Civil Engineering Infrastructures Journal xx,xx

DOI: 10.22059/ceij.2025.388866.2226

Introduction of Peripheral-Perpendicular Optimization with application in structural engineering

Roudak, M.A.[1]*, Shayanfar, M.A.[2], Farahani, M.[3], Karamloo, M.[4], Yavarikhah, M.[3], Ebrahimpour, E.[3]

[1]Assistant Professor, Department of Civil Engineering, Faculty of Engineering, Alzahra University, Tehran, Iran.
[2]Associate Professor, School of Civil Engineering, Centre of Excellence for Fundamental Studies in Structural Engineering, Iran University of Science and Technology, Tehran, Iran.
[3]Graduate Student, Department of Civil Engineering, Faculty of Engineering, Alzahra University, Tehran, Iran.
[4]PhD, Department of Civil Engineering, Shahid Rajaee Teacher Training University, Lavizan, Tehran, Iran.

Received: 15/01/2025

Revised: 30/04/2025

Accepted: 21/05/2025

ABSTRACT

In this paper, a swarm-based metaheuristic optimization algorithm is proposed. The optimization process of this algorithm is conducted by a specific number of defined agents. These agents move through the search space based on their distance from the best candidate and using the combination of tangential- and perpendicular-direction movements. It dynamically adapts the movements to improve the search for optimal results. The agents explore a circular region to uncover potentially better solutions. The radius of this circle decreases gradually to provide a proper balance between exploration and exploitation. In order

* Corresponding author E-mail: a.roudak@alzahra.ac.ir

to validate the performance and efficiency of the presented algorithm, several mathematical and constrained engineering problems are analyzed. The performance of the algorithm is compared against other optimization methods. Based on the examples, the proposed method shows strong exploration and exploitation ability, while many other methods lack at least one of them. Moreover, the proposed method does not have many parameters to be highly sensitive to them. On the other hand, in all mathematical, engineering, and structural examples, the proposed method could successfully handle the local optima due to the combination of peripheral and perpendicular movements. These features together make the proposed method an efficient choice for solving optimization problems.

Keywords: *Metaheuristics; Swarm-based Optimization; Constrained Optimization; Perimeter-Perpendicular Optimization*

1. Introduction

Optimization algorithms can be classified into classical and non-classical (heuristic) categories. Most classical algorithms work based on the gradient or Hessian matrix and search for the optimal point in the neighboring region of the starting point. Hence, these methods can increase the computational cost, especially for problems with a large number of variables. On the other hand, these algorithms can get stuck in local optima, or the convergence to the best solution may be slow. Moreover, when the objective function or constraints are non-differentiable or discontinuous, the applicability of the classical algorithms is limited. These problems led to the consideration of heuristic and metaheuristic optimization methods (Almufti, 2019; safarkhani and madhkhan, 2025). These methods generally combine simple rules to provide a solution in a reasonable time. None-classical algorithms also use the information from previous steps to preserve the search experience and reduce calculation costs (Abdel-Basset et al., 2018).

Three important categories of metaheuristic methods are as follows: evolutionary methods, inspired by the principles of natural selection and biological evolution, such as genetic algorithm (GA); swarm intelligence, inspired by the collective behavior of living organisms, such as particle swarm optimization (PSO); physics-based methods, inspired by physical principles and phenomena observed in nature, such as the gravitational search algorithm (GSA) (Hashemi et al., 2021).

Evolutionary algorithms (EAs) use simulated evolution based on the patterns of reproduction and mutation found in nature. Inspired by Darwinian evolutionary concepts, EAs demonstrate the gradual change of a system over time. In complex cases where other methods may fail, EAs are considered a simple and flexible approach. They belong to a set of random search algorithms and are classified as metaheuristic methods because they provide good solutions, although they do not guarantee the optimal solution. EA includes three algorithms: evolutionary programming, evolutionary strategies, and genetic algorithms.

The most popular type of EA is GA, which was first proposed by Holland (Holland, 1992) in 1970 and later extended by Goldberg in 1989 (Goldberg, 1992). It is often used for the optimization of discontinuous and non-linear problems. Recombinant operators are used to search and design the adaptive system. In this method, the entire population is considered as candidate solutions for the problem, and a population called parents are randomly selected from the current population and used to produce children for the next generation. Over successive generations, an optimal solution evolves through natural selection, as unfavorable traits are removed from the population (Ezugwu et al., 2021; Haldurai et al., 2016).

Evolution strategy (ES) was introduced by Rechenberg (Rechenberg, 1978) and is considered an optimization method for complex and multifaceted functions. The method involves selectively evaluating data and removing redundant codes until the final solutions are represented by number vectors (Coello Coello, 2005).

Evolutionary programming (EP) was first proposed by Fogel (Fogel, 1998) as a method to achieve artificial intelligence. Unlike ES, there are no restrictions on the type of data used. This method has a fixed structure and allows numerical parameters to evolve.

Swarm intelligence, as a subset of metaheuristic methods, have the capability to explore parameter spaces to identify optimal values for specific problems. This algorithm is a group of optimization methods based on self-organizing and interactive approaches inspired by the collective behavior of living organisms such as flowers, ants, bees, birds, and other social creatures to solve optimization problems (Kennedy, 2006).

In swarm algorithms, agents or particles initiate their movements from a starting point in the search space, continuously change their positions, and adjust themselves. This process of adjustment is controlled by a set of rules inspired by social or evolutionary concepts, such as goal orientation and interaction with peers. Unlike evolutionary methods that operate on an individual basis (one-to-one), swarm methods use group interactions among agents to reach the optimal solution.

One of the popular swarm algorithms is PSO, introduced by Kennedy and Eberhart (Kennedy and Eberhart, 1995). PSO, inspired by the social behavior of bird flocks, models the movement of particles in a multidimensional search space to search for optimal solutions. Through iterative updates of particle positions and velocities, PSO effectively balances exploration and exploitation to converge toward optimal solutions. The artificial bee colony (ABC) algorithm, presented by Karaboga and developed by Basturk and Karaboga (Akay and Karaboga, 2012; Karaboga and Akay, 2009), is another swarm-based method. This algorithm imitates the behavior of a bee colony without the limitation of the non-linearity of functions. In this algorithm, new bees find food sources (candidate solutions) based on information from old bees that visited these sources. Many researchers developed other different swarm-based algorithms such as whale optimization algorithm (Mirjalili and Lewis, 2016), firefly algorithm

(Rokh et al., 2024), bat algorithm (Umar et al., 2024), cuckoo search (Abualigah et al., 2024), glowworm swarm optimization (Mohd Shahrom et al., 2023), etc.

Physics-based optimization methods are gaining recognition as a new approach. Some well-known Physics-based algorithms are electromagnetic field optimization (EFO) which is based on electromagnetics(Abedinpourshotorban et al., 2016), gravitational search algorithm (GSA) which is based on the laws of gravity and motion (Rashedi et al., 2009), Simulated Annealing (SA) (Rutenbar, 1989), which imitates the cooling process of metals, the harmony search (HS) method which is inspired by the improvisation process of music (Lee and Geem, 2005; Geem et al., 2001), etc.

In recent years, different classes of metaheuristics have extremely developed. Among swarm algorithms, there are well-known ones such as Ant Colony Optimization (Dorigo et al., 2007), Cuckoo Search (Yang and Deb, 2009), Whale Optimization Algorithm (Mirjalili and Lewis, 2016), Seagull Optimization Algorithm (Dhiman and Kumar, 2019), and Butterfly Optimization Algorithm (Arora and Singh, 2019). Among evolutionary algorithms, there are also other developed methods. Apart from GA, Evolutionary Programming, Differential Evolution (DE) (Storn and Price, 1997), Evolution Strategies (Hansen et al., 2003) can be named. Enhanced Genetic Algorithm (EGA) (Roudak et al., 2024) is another method in this group, which has been recently proposed. Among physics-based algorithms, Simulated Annealing (SA) (Kirkpatrick et al., 1983) and Gravitational Search Algorithm (GSA) (Rashedi et al., 2009) can be mentioned. Other recent methods of this category are Black Hole algorithm (Hatamlou, 2013), Henry Gas Solubility Optimization (Hashim et al., 2019), and Equilibrium Optimizer (Faramarzi et al., 2020). Recently, another group of metaheuristics based on social behavior of humans have developed. In this group fall Bus Transportation Algorithm (Bodaghi and Samieefar, 2019), Political Optimizer (Askari et al., 2020), Group Teaching Optimization Algorithm (Zhang and Jin, 2020), and Student Psychology Based Optimization (Das et al., 2020). Among other well-known methods, Archimedes Optimization Algorithm (AOA) is inspired by the Archimedes' Principle (Hashim et al., 2021). Honey Badger algorithm (HBA) is presented by formulating the digging behavior to represent the exploration stage while exploitation is represented by the process of finding honey (Hashim et al., 2022). War Strategy Optimization (WSO) mimics defense or attack of troops in wars (Ayyarao et al., 2022). Single Candidate Optimizer (SCO) is another method which has been presented recently (Shami et al., 2024).

Although the above-mentioned optimization algorithms have taken large steps of improvement, they present the best performance in some problems and they fail to success in many other ones. Many algorithms show strong exploration while their exploitation is questioned. Many other methods are completely opposite. Some methods fall in local optima. Some methods require a large number of function evaluations. Some methods have several parameters where the methods are highly sensitive to these parameters. These all, necessitate a robust efficient algorithm, on which one can rely to solve a large variety of optimization problems (Shami et al., 2024). In this study, a new swarm-based metaheuristic optimization algorithm is proposed. In this algorithm, a specific number of agents explore the search space to find the best direction of movement to reach the global minimum. To ensure proper exploration and exploitation, adaptive parameters are defined to control the movement of the population. The details of the proposed algorithm and the validation process are described in the subsequent sections. The proposed algorithm offers limited number of parameters to escape from sensitivity of parameters. Besides, it has been tested in various mathematical, structural and engineering problems to check its performance under different conditions. Based on the extracted results and by comparing with other methods of literature, the proposed method has shown that it can more robustly and efficiently converge to the solutions, avoiding local optima.

2. Proposed method

In this paper, a novel metaheuristic optimization algorithm is proposed. This algorithm operates by defining a specific number of agents and moving them through the search space. These agents determine that the approach reaches the global minimum by deciding on the direction of movement. The proposed algorithm can be used for a wide range of optimization problems due to its simple structure, adaptability, and balance between local and global searches, which prevents it from becoming trapped in local optima.

In this section, the mechanism of the algorithm is presented step by step. Then, the exploration and exploitation ability of the algorithm is discussed in detail.

2.1. Mechanism of the algorithm

In this section, the process of the algorithm is presented elaborately. The algorithm consists of the following steps. Each step will be explained in detail in the subsequent sections.

1. Generation of the initial population and random agents
2. Evaluation of the objective function in the generated population, identifying the initial minimum point and considering it as the center.
3. The movement of the agents based on the distance from the center with a certain criterion and the evaluation of the objective function in the updated position of the agents.
4. Determining the new agents, updating the position of the center, and repeating the previous steps until convergence

2.1.1. Initialization

The algorithm is initialized randomly with a specific size in the range of variables. The initial population consists of a set of uniformly distributed random points across the search space. Then, the objective function is evaluated for the population. The point with the minimum value of the objective function is selected from the initial population. This point will be considered as the basis of generating the next population. Then a specific number of points are randomly selected as the agents. These points have a fundamental role in reaching the global minimum by moving through the search space at each iteration.

