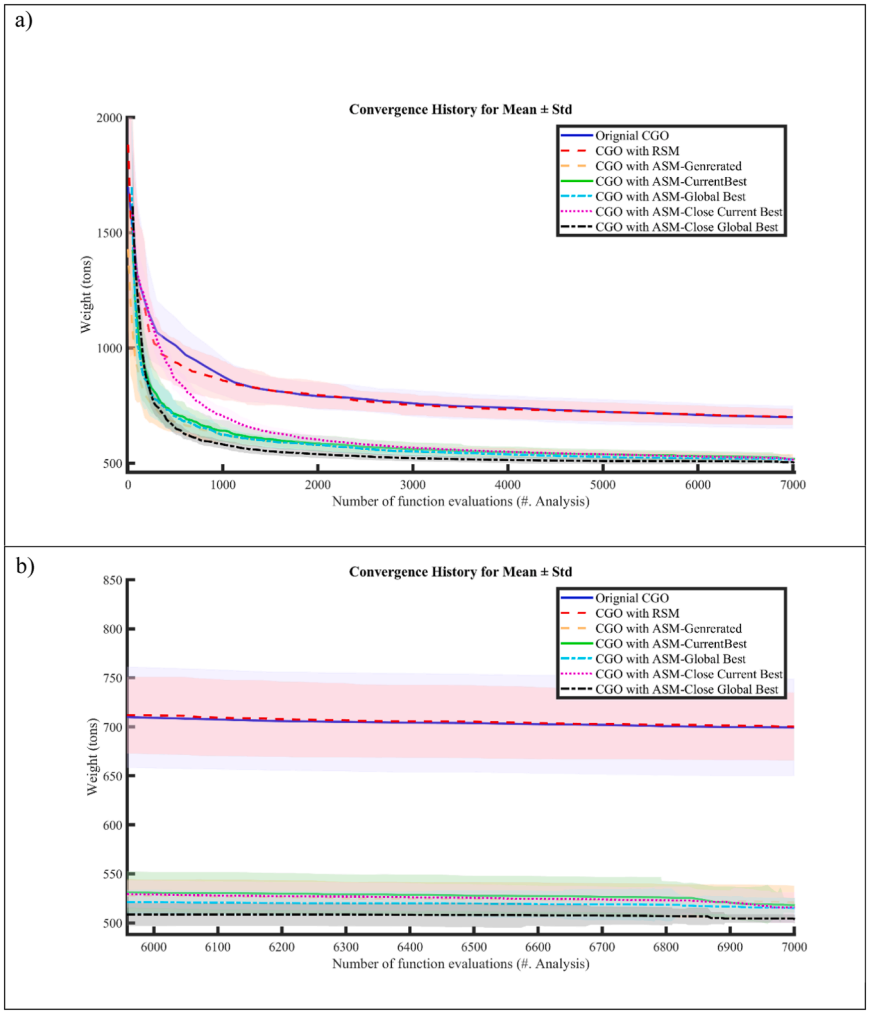


Fig. 7. Average Performance (*AveP*) indicators for the 10-story structure: (a) full range comparison, (b) zoomed-in view.

consistency. For instance, the ASM-Close Global Best, ASM-Close Current Best, and ASM-Global Best variants achieve dramatic reductions in standard deviation, down to 4.4 tons, 15.8 tons, and 10.6 tons, respectively, representing improvements of approximately 91%, 70%, and 80% over the CGO algorithm. These results confirm the effectiveness of the ASM designs, especially those employing close-based filtering combined with the best global or best current switching strategies, in consistently delivering both high-quality and reliable solutions.

Fig. 8 presents the *CI* plots for these methods, providing a statistical representation of the reliability of the obtained results. The CI plots show the 95% confidence bounds around the mean performance for each method, offering insights into the precision of the estimated averages. Narrower confidence intervals indicate higher reliability and less variability across runs, while wider intervals reflect greater uncertainty or inconsistent behaviour. The results in Fig. 8 reinforce the trends observed in Fig. 7, confirming that ASM-based methods not only achieve better mean performance but also exhibit greater statistical reliability and reduced variance, particularly evident in the ASM-Close and ASM-Best variants.

The *BLI* plot is presented in Fig. 9. According to this figure, it is evident that almost all algorithms begin very strongly, especially during the first two intervals, where they achieve the most significant improvements. However, starting from the third interval, the performance of the base CGO and CGO-RSM methods declines considerably, while the ASM-based methods maintain strong and stable performance.

The best performance is achieved by ASM-Close Global Best, which demonstrates a remarkable 75% improvement over the initial solution within the first 1,000 analyses, compared to approximately 65% for the base CGO method. The other ASM variants also show high initial performance, reaching or exceeding 70% improvement during the same interval. In the second interval, the ASM-Close Global Best method again leads with a 40% improvement, while the other ASM-Global Best and ASM-Current Best methods achieve around 35%–37%. In all subsequent intervals, the ASM-Close Global Best method continues to outperform others, demonstrating its consistent ability to enhance the solution quality. The only exception is in the final interval, where the ASM-Global Best slightly

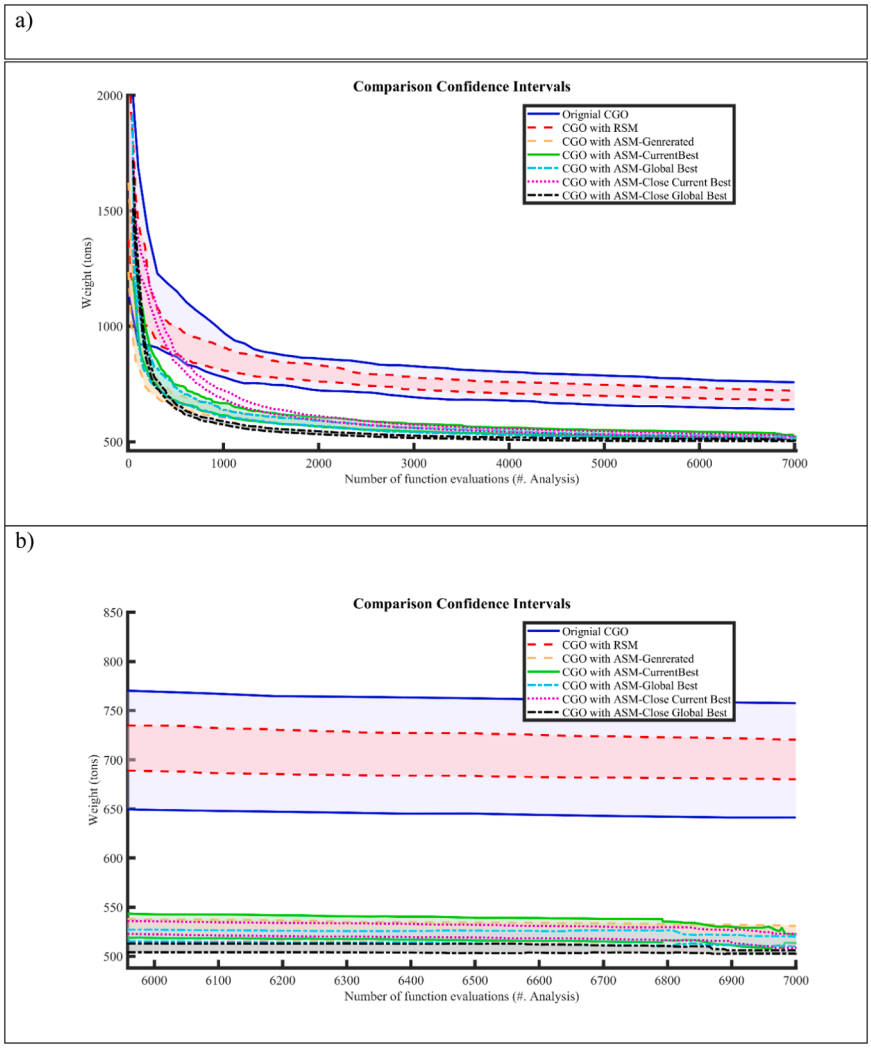


Fig. 8. Confidence Interval (*CI*) plots for the 10-story structure: (a) full range comparison, (b) zoomed-in view.

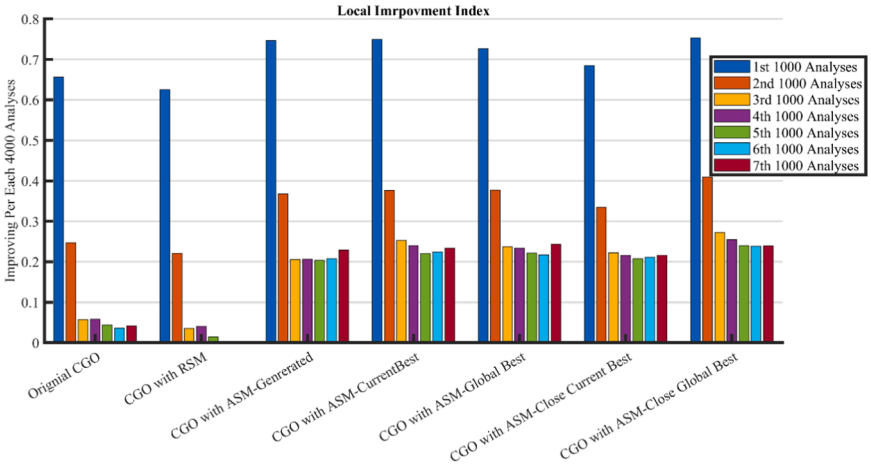


Fig. 9. Best Local Improvement (*BLI*) for the 10-story structure.

surpasses it, with a 24.3% improvement compared to 24% for ASM-Close Global Best. The main key observation is the consistent improvement trend exhibited by all ASM-based methods, clearly indicating their ability to continuously enhance solution quality over time.

Fig. 10 presents the *GMI* curves over time. GMI quantifies the overall average percentage of improvement relative to the initial solution, offering a clear view of both the performance level and the convergence consistency achieved by each algorithm. According to the figure, the ASM-based methods consistently outperform the base CGO and CGO-RSM variants in terms of global mean improvement throughout all analysis intervals. The ASM-Close Global Best method shows the highest GMI values at nearly every stage, ultimately reaching approximately 79.0% improvement by the final interval. This reflects not only strong local exploitation but also a globally guided improvement trajectory made possible through adaptive switching logic. The ASM-Global Best and ASM-Close Current Best methods also perform closely, achieving 78.3% and 78.5% improvements at the final stage, respectively. The minimal performance gap among the top ASM variants demonstrates their robustness and consistency across runs. In contrast, the base CGO and CGO-RSM variants achieve significantly lower improvements, with final GMI values of approximately 70.85% and 70.8%, respectively. While RSM contributes marginally to convergence stability, it does not offer a substantial improvement in global performance when compared to ASM-based strategies. The consistent upward trend in the GMI curves of ASM variants indicates sustained improvement behaviour, confirming their capacity for ongoing optimization even in the later stages of the analysis. This is in contrast with the CGO-based approaches, which plateau earlier and exhibit limited gains in the final intervals.

5.3.2. *Numerical results for the X-braced 20-story structure*

The second example is the structural optimization of an X-braced 20-story building, a large-scale problem used to benchmark the performance of the proposed CGO-based algorithms against several well-known metaheuristics from the literature, including those reported by [16,17]. Among the previously published results (Table 5), the best-known solution (GBS) was achieved by CSS2 with a total weight of 2713.57 tons [16]. However, the proposed ASM-based variants, particularly the ASM-Close Global Best method, outperformed all existing methods by obtaining a GBS result of 2407 tons. This represents a significant reduction in structural weight (approximately 11% improvement compared to the best previous report) and highlights the superior optimization capability of the developed framework. Other top-performing variants also include ASM-Close Current Best and ASM-Global Best, with GBS results of 2530 tons and 2511 tons, respectively. These results further confirm the robustness and effectiveness of the ASM strategies in achieving high-quality solutions.

In terms of mean results, which better reflect algorithmic consistency and reliability, the ASM-Close Global Best method again outperformed all other techniques, achieving a mean structural weight of 2503 tons. These results are substantially better than the best mean reported in earlier studies, such as CSS2's 2803.92 tons. The next best performer was ASM-Close Current Best with 2644 tons, followed by ASM-Global Best with 3160 tons. Notably, this performance was achieved with only 7,000 structural analyses, compared to the 17,500 (or 50,000) analyses used by all the methods reported in the earlier studies.

Fig. 11 presents the *CHPs* for the best run of each algorithm applied to the X-braced 20-story structure. The results reveal three distinct behavioural patterns among the proposed methods. In the initial phase (first 2,000 analyses), the ASM-Close Global Best and ASM-Current Best variants demonstrate rapid early improvements, securing over 90% of their total objective reduction early in the process. These methods exhibit highly efficient exploration capabilities, making them the most effective in early-stage convergence.

The second group, including the RSM and ASM-Generated variants, shows moderate performance in the beginning. While they outperform the original CGO in early iterations, they are unable to maintain their momentum in later stages, eventually being overtaken by other ASM variants. The third group, comprising ASM-Close Current Best and ASM-Global Best, displays relatively slow initial convergence. However, their performance improves significantly in the mid-phase (between 6,000–8,000 analyses), allowing them to catch up with and eventually surpass the second group.

