Hybrid teaching–learning-based optimization for solving engineering and mathematical problems

Mohammadhossein Dastan1 · Saeed Shojaee1 · Saleh Hamzehei-Javaran1 · Vahid Goodarzimehr1

Received: 6 March 2022 / Accepted: 26 July 2022 / Published online: 25 August 2022
© The Author(s), under exclusive licence to The Brazilian Society of Mechanical Sciences and Engineering 2022

Abstract
In this work, a new and effective algorithm called hybrid teaching–learning-based optimization (TLBO) and charged system search (CSS) algorithms (HTC) are proposed to solve engineering and mathematical problems. The CSS is inspired by Coulomb and Gauss's electrostatic laws of physics as well as the Newtonian mechanic laws of motion. The TLBO is inspired by the interaction between teacher and student in a classroom. Usually, the TLBO gets trapped in the local optimal due to the lack of a system for measuring the distance between the student and the optimal point. In order to solve this problem, the CSS algorithm, which is based on the electrical physics laws, is utilized. In the CSS algorithm, each factor is stored under the influence of the best local and global positions, and it is used in subsequent iterations as the possible optimal answers. In fact, this leads to a better balance between exploration and exploitation. In order to validate the proposed method, the CEC2021 and CEC2005 mathematical functions are optimized. Additionally, to show the applicability of the proposed algorithm and to evaluate its performance and convergence rate, several benchmark truss structures are optimized. The weight of the structural elements is taken into account as the objective function, which is optimized under displacement and stress constraints. The results of the proposed algorithm are compared with some other well-known meta-heuristic methods. The results show that the hybrid HTC algorithm improved the convergence rate and quickly obtained the optimal and desired design. The hybrid HTC algorithm can be adapted to solve other complex mathematical and optimization problems.

Keywords Meta-heuristic algorithms · Optimization of structures · Charged system search algorithm · Teaching–learning-based optimization · Hybrid charged system search based on TLBO algorithm

1 Introduction
Optimization is an important and decisive process in solving mathematical and structural design problems in civil engineering that have attracted the attention of many researchers nowadays. Researchers and designers can better solve complex problems and come up with better designs with optimization methods, which leads to saving time and money. The goal of optimization is to find the best acceptable solution, given the limitations and needs of the problem. There may be different solutions to a problem, and a function called the objective function is defined to compare them and select the optimal one. Selecting the right objective function is one of the most important steps in the optimization process. Sometimes, several objective functions are considered simultaneously in the optimization process. Such optimization problems, which involve multiple objective functions, are called multi-objective problems.

Optimization problems are divided into two categories. The first category is the unlimited optimization problem, where the goal is to maximize or minimize the objective function without any constraints on the design variables. The second category is constrained optimization problems. Optimization of most of the practical problems, including structural analysis in civil engineering, is performed by considering constraints on the behavior and operation of a system, which are called behavioral constraints, and constraints on physics and geometry, which are called geometric or lateral constraints.

In the last three decades, different methods have been proposed by researchers to solve different types of optimization
problems. In the past, there was no effective optimization method and limited stress and ultimate strain methods, which are considered traditional optimization methods, had certain shortcomings. Generally, optimization methods are divided into two categories. The first category is gradient-based methods that lead to accurate optimization, but in cases with complex problems, they have low speed and cannot calculate the derivative of functions. Given the new needs of human beings and the complexity of optimization problems, researchers need powerful methods to solve them. For this reason, many meta-heuristic methods have been proposed over the past decade that is inspired by nature and the laws of physics and chemistry. These methods do not require basic calculations such as calculating the gradient of functions, so they perform faster in solving optimization problems than conventional algorithms. In heuristic approaches, there are two essential components of exploration and exploitation that are directly related to the searchability of an algorithm. The exploration component leads to more exploration of the search space to find better solutions and improve their diversity. The exploitation component also improves the quality of solutions by increasing the search in local areas around them.