2.1.2. Agents' movement criterion

As shown in Fig. 1, first, the distance between agents and the best solution (R_i) is obtained. The agents are considered to be on a circle centered on the best solution with radius R_i. Then, a random unit vector λ is multiplied by R_i to move the agents along the circumference of each circle. The unit vector determines the direction on which the new agent is to be located. The agents have to decide whether to move closer to the best solution or further away from it. To achieve this, the agents move inside or outside the circle by a distance d in the direction of λ vector. d is used as a parameter to control the movement of the agents. In fact, d acts as a scalar value that specifies the distance to which the agents should move. As shown in Fig. 1, using λ the agent moves on the perimeter of the circle centered at the best solution with radius R_i. Then, the agent moves perpendicular to this circle with the step size d. Therefore, the proposed method presents a combination of perimeter-perpendicular movements. By adjusting the value of d, the algorithm can regulate the agents' step size, which influences how far the agents explore in the search space during each iteration. d is

$$d = (X_{\max} - X_{\min}) \times (\frac{1}{it^{\alpha}}) \qquad (1)$$

where it is the number of iteration and α determines the rate of decrease in the value of d in each iteration. It is considered a random number between 0.95 and 1.05.

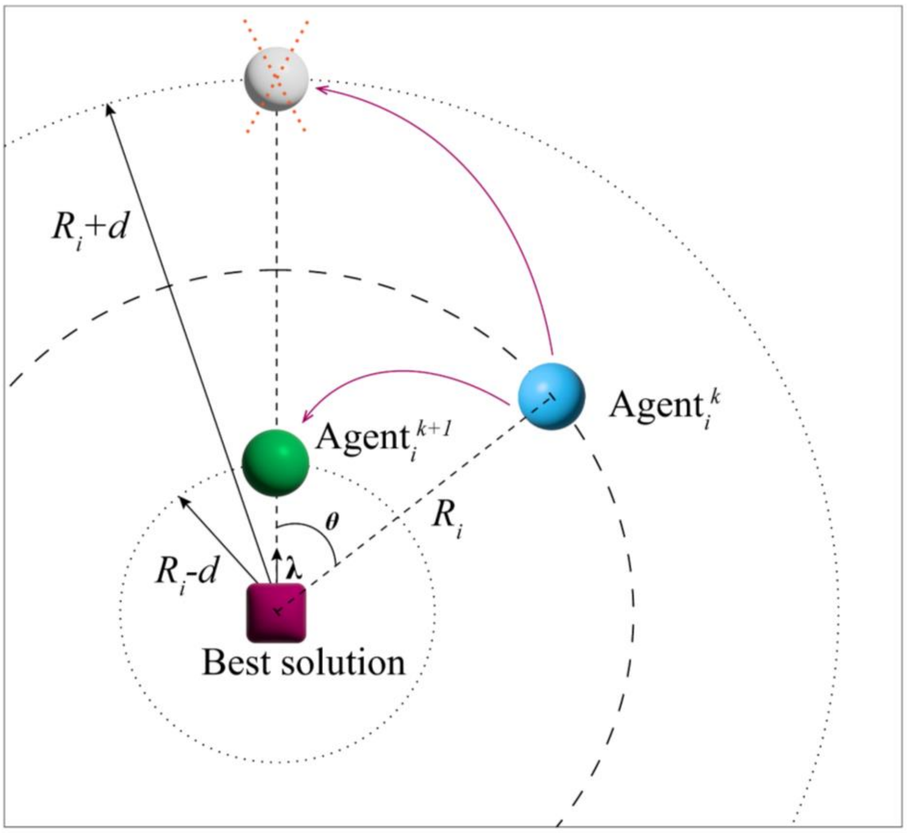


Fig. 1. Selection of agents in each iteration

Based on the defined R_i and d, two points $\mathbf{X}_1$ and $\mathbf{X}_2$ are determined for each agent according to Eq. 2, and the value of the objective function is evaluated at each of them.

$$\mathbf{X}_1^i = \mathbf{X}_c + \boldsymbol{\lambda}(R_i + d) \qquad i = 1, 2, 3, \ldots, N_A \qquad (2)$$

$$\mathbf{X}_2^i = \mathbf{X}_c + \boldsymbol{\lambda}(R_i - d)$$

Where $\mathbf{X}_c$ is the center of the circles which is the best solution at each iteration, $\boldsymbol{\lambda}$ is a random unit vector defined to set the location of the agents on the circumference of the circles, and N_A is the number of agents. Between points $\mathbf{X}_1^i$ and $\mathbf{X}_2^i$, the point with the smaller value of the objective function is considered as the new position of the agent$_i$. Fig.1 indicates the process of agent selection at iteration k+1 based on its location at iteration k.

2.1.3. Position updating procedure and convergence

As mentioned, the agents are moved in the search space by the specified criterion in each iteration. This criterion determines how the agents move in search of optimal solutions. In the next step, among all the new positions of the agents, the one that results in the smallest objective function value is identified and replaces the center of the circles. This process helps in progressively moving the center toward an optimal solution. This procedure during three iterations is illustrated in Fig. 2.

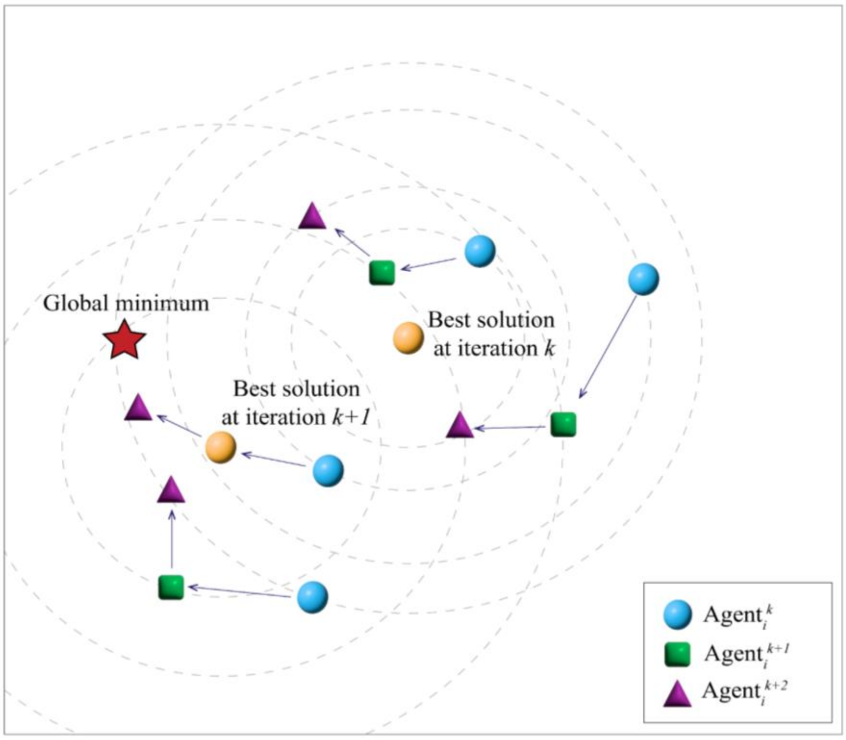


Fig. 2. The movement of agents toward the global minimum during three iterations

By changing the location of the center and updating the position of the agents, new circles are created around the replaced center, and the previous steps are repeated. In the subsequent iterations, as the global minimum is approached, the radius of the circle is gradually reduced until convergence occurs.

2.1.4. Escaping from local optima

The parameters d and λ provide an acceptable exploration ability. To further improve the algorithm ability to escape from local optima, a specific threshold is defined to determine whether the radius (R_i) should be changed from its original value. This altering helps in diversifying the search process and avoiding premature convergence to local optima. At each iteration, a uniformly distributed random number within (0,1) is compared to the predefined threshold. If this random number is smaller than the threshold, the radius changes based on the following equation.

$$R = \left(\frac{MaxIt}{MaxIt + 20 \times it}\right) \times (X_{\max} - X_{\min}) \times a \tag{3}$$

Where a is a random number with uniform distribution. By occasionally changing the radius, the algorithm can explore new areas of the solution space to increase the chance of escaping from local optima.

2.2. Exploration and exploitation ability

2.2.1. Exploration

Exploration refers to the process of searching through a broad range of possible solutions to discover potentially better ones. The goal of exploration is to avoid getting stuck in local optima and examining a wide range of options instead of focusing on a local optimum that might incorrectly seem to be the best solution. In the proposed algorithm, the defined parameters and their variation process provide effective exploration. The initial agents are randomly selected. Therefore, R_i can have large values in the early steps. $\boldsymbol{\lambda}$ is a random unit vector that moves the subsequent agents in random directions. In fact, $\boldsymbol{\lambda}$ guides the agents to move along the circumference of the circles, leading to the coverage of the entire search space during the initial iterations. The parameter d also has a larger value in the first iterations, ensuring that both the inside and outside areas of the circles are explored. Additionally, considering an occasional sudden change in the size of R_i helps to escape potential local optima.

2.2.2. Exploitation

Exploitation involves concentrating on areas near the optimal solutions in the search space. In optimization algorithms, it is essential to narrow the search area as the minimum is approached to refine and improve the existing good solutions.

In the proposed algorithm, the selection of parameters is performed in a way that sufficiently exploits candidate solutions. Since the agents move toward the best solutions at each iteration, exploitation is inherently achieved by the gradual reduction of R_i. Additionally, according to Eq. (1), as the algorithm progresses, the value of d decreases. This reduction allows the algorithm to focus more precisely on the area near the optimal solutions.

3. Results and discussion

In this section, several mathematical and constrained engineering problems are provided to show the validity and effectiveness of the proposed method. To analyze the performance of the presented algorithm, the results of these examples have been compared with five different methods including GA, PSO, ABC, HS, AVOA (Abdollahzadeh et al., 2021), CRY (Talatahari et al., 2021), CBBO (Kaveh and Yousefpoor, 2022), and AHA (Zhao et al., 2022). A summary of the parameter settings for the compared benchmark algorithms is presented in Table 1.

It should be noted that the recommended value of α is for improving efficiency of the proposed method. Besides, like many other meta-heuristic methods, it does not have mathematical convergence proof or guarantee. However, in all following examples the proposed method could robustly and efficiently converge to the correct solution.