In the final phase (8,000–12,000 analyses), the first group continues to improve and maintains a clear lead over all other methods,

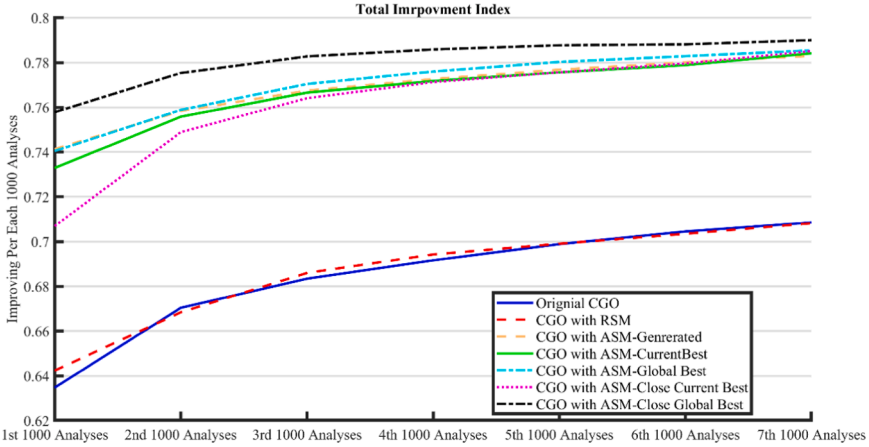


Fig. 10. Global Mean Improvement (*GMI*) curves for the 10-story structure.

Table 5
Final optimum results reported by other researchers compared to the developed methods for the X-braced 20-story structure.

				[16]					
Methods	GA1	PSO1	ACO1	SOS1	ICA1	WOL1	ISA1	FA1	CSS1
Best Result (tons)	3985.43	3066.03	3155.36	2818.76	3106.56	2994.61	2918.67	3077.17	2822.70
Mean Result	5626.49	4683.78	4654.03	3259.86	4150.00	4125.60	3701.93	4116.53	3148.67
No. Analyses	17,500	17,500	17,500	17,500	17,500	17,500	17,500	17,500	17,500
				[16]					
	GA2	PSO2	ACO2	SOS2	ICA2	WOL2	ISA2	FA2	CSS2
Best Result (tons)	3614.55	2993.51	3112.13	2775.39	2913.79	2921.28	2817.29	3036.29	2713.57
Mean Result	4351.38	3983.34	4293.17	3003.12	3675.29	3963.89	3116.21	3465.44	2803.92
No. Analyses	17,500	17,500	17,500	17,500	17,500	17,500	17,500	17,500	17,500
	[17]				Current Work (CGO)				
	EBB-BC	MCC-EB	Original	RSM	ASM-Generated Based	ASM- Current Best	ASM- Global Best	ASM-Close Current Best	ASM-Close Global Best
Best Result (tons)	3021.48	2974.69	2744.73	2745.38	2747.25	2553.07	2511.35	2530.67	**2407.04**
Mean Result	3322.84	3245.55	2964.74	2893.19	3166.15	3114.32	3160.62	2644.81	**2503.98**
No. Analyses	50,000	50,000	50,000	50,000	**7,000**	**7,000**	**7,000**	**7,000**	**7,000**

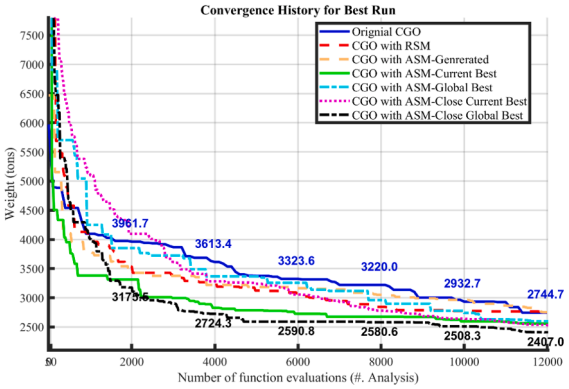


Fig. 11. Convergence history Plot (*CHP*) for the X-braced 20-story building.

achieving the best final objective values. In contrast, the second group, despite a promising start, stagnates and even falls behind or near the original CGO. Meanwhile, the third group maintains steady improvement and secures the second-best overall performance after the ASM-Close Global Best. This analysis highlights the robust performance of both ASM-Close Global Best and ASM-Current Best across all phases of the optimization process, effectively combining rapid initial progress with consistent refinement. It also underscores the strength of ASM-Close Current Best and ASM-Global Best in the exploitation phase, as they compensate for their slower starts with sustained improvements in the later iterations.

Fig. 12 illustrates the *AveP* profiles of each optimization strategy for the X-braced 20-story structure, using both mean and standard deviation values to characterize performance across multiple runs. As shown, the RSM variant offers a clear improvement over the original CGO. It reduces the average structural weight from 2964 tons (Original CGO) to 2893 tons, while also lowering the standard deviation from 296.5 to 286.1 tons. This indicates that RSM enhances slightly both solution quality and consistency. However, both methods still exhibit relatively high mean values and large standard deviations, reflecting unstable convergence behaviour and limited reliability when compared to the ASM-based strategies.

In contrast, ASM-Close-based methods demonstrate substantial superiority in both optimization performance and robustness. For instance, ASM-Close Global Best achieves the lowest average weight of 2503 tons, accompanied by a remarkably low standard deviation of 32.7 tons. Similarly, ASM-Close Current Best delivers an average weight of 2644 tons with a standard deviation of 44.2 tons, further emphasizing the consistent performance of close-based mechanisms. These results highlight the dominance of these two methods in navigating the complex design space of the large-scale X-braced structure.

On the other hand, although ASM-Current Best and ASM-Global Best showed promising results in earlier examples, their performance deteriorates significantly in this larger problem. With mean weights of 3114 tons and 3160 tons and corresponding standard deviations of 336.58 tons and 315.98 tons, their convergence behavior appears highly inconsistent and less effective. This suggests that while these methods may perform well in small- to medium-scale problems, they struggle to maintain quality and stability when applied to more complex structural optimization tasks.

Fig. 13 presents the *CI* plots for all optimization methods, offering a statistical perspective on the reliability and repeatability of the

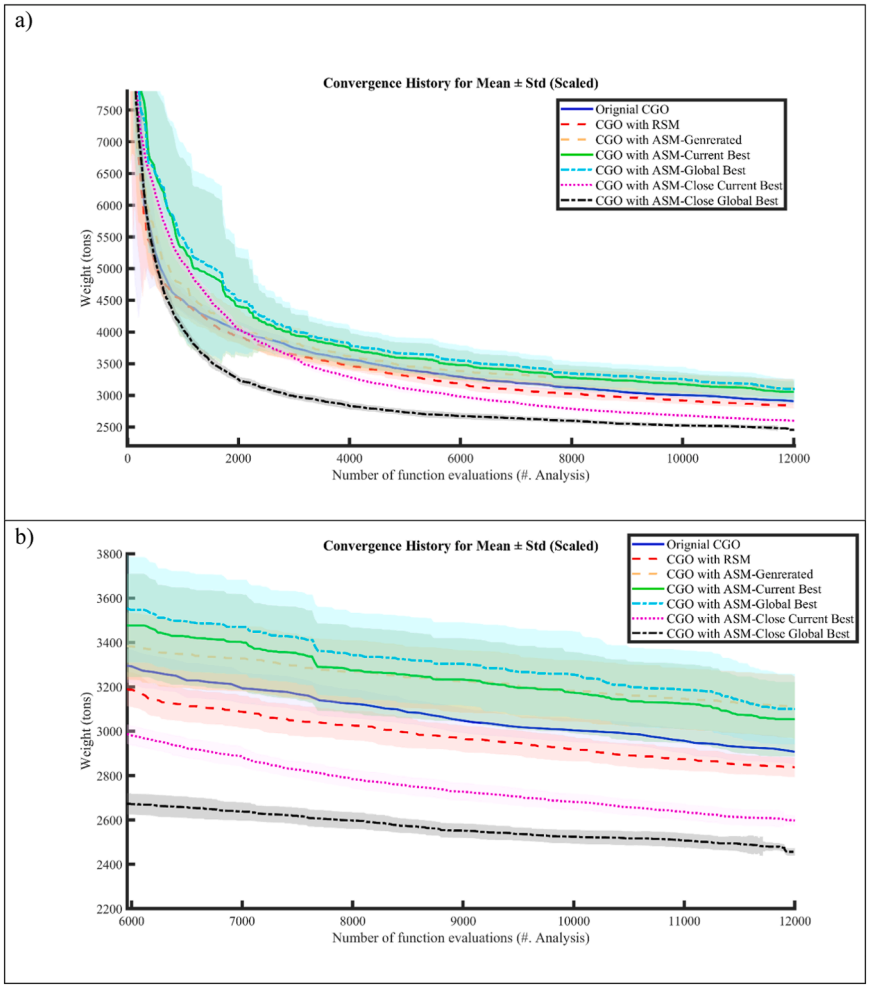


Fig. 12. Average Performance (*AveP*) indicators for the X-braced 20-story building: (a) full range comparison, (b) zoomed-in view.

obtained solutions. The results clearly show that ASM-Close Global Best and ASM-Close Current Best exhibit the tightest confidence intervals, reinforcing their dominance not only in terms of best mean performance but also in consistency across runs. These methods consistently produce reliable and repeatable results, making them the most trustworthy options among the tested strategies. Interestingly, RSM ranks third, offering moderately narrow confidence bounds and further supporting its role as a stable improvement over the original CGO. By contrast, the remaining ASM variants, specifically ASM-Current Best and ASM-Global Best, display much wider intervals, indicating inconsistent behavior in this more complex design problem. The original CGO also shows broad confidence bounds, confirming its lower reliability.

These findings reinforce the conclusions drawn from the previous figures, confirming that the ASM-Close frameworks not only achieve superior mean performance but also demonstrate strong statistical robustness. This combination of accuracy and consistency makes them particularly well-suited for high-dimensional structural design problems, where both solution quality and reliability are essential.

Fig. 14 illustrates the *BLI* trends across six successive intervals, each representing 2,000 structural analyses. In the first interval, the ASM-Close Global Best method stands out as the clear leader, achieving the highest improvement by a significant margin, approximately 3.5% better than ASM-Close Current Best, which holds the second position. This substantial lead becomes even more pronounced when compared to ASM-Global and RSM, with ASM-Close Global Best surpassing them by approximately 8-9%. Notably, the original method exhibits the smallest improvement, consistently lagging by a substantial margin and reflecting its overall inferior performance across all intervals.

Throughout the subsequent intervals, ASM-Close Global Best maintains its dominance, continuing to lead with the highest improvement in each interval. The consistency of this high performance is critical, as it shows the method's ability to maintain strong progress over time, while the other methods, particularly ASM-Global and RSM, exhibit more fluctuation in their improvement rates. ASM-Close Global Best not only stays ahead but also consistently produces the highest improvement values across all intervals, reinforcing its effectiveness in maintaining superior performance.

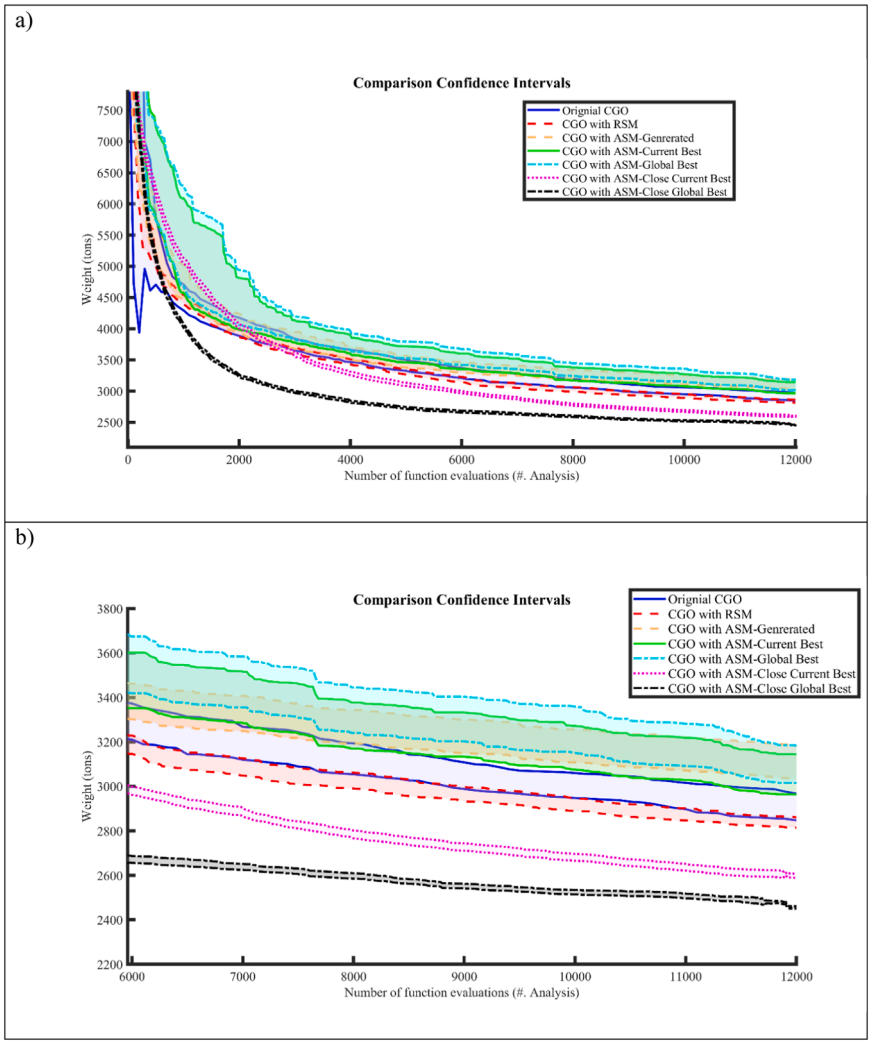


Fig. 13. Confidence Interval (*CI*) plots for the X-braced 20-story building: (a) full range comparison, (b) zoomed-in view.