Based on the source of inspiration, the meta-heuristic algorithms can be divided into four categories: evolutionary algorithms (EAs), swarm intelligence (SI), physics / chemistry-based algorithms, and human-based algorithms. The EAs are inspired by Darwin’s evolution theory in nature and mimic biological evolutionary behaviors such as recombination, mutation, and selection. These algorithms often provide good approximate solutions to a variety of problems because, ideally, they make no assumptions about the fundamental fitness perspective. The techniques of evolutionary algorithms used to model biological evolution are generally limited to exploring micro-evolutionary processes and programming models based on cellular processes. In most real applications of EAs, computational complexity is a deterrent. In fact, this computational complexity is due to the assessment of fitness performance. Among the researchers who have proposed methods in this field are Goldberg and Holland [1], who proposed the genetic algorithm (GA). This algorithm generates a new generation by using Darwin’s theory and utilizing mutation mechanisms. Simon [2] proposed the biogeography-based optimization algorithm (BBO) based on the distribution of vital species in different regions. Storn and Price [3] produced a new population by utilizing the mutation process and adding the weight difference of two population vectors to the third one. After being inspired by the coronavirus, which is an unknown animal virus, Salehan and Deldari [4] developed a new algorithm called the Corona Virus optimization algorithm (CVO), which is used to solve a number of benchmark functions as well as discrete and continuous problems. Lee and Tom [5] also introduced virus-spread optimization (VSO), which is inspired by the spread of viruses between individuals.

Swarm intelligence (SI) algorithms are derived from the behaviors of animal groups (food search, mating, migration, etc.). SI is the collective behavior of decentralized and self-organized systems, which can be natural or artificial. This concept has been used to work on artificial intelligence. The term was introduced by Gerardo Benny and Jing Wang in 1989 in the field of cellular robotic systems. The SI systems typically include a population of simple agents that interact locally with each other and with their environment. After being inspired by the collective behavior of fish and birds, Kennedy and Eberhart [6] proposed particle swarm optimization (PSO). Dricoi et al. [7] presented the ant colony optimization algorithm (ACO) after observing and being inspired by the collective behavior of ants to find the closest path to food as a result of the activity of a chemical substance called pheromone. Karaboga and Basturk [8] presented the artificial bee colony algorithm (ABC) using the interaction between worker, guard, and queen bees to find the food source. Chou et al. [9] proposed cat swarm optimization (CSO) after studying the behavior of cats in search, tracing, and finding prey. Mirjalili et al. [10] proposed a meta-heuristic algorithm called the Grasshopper optimization algorithm (GOA), which is inspired by the social behavior of grasshoppers and their influence on their environment. After being inspired by the unique mating behavior of certain species of spiders, Hayolalam and Pourhaji Kazem [11] proposed a new algorithm called the Black Widow optimization algorithm (BWO). The behavior of these spiders is toward the elimination of inappropriate responses which leads to the convergence of the new algorithm.

Researchers interested in nature-inspired calculations have increased significantly by studying various phenomena in nature and the basic principles of physics, chemistry, and biology. The algorithms based on physics and chemistry simulate some physical and chemical laws, such as electric charges, gravity, river systems, and motion. These algorithms provide a comprehensive introduction to methods and algorithms in nature-inspired computations, with an emphasis on applications for real engineering problems.

Kaveh and Talatahari [12] presented the charged system search algorithm (CSS) based on Newton’s laws of mechanics and Coulomb’s laws. Hatamlou [13] introduced the black hole optimization algorithm (BH), which is inspired by the black hole phenomenon in physics. In this method, the best particle is selected as a black hole and the stars that get too close to the black hole will be swallowed by it. Rashedi et al. [14] proposed the gravitational search algorithm (GSA) based on Newton’s laws of gravitation. Crick Patrick et al. [15] presented the simulated annealing (SA) algorithm based on the laws of physics. The Big Bang–Big Crunch algorithm (BB-BC) was presented by Erol and Eksin [16]. This
algorithm was inspired by the Big Bang and Big Crunch theories. Chemical reaction optimization (CRO) was introduced by Lam and Li [17], which is inspired by a natural process of transforming unstable substances into stable ones. Also, Kaveh and Khayatzadeh [18] and Kaveh and Ghazaan [19] proposed two algorithms of Ray optimization (RO) and vibrating particle system (VPS), respectively. The RO algorithm is proposed based on the refraction and change of direction of light rays when entering and leaving different environments and the vibrating particle system algorithm is inspired by vibrations of a single degree of freedom systems.

Finally, human-based algorithms, as the last category of meta-heuristic algorithms, are inspired by human behaviors and characteristics. For example, the teaching–learning-based optimization algorithm (TLBO) [20] was proposed by Rao et al. and is inspired by the relationship between a teacher and his students. Zhang and Jin presented the group teaching optimization algorithm (GTOA) [21] after being inspired by studying human behavior in group training. Also, the human behavior-based optimization (HBBO) [22], which was inspired by human behaviors to succeed in their major field, was presented by Ahmadi. Figure 1 shows the classification of algorithms based on the source of inspiration and their subsets in the form of a flowchart.