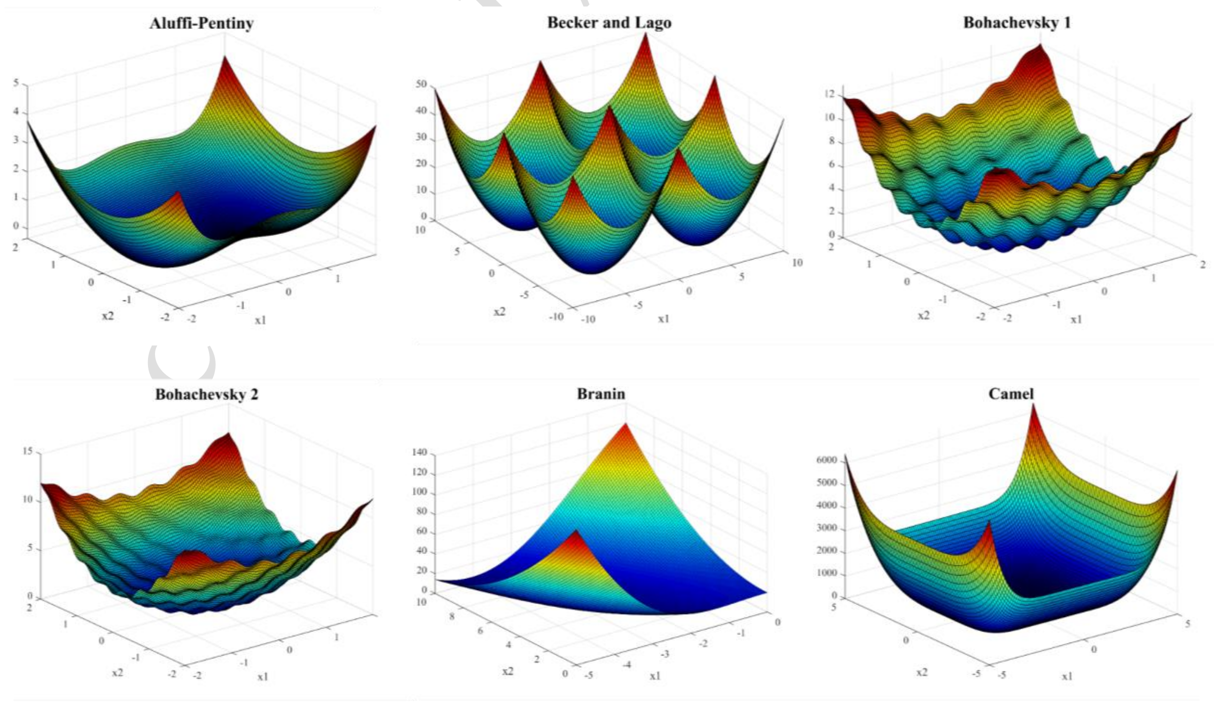
Table 1. Parameters setting of the proposed algorithm and benchmark algorithms for comparison

Algorithms	Parameters	Values
GA	N_{pop} (Population Size)	100
	P_c (Crossover Percentage)	0.7
	P_m (Mutation Percentage)	0.3
	μ (Mutation Rate)	0.1
PSO	N_{pop} (Swarm size)	10
	w (Inertia weight)	1
	C_1 (individual-best acceleration factor)	1.5
	C_2 (global-best acceleration factor)	2

ABC	N_{pop} (Colony Size)	30
	N_o (Number of Onlooker Bees)	30
	a (Acceleration Coefficient Upper Bound)	1
HS	HMS (Harmony Memory Size)	25
	N_{new} (Number of new Harmonies)	20
	$HMCR$ (Harmony Memory Consideration Rate)	0.9
	PAR (Pitch Adjustment Rate)	0.1
	FW_{damp} (Fret Width Damp Ratio)	0.995

3.1. Mathematical examples

In this section, a number of mathematical benchmark functions are selected to evaluate the effectiveness of the proposed optimization algorithm. The specifications of these functions, including their mathematical form, intervals of the variables, and the exact value of the global minimum are presented in Table 2. Fig. 3 shows the 3d view of the bivariate mathematical functions. The number of required function evaluations compared with other selected algorithms is presented in Table 3. The algorithms run 10 times for each function and the mean of the results is reported.



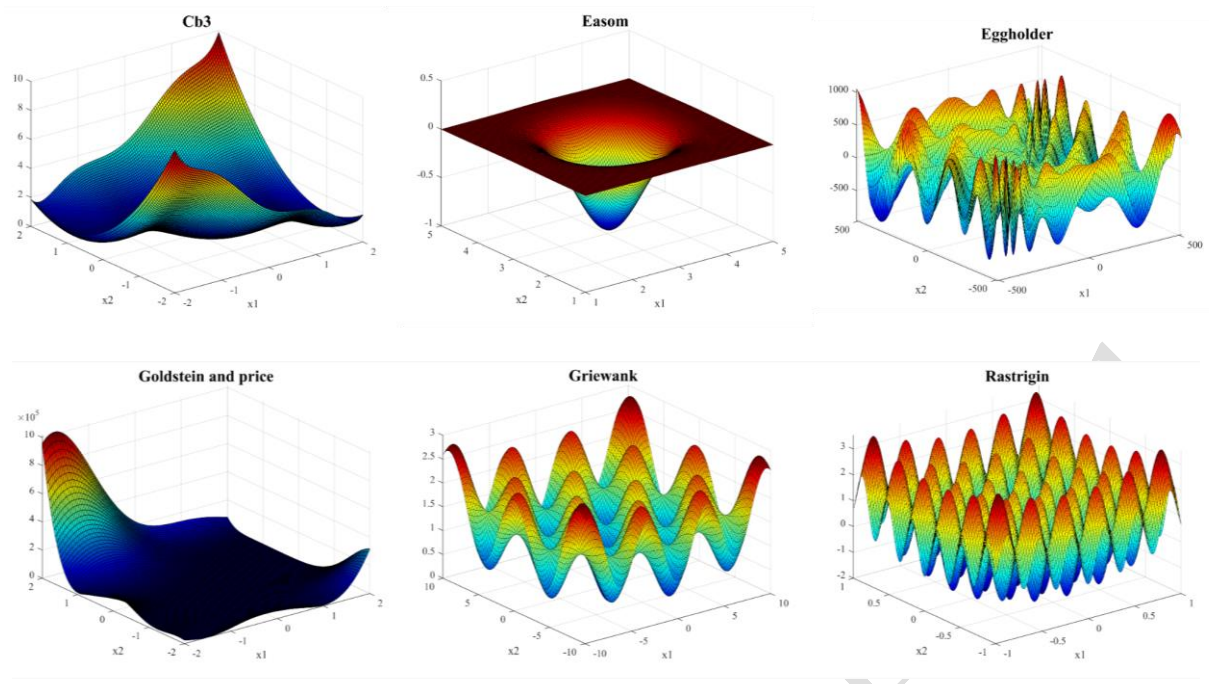


Fig.3. 3D view of the benchmark problems

Table 2. Specifications of the benchmark functions

Function name	**Function**	**Interval**	**Global minimum**
Aluffi-Pentiny	$f(\mathbf{x}) = \frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 + \frac{1}{10}x_1 + \frac{1}{2}x_2^2$	[−10,10]	−0.352386
Becker and Lago	$f(\mathbf{x}) = (\lvert x_1 \rvert - 5)^2 + (\lvert x_2 \rvert - 5)^2$	[−10,10]	0.0
Bohachevsky 1	$f(\mathbf{x}) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$	[−100,100]	0.0
Bohachevsky 2	$f(\mathbf{x}) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$	[−50,50]	0.0
Branin	$f(\mathbf{x}) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$	[−5,5]	0.397887
Camel	$f(\mathbf{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	[−5,5], [10,15]	−1.0316
Cb3	$f(\mathbf{x}) = 2x_1^2 - 1.05x_1^5 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$	[−5,5]	0.0
DeJoung	$f(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2$	[−5.12,5.12]	0.0
Easom	$f(\mathbf{x}) = (-\cos(x_1)\cos(x_2)) \times (e^{(-x_1-\pi)^2 - (x_2-\pi)^2})$	[−100,100]	−1
Eggholder	$f(\mathbf{x}) = (((-x_2 + 47) \times \sin(\sqrt{\frac{x_1 + x_2}{49}})) + ((-x_1) \times \sin(\sqrt{x_1 - (x_2 + 47)}))$	[−512, 512]	−959.64
Exponential	$f(\mathbf{x}) = -\exp(-0.5\sum_{i=1}^{n} x_i^2)$	[−1, 1]	−1

Goldstein and price	$f(\mathbf{x})=[1+(x_1+x_2+1)^2+(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2]\times$ $[30+(2x_1-3x_2)^2(18-32x_1-12x_1^2+48x_2-36x_1x_2+27x_2^2]$	[−2, 2]	3.0
Griewank	$f(\mathbf{x})=1+\frac{1}{200}\sum_{i=1}^{2}x_i^2-\prod_{i=1}^{2}\cos(\frac{x_i}{\sqrt{i}})$	[−100, 100]	0.0
Hartman 3	$f(\mathbf{x})=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2)$ $a=\begin{bmatrix}3&10&30\\0.1&10&35\\3&10&30\\0.1&10&35\end{bmatrix}$, $c=\begin{bmatrix}1\\1.2\\3\\3.2\end{bmatrix}$ and $p=\begin{bmatrix}0.3689&0.117&0.2673\\0.4699&0.4387&0.747\\0.1091&0.8732&0.5547\\0.03815&0.5743&0.8828\end{bmatrix}$	[0, 1]	−3.862782
Hartman 6	$f(\mathbf{x})=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2)$ $a=\begin{bmatrix}10&3&17&3.5&1.7&8\\0.05&10&17&0.1&8&14\\3&3.5&17&10&17&8\\17&8&0.05&10&0.1&14\end{bmatrix}$, $c=\begin{bmatrix}1\\1.2\\3\\3.2\end{bmatrix}$ and $p=\begin{bmatrix}0.1312&0.1696&0.5569&0.0124&0.8283&0.5886\\0.2329&0.4135&0.8307&0.3736&0.1004&0.9991\\0.2348&0.1451&0.3522&0.2883&0.3047&0.6650\\0.4047&0.8828&0.8732&0.5743&0.1091&0.0381\end{bmatrix}$	[0, 1]	−3.322368
Michalewicz	$f(\mathbf{x})=-\sin(x_i)(\sin(\frac{i\times x_i^2}{\pi})^{20}$	[0, π]	−1.8013
Rastrigin	$f(\mathbf{x})=\sum_{i=1}^{2}(x_i^2-\cos(18x_i))$	[−1,1]	−2.0
Rosenbrock	$f(\mathbf{x})=\sum_{i=1}^{n-1}100(x_{i+1}-x_i^2)^2+(x_i-1)^2$	[−30,30]	0.0

As shown in Table 3, in most cases the proposed method achieves convergence to the optimal solution with fewer function evaluations, compared to other methods. As seen, although PSO has slightly better performance in five cases, the overall results indicate the superiority of the presented algorithm over the other investigated methods. In fact, due to its enhanced exploration and exploitation capabilities, the proposed optimization algorithm provides effective and efficient performance compared to common algorithms.

Table 3. Performance comparison for the benchmark functions

		PSO	GA	ABC	HS	Present study
Aluffi-Pentiny	NFE	534	1000	1263	5260	790
	CPU Time	0.21	0.45	0.38	2.24	0.32

Becker and Lago	NFE	444	850	1046	390	376
	CPU Time	0.18	0.39	0.45	0.15	0.14
Bohachevsky1	NFE	718	950	1507	5228	703
	CPU Time	0.32	0.45	0.67	2.27	0.28
Bohachevsky2	NFE	654	850	1316	4200	680
	CPU Time	0.28	0.35	0.53	1.86	0.30
Branin	NFE	606	700	2975	4104	428
	CPU Time	0.24	0.31	1.15	1.91	0.17
Camel	NFE	494	640	1144	1540	291
	CPU Time	0.15	0.21	0.42	0.53	0.09
Cb3	NFE	322	340	887	544	260
	CPU Time	0.13	0.16	0.35	0.24	0.19
DeJoung	NFE	422	850	925	664	406
	CPU Time	0.18	0.39	0.43	0.27	0.11
Easom	NFE	686	1590	4280	9188	788
	CPU Time	0.23	0.56	2.27	4.03	0.31
Eggholder	NFE	636	19870	7164	3600	326
	CPU Time	0.19	8.67	2.84	1.94	0.12
Exponential	NFE	336	240	716	420	193
	CPU Time	0.10	0.09	0.21	0.13	0.05
Goldstein and price	NFE	590	850	2506	2108	564
	CPU Time	0.13	0.28	1.17	1.16	0.10
Griewank	NFE	512	1300	2280	2280	691
	CPU Time	0.12	0.26	0.49	0.45	0.15
Hartman 3	NFE	518	1400	1293	880	329
	CPU Time	0.14	0.31	0.29	0.19	0.08
Hartman 6	NFE	802	2500	4872	2908	880
	CPU Time	0.35	0.62	1.26	0.53	0.39
Michalewicz	NFE	530	700	1855	1144	437
	CPU Time	0.21	0.29	0.34	0.31	0.13
Rastrigin	NFE	518	850	2539	1800	494
	CPU Time	0.20	0.24	0.53	0.35	0.16
Rosenbrock	NFE	884	1500	4699	9124	842
	CPU Time	0.26	0.31	1.35	2.89	0.20

3.2. Constrained engineering problems

The performance of the proposed algorithm was analyzed in the previous section. In this section, the ability of the algorithm to solve constrained problems is investigated. This section includes optimizing various mechanical and structural problems under specific constraints. The number of agents is considered to be five, and the penalty approach is applied to handle the constraints of each problem.