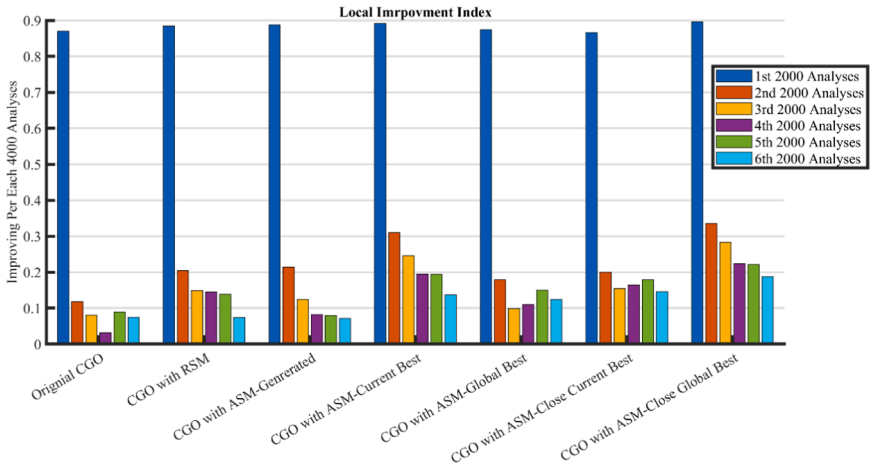


Fig. 14. Best Local Improvement (*BLI*) for the X-braced 20-story building.

In comparison, ASM-Close Current Best shows strong results as well, but it never quite matches the consistently higher improvements of ASM-Close Global Best, particularly in the later intervals. The gap between the two methods remains stable, with ASM-Close Global Best maintaining a lead of around 5-7% in each interval. Meanwhile, the performance of ASM-Global and RSM remains more variable, with both methods struggling to achieve the same level of sustained improvement, especially in the later intervals. RSM, in particular, shows less consistent progress, remaining at a lower level of improvement throughout.

Fig. 15 presents the *GMI* curves over time for the different algorithms. The continuous high GMI values for the ASM-based methods in all intervals underscore their robustness and consistent optimization capability. According to the figure, the ASM-Close Global Best method consistently outperforms all other methods, showing the highest GMI at every interval. In the first interval, the ASM-Close Global Best method shows the highest improvement, with a significant increase of 89%, while all other methods remain below 87%. As the intervals progress, ASM-Close Global Best continues to lead, consistently showing the highest improvement throughout the optimization process. By the final interval, ASM-Close Global Best reaches a peak improvement of 91.8%.

5.3.3. Numerical results for the mega-braced tubed high-rise structure

The last example is the mega-braced tubed high-rise structure that can be considered as a very large-scale example. Table 6 provides a comparative summary of the GBS results, standard deviations, and feasibility status for a total of 16 methods. This includes 9 methods from the literature [16], labeled CSS 1 to CSS 9, and 7 methods developed in the current study.

When focusing only on the feasible solutions as the acceptable ones, the RSM method from the current work achieved the lowest best result of 6251 tons, followed by the Original CGO with 6359 tons. These results outperform all the feasible CSS methods from the reference, including CSS 2 (6720 tons) and CSS 9 (6779 tons), which were the best among the classical approaches. Moreover, the ASM-Close Global Best (5819 tons) and ASM-Close Current Best (5843 tons) variants proposed in this paper also generated feasible solutions and achieved better performance than all CSS methods, as well as better results than RSM and Original CGO.

While the standard deviation values for the ASM-Close Global Best and ASM-Close Current Best methods in the final iteration (175.4 and 197.0 tons, respectively) are higher than that of the Original CGO (159.4 tons), both methods still demonstrate strong performance in terms of consistency when compared to all other approaches.

To better evaluate their effectiveness, Fig. 16 presents the average performance trends, offering a more comprehensive comparison across the optimization process. From this figure, it becomes evident that the algorithms can be categorized into two distinct classes: high-performing and low-performing groups. The high-performing class includes ASM-Close Global Best, ASM-Close Current Best, Original CGO, and RSM, arranged in descending order of performance. These methods not only produce feasible solutions but also maintain favourable average values for both mean performance and standard deviation. In contrast, the second group, comprising ASM-Generated, ASM-Global Best, and ASM-Current Best, exhibits poor performance on both metrics and, more critically, fails to generate feasible results for this structure.

A comparative analysis across three benchmark problems reinforces this trend. While the underperforming methods showed reasonable effectiveness in the first problem, their performance declined significantly in the second problem and became completely inadequate for the current, more complex problem. On the other hand, ASM-Close Global Best, and to a slightly lesser extent, ASM-Close Current Best, consistently achieved superior results and demonstrated robust average performance throughout the optimization iterations. These findings highlight the strength of the proposed adaptive "Close"-filtering-based variants, particularly ASM-Close Global Best, in generating competitive, stable, and feasible solutions, especially as problem complexity increases.

Fig. 17 presents the *CHPs* for the developed methods. Similar to the previous example, the ASM-Close Global Best exhibits the steepest decline in objective values during the initial iterations, indicating a strong capacity for early-stage exploration. More importantly, it continues to consistently improve throughout the middle and later stages of the optimization process, ultimately achieving the best performance among all tested methods. This demonstrates its robustness in both the exploration and exploitation phases, maintaining a steady convergence trend until the final iterations.

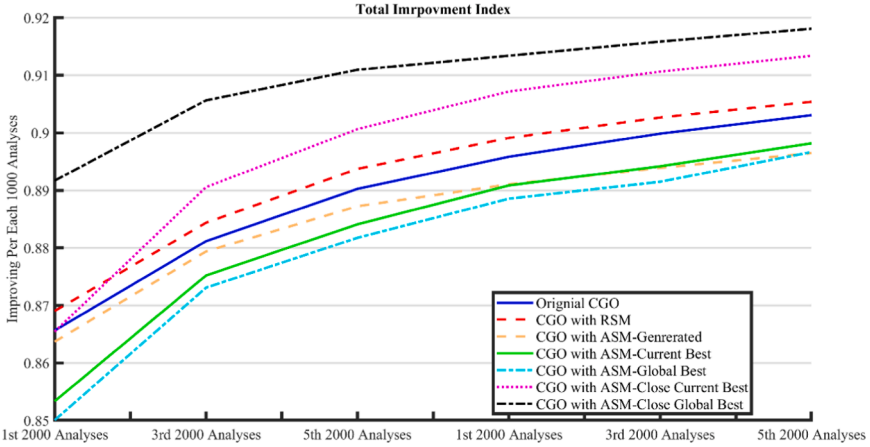


Fig. 15. Global Mean Improvement (*GMI*) curves for the X-braced 20-story building.

Table 6
Final optimum results reported by other researchers compared to the developed methods for the mega-braced tubed high-rise structure.

References	Method	Best Result (tons)	Standard Deviation	Feasible?
[16]	CSS 1	7158.23	930.35	Yes
	CSS 2	6720.45	108.30	Yes
	CSS 3	6453.09	871.51	No
	CSS 4	7044.32	577.81	Yes
	CSS 5	6975.29	445.14	No
	CSS 6	7031.45	649.09	Yes
	CSS 7	7052.44	508.46	Yes
	CSS 8	7158.43	818.38	Yes
	CSS 9	6779.56	202.12	Yes
	Original CGO	6359.19	**159.46**	Yes
	RSM	6251.41	185.81	Yes
Current work	ASM-Generated	4873.80	610.80	No
	ASM-Current Best	5359.26	915.65	No
	ASM-Global Best	5359.26	332.99	No
	ASM-Close Current Best	5843.36	197.01	Yes
	ASM-Close Global Best	**5819.63**	175.41	Yes

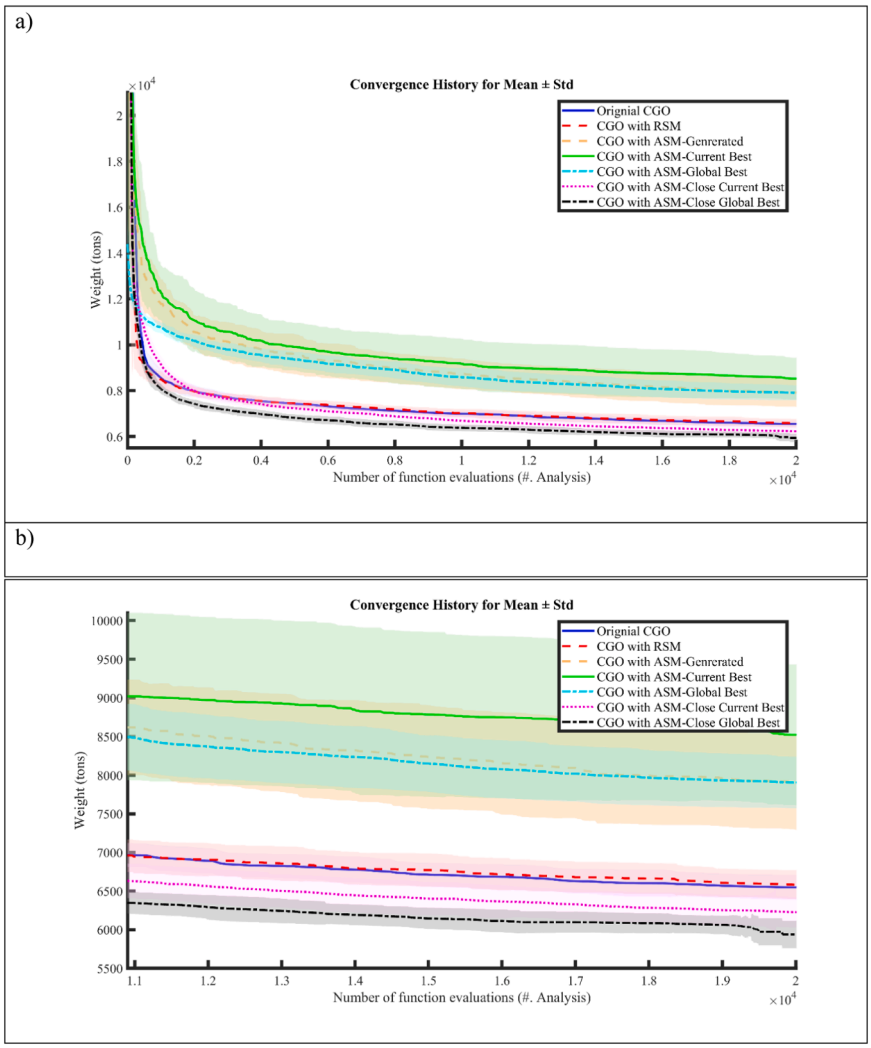


Fig. 16. Average Performance (*AveP*) indicators for the mega-braced tubed high-rise structure: (a) full range comparison, (b) zoomed-in view.

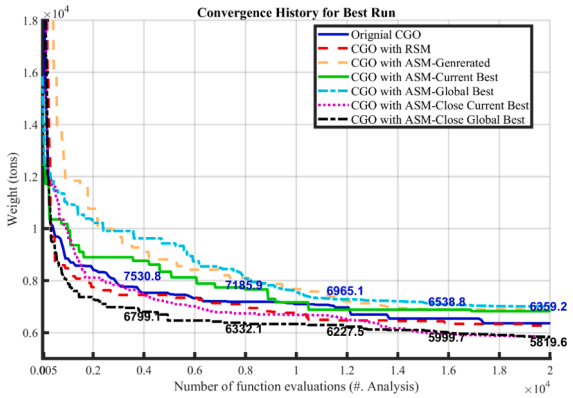


Fig. 17. Convergence history Plot (*CHP*) for the mega-braced tubed high-rise structure.

The ASM-Close Current Best method, while slower in its initial progress, exhibits a notable acceleration during the middle and final stages. This delayed but effective convergence enables it to secure the second-best performance by the end of the optimization process. Its ability to improve results in later stages suggests that the algorithm benefits from a more refined search behaviour over time, which

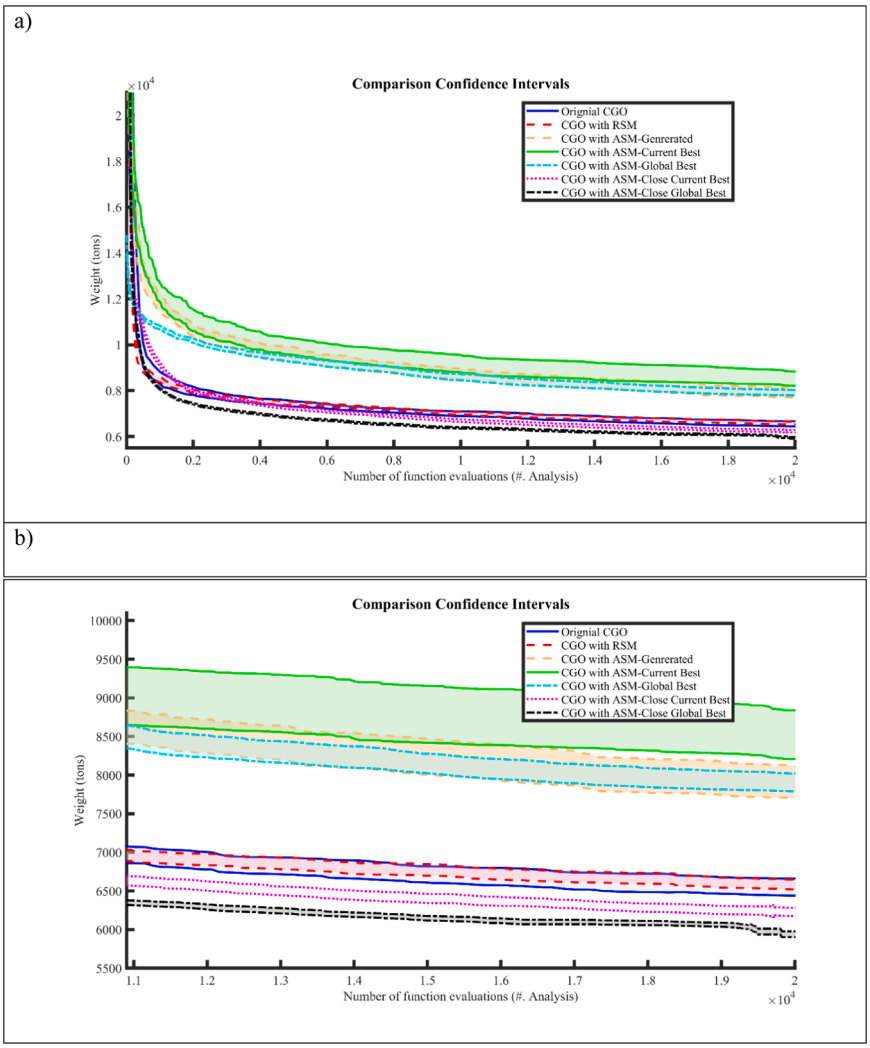


Fig. 18. Confidence Interval (*CI*) plots for the mega-braced tubed high-rise structure: (a) full range comparison, (b) zoomed-in view.

helps in escaping local optima and reaching better feasible solutions.