One optimization method is to use more than one algorithm in a hybrid process. This operation is a relatively good way to solve difficult problems. In this method, by examining two or more meta-heuristic algorithms and identifying their strengths and weaknesses, a new hybrid algorithm is proposed. The common method is to use the global search algorithm to find an almost optimal solution and then use the results as a starting point for a local optimizer. The combination of optimization algorithms is an approach toward balancing the ability to explore and exploit. For example, the balance between exploration and exploitation is one of the problems of the PSO algorithm. Over the past decade, many hybrid algorithms have been proposed and developed by researchers. Maheri et al. [23] developed the hybrid genetic and particle swarm optimization (GA-PSO) to optimize the size and geometry of structures. Talatahari et al. [24] proposed the hybrid teaching–learning-based optimization and harmony search algorithm (TLBO-HS) for optimizing large-scale structures. Also, Talatahari et al. [25] used the symbiotic organisms search and harmony search algorithms for discrete optimization of structures. Omidinsab and Goodarzimehr [26] proposed the hybrid particle swarm optimization and genetic (PSO-GA) algorithm for the optimal design of truss structures with discrete variables. Kaveh and Talatahari [27] presented the hybrid charged system search and

![Flowchart of the meta-heuristic algorithms categorization based on the source of inspiration](image_url)
particle swarm optimization (CSS-PSO) algorithm for the optimal design of structures.

In recent years, researchers have used many meta-heuristic algorithms to optimal design structures. Awad [28] proposed the political optimizer (PO) algorithm, which is inspired by the multi-phased political process in parliamentary democracies. Attar and Carbas [29] used teaching–learning-based optimization (TLBO) and biogeography-based optimization (BBO) algorithms to examine the optimum discrete sizing design of steel truss steel bridges for minimizing the structural weights. Talatahari and Azizi [30] proposed the tribe-interior search algorithm for optimum design of building structures, in which the search phase of the algorithm is divided into three phases. Jafari et al. [31] proposed an efficient hybrid algorithm based on the particle swarm optimizer (PSO) and the cultural algorithm (CA) for the optimal design of truss structures. In this method, the cultural space defined by the CA has been used to improve the PSO method. Baykasoglu and Baykasoglu [32] proposed a weighted superposition attraction–repulsion (WSAR) algorithm for truss optimization with multiple frequency constraints. This algorithm is a recent swarm intelligence-based metaheuristic algorithm. Jawad et al. [33] proposed a heuristic dragonfly algorithm for the optimal design of truss structures with discrete variables, which is inspired by the DA that emerges from behaviors of static and dynamic swarming of dragonflies in nature. Also, Jawad et al. [34] used a swarm intelligence-based optimization technique called artificial bee colony algorithm (ABC) for sizing and layout optimization of truss structures.

The purpose of this paper is to present an efficient algorithm that improves the weaknesses of the TLBO and CSS algorithms and has a higher convergence rate in the same amount of time. The TLBO algorithm is trapped in the local optimization due to the lack of a system for measuring the distance between the student and the optimal point. In order to solve this problem, the CSS algorithm, which is based on the laws governing electrical physics, is used. In the CSS algorithm, each factor is stored under the influence of the best local and global positions and is used in subsequent iterations as the response to optimal optimization. This fact has led to an increase in the utilization rates as well as more exploration of the exploration space. As a result, the convergence rate of the proposed algorithm is higher than other innovative algorithms. Now, considering the mentioned facts, the question is whether the performance of this algorithm is appropriate or not? Answering this question requires scientific and research work. Therefore, in this paper, some benchmark functions and practical examples such as truss structures are solved by the proposed algorithm, and its results are compared with the outcomes of some other innovative algorithms.

The outline of this article is as follows: The common CSS and TLBO algorithms are summarized in Sect. 2, and the basic principles and ideas of the proposed method are reviewed in this section. In Sect. 3, the optimization of two types of benchmark performance and the design of truss structures are discussed. The efficiency and performance of the proposed method are assessed and confirmed through the numerical examples given in Sect. 4, and finally, the conclusions of this method are summarized in Sect. 5.