Problem 1. Tension/compression spring

This problem involves designing a tension/compression spring to minimize its weight, subject to constraints on shear stress, surge frequency, and deflection. The schematic of this spring is shown in Fig. 4.

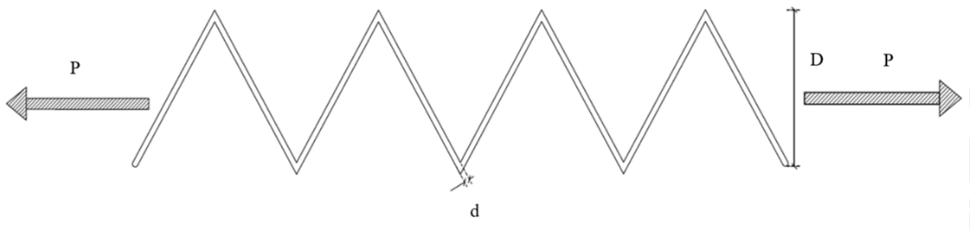


Fig. 4. Schematic of the tension/compression spring

The wire diameter (*d*), the mean coil diameter (*D*), and the number of active coils (*N*) are considered as design variables.

The mathematical formulation of the cost function is shown below

$$f_{cost}(d, D, N) = (N+2)Dd^2 \tag{4}$$

Subject to

$$g_1(d, D, N) = 1 - \frac{DN^3}{71785d^4} \le 0$$

$$g_2(d,D,N) = \frac{4D^2 - dD}{12566 \times (Dd^3 - d^4)} + \frac{1}{5108d^2} - 1 \le 0 \tag{5}$$

$$g_3(d,D,N) = 1 - \frac{140.45d}{D^2N} \le 0$$

$$g_4(d,D,N) = \frac{D+d}{1.5} - 1 \le 0$$

The range of design variables is as follows

$$0.05 \le d \le 2$$

$$0.25 \le D \le 1.3 \tag{6}$$

$$2 \le N \le 15$$

Table 4. Results of the spring design

Algorithm	Optimal design variable			f_{cost}
	x_1	x_2	x_3	
GA	0.0533	0.3988	9.1886	0.012720
PSO	0.0522	0.3684	10.7072	0.012755
ABC	0.0500	0.3166	14.1636	0.012794
HS	0.0523	0.3716	10.4744	0.012689
Present work	0.0522	0.3688	10.6251	0.012688

Table 5. Statistical results of the spring design

Algorithm	Statistical result				
	Best	Average	Worst	Std. Dev.	CPU Time (s)
GA	0.012720	0.013750	0.017853	1.1012E-03	0.235
PSO	0.012755	0.012868	0.013354	2.3936E-04	0.113
ABC	0.012794	0.012981	0.013221	1.8872E-04	0.315
HS	0.012689	0.013671	0.015710	1.1157E-03	0.296
Present work	0.012688	0.012651	0.012740	9.0142E-06	0.108

The obtained optimal weight and its corresponding variables are reported in Table 4. As shown, the presented method achieved the best parameter values among the competitive algorithms. According to Table 5, the performance of the proposed algorithm is clearly superior to that of the other algorithms. According to the tables, there is not significant difference between some methods in this example. This shows that in this example other methods display good performance, too. In fact, since the other methods find the solution appropriately, the proposed method could not improve them significantly in this example. But considering all examples together, the proposed method is an improvement.

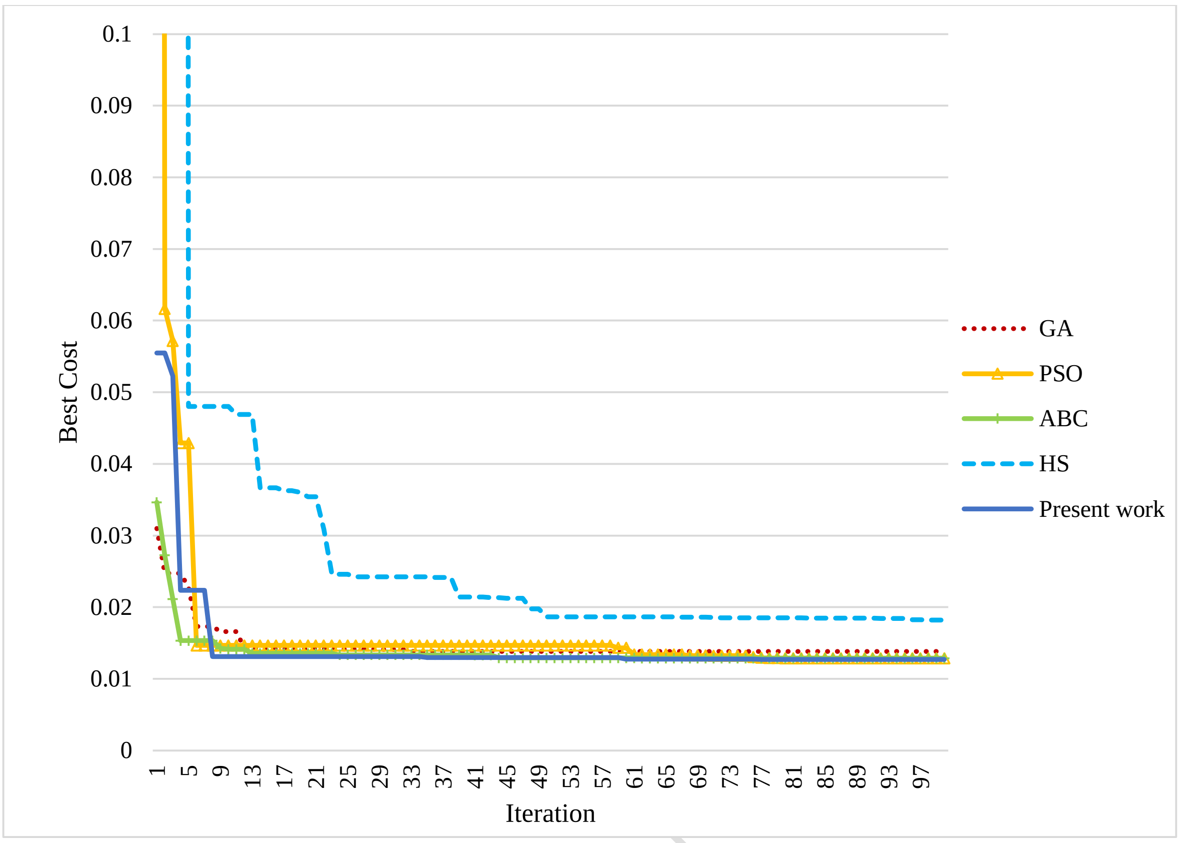


Fig. 5. Convergence curve of the proposed algorithm for the tension/compression spring

Problem 2. Design of a cantilever beam

This problem aims to minimize the weight of a cantilever beam consisting of five hollow square blocks. As shown in Fig. 6, the first block is rigidly supported while a vertical load is applied to the fifth block.

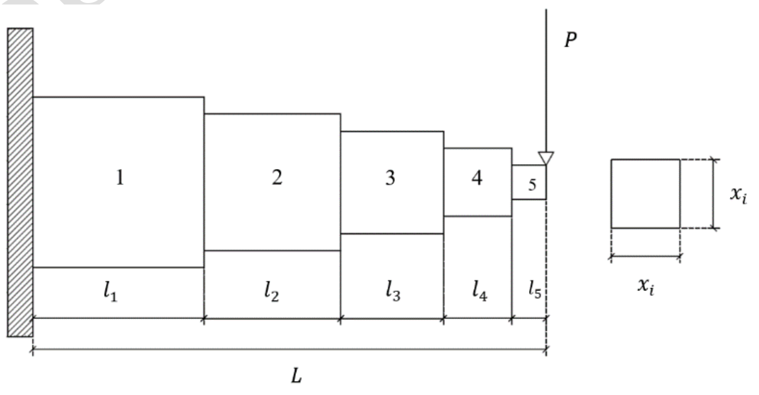


Fig. 6. Schematic of the Schematic of the cantilever beam

Dimensions of the cross-section $\{x_1, x_2, x_3, x_4, x_5\}$ are the design variables. The formulation of the problem, using classical beam theory is as follows

$$f_{cost}(\mathbf{x}) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5) \tag{7}$$

Subjected to

$$g_1(\mathbf{x}) = \frac{61}{x_1^3} + \frac{27}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \le 0 \tag{8}$$

The range of design variables are

$$0.01 \le x_{1,2,3,4,5} \le 100 \tag{9}$$

Table 6. Results of the cantilever beam

Algorithm	Optimal design variable					
	x_1	x_2	x_3	x_4	x_5	f_{cost}
GA	5.9943	4.9034	4.4389	3.4784	2.1346	1.3073
PSO	6.1319	4.7274	4.3862	3.5558	2.1614	1.3081
ABC	6.0888	4.7437	4.4530	3.5133	2.1569	1.3076
HS	5.9826	4.7628	4.4742	3.4160	2.3692	1.3107
Present work	6.2115	4.5254	4.6811	3.4912	2.2135	1.3056

Table 7. Statistical results of the cantilever beam

Algorithm	Statistical result				
	Best	Average	Worst	Std. Dev.	CPU Time (s)
GA	1.30854	1.3791776	1.8765196	1.7521E-01	1.235
PSO	1.30788	1.3082587	1.3148445	2.5014E-03	0.194
ABC	1.30763	1.3080077	1.3093857	7.6484E-04	0.415
HS	1.30782	1.3094487	1.3113736	1.3126E-03	0.317
Present work	1.30660	1.3066241	1.3066074	8.4795E-06	0.128

Table 6 presents the optimal solution for the cantilever beam. As shown, the proposed method reached the optimal parameter values, achieving the best cost function value of 1.3056. The statistical results are reported in Table 7, which indicates that the proposed method outperforms other algorithms in optimizing this problem. Like the previous example, there is not significant difference between some methods in this example. Thus, the performance of all methods is acceptable in this example. It should be noted that even a strong method might reach the solution less efficiently or accurately in a limited number of examples, but generally in most examples it shows better performance.