The RSM method starts with relatively strong performance, even outperforming CGO in the early and middle stages. However, in the later iterations, its convergence stagnates, and it fails to achieve further significant improvements. Consequently, it is overtaken by the ASM-Close Current Best and is eventually unable to outperform the Original CGO in the final solution quality. Interestingly, the performance trajectories of CGO, RSM, and ASM-Close Current Best remain closely aligned until approximately 4,000 analyses. After this point, RSM and ASM-Close Current Best diverge from CGO and follow distinct paths of improvement. Around the 12,000th analysis, RSM reaches a plateau, whereas ASM-Close Current Best continues its upward trend, confirming its superior long-term convergence behaviour.

In contrast, the other three methods, ASM-Generated, ASM-Global Best, and ASM-Current Best, exhibit poor convergence behaviour throughout the optimization process. Their improvement rates remain low across all stages, and more critically, the solutions they produce remain infeasible. This suggests a fundamental limitation in their ability to navigate the feasible search space for this problem instance. These findings underline their unsuitability for handling more complex structural design problems, in contrast to the adaptive “Close” variants, which not only deliver feasible solutions but also excel in performance and convergence speed.

The *CI* plots are presented in Fig. 18, offering a statistical perspective on the stability and reliability of each method’s performance. As shown, the ASM-Close Global Best demonstrates the narrowest confidence interval, indicating not only high-quality solutions but also strong consistency across multiple runs. Closely following is the ASM-Close Current Best, which also exhibits a tight CI, reinforcing its robustness and stable optimization behaviour despite a slightly wider spread compared to the global best variant.

The RSM method ranks next, showing relatively acceptable CI bounds, suggesting a fair balance between performance quality and variability. The Original CGO method, although delivering feasible solutions, reveals a wider confidence range, highlighting a greater degree of fluctuation in its outcomes. While still within acceptable limits, its performance stability is inferior to the top-performing "Close" variants and RSM.

In contrast, the ASM-Global Best and ASM-Generated methods display considerably wider confidence intervals, reflecting high variability and reduced reliability in producing competitive solutions. Finally, the ASM-Current Best records the widest and most unstable CI, signifying very poor performance and significant inconsistency. This result confirms its unsuitability for the large problem at hand. Overall, the CI analysis strongly supports the superiority of the Close variants, particularly the Global Best, in delivering both accurate and dependable optimization results.

Fig. 19 illustrates the *BLI* trends of seven developed variants across five successive evaluation intervals, each representing a 4,000-iteration segment. Among all variants, the ASM-Close Global Best method consistently delivers the highest *BLI* across all intervals, culminating in a particularly strong advantage during the later stages where fine-tuning becomes critical. Closely trailing is the ASM-Close Current Best, which also leverages proximity-based adaptation but with a current-best reference. While slightly less dominant, this method exhibits a similarly steady and high rate of local improvements, demonstrating that spatial closeness in strategy adaptation is a key driver of local refinement success. Together, these two variants underscore the effectiveness of targeted, context-aware switching mechanisms.

In the middle-performance tier, the Original CGO and CGO with RSM offer moderate and relatively stable improvements. Lower performance is observed in the ASM-Generated, ASM-Current Best, and ASM-Global Best variants. In other words, their *BLI* values decline more sharply with each interval, suggesting they are less capable of maintaining improvement as the search progresses. Their relatively poor performance highlights the limitations of broad, untailored switching approaches in scenarios where local enhancement is critical.

Fig. 20 illustrates the *GMI* trends across five intervals, showing how the overall solution quality evolves for each method. The ASM-Close Global Best method consistently outperforms the others, achieving the highest improvements at every interval. Starting at 75.6% in the first interval, its performance steadily increases to 79.2% in the final interval, reflecting a strong and consistent optimization

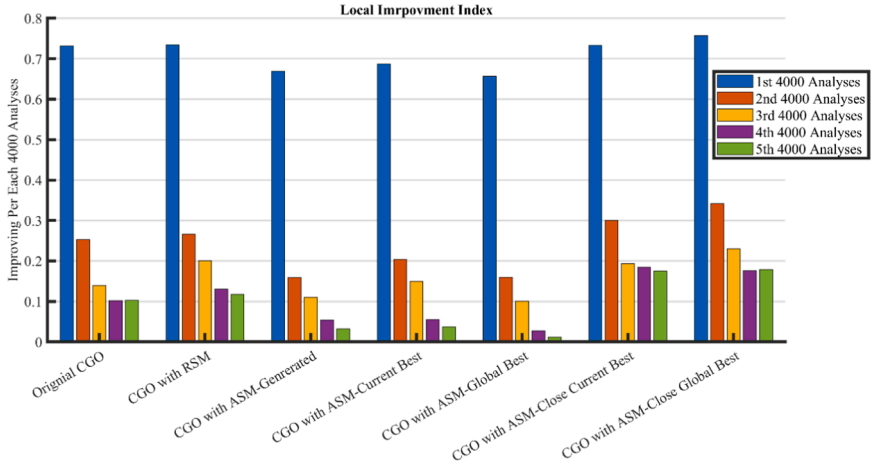


Fig. 19. Best Local Improvement (*BLI*) for the mega-braced tubed high-rise structure.

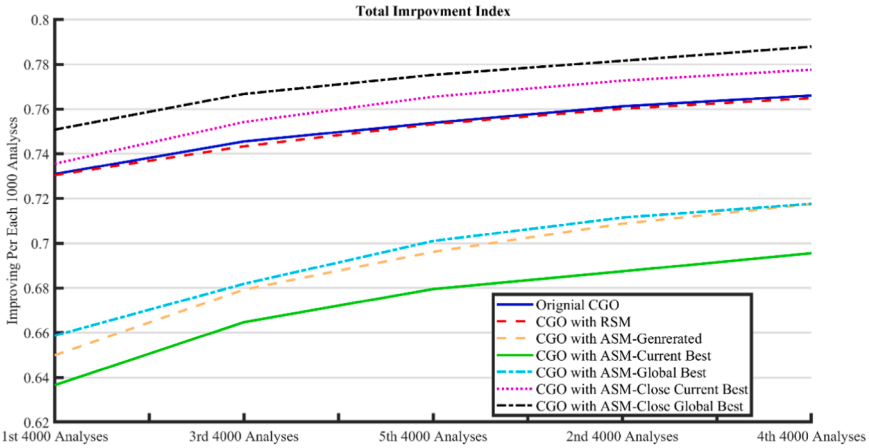


Fig. 20. Global Mean Improvement (*GMI*) curves for the mega-braced tubed high-rise structure.

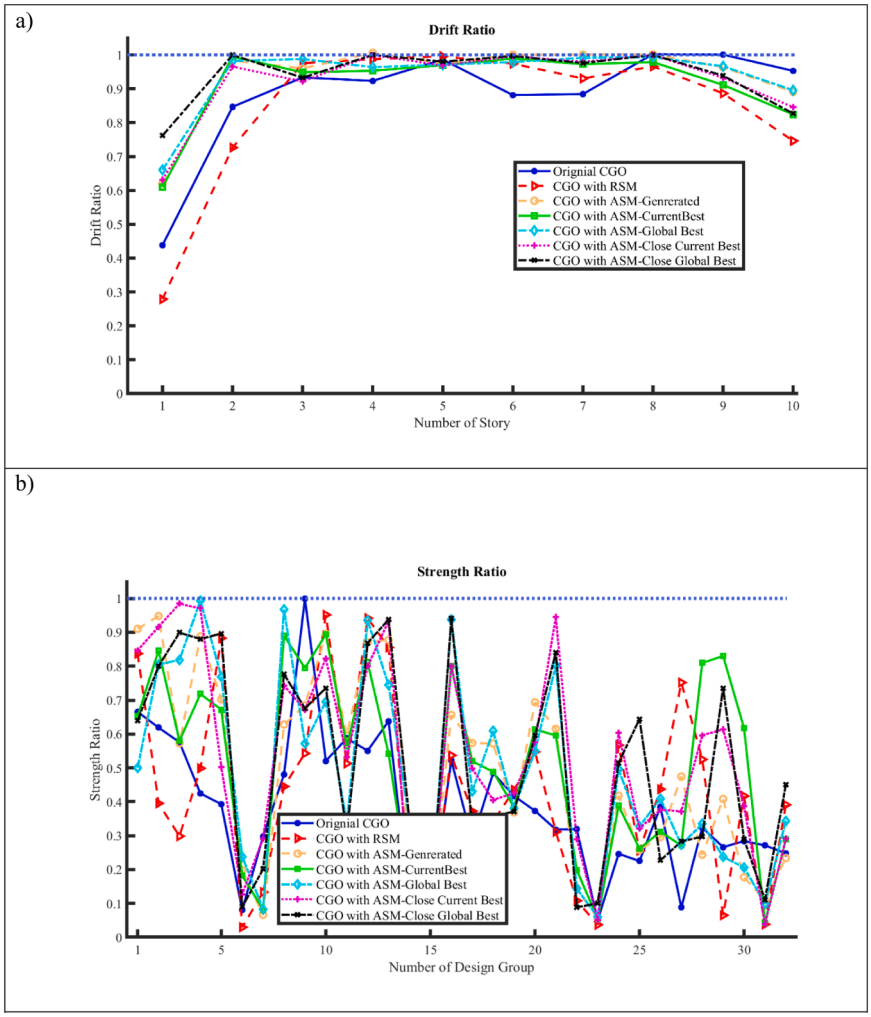


Fig. 21. Drift and Strength Indices of the final design solutions found using the developed methods: a) Drift Indices for the 10-story structure; b) Strength Indices for the 10-story structure; c) Drift Indices for the Xbraced 20-story structure; d) Strength Indices for the Xbraced 20-story structure; e) Drift Indices for the mega-braced tubed high-rise structure; f) Strength Indices for the mega-braced tubed high-rise structure.

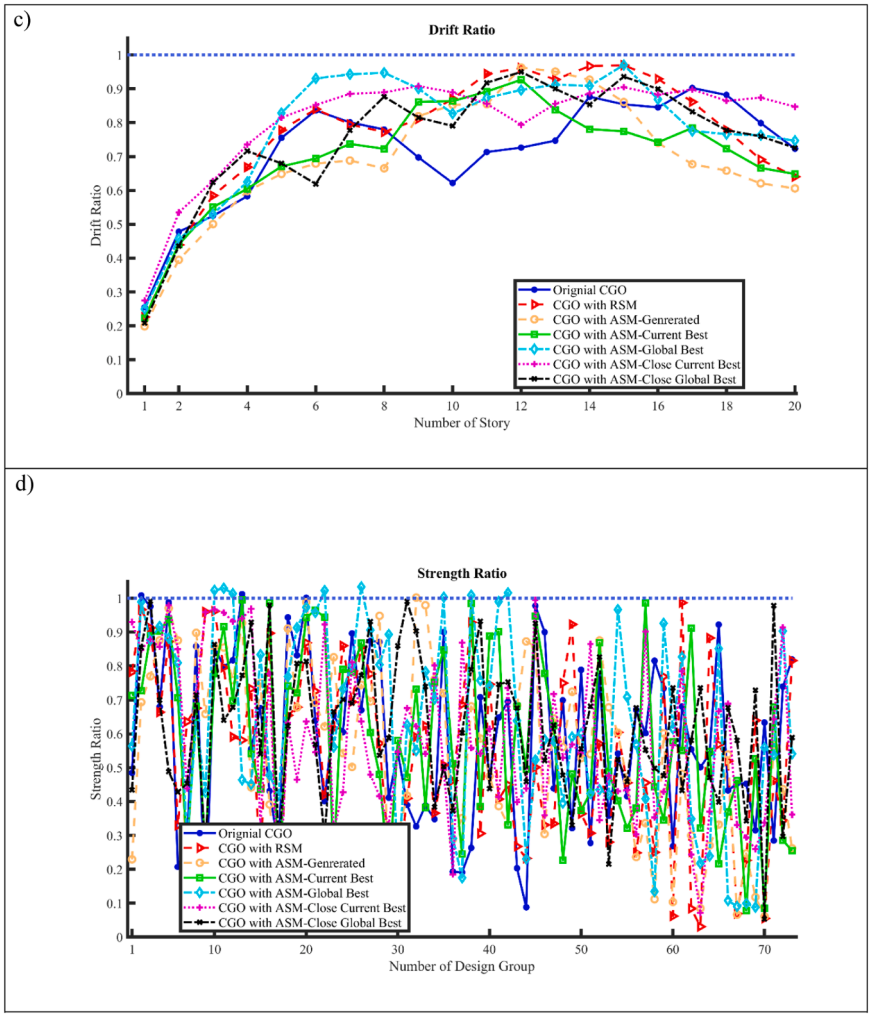


Fig. 21. (*continued*).

process. Similarly, the ASM-Close Current Best method also maintains high performance, reaching 78.2% in the last interval, though it is slightly behind the top-performing method by about 1%.