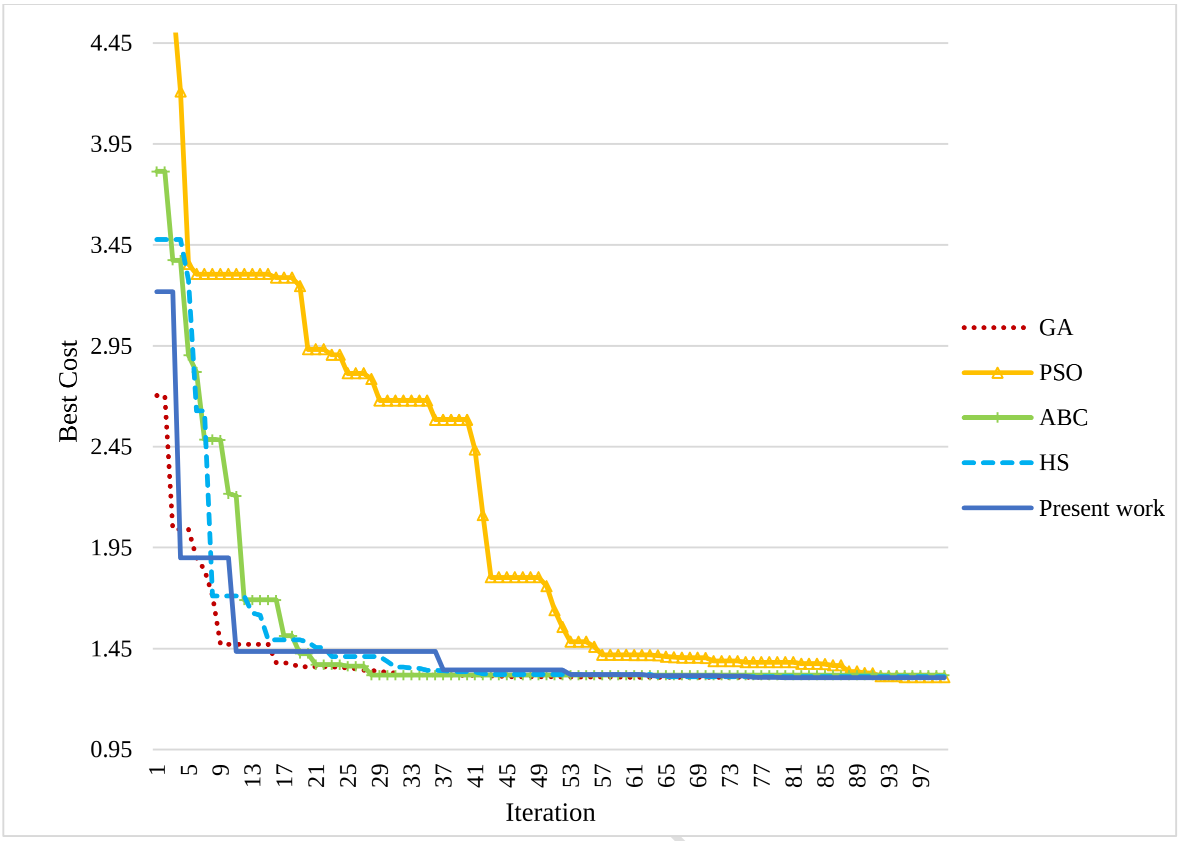


Fig. 7. Convergence curve of the proposed algorithm for the cantilever beam

Problem 3. A 25-bar truss

The weight optimization of a 25-bar spatial truss, shown in Fig. 8, has been studied in this problem. The material has a modulus of elasticity of 10,000 ksi and a density of 0.1 lb/in³. Nodes have displacement limits of ±0.35 in, in all three directions, and all members have stress limits of ±40 ksi. The structure with 25 members, is categorized into 8 groups: (1) A_1, (2) A_2–A_5, (3) A_6–A_9, (4) A_{10}– A_{11}, (5) A_{12}–A_{13}, (6) A_{14}–A_{17}, (7) A_{18}–A_{21} and (8) A_{22}–A_{25}. The loading conditions are listed in Table 8, and shown in Fig. 6 as $P_{1\text{-}7}$.

Table 8. Loading conditions for the 25-bar truss

Node	Loads (Kips)		
	P_x	P_y	P_z
A	0.6	0	0
B	0.5	0	0
C	1	-10	10
D	0	-10	10

Table 9 indicates that the proposed algorithm obtains the optimal weight of the structure, with the lowest standard deviation. The optimal values of the cross-sectional areas for the 25-bar truss structure are also reported.

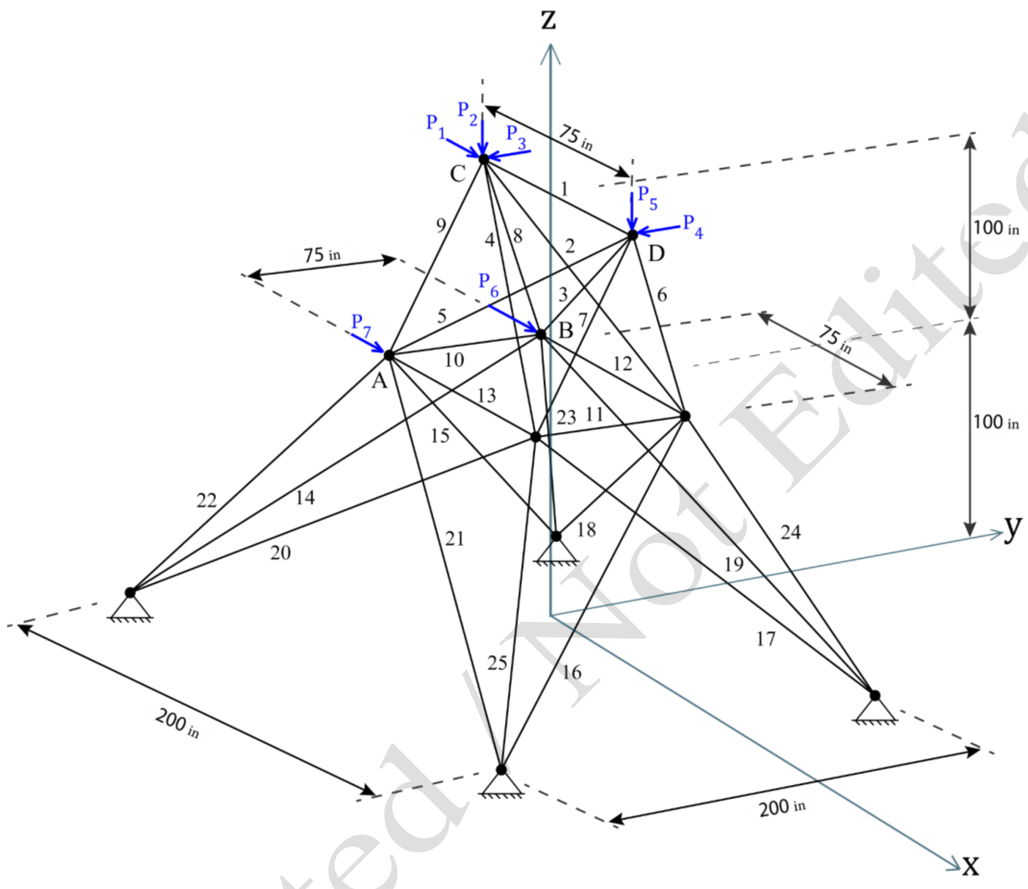


Fig. 8. A 25-bar spatial truss structure

Table 9. Results of the 25-bar truss

Variables	Optimal design variable						
	GA	PSO	ABC	HS	AVOA	CRY	Present work
A_1	1.48	1.21	0.01	0.26	0.07	1.12	1.95
A_2-A_5	2.23	1.24	1.04	1.12	2.18	2.00	1.25
A_6-A_9	2.06	3.29	3.40	3.07	2.83	3.08	3.02
A_{10}-A_{11}	0.93	0.08	0.01	0.02	0.01	0.02	0.04
A_{12}-A_{13}	1.48	1.05	0.80	0.73	0.01	0.24	1.08
A_{14}-A_{17}	1.16	0.86	1.10	1.06	0.66	0.72	0.78
A_{18}-A_{21}	1.04	0.38	0.52	0.71	1.59	1.81	0.53
A_{22}-A_{25}	3.03	3.38	3.37	3.23	2.74	2.41	1.39
Best weight (lb)	538.25	503.29	510.06	505.65	545.96	578.75	501.53
Average weight (lb)	555.69	506.70	525.99	514.30	549.37	615.80	503.65
Std. Dev. (lb)	13.05	3.44	10.95	10.94	4.44	20.68	2.33
CPU Time (s)	315	295	356	346	-	-	286

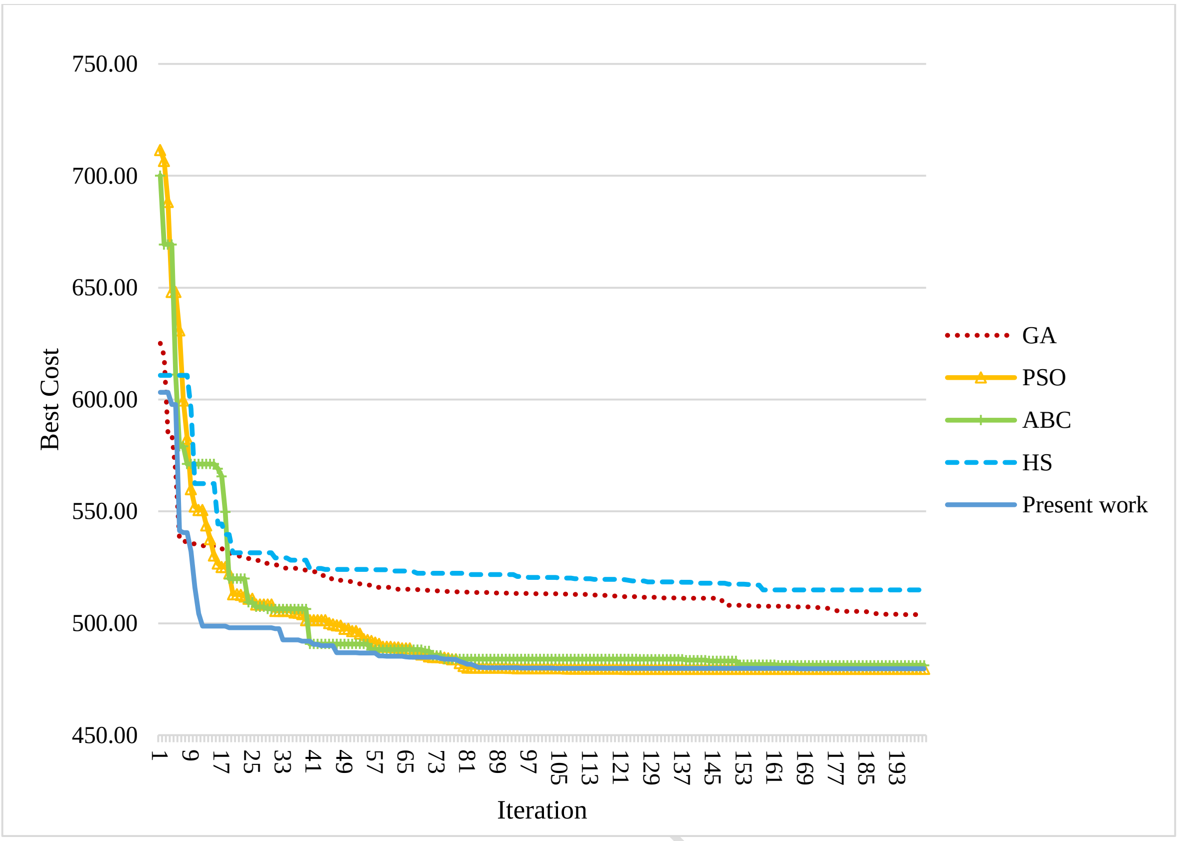


Fig. 9. Convergence curve of the proposed algorithm for the 25-bar truss

Obviously, the proposed algorithm achieves an optimal design with less weight and fewer function evaluations compared to other metaheuristic approaches, indicating its strong performance and efficiency. The convergence curve shown in Fig. 9 further supports this, clearly showing its advantage over the alternatives. A key factor contributing to this performance is the algorithm's ability to explore widely in the early stages and then gradually focus on promising regions. This smooth shift from exploration to exploitation helps the algorithm avoid getting trapped in local minima and improves its ability to find the global optimum.