The Original CGO and CGO-RSM methods occupy the third position. Their performance remains similar across the intervals, with the Original CGO starting at 73.6% in the first interval and ending at 74.9% in the final interval. The CGO with RSM starts at 73.6% and ends at 74.7%. These methods show steady but comparatively lower improvements compared to the ASM-Close-based methods, trailing significantly behind the ASM-Close Global Best, indicating they are less effective in exploring the solution space.

On the other hand, the ASM-Generated, ASM-Global Best, and ASM-Current Best methods show significantly lower mean improvements, reaching very low values at the end of the intervals. The ASM-Generated starts at 65.7% and ends at 72.3%, while the ASM-Global Best method starts at 64.4% and ends at 70.2%. Finally, the ASM-Current Best starts at 67.2% and ends at 72.8%. These methods struggle to improve over time, suggesting that their optimization performance is not as consistent or efficient as the top-performing methods for this example.

5.3.4. Feasibility investigation

One of the most critical aspects of optimizing structural problems is ensuring that the obtained solutions are practical or feasible. A feasible design must satisfy all relevant constraints, including drift limits and strength requirements. To evaluate this essential criterion, Fig. 21 presents both the drift and strength indicators for all developed methods across the three benchmark examples. In this context, indicator values below 1.0 signify feasibility, while values above 1.0 denote constraint violations, rendering the solution infeasible. Notably, a single violation of any constraint results in the entire design being classified as impractical.

To offer a comprehensive view, Table 7 summarizes the maximum drift and strength indicators for each method. From the table, several consistent trends can be observed:

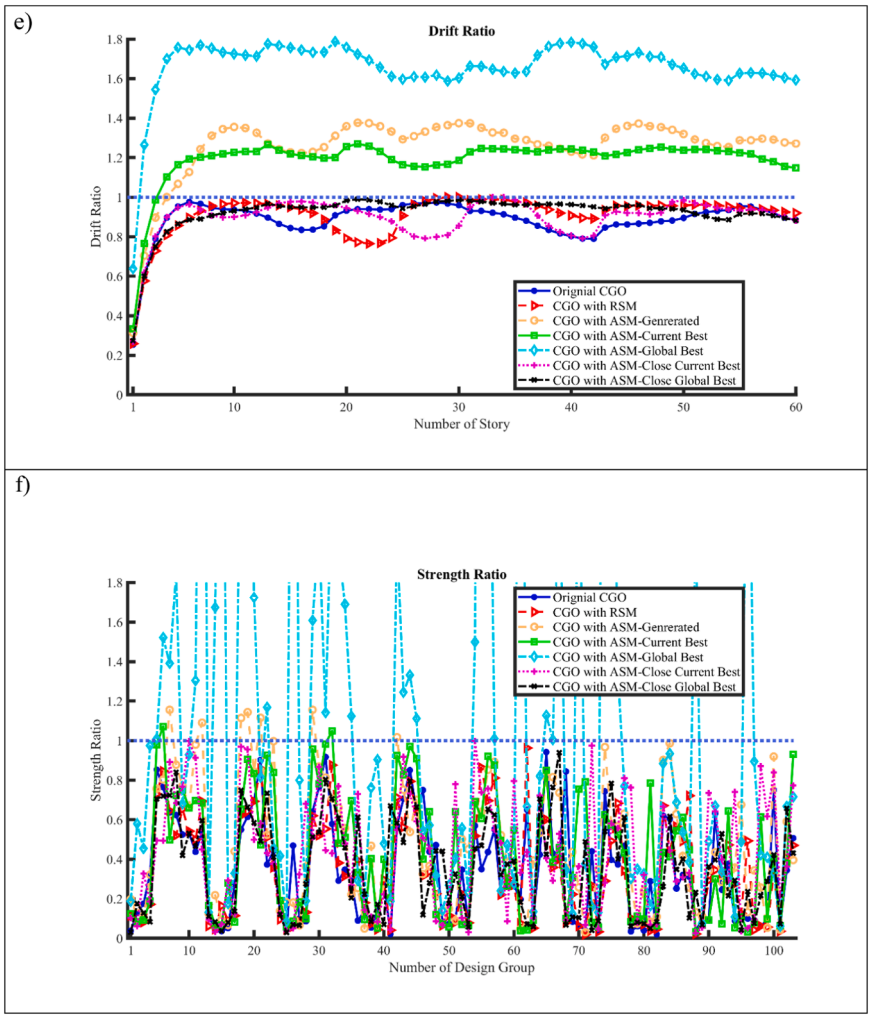


Fig. 21. (*continued*).

Table 7
Maximum drift and strength indices of the final design solutions found using the developed methods.

	Original CGO	RSM	ASM-Generated	ASM- Current Best	ASM- Global Best	ASM-Close Current Best	ASM-Close Global Best
Example 1							
Maximum number of Analyses	1.00	0.99	1.00	0.99	0.99	0.99	1.00
Computational time (sec)	0.99	0.95	0.95	0.89	0.99	0.98	0.94
Example 2							
Maximum number of Analyses	0.90	0.97	0.96	0.93	0.97	0.91	0.95
Computational time (sec)	1.01	0.99	1.00	0.99	1.03	0.99	0.99
Example 3							
Maximum number of Analyses	0.97	1.00	1.37	1.27	1.78	0.99	0.98
Computational time (sec)	0.94	0.96	1.15	1.07	6.54	0.99	0.93

1. ASM-Close Current Best and ASM-Close Global Best consistently provide feasible solutions across all examples, while also achieving the best optimization results in terms of performance.
2. The Original CGO and RSM methods also produce feasible solutions; however, their optimization quality is comparatively lower.
3. On the other hand, the ASM-Generated, ASM-Global Best, and ASM-Current Best methods show a declining trend. Although they generate feasible solutions for the first and second examples, with ASM-Global Best even producing the best result in the first

example, their performance significantly degrades in the third case, where they yield entirely infeasible solutions. This clearly reflects a failure to adequately explore the feasible design space as the problem complexity increases.

This trend indicates that robustness and scalability in maintaining feasibility across large-scale problem settings are key strengths of the ASM-Close variants, particularly under tighter or more complex constraint scenarios.

5.3.5. *Strategy dynamics analysis*

Fig. 22 presents the Solution Distribution Plot (CDP) for the selected problems. By comparing the performance curves, it is evident that ASM-Close Global Best, identified as the most effective method, exhibits a more stable and strategically adaptive behavior across all problem scales. In the first example, which represents a medium-scale problem, this method maintains a relatively steady exploration level during the early stages and begins to significantly reduce the error only after approximately 4,000 analyses. This indicates a deliberate delay in convergence to preserve diversity and avoid premature exploitation. In contrast, for the two large-scale problems (Fig. 22(b) and (c)), ASM-Close Global Best maintains a high and nearly constant distribution distance throughout most of the search process. This suggests enhanced exploration capacity in high-dimensional spaces, leading to a more effective search over the broader

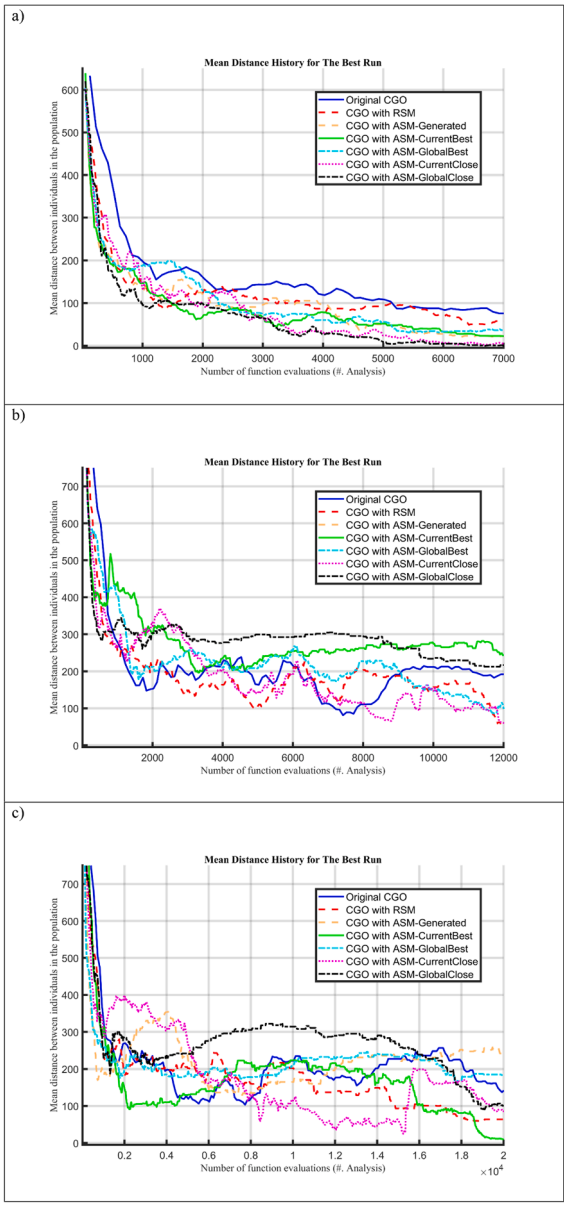


Fig. 22. Solution Distribution Plot (CDP) for: a) the 10-story structure; b) the X-braced 20-story structure; c) the mega-braced tubed high-rise structure.

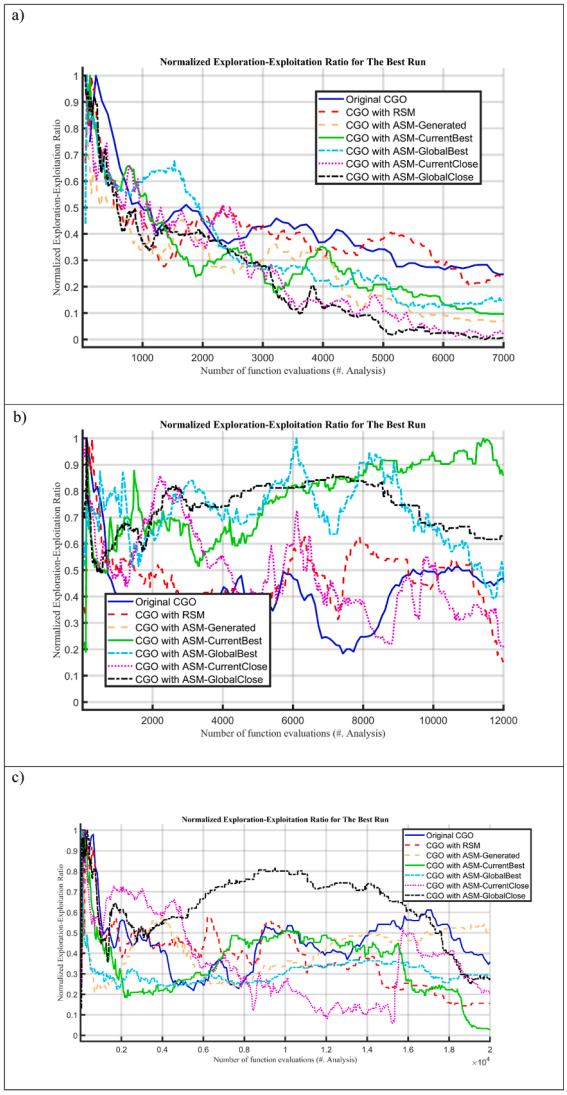
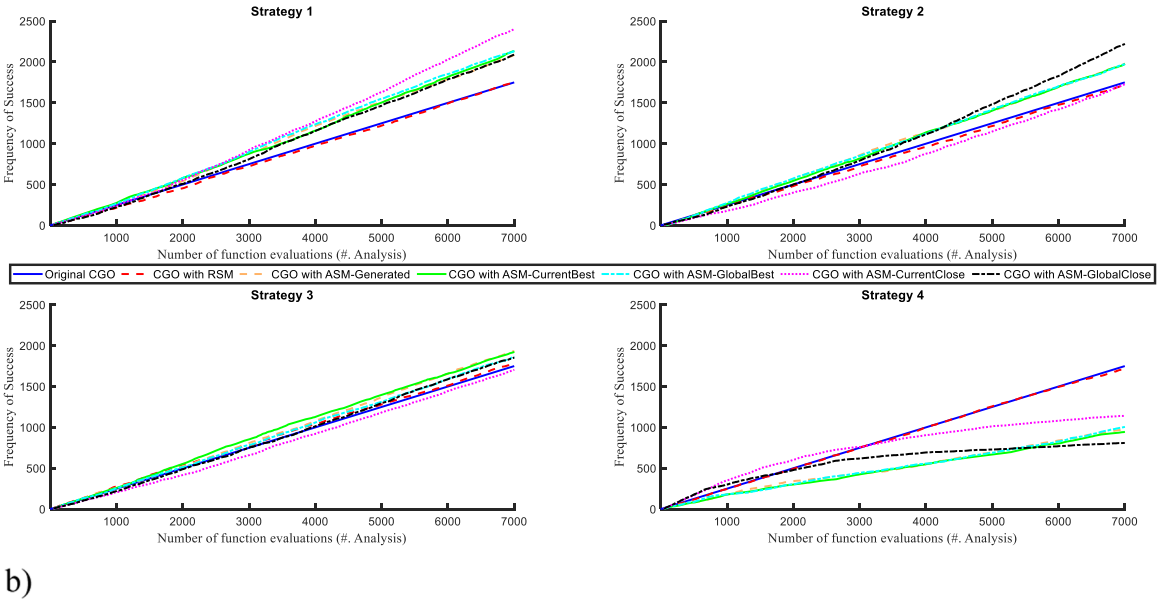


Fig. 23. Exploration–Exploitation Ratio (EER) plots for: a) the 10-story structure; b) the Xbraced 20-story structure; c) the mega-braced tubed high-rise structure.

landscape. Notably, a controlled reduction in distribution appears only in the final stages, reflecting a focused convergence behavior. On the other hand, the other methods, such as the original CGO, demonstrate more erratic patterns. Their distribution plots reveal non-smooth, unstable progression with fluctuations and sudden shifts, particularly in the final iterations, often increasing again after earlier convergence trends. This instability may partially explain occasional success in finding good solutions, but it lacks the consistency and predictability of the best ASM-based method.

Fig. 23 illustrates the normalized Exploration–Exploitation Ratio (EER) over the optimization process for the three structures. The EER captures the dynamic balance between global search (exploration) and local refinement (exploitation), offering deeper insight into the internal behavior of each method. Among all methods, the ASM-Close Global Best strategy demonstrates a remarkably steady EER profile. This consistency reflects its controlled transition between exploration and exploitation phases, indicating a robust internal mechanism that avoids erratic shifts in search behavior. While the EER for ASM-Close Global Best remains relatively high throughout much of the process, this is not a sign of inefficiency; instead, it suggests that the method actively preserves exploration capacity for a longer duration, which is especially valuable in large and complex design spaces. Interestingly, although this method shows a higher EER value compared to others for much of the search process, it successfully reduces this ratio in the final iterations, signaling a smooth and deliberate transition toward convergence. This gradual reduction, rather than a sudden collapse, supports a more refined local search in the closing stages, increasing the likelihood of identifying high-quality solutions. In contrast, other methods, particularly the original CGO, exhibit more volatile EER trajectories. These are marked by abrupt drops and spikes that imply a lack of strategic control over when to explore versus when to exploit. Such behavior can lead to inefficient search patterns, where premature exploitation or excessive wandering prevents optimal convergence.



b)

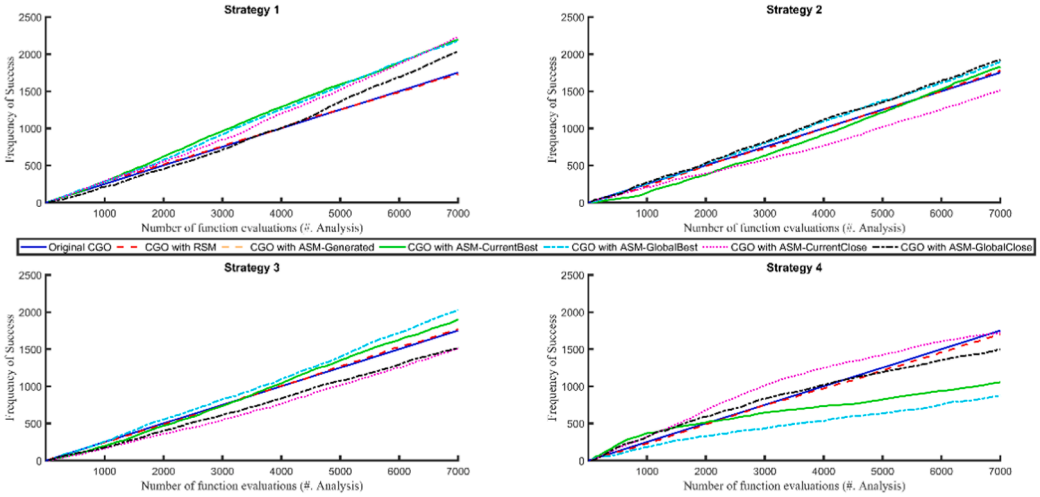


Fig. 24. Strategy-Switch Dynamic (SSD) Graph for: a) the 10-story structure; b) the X-braced 20-story structure; c) the mega-braced tubed high-rise structure.

Fig. 24 presents the SSD Graph, which visualizes how frequently each search strategy is selected throughout the optimization process. The graph offers a comparative view of how different algorithms adapt their internal mechanisms in response to problem characteristics. For both Original CGO and CGO-RSM, the strategy selection pattern remains largely uniform across iterations. This indicates that these methods do not actively adjust their strategy preferences over time, relying instead on a static or pre-defined balance among strategies. Such fixed behavior may limit their adaptability, especially when solving problems with varying landscapes or constraints.

In contrast, ASM-enhanced methods, particularly ASM-Close Global Best, demonstrate a dynamic and responsive use of strategies. A key observation is that the frequency with which each strategy is selected changes notably throughout the optimization process, and this change varies across different problem instances. For instance, Strategy 4 usage in ASM-Close Global Best:

1. Decreases gradually over iterations for Example 1,
2. Remains relatively stable in Example 2, and
3. Increases significantly in Example 3.

c)

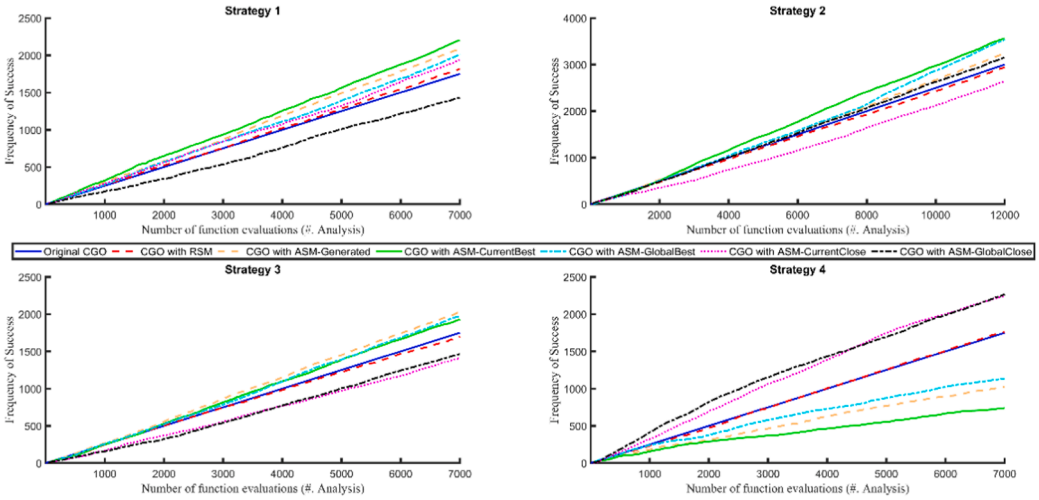


Fig. 24. (*continued*).

These patterns highlight two major advantages of ASM:

Problem-specific adaptation: The framework tunes its strategy preference based on the nature and complexity of the structural problem being solved.

Search-phase awareness: ASM manages the use of strategies dynamically, adjusting selections as the search progresses, emphasizing some strategies early for exploration and shifting toward others later for refined exploitation.

5.3.6. Sensitivity analysis on population size

To assess the influence of population size on performance and ensure fairness in comparison, we conducted a sensitivity analysis of the best-performing variant, ASM-Close Global Best, across three different population sizes: 50, 75, and 100, in addition to the baseline value of 25 used throughout the main experiments. All three structural benchmarks were re-optimized using these larger population sizes, and results are summarized in Table 8.

The results indicate that increasing the population size does not lead to consistent or significant improvements in solution quality. While minor fluctuations in best, mean, and worst results are observed across different cases, the overall performance remains relatively stable. For instance, in the 10-story structure example, increasing the agent count from 25 to 75 or 100 does not yield a statistically significant improvement in the mean solution quality. Similarly, for the X-braced and mega-braced structures, higher population sizes sometimes lead to marginal degradation in performance, particularly for the worst-case outcomes.

To statistically verify this, Wilcoxon Signed-Rank tests were performed to compare the performance of larger agent sizes (50, 75, and 100) against the baseline (25). The *p*-values presented in Fig. 25 are predominantly greater than the 0.05 significance threshold, indicating no statistically significant performance differences for the 10-story and X-braced 20-story structures across varying population sizes. Notable exceptions occur in the case of the mega-braced high-rise structure, where $Npop = 100$ yields *p*-values below 0.01. While these values indicate a statistically significant difference, they do not imply superior performance, as shown in Table 8, the results for $Npop = 100$ in this case yield worse best and mean outcomes compared to smaller population sizes.

These findings suggest that a smaller agent size (25) is sufficient to ensure competitive and stable performance in the proposed framework, justifying its use in the original comparative experiments while maintaining computational efficiency.

6. Efficiency analysis, conceptual and theoretical aspects

6.1. Computational cost, efficiency analysis, and complexity investigation

In structural optimization problems, the most computationally expensive step is typically the repeated finite element method (FEM) analysis, particularly the solution of large systems of linear equations. In the worst-case scenario, solving these equations using dense direct solvers has a complexity of $O(n^3)$, especially for dense or ill-conditioned systems [43,44]. Even with sparse or iterative solvers, the computational cost remains superlinear and usually dominates the total optimization time. In the ASM-CGO framework, although multiple candidate solutions are generated per seed, only one is selected and evaluated per iteration. The associated operations, solution generation, filtering, strategy switching, and updating involve basic vector computations and comparisons, which are computationally negligible. Thus, despite embedding multiple strategies, ASM introduces minimal overhead while substantially

Table 8
Effect of population size on ASM-close global best performance.

Example	N_{pop}	Best Result (tons)	Mean Result (tons)	Worst Result (tons)
10-story structure	25	494.41	511.27	534.08
	50	490.72	509.17	525.49
	75	488.80	513.13	539.32
	100	493.76	517.02	544.89
X-braced 20-story	25	2407.04	2503.98	2625.24
	50	2427.14	2503.12	2640.57
	75	2426.75	2508.49	2625.16
	100	2501.97	2574.55	2711.30
Mega-braced tubed high-rise structure	25	5819.63	5939.34	6438.00
	50	5816.01	6003.56	6306.30
	75	5919.52	6176.24	6367.51
	100	6090.94	6275.50	6427.49

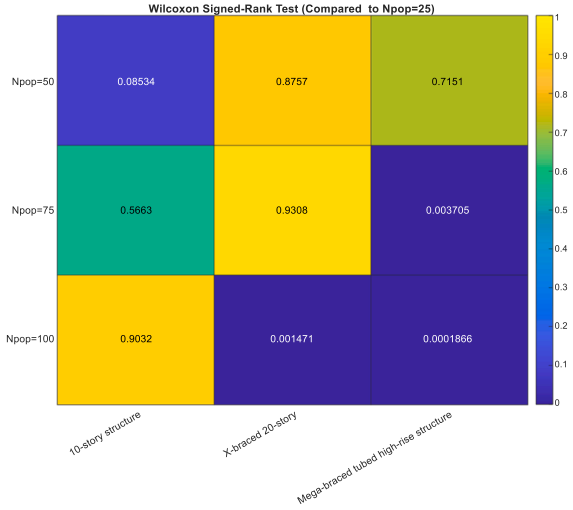


Fig. 25. Wilcoxon test results for population size sensitivity.

Table 9
Complexity evaluation of CGO and CGO-ASM.

Step	CGO	CGO-ASM
Generate Solution	$\sim 4 \times O(Dim)$	$\sim 4 \times O(Dim)$
FEM Evaluation*	$\sim 4 \times O(n^3)$ or $O(n^2)$ or $O(n{\cdot}k)$	$\sim 1 \times O(n^3)$ or $O(n^2)$or $O(n{\cdot}k)$
Objective evaluation	$\sim 4 \times O(Dim)$	$\sim 4 \times O(Dim)$
Filtering	0	$\sim O(4)$
Switching	0	$\sim O(1)$
Updating	0	$\sim 1 \times O(Dim)$
Penalty function	$\sim 4 \times O(Nel)$	$\sim 1 \times O(Nel)$
Global Updating	$O(1)$	$O(1)$

*FEM (Dense Direct solver: $O(n^3)$, FEM (Sparse): $O(n^{1.5–2})$, FEM (Iterative): O(n·k).

reducing redundant FEM evaluations, keeping the overall complexity dominated by the structural analysis step.

To clarify the computational implications, Table 9 provides a detailed step-by-step comparison of the complexity associated with standard CGO and CGO-ASM, evaluated for a single iteration and a single seed. In this table:

- *Dim* refers to the number of design variables (or groups),
- n is the number of degrees of freedom (DoFs) (for a 3D structure, $n = 6 \times$ number of joints),
- *Nel* is the number of elements,
- k represents the iteration number for an iterative FEM solver.

It is important to note that in large-scale structural optimization problems, the number of degrees of freedom (denoted by n) is typically an order of magnitude larger than the number of design variables (Dim). For example, in the first benchmark problem, n =

2310 (based on 6 DOFs per joint × 385 joints), Dim = 32 (design groups), and Nel = 1026 (elements). This disparity means that computational steps dependent on n (such as FEM analyses) dominate the total cost, while operations tied to Dim or constant-time steps are comparatively negligible.

The ASM framework is specifically designed to reduce the number of redundant, high-cost FEM evaluations. While it introduces a few low-cost operations, such as filtering (~O(4)) and switching (~O(1)), these are constant-time steps that add negligible overhead. In contrast, ASM significantly reduces the most expensive part of the process: the FEM analysis. For example, rather than performing four FEM evaluations per seed per iteration (as in standard CGO), ASM reduces this to just one, effectively lowering this step's complexity from ~4 × ($O(n^2)$ or $O(n^3)$) to ~1 × ($O(n^2)$ or $O(n^3)$). Similarly, the penalty function evaluation, which depends on Nel, is also reduced from 4 evaluations per iteration to just 1. Therefore, the small increase in low-cost operations is vastly outweighed by the substantial savings in high-cost computations, making ASM a computationally efficient enhancement, especially as the problem size increases.