Problem 4. A 120-bar dome-shaped truss

In this problem, a dome-shaped truss, illustrated in Fig. 10, is selected for structural optimization. Due to geometric symmetry, the truss members are classified into seven distinct groups. All members share the same material properties, with a density of 7971.81 kg/m³ and a Young's modulus of 210 GPa. The cross-sectional area of each member is constrained to lie within the range of 1 cm² to 129.3 cm². The loading conditions include a concentrated load of 3000 kg at node 1,500 kg at nodes 2 through 13, and 100 kg at all remaining nodes. Additionally, the natural frequency constraints are

$f_1 \geq 9$ Hz and $f_2 \geq 11$ Hz

Table 10. Results of the 120-bar dome-shaped truss

Variables	Optimal design variable (cm^2)						
	GA	PSO	ABC	HS	CBBO-23	CBBO-31	Present work
A_{G1}	19.85	20.44	20.12	19.64	19.47	19.63	19.75
A_{G2}	38.01	37.84	37.75	37.92	40.02	40.24	40.11
A_{G3}	11.47	10.61	10.39	10.96	10.74	10.62	10.59
A_{G4}	21.58	21.21	21.35	21.05	21.19	21.13	21.32
A_{G5}	9.14	9.63	9.39	9.57	10.10	9.64	9.65
A_{G6}	12.52	11.49	11.76	11.91	11.71	11.73	11.73
A_{G7}	15.00	15.31	15.15	15.44	14.71	14.90	14.63
Best weight (kg)	8824.521	8726.367	8797.571	8816.354	8710.1	8709.3	8708.5
Average weight (kg)	8951.213	8921.651	8863.857	8934.254	8732.1	8711.4	8709.6
Std. Dev.	17.65	11.26	9.65	16.52	-	-	8.56
CPU Time (s)	397	323	405	431	-	-	309

In this example, the proposed algorithm shows superior performance compared to other metaheuristics algorithms. As reported in Table 11, the algorithm achieved a lighter optimal design with fewer function evaluations, confirming its efficiency and effectiveness. The dynamic control of the agent-to-solution distance allows for extensive exploration in the early iterations and a gradual transition to exploitation later on. This balance prevents premature convergence and enables the algorithm to efficiently find the global minimum. The convergence curve in Fig. 11 clearly indicates the superiority of the proposed algorithm over the other methods.

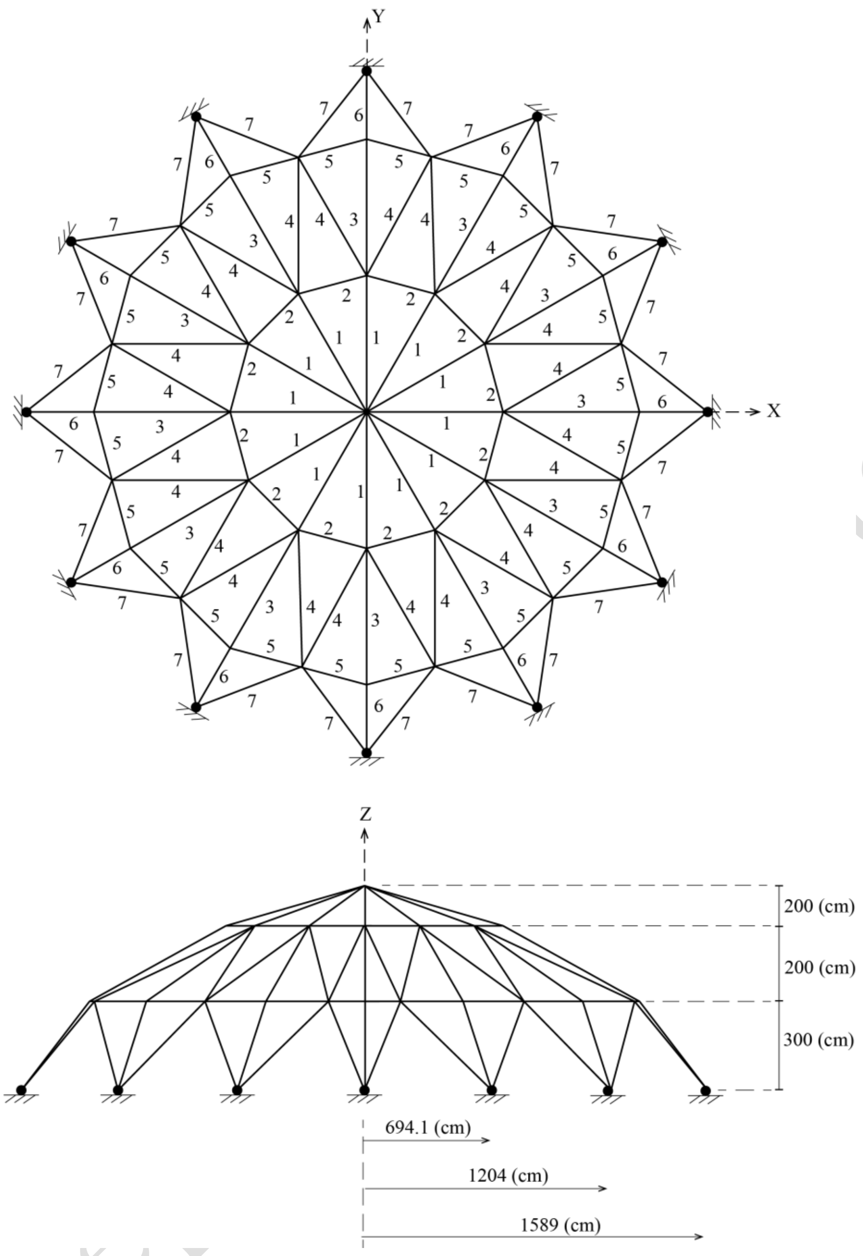


Fig. 10. The 120-bar dome-shaped truss

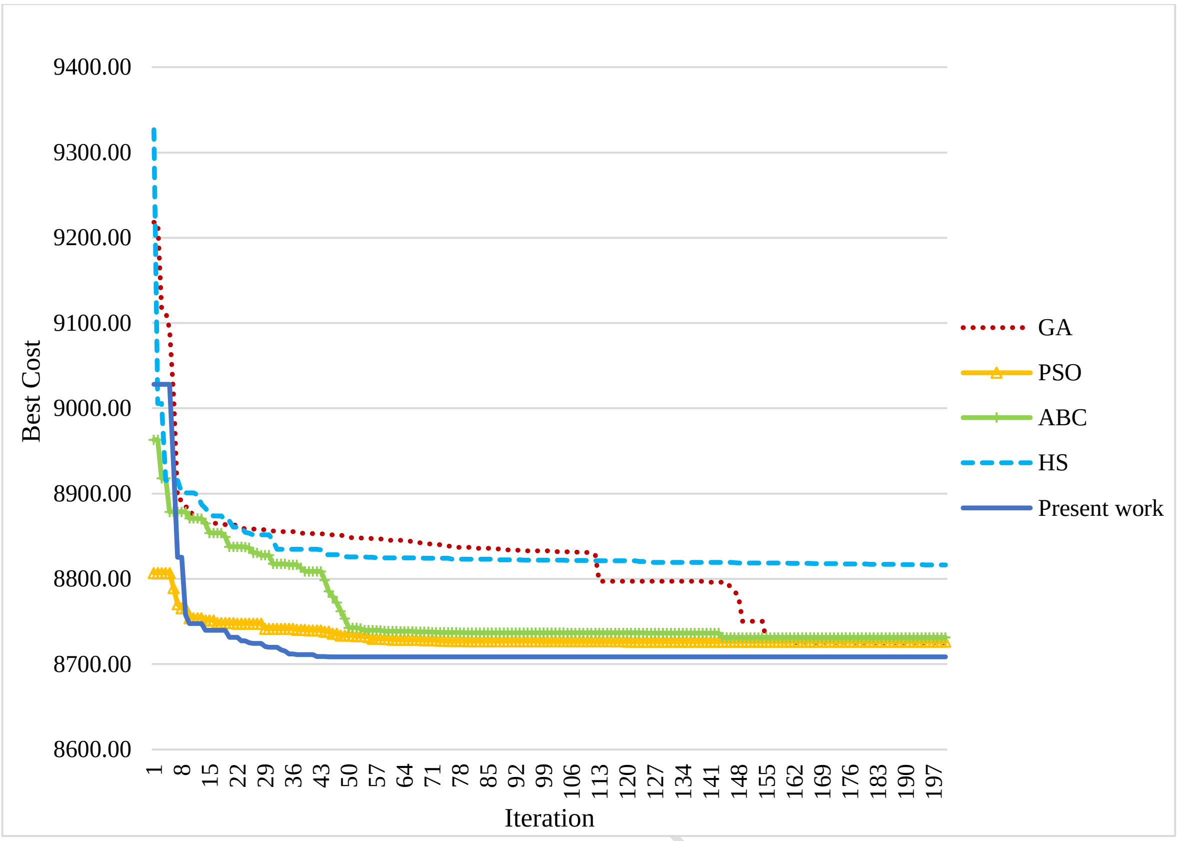


Fig. 11. Convergence curve of the proposed algorithm for the 120-bar dome-shaped truss

Problem 5. A 200-bar planar truss

An optimal design is sought for the 200-bar planar truss illustrated in Fig. 12. The design constraint requires that the stress in each member remain within ±10 ksi. The material properties assigned to all members include a Young's modulus of 3×10^7 psi and a density of 0.273 lb/in³. A minimum cross-sectional area of 0.1 in² was imposed for the members. To simplify the design, the members were grouped into 29 categories, with all members within a group sharing the same cross-sectional area. As shown in Table 11, the truss was subjected to three distinct loading scenarios. The final designs were then compared with those obtained by other optimization methods, as presented in Table 12.