To further validate the efficiency of the proposed ASM framework, we conducted a numerical comparison focusing on computational overhead and performance gains. Table 10 summarizes the total computational time required by each algorithm variant, with and without the ASM component, across all three structural optimization problems. Despite incorporating additional operations such as filtering, switching, and updating, the increase in total runtime due to ASM is low. In most cases, ASM variants incur less than a 10% increase in computational time relative to their non-ASM counterparts. This minor overhead is well-justified by the substantial performance improvements ASM provides, especially in guiding the search more effectively. In particular, the ASM-Close Global Best variant, which consistently delivered superior solutions across benchmarks, demonstrates a strong balance between solution quality and computational time. While it introduces a slightly higher runtime compared to the original CGO or RSM approaches, this increase remains within a reasonable margin and does not undermine the scalability of the framework.

To better illustrate the practical efficiency of ASM-based methods, Table 11 presents a comparative analysis of the mean time and number of analyses required to reach predefined performance levels. These levels are expressed as percentages above the best-known solution (e.g., 130%, 120%, 110%, and the final optimum). The results show that the ASM variant reaches much better levels of performance, significantly faster and with fewer function evaluations. In many cases, the ASM-based strategy requires less than half the runtime and analysis effort compared to the original CGO to reach comparable or better performance thresholds. Moreover, for higher-precision targets (e.g., within 10% of the optimum), the original CGO often fails to converge within the maximum allotted budget, while ASM-Close Global Best achieves these targets well within the available resources. This demonstrates that ASM integration not only maintains computational efficiency but also accelerates convergence and enhances solution quality, especially in challenging structural optimization scenarios.

6.2. Scalability of ASM

The scalability of the developed framework is one of the most important aspects to evaluate, particularly in large-scale structural optimization. This section assesses the scalability of ASM with respect to problem dimensionality, number of embedded strategies, and filtering thresholds, supported by both analytical reasoning and numerical results.

- **Problem Dimensionality (Dim):** ASM introduces negligible additional cost as problem dimensionality increases. The complexity of candidate generation, filtering, and switching is either $O(Dim)$ or constant ($O(1)$), involving only simple vector operations or comparisons. By contrast, the dominant computational cost lies in structural analysis (FEM), which typically scales as $O(n^2)$ to $O(n^3)$ depending on solver type and matrix sparsity. Since $n \gg Dim$ in large-scale structures, the relative overhead introduced by ASM remains insignificant as provided in Table 11.
- **Number of Embedded Strategies (S):** While ASM considers multiple strategies per seed, only one candidate is evaluated via FEM analysis. The cost of selecting this candidate based on basic comparison or filtering is linear in the number of strategies but effectively remains constant ($O(1)$) due to the small and fixed set of strategies (e.g., 4 in CGO). Therefore, robustness can be increased by adding strategies with minimal impact on computational load.

Table 10
Total computational times of the developed algorithms with and without ASM.

	Original CGO	RSM	ASM-Generated	ASM- Current Best	ASM- Global Best	ASM-Close Current Best	ASM-Close Global Best
Example 1							
Maximum number of Analyses	7,000	7,000	7,000	7,000	7,000	7,000	7,000
Computational time (sec)	238.9	224.5	235.6	255.9	265.2	245.4	255.4
Example 2							
Maximum number of Analyses	12,000	12,000	12,000	12,000	12,000	12,000	12,000
Computational time (sec)	1,341	1,286	1,366	1,377	1,365	1,677	1,421
Example 3							
Maximum number of Analyses	20,000	20,000	20,000	20,000	20,000	20,000	20,000
Computational time (sec)	3,920	3,541	4,029	3,590	3,584	3,634	4,343

Table 11
Time/computation required to reach predefined performance thresholds.

	Original CGO	ASM-Close Global Best	Original CGO	ASM-Close Global Best	Original CGO	ASM-Close Global Best	Original CGO	ASM-Close Global Best
Example 1	130% of the best result		120% of the best result		110% of the best result		The final best result	
Optimum value	642.2 tons		593.5 tons		544 tons		494.4 tons	
Required number of Analyses	3449	543	~8000*	1207	~15,000*	1,777	21,000*	7,000
Computational runtime (sec)	117.7	19.8	273.2	44.0	512.0	64.8	716.7	255.4
Example 2								
Optimum value	3129 tons		2888 tons		2648 tons		2407 tons	
Required number of Analyses	8,800	2,015	11,075	3,071	~13,000*	4,647	~17,000*	12,000
Computational runtime (sec)	983	238	1,237	364	1,452	550	1,899	1,421
Example 3								
Optimum value	7565 tons		6983 tons		6401 tons		5819 tons	
Required number of Analyses	3,877	1,444	11,558	2,549	17,386	7,103	~34,000*	20,000
Computational runtime (sec)	760	313	2,265	553	3,407	1,542	6,664	4,343

* The method cannot reach to the specific value.

- **Filtering Thresholds:** The filtering mechanism in the ASM-CGO framework is based on objective function values and uses straightforward comparison rules. Its computational complexity remains constant or linear, depending on the specific logic employed (e.g., selecting the best or a random candidate). Importantly, this mechanism does not require any manually adjustable thresholds or tuning parameters. Instead, the filtering thresholds are automatically updated during the search process, guided by improvements in either the current best or global best solutions. This design ensures a fully adaptive behavior while maintaining low computational overhead. Future enhancements may explore more sophisticated learning-based filters, but the current implementation prioritizes simplicity and efficiency.
- **Potential Bottlenecks and Mitigation:** One potential bottleneck in the ASM framework arises from the use of multiple solution-generation strategy modules, which can produce an excessive number of candidate solutions. If not properly managed, this may lead to increased computational overhead. However, this issue is effectively addressed in the CGO-ASM framework through its built-in filtering and selection mechanism, which generates four solutions per seed but ensures that at most one candidate per seed is evaluated per iteration, regardless of how many are produced. This selective evaluation keeps the number of FEM calls small and controlled, thereby maintaining the computational cost per iteration and preserving the scalability and efficiency of the framework.

6.3. Grossone-Inspired Mathematical Modeling

To formalize the hierarchical logic underlying the ASM framework, we draw on the grossone-based numeral system, a mathematical tool designed to represent infinite and infinitesimal quantities in a structured numerical format [45]. The grossone (denoted as ①) allows infinity to be treated as a number rather than as a symbolic concept like ∞. For example, the following strict ordering holds:

$$① > ①^{-1} > ①^{-2}\ ,\ \textit{and}\ ① + 2 > ① + 1$$

Also, similar to natural numbers, grossone can be used as a base to represent numbers hierarchically:

$$\textit{value} = a_0①^{0} + a_1①^{-1} + a_2①^{-2} + \ldots \tag{16}$$

In this representation, the coefficient a_0 has the greatest significance, followed by a_1, and so on, imposing a natural hierarchy of importance among components. This hierarchy can be exploited to model prioritized multi-objective optimization problems. For instance, we can define the following structured objective [45]:

$$\min\left(f_1(x)①^{0} + f_2(x)①^{-1} + f_3(x)①^{-2}\right) \tag{17}$$

In this formulation, $f_1(x)$ must be optimized first. Once it reaches an acceptable or optimal level, attention can then shift to optimizing $f_2(x)$, followed by $f_3(x)$. This ensures that priority among objectives is strictly maintained.

Given that the ASM framework is built upon three core mechanisms, Filtering, Switching, and Updating, this grossone-inspired hierarchy provides a natural way to model ASM's internal logic. Accordingly, the framework's overall goal can be expressed as:

$$\min\left(\textit{filter}(x)①^{0} + \textit{Switch}(x)①^{-1} + \textit{Update}(x)①^{-2}\right) \tag{18}$$

where, $\textit{filter}(x)$, $\textit{Switch}(x)$, and $\textit{Update}(x)$ represent the objective contributions of each respective step in ASM's decision process, applied to a generated solution x.

This hierarchical model ensures that:

- A solution must first pass the filtering criterion (highest priority),
- Then it may be considered for strategy switching, and
- Finally, if eligible, it proceeds to the updating stage.

This structured hierarchy guides the search process toward feasible and high-quality solutions, thereby reinforcing the robustness and convergence behavior of the ASM framework. The next section uses graphical and simplified conceptual explanations to illustrate how this hierarchical logic plays out in practice.

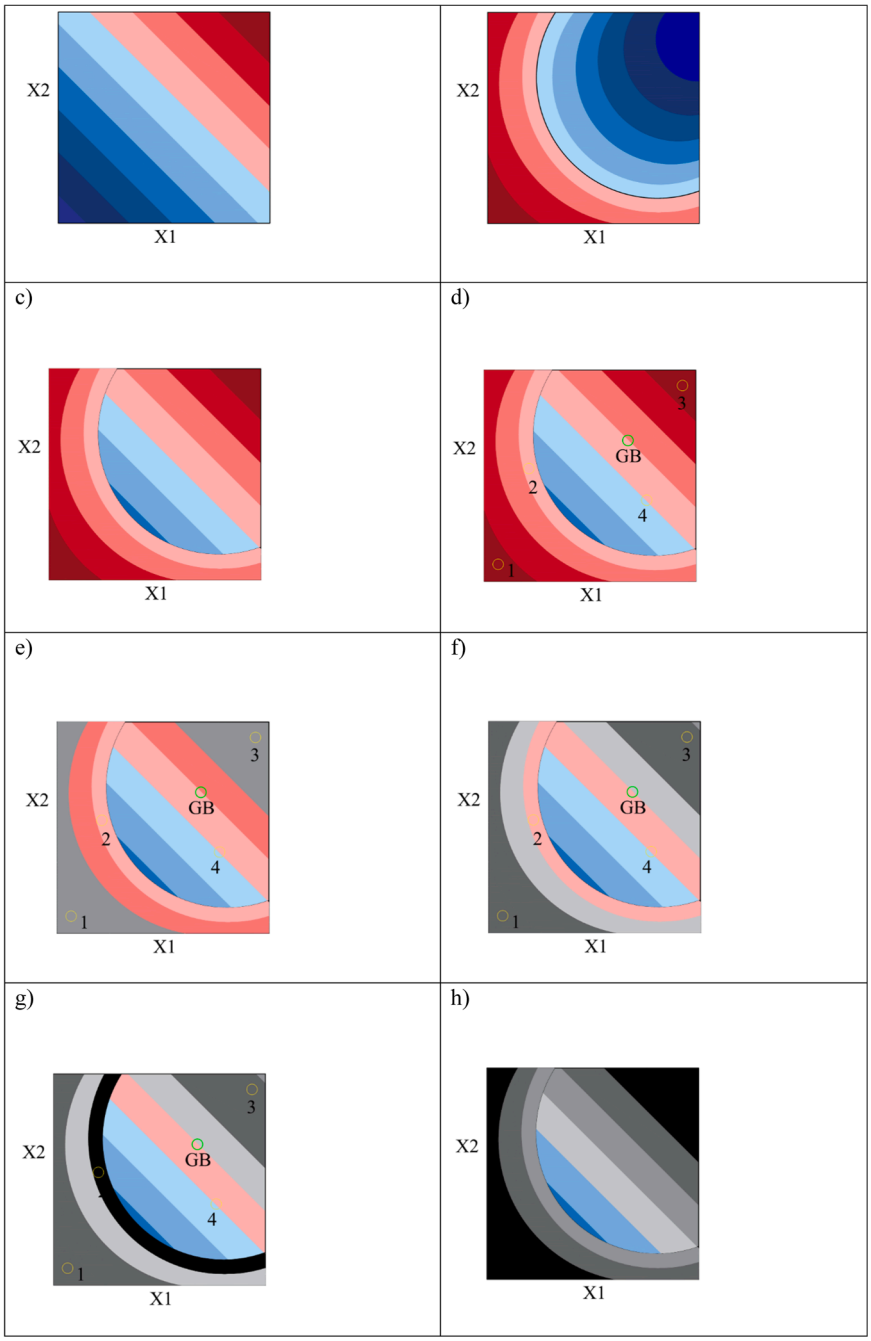


Fig. 26. Graphical Interpretation of ASM Search Dynamics in 2D Design Space: (a) Objective function distribution, (b) Constraint space division, (c) Penalty function mapping, (d) Initial candidate solutions, (e) Switching based on global best, (f) Filtering and updating applied, (g–h) Progressive refinement of the feasible search region.

6.4. Theoretical insights on ASM dynamics

To better illustrate the conceptual behavior of the proposed ASM framework, we begin with a graphical interpretation of the search space in structural optimization. For simplicity and visual clarity, we consider a representative problem with two design variables (see Fig. 26), though the underlying concepts are equally applicable to higher-dimensional problems.

Fig. 26(a) illustrates the design space with respect to the objective function, which exhibits a linear increase from the origin to the upper bounds of the domain. The constraint boundaries divide the space into feasible and infeasible regions, as shown in Fig. 26(b): the feasible region is highlighted in blue, while the infeasible region is shaded in red. As the distance from the constraint boundaries increases, the magnitude of constraint violation grows, represented by darker shades of blue (deeply feasible) or red (deeply infeasible). Combining both the objective and constraint functions produces a penalty-based search landscape, illustrated schematically in Fig. 26(c). According to standard penalty function formulations, the feasible region retains the shape of the original objective landscape, while the infeasible region is distorted according to the severity of constraint violations.

Now, let us consider how ASM operates within this landscape, particularly through the lens of the grossone-inspired model. We focus on ASM–Close Global Best, which consistently delivers top performance across various examples. Initially, CGO generates four candidate solutions (Fig. 26(d)). According to the grossone-inspired model, the filtering step must be satisfied first. Solutions located in "far" regions, those with poor objective values, are discarded, indicated by greying out in the figure (e.g., solutions 1 and 3). In the second step, switching is performed. Only candidates that outperform the current global best (GB) are considered. As illustrated in Fig. 26(e), regions with worse performance than GB are excluded from further evaluation. Finally, the updating step is activated. At this stage, constraint evaluation is permitted. By applying a penalty to infeasible solutions, their fitness is degraded relative to feasible ones, leading the algorithm to prioritize feasible regions (Fig. 26(f)). As the optimization progresses, the search narrows further. Fig. 26 (g) and (h) show how the focus region becomes increasingly refined. This adaptive narrowing ensures two key outcomes: the final solution is highly likely to be feasible, and the final solution closely approximates the true global optimum, based on the capacity of the selected optimizer.

One of the notable features of ASM is its robustness to penalty coefficient tuning, which is often a challenge in constrained optimization. Here's why:

- In early iterations, ASM naturally filters out grossly infeasible or overly costly solutions.
- As the algorithm focuses on the narrow band near constraint boundaries, the variation in constraint violation becomes small.
- Consequently, even a simple penalty function suffices, since all candidates are already near-feasible. The impact of the penalty coefficient diminishes.
- In later iterations, ASM consistently refines the feasible region, ensuring that the final solution is feasible, even without aggressive penalty tuning.

7. Conclusion and future works

This paper introduces a Strategy Management (SM) framework that considers the features of the problem to switch between generated solutions in order to maximize the benefits of performing computationally expensive evaluations. The ASM framework operates by adaptively switching between different guiding strategies, i.e. local and global, based on real-time performance feedback. To achieve this, three main steps are defined: filtering, switching, and updating. In the filtering step, the framework selects the most appropriate solution for further implementation. In the switching step, the filtered solution is switched to either an active or passive state. Finally, in the updating step, the framework decides whether to evaluate (or update) the selected solution.

Based on the different mechanisms selected in the filtering and switching steps, different SM-based methods are developed. Filtering can be performed randomly or based on reference solutions such as the current best or global best. During the switching step, the selected solution may switch directly or based on guidance from the global best or local best solution. Therefore, several SM variants are developed, including RSM, ASM-Generated, ASM-Current Best, ASM-Global Best, ASM-Close Current Best, and ASM-Close Global Best, each employing different policies for choosing active solutions during the optimization process. Among these, the ASM-Close Global Best variant integrates the benefits of global-best knowledge with a close proximity filter, leading to a more informed and focused search direction.

In addition to the strategy-level improvements, this study also introduces some modifications to the CGO algorithm's update equations. The work focuses on addressing CGO's limitations in maintaining a consistent balance between exploration and exploitation across the search process by using a memory-based concept that does not impose extra computational costs on the model. The performance of the proposed methods was evaluated using metrics such as computational cost, Global Best Solution (GBS), Global Mean Improvement (GMI), Success Rate (SR), iterative improvement behaviour, and others. The results consistently showed that ASM-based variants significantly outperform both the original CGO and the CGO-RSM hybrid across the medium-size benchmark. For large-scale problems, close-based filtering methods outperform not only all variants developed in this paper but also other state-of-the-art algorithms from the literature. Specifically, the ASM-Close Global Best method delivered the highest improvements across all evaluation intervals, demonstrating strong convergence characteristics and robust optimization capability. The improvements were not only consistent across metrics but also showed meaningful practical benefits by achieving considerably higher global improvements and consistently maintaining top ranks in both the intermediate and final stages of optimization. To sum up, this work has demonstrated that integrating adaptive strategic guidance and modifying key algorithmic components for large-scale structural optimization problems can lead to substantial performance gains.

Future research will explore several directions to build upon the promising results of the proposed ASM framework. One potential avenue is to integrate advanced switching mechanisms, such as reinforcement learning-based strategy selection, where the optimizer learns optimal switching behavior through interactions with the search landscape. In the present study, we employed a simple rule-based mechanism for filtering and switching. Incorporating learning-based approaches could further enhance adaptability and overall performance across a broader range of problem types.

While this study focuses on single-objective structural optimization, the SM framework is modular and holds strong potential for extension to multi-objective and many-objective optimization problems. This is particularly relevant when the computational costs of evaluating different objectives vary significantly, as SM can help allocate resources more efficiently. Future research could explore how SM performs in handling trade-offs among conflicting objectives, such as weight versus displacement, or cost versus reliability, in structural design. As an example, ASM's strategy management can be adapted by incorporating feedback from Pareto dominance relations and diversity indicators. This enables dynamic switching between exploration and exploitation strategies, tailored to the evolving shape of the Pareto front. Furthermore, the application of SM-based methods may be expanded beyond structural optimization to other computationally expensive and constrained domains, such as energy system design, fluid-structure interaction, or biomedical optimization. These extensions would serve to evaluate the generalizability and practical utility of the SM framework across diverse problem spaces.

CRediT authorship contribution statement

Siamak Talatahari: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Behnaz Nouhi:** Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis. **Amin Beheshti:** Writing – review & editing, Writing – original draft, Visualization, Supervision. **Fang Chen:** Writing – review & editing, Writing – original draft, Supervision. **Amir H. Gandomi:** Writing – review & editing, Writing – original draft, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors acknowledge that this research is supported by the Australian Government Research Training Program Scholarship.

Data availability

Data will be made available on request.

References

[1] N.D. Lagaros, V. Plevris, N.A. Kallioras, The mosaic of metaheuristic algorithms in structural optimization, Arch. Comput. Methods Eng. 29 (2022) 5457–5492.
[2] J. Wu, O. Sigmund, J.P. Groen, Topology optimization of multi-scale structures: a review, Struct. Multidiscip. Optim. 63 (2021) 1455–1480.
[3] X.S. Yang, S. Talatahari, A.H. Alavi, Metaheuristic Applications in Structures and Infrastructures, Newnes, 2013.
[4] A.R. Kashani, C.V. Camp, M. Rostamian, K. Azizi, A.H. Gandomi, Population-based optimization in structural engineering: a review, Artif. Intell. Rev. (2022) 1–108.
[5] M. Soori, F.K.G. Jough, Artificial intelligent in optimization of steel moment frame structures: a review, Int. J. Struct. Constr. Eng. (2024).
[6] L. Velasco, H. Guerrero, A. Hospitaler, A literature review and critical analysis of metaheuristics recently developed, Arch. Comput. Methods Eng. 31 (2024) 125–146.
[7] R.E. Perez, K. Behdinan, Particle swarm approach for structural design optimization, Comput. Struct. 85 (2007) 1579–1588.
[8] J. Leite, B. Topping, Parallel simulated annealing for structural optimization, Comput. Struct. 73 (1999) 545–564.
[9] J.V. Martí, F. Gonzalez-Vidosa, V. Yepes, J. Alcalá, Design of prestressed concrete precast road bridges with hybrid simulated annealing, Eng. Struct. 48 (2013) 342–352.
[10] V. Machairas, A. Tsangrassoulis, K. Axarli, Algorithms for optimization of building design: a review, Renew. Sustain. Energy Rev. 31 (2014) 101–112.
[11] C. Wang, Z. Zhao, M. Zhou, O. Sigmund, X.S. Zhang, A comprehensive review of educational articles on structural and multidisciplinary optimization, Struct. Multidiscip. Optim. (2021) 1–54.
[12] H. Alkhraisat, L.M. Dalbah, M.A. Al-Betar, M.A. Awadallah, K. Assaleh, M. Deriche, Size optimization of truss structures using improved grey wolf optimizer, IEEE Access 11 (2023) 13383–13397.
[13] R. Awad, Sizing optimization of truss structures using the political optimizer (PO) algorithm, Structures (2021) 4871–4894.
[14] A. Baykasoğlu, F.B. Özsoydan, Adaptive firefly algorithm with chaos for mechanical design optimization problems, Appl. Soft Comput. 36 (2015) 152–164.
[15] G.G. Tejani, V.J. Savsani, V.K. Patel, Adaptive symbiotic organisms search (SOS) algorithm for structural design optimization, J. Comput. Des. Eng. 3 (2016) 226–249.
[16] N. Sadrekarimi, S. Talatahari, B.F. Azar, A.H. Gandomi, A surrogate merit function developed for structural weight optimization problems, Soft Comput. 27 (2023) 1533–1563.
[17] S.K. Azad, Design optimization of real-size steel frames using monitored convergence curve, Struct. Multidiscip. Optim. 63 (2021) 267–288.
[18] C.A.C. Coello, Constraint-handling techniques used with evolutionary algorithms, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2022, pp. 1310–1333.
[19] W.K. Hong, M.C. Nguyen, AI-based Lagrange optimization for designing reinforced concrete columns, J. Asian Archit. Build. Eng. 21 (2022) 2330–2344.

[20] H. Liao, W. Yuan, S. Ai, X. Yuan, An univariate method for multi-material topology optimization and its application to engineering structures with unstructured meshes, Comput. Methods Appl. Mech. Eng. 437 (2025) 117749.
[21] Y. Liu, Z. Yang, D. Xu, H. Qiu, L. Gao, A surrogate-assisted differential evolution for expensive constrained optimization problems involving mixed-integer variables, Inf. Sci. 622 (2022) 282–302.
[22] A. Goli, M. Alaghmandan, F. Barazandeh, Parametric structural topology optimization of high-rise buildings considering wind and gravity loads, J. Archit. Eng. (2021).
[23] G. Kazakis, I. Kanellopoulos, S. Sotiropoulos, N.D. Lagaros, Topology optimization aided structural design: interpretation, computational aspects and 3D printing, Heliyon 3 (2017).
[24] S. Mukherjee, D. Lu, B. Raghavan, P. Breitkopf, S. Dutta, M. Xiao, W. Zhang, Accelerating large-scale topology optimization: state-of-the-art and challenges, Arch. Comput. Methods Eng. (2021) 1–23.
[25] N. Sukulthanasorn, J. Xiao, K. Wagatsuma, R. Nomura, S. Moriguchi, K. Terada, A novel design update framework for topology optimization with quantum annealing: application to truss and continuum structures, Comput. Methods Appl. Mech. Eng. 437 (2025) 117746.
[26] F. Ferrari, O. Sigmund, Towards solving large-scale topology optimization problems with buckling constraints at the cost of linear analyses, Comput. Methods Appl. Mech. Eng. 363 (2020) 112911.
[27] P. Hao, D. Liu, H. Liu, S. Feng, B. Wang, G. Li, Intelligent optimum design of large-scale gradual-stiffness stiffened panels via multi-level dimension reduction, Comput. Methods Appl. Mech. Eng. 421 (2024) 116759.
[28] S.K. Azad, S. Aminbakhsh, High-dimensional optimization of large-scale steel truss structures using guided stochastic search, Structures (2021) 1439–1456.
[29] A. Kaveh, S. Talatahari, A general model for meta-heuristic algorithms using the concept of fields of forces, Acta Mech. 221 (2011) 99–118.
[30] R. Falcone, C. Lima, E. Martinelli, Soft computing techniques in structural and earthquake engineering: a literature review, Eng. Struct. 207 (2020) 110269.
[31] E.H. Houssein, M.K. Saeed, G. Hu, M.M. Al-Sayed, Metaheuristics for solving global and engineering optimization problems: review, applications, open issues and challenges, Arch. Comput. Methods Eng. (2024) 1–35.
[32] S. Talatahari, M. Azizi, Optimization of constrained mathematical and engineering design problems using Chaos Game Optimization, Comput. Ind. Eng. 145 (2020) 106560.
[33] S. Talatahari, M. Azizi, Chaos Game Optimization: a novel metaheuristic algorithm, Artif. Intell. Rev. 54 (2021) 917–1004.
[34] R. Oueslati, G. Manita, A. Chhabra, O. Korbaa, Chaos Game Optimization: a comprehensive study of its variants, applications, and future directions, Comput. Sci. Rev. 53 (2024) 100647.
[35] 2001. Manual of Steel Construction–Load Resistance Factor Design. Chicago: AISC.
[36] S. Talatahari, F. Chen, A.H. Gandomi, Developing a robust machine learning framework for predicting the behavior of large-scale structure, J. Build. Eng. 105 (2025) 112204.
[37] A.V. Holden, Chaos, Princeton University Press, 2014.
[38] E.Y. Chan, R.M. Corless, Chaos game representation, SIAM Rev. 65 (2023) 261–290.
[39] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2001) 60–68.
[40] T. Zhang, Z.W. Geem, Review of harmony search with respect to algorithm structure, Swarm Evol. Comput. 48 (2019) 31–43.
[41] S.K. Azad, O. Hasançebi, Upper bound strategy for metaheuristic based design optimization of steel frames, Adv. Eng. Softw. 57 (2013) 19–32.
[42] N. Khodadadi, S. Talatahari, A.H. Gandomi, ANNA: advanced neural network algorithm for optimisation of structures, in: Proceedings of the Institution of Civil Engineers-Structures and Buildings 177, 2023, pp. 529–551.
[43] George, A., Liu, J. & Ng, E. 1994. Computer Solution of Sparse Linear Systems. Oak Ridge National Laboratory.
[44] A.J. Hoffman, M.S. Martin, D.J. Rose, Complexity bounds for regular finite difference and finite element grids, SIAM J. Numer. Anal. 10 (1973) 364–369.
[45] Y.D. Sergeyev, R. De Leone, Numerical Infinities and Infinitesimals in Optimization, Springer, 2022.