Table 11. Load cases for the planar 200-bar truss

Case	Load (lb)	Direction	Nodes
1	1000	X	1, 6, 15, 20, 29, 34, 43, 48, 57, 62, 71
2	10000	Y	1-6, 8, 10, 12, 14-20, 22, 24, 26, 28-34, 36, 38, 40, 42-48, 50, 52, 54, 56-62, 64, 66, 68, 70-75
3			Load cases 1 and 2 acting simultaneously

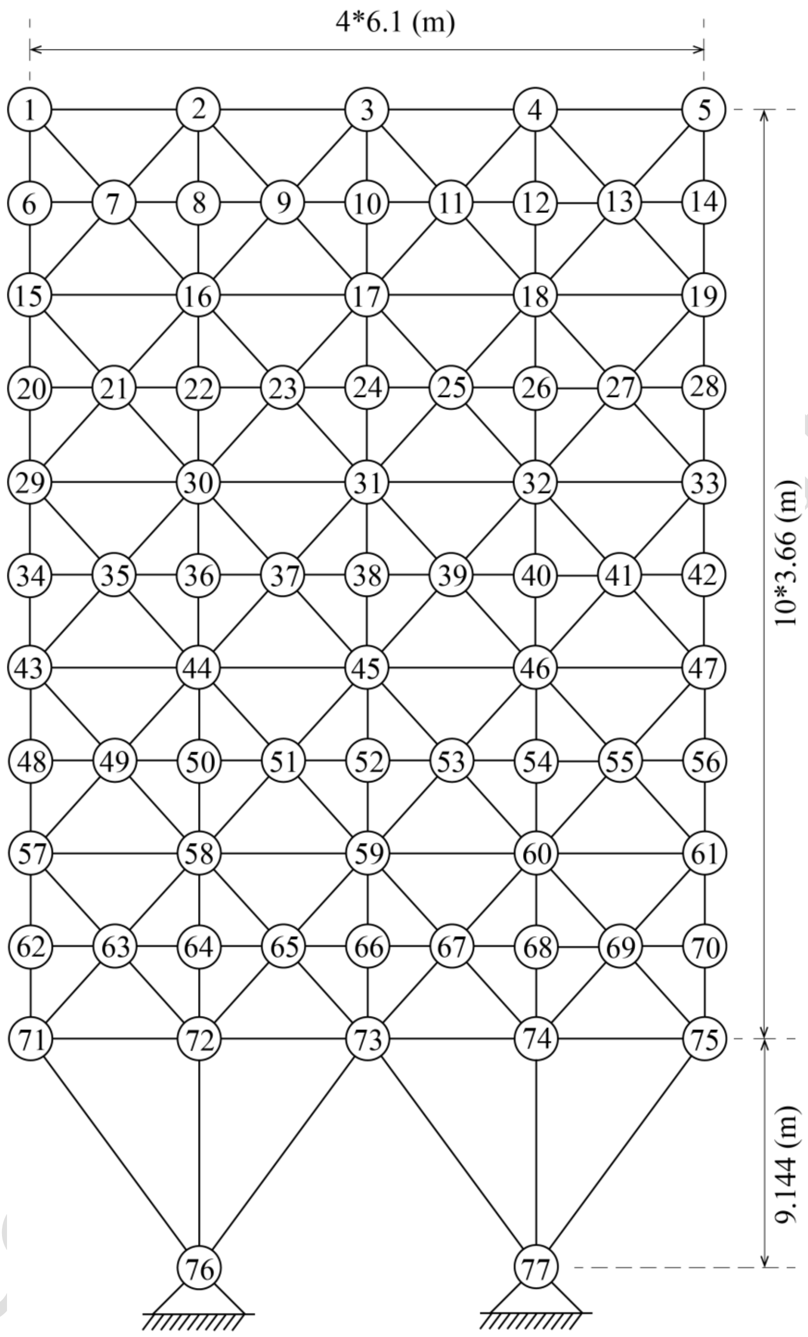


Fig. 12. The 200-bar planar truss

Table 12. Results of the 200-bar planar truss

Variables	Optimal design variable (cm^2)						
	GA	PSO	ABC	HS	SO	AHA	Present work
A_{G1}	0.11	0.14	0.15	0.14	0.11	0.28	0.14
A_{G2}	0.94	0.94	0.94	0.96	1.03	0.82	0.94
A_{G3}	0.13	0.10	0.10	0.10	0.20	0.10	0.10
A_{G4}	0.10	0.10	0.10	0.10	0.10	0.10	0.10
A_{G5}	2.03	1.94	1.94	1.95	1.92	1.82	1.94
A_{G6}	0.31	0.29	0.30	0.29	0.30	0.55	0.29
A_{G7}	0.16	0.10	0.10	0.11	0.10	0.12	0.10

A_{G8}	3.15	3.10	3.10	3.11	3.01	3.12	3.11
A_{G9}	0.10	0.10	0.10	0.10	0.40	0.10	0.10
A_{G10}	4.10	4.10	4.10	4.11	4.01	4.12	4.11
A_{G11}	0.43	0.40	0.40	0.41	0.55	0.35	0.40
A_{G12}	0.11	0.19	0.19	0.18	0.11	0.14	0.11
A_{G13}	5.38	5.42	5.42	5.45	5.42	5.32	5.38
A_{G14}	0.16	0.10	0.10	0.10	0.36	0.17	0.10
A_{G15}	6.41	6.42	6.42	6.45	6.40	6.33	6.38
A_{G16}	0.56	0.57	0.58	0.58	0.70	0.52	0.53
A_{G17}	0.40	0.13	0.15	0.15	0.16	0.14	0.39
A_{G18}	7.97	7.97	7.97	8.01	8.11	7.72	7.94
A_{G19}	0.10	0.10	0.10	0.10	0.10	0.15	0.10
A_{G20}	9.01	8.97	8.97	9.01	9.11	8.68	8.94
A_{G21}	0.86	0.70	0.71	0.73	0.78	0.68	0.83
A_{G22}	0.22	0.42	0.42	0.78	0.27	2.28	0.15
A_{G23}	11.02	10.86	10.89	11.17	11.21	10.88	10.94
A_{G24}	0.13	0.10	0.10	0.14	0.11	0.89	0.10
A_{G25}	12.03	11.86	11.88	12.17	12.13	11.87	11.94
A_{G26}	1.00	1.03	1.04	1.34	0.99	2.62	0.89
A_{G27}	6.57	6.68	6.64	5.48	6.39	3.98	6.84
A_{G28}	10.72	10.81	10.80	10.13	10.57	10.12	10.88
A_{G29}	13.96	13.84	13.87	14.52	14.08	15.60	13.74
Best weight (lb)	25681.3	25459.9	25495.4	25519.4	25835.6	26764.1	25452.3
Average weight (lb)	26613.4	25547.6	25610.2	25543.5	27092.3	30823.5	25495.6
Std. Dev. (lb)	615.80	129.09	168.85	23.21	795.17	2268.35	123.42
CPU Time (s)	498	421	568	563	437	418	403

This problem shows the effectiveness of the proposed algorithm in handling high-dimensional structural design problems. Unlike conventional metaheuristic algorithms, which often struggle with maintaining solution quality as problem size increases, the proposed method consistently converges to the best solution with remarkable stability. The results presented in Table 12 reveal that it consistently outperformed the compared algorithms in terms of achieving the best solution. The adaptive nature of the distance parameter provides strong exploration and exploitation capabilities, allowing the algorithm to converge more quickly and accurately to the optimal solution. The convergence plot in Fig. 13 shows the efficient progression of the proposed algorithm toward the global minimum compared to other methods.

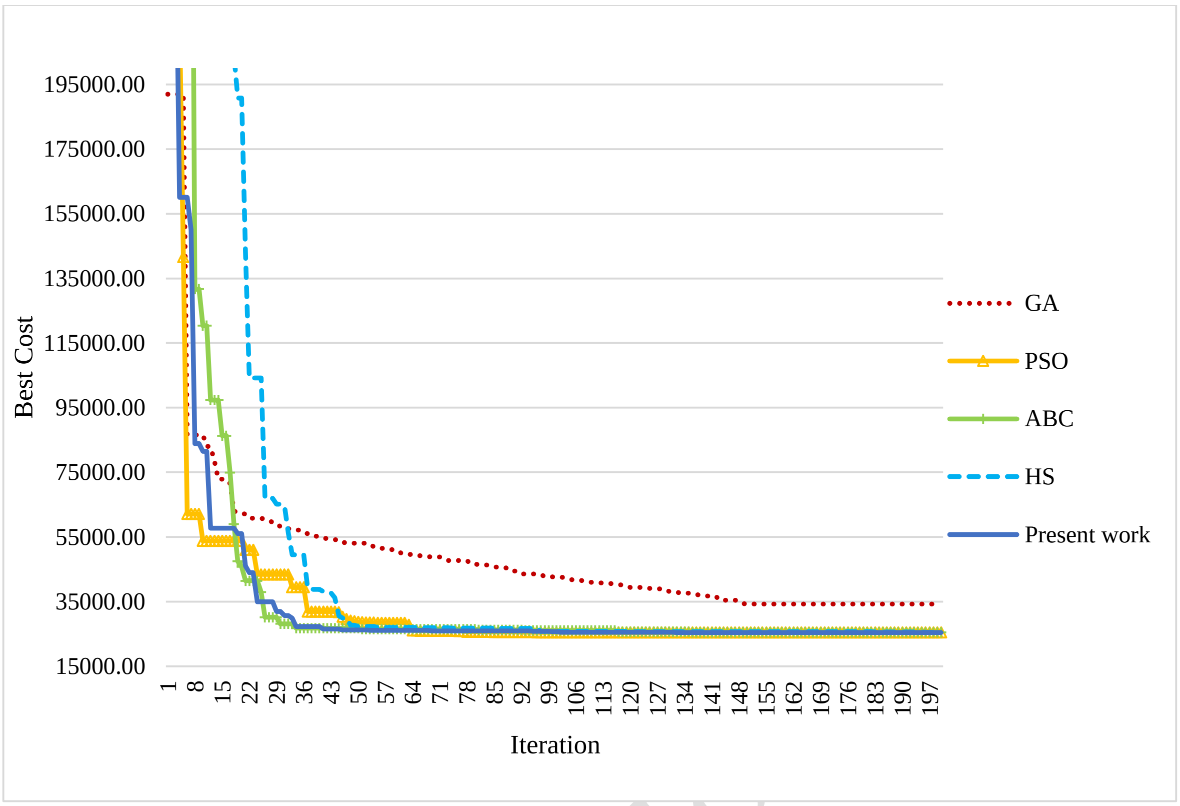


Fig. 13. Convergence curve of the proposed algorithm for the 200-bar planar truss

4. Conclusions

In this paper, a new optimization algorithm is developed to achieve an optimal solution, efficiently. This algorithm operates based on the movement of the specified agents in the search space. These agents explore the search space at a specific distance from the best solutions. Determining this distance plays a fundamental role in the exploration and exploitation abilities of the algorithm. This distance is defined using specific dynamic parameters based on the position of the agents and the best solutions. It is a large distance at first, ensuring good exploration, and gradually decreases during iterations leading to proper exploitation. The performance of the proposed algorithm is assessed in several mathematical and engineering problems and is compared with various optimization algorithms such as GA, PSO, ABC, and HS. Different structures and engineering problems are also investigated to signify the capabilities of these algorithms in structural problems. The results demonstrate the effectiveness and efficiency of the proposed algorithm in converging to the global minimum.

Th fact that the proposed method reaches the solution efficiently in a wide range of examples, is a sign of robustness of the proposed method.

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

Abdel-Basset, M., Abdel-Fatah, L. and Sangaiah, A. K. (2018). "Metaheuristic algorithms: A comprehensive review", *Computational intelligence for multimedia big data on the cloud with engineering applications*, 185–231, https://doi.org/10.1016/B978-0-12-813314-9.00010-4.

Abdollahzadeh, B., Gharehchopogh, F. S. and Mirjalili, S. (2021) "African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems", *Computers & Industrial Engineering*, 107408, https://doi.org/10.1016/j.cie.2021.107408.

Abedinpourshotorban, H., Mariyam Shamsuddin, S., Beheshti, Z. and Jawawi, D. (2016). "Electromagnetic field optimization: a physics-inspired metaheuristic optimization algorithm", Swarm Evol Comput 26, 8–22, https://doi.org/10.1016/j.swevo.2015.07.002.

Abualigah, L., Ababneh, A., Ikotun, A. M., Zitar, R. A., Alsoud, A. R., Khodadadi, N., Ezugwu, A. E., Hanandeh, E. S. and Jia, H. (2024). "A Survey of cuckoo search algorithm: optimizer and new applications", *Metaheuristic optimization algorithms*, Elsevier, 45–57, https://doi.org/10.1016/B978-0-443-13925-3.00018-2.

Akay, B. and Karaboga, D. (2012). "Artificial bee colony algorithm for large-scale problems and engineering design optimization", *Journal of intelligent manufacturing*, 23, 1001–1014, https://doi.org/10.1007/s10845-010-0393-4.

Almufti, S. M. (2019). "Historical survey on metaheuristics algorithms", *International Journal of Scientific World*, 7(1), 1, https://doi.org/10.14419/ijsw.v7i1.29497.

Arora, S. and Singh, S. (2019). "Butterfly optimization algorithm: a novel approach for global optimization", *Soft computing*, 23, 715–734, https://doi.org/10.1007/s00500-018-3102-4.

Askari, Q., Younas, I. and Saeed, M. (2020). "Political Optimizer: A novel socio-inspired meta-heuristic for global optimization", *Knowledge-Based Systems*, 195, 105709, https://doi.org/10.1016/j.knosys.2020.105709.

Ayyarao, T. S., Ramakrishna, N., Elavarasan, R. M., Polumahanthi, N., Rambabu, M., Saini, G., Khan, B. and Alatas, B. (2022). "War strategy optimization algorithm: a new effective metaheuristic algorithm for global optimization", *IEEE Access*, 10, 25073–25105, https://doi.org/10.1109/ACCESS.2022.3153493.

Bodaghi, M. and Samieefar, K. (2019). "Meta-heuristic bus transportation algorithm", *Iran Journal of Computer Science*, 2, 23–32, https://doi.org/10.1007/s42044-018-0025-2.

Coello Coello, C. A. (2005). "An introduction to evolutionary algorithms and their applications", *International Symposium and School on Advancex Distributed Systems*, Springer, 425–442, https://doi.org/10.1007/11533962_39.

Das, B., Mukherjee, V. and Das, D. (2020). "Student psychology based optimization algorithm: A new population based optimization algorithm for solving optimization problems", *Advances in Engineering software*, 146, 102804, https://doi.org/10.1016/j.advengsoft.2020.102804.

Dhiman, G. and Kumar, V. (2019). "Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems", *Knowledge-Based Systems*, 165, 169–196, https://doi.org/10.1016/j.knosys.2018.11.024.

Dorigo, M., Birattari, M. and Stutzle, T. (2007). "Ant colony optimization", *IEEE computational intelligence magazine*, 1(4), 28–39, https://doi.org/10.1109/MCI.2006.329691.

Ezugwu, A. E., Shukla, A. K., Nath, R., Akinyelu, A. A., Agushaka, J. O., Chiroma, H. and Muhuri, P. K. (2021). “Metaheuristics: a comprehensive overview and classification along with bibliometric analysis”, *Artificial Intelligence Review*, 54, 4237–4316, https://doi.org/10.1007/s10462-020-09952-0.

Faramarzi, A., Heidarinejad, M., Stephens, B. and Mirjalili, S. (2020). “Equilibrium optimizer: A novel optimization algorithm”, *Knowledge-Based Systems*, 191, 105190, https://doi.org/10.1016/j.knosys.2019.105190.

Fogel, D. B. (1998). *“Evolutionary computation: the fossil record”*, John Wiley & Sons, https://doi.org/10.1109/9780470544600.

Geem, Z. W., Kim, J. H. and Loganathan, G. V. (2001). “A new heuristic optimization algorithm: harmony search”, *simulation*, 76(2), 60–68, https://doi.org/10.1177/003754970107600201.

Goldberg, D. E. (1992). “Genetic Algorithms in Search, Optimization &amp; Machine Learning, 401pp., Addison-Wesley (1989)”, *Journal of the Japanese Society for Artificial Intelligence*, 7(1), 168–168, https://doi.org/10.11517/jjsai.7.1_168.

Haldurai, L., Madhubala, T. and Rajalakshmi, R. (2016). “A study on genetic algorithm and its applications”, *Int. J. Comput. Sci. Eng*, 4(10), 139–143, https://www.doi.org/10.56726/IRJMETS32980

Hansen, N., Müller, S. D. and Koumoutsakos, P. (2003). “Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)”, *Evolutionary computation*, 11(1), 1–18, https://doi.org/10.1162/106365603321828970.

Hashemi, A., Dowlatshahi, M. B. and Nezamabadi-Pour, H. (2021). “Gravitational search algorithm: Theory, literature review, and applications”, *Handbook of AI-based Metaheuristics*, 119–150, http://doi.org/10.1201/9781003162841-7.

Hashim, F. A., Houssein, E. H., Hussain, K., Mabrouk, M. S. and Al-Atabany, W. (2022). “Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems”, *Mathematics and Computers in Simulation*, 192, 84–110, https://doi.org/10.1016/j.matcom.2021.08.013.

Hashim, F. A., Houssein, E. H., Mabrouk, M. S., Al-Atabany, W. and Mirjalili, S. (2019). “Henry gas solubility optimization: A novel physics-based algorithm”, *Future Generation Computer Systems*, 101, 646–667, https://doi.org/10.1016/j.future.2019.07.015.

Hashim, F. A., Hussain, K., Houssein, E. H., Mabrouk, M. S. and Al-Atabany, W. (2021). “Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems”, *Applied intelligence*, 51, 1531–1551, https://doi.org/10.1007/s10489-020-01893-z.

Hatamlou, A. (2013). “Black hole: A new heuristic optimization approach for data clustering”, *Information Sciences*, 222, 175–184, https://doi.org/10.1016/j.ins.2012.08.023.

Holland, J. H. (1992). *“Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence”*, MIT press, https://doi.org/10.7551/mitpress/1090.001.0001.

Karaboga, D. and Akay, B. (2009). “A comparative study of artificial bee colony algorithm”, *Applied mathematics and computation*, 214(1), 108–132, https://doi.org/10.1016/j.amc.2009.03.090.

Kaveh, A. and Yousefpoor, H. (2022). “Chaotically enhanced meta-heuristic algorithms for optimal design of truss structures with frequency constraints”, *Periodica Polytechnica Civil Engineering*, 66(3), 900–921, https://doi.org/10.3311/PPci.20220.

Kennedy, J. (2006). “Swarm intelligence”, *Handbook of nature-inspired and innovative computing: integrating classical models with emerging technologies*, Springer, pp.187–219. https://doi.org/10.1007/0-387-27705-6_6.

Kennedy, J. and Eberhart, R. (1995) “Particle swarm optimization”, *Proceedings of ICNN'95-international conference on neural networks*, ieee, 1942–1948, http://doi.org/10.1109/ICNN.1995.488968.

Kirkpatrick, S., Gelatt Jr, C. D. and Vecchi, M. P. (1983). “Optimization by simulated annealing”, *science*, 220(4598), 671–680, http://doi.org/10.1126/science.220.4598.671.

Lee, K. S. and Geem, Z. W. (2005). “A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice”, *Computer Methods in Applied Mechanics and Engineering*, 194(36-38), 3902–3933, https://doi.org/10.1016/j.cma.2004.09.007.

Mirjalili, S. and Lewis, A. (2016). “The whale optimization algorithm”, *Advances in Engineering software*, 95 51–67, https://doi.org/10.1016/j.advengsoft.2016.01.008.

Mohd Shahrom, M. A., Zainal, N., Ab Aziz, M. F. and Mostafa, S. A. (2023). “A Review of Glowworm Swarm Optimization Meta-Heuristic Swarm Intelligence and its Fusion in Various Applications”, *Fusion: Practice & Applications*, 13(1), https://doi.org/10.54216/FPA.130107.

Rashedi, E., Nezamabadi-pour, H. and Saryazdi, S. (2009). “GSA: A Gravitational Search Algorithm”, *Information Sciences*, 179(13), 2232–2248, https://doi.org/10.1016/j.ins.2009.03.004.

Rechenberg, I. (1978) “Evolutionsstrategien”, *Simulationsmethoden in der Medizin und Biologie: Workshop, Hannover, 29. Sept.–1. Okt. 1977*, Springer, 83–114, https://doi.org/10.1007/978-3-642-81283-5_8.

Rokh, B., Mirvaziri, H. and Olyaee, M. (2024). “A new evolutionary optimization based on multi-objective firefly algorithm for mining numerical association rules”, *Soft computing*, 28(9), 6879–6892, https://doi.org/10.1007/s00500-023-09558-y.

Roudak, M., Shayanfar, M., Farahani, M., Badiezadeh, S. and Ardalan, R. (2024). “An enhanced genetic algorithm based on the introduction of fixed station groups and a new variable multi-parent crossover technique”, *Iran University of Science & Technology*, 14(2), 189–210, https://doi.org/10.22068/ijoce.2024.14.2.582.

Rutenbar, R. A. (1989). “Simulated annealing algorithms: An overview”, *IEEE Circuits and Devices magazine*, 5(1), 19–26, https://doi.org/10.1109/101.17235.

safarkhani, m. and madhkhan, m. (2025). “Multi-objective optimization of outriggers and belt walls location in high-rise concrete structures using the Genetic-Descent Gradient integrated method”, *Civil Engineering Infrastructures Journal*, https://doi.org/10.22059/ceij.2025.382260.2152.

Shami, T. M., Grace, D., Burr, A. and Mitchell, P. D. (2024). “Single candidate optimizer: a novel optimization algorithm”, *Evolutionary Intelligence*, 17(2), 863–887, https://doi.org/10.1007/s12065-022-00762-7.

Storn, R. and Price, K. (1997). “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces”, *Journal of global optimization*, 11, 341–359, https://doi.org/10.1023/A:1008202821328.

Talatahari, S., Azizi, M., Tolouei, M., Talatahari, B. and Sareh, P. (2021). “Crystal structure algorithm (CryStAl): a metaheuristic optimization method”, *IEEE Access*, 9, 71244–71261, https://doi.org/10.1109/ACCESS.2021.3079161.

Umar, S. U., Rashid, T. A., Ahmed, A. M., Hassan, B. A. and Baker, M. R. (2024). “Modified Bat Algorithm: a newly proposed approach for solving complex and real-world problems”, *Soft computing*, 28(13), 7983–7998, https://doi.org/10.1007/s00500-024-09761-5.

Yang, X.-S. and Deb, S. (2009). “Cuckoo search via Lévy flights,[in:] 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)”, *Coimbatore, India*. 210–214, https://doi.org/10.1109/NABIC.2009.5393690.

Zhang, Y. and Jin, Z. (2020). “Group teaching optimization algorithm: A novel metaheuristic method for solving global optimization problems”, *Expert Systems with Applications*, 148, 113246, https://doi.org/10.1016/j.eswa.2020.113246.

Zhao, W., Wang, L. and Mirjalili, S. (2022). “Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications”, *Computer Methods in Applied Mechanics and Engineering*, 388, 114194, https://doi.org/10.1016/j.cma.2021.114194.