2 Introduction to CSS and TLBO algorithms

The proposed algorithm in this research is the result of the hybridization of the CSS and TLBO algorithms. Among the unique features of this algorithm, the high convergence rate and the ability to balance between exploration and exploitation can be pointed out. Each of these algorithms, which is briefly described below, has advantages and disadvantages that can be mitigated by combining them.

2.1 The charged system search (CSS) algorithm

The charged system search (CSS) algorithm [12] is an efficient, population-based meta-heuristic technique used to optimize a variety of problems and structures. The CSS algorithm uses Coulomb’s laws of electrostatics and Newton’s laws of mechanics. In this algorithm, each factor is a charged particle with a predetermined radius. The amount of charge on the particles is considered based on their quality. Each particle creates an electric field, which exerts a charge on other electrically charged objects. The quantity of the resulting force is determined using the laws of electrostatics and the quality of motion is determined using the Newton laws of mechanics. In order to introduce the CSS algorithm, some rules are described given below.

Rule (1): Similar to other natural evolution algorithms, the CSS algorithm considers a number of charged particles (CP) as the initial population. Each CP has a charge amount \( q_i \), and thus, creates an electric field around its space. The amount of charge (force) according to the quality of its solution procedure is defined as follows:

\[
q_i = \frac{\text{fit}(i) - \text{fitworst}}{\text{fitbest} - \text{fitworst}}, \quad i = 1, 2, ..., N
\]

(1)

where fitbest and fitworst are the best and worst proportions of \( X_i \), respectively.

where fitbest and fitworst are the best and worst proportions of between all particles. \( \text{fit}(i) \) represents the value of the objective function, and \( N \) is the number of total CPs. Moreover, the separation distance, denoted by \( r_{ij} \), between two charged particles is defined as follows:

\[
r_{ij} = \frac{|X_i - X_j|}{\left(\frac{(X_i - X_j)}{2} - X_{best}\right)^2 + \varepsilon}
\]

(2)

in which, \( X_i \) and \( X_j \) represent the position of \( i \)-th and \( j \)-th CPs, respectively. \( X_{best} \) denotes the best position of the current CP and \( \varepsilon \) is a positive number.
Rule (2): The initial positions of CPs are randomly determined in the search space:

\[
x_{i,j}^{(0)} = x_{i,min} + rand \cdot (x_{i,max} - x_{i,min}), \quad i = 1, 2, \ldots, n,
\]

in which, \(x_{i,j}^{(0)}\) specifies the initial variable \(i\) for the \(j\)-th CP. \(x_{i,min}\) and \(x_{i,max}\) are the minimum and maximum allowable values for the variable \(i\), respectively. \(rand\) is a random number in the range \([0,1]\) and \(n\) is the number of variables. Moreover, the initial velocity of the charged particles is zero.

\[
v_{i,j}^{(0)} = 0, \quad i = 1, 2, \ldots, n,
\]

Rule (3): Three conditions can be considered for different types of gravitational forces:

- Each CP can affect another one. For example, a bad CP can affect a good CP and vice versa (\(p_{ij}=1\)).
- If the amount of electric charge of a CP is better than the other, it can absorb the other. In other words, a good CP absorbs a bad CP:

\[
p_{ij} = \begin{cases} 1 & \text{if} \{ j \} > \text{fit}(i) \\ 0 & \text{else}, \end{cases}
\]

- All good CPs can absorb bad CPs and only some bad agents absorb good agents according to the following probability function:

\[
p_{ij} = \begin{cases} 1 & \text{if} \{ j \} > \text{fit}(i) > \text{fit}(i), \\ 0 & \text{else}, \end{cases}
\]

Rule (4): The resultant value of the electric forces on each CP is calculated using Eq. (7):

\[
F_j = q_i \sum_{i\neq j} \left( \frac{q_i}{4\pi\epsilon_0} r_{ij} \cdot i_1 + \frac{q_i}{4\pi\epsilon_0} r_{ij} \cdot i_2 \right) p_{ij} (X_i - X_j),
\]

\[
\begin{align*}
& j = 1, 2, \ldots, N, \\
& i_1 = 1, i_2 = 0 \iff r_{ij} < a, \\
& i_1 = 0, i_2 = 1 \iff r_{ij} \geq a,
\end{align*}
\]

since each particle is considered a sphere, the parameter \(a\) represents the radius of this sphere, which is calculated from Eq. (8):

\[
a = 0.1 \times \max \left\{ \left\{ x_i, \max - x_i, \min \mid i = 1, 2, \ldots, n \right\} \right\}
\]

Rule (5): The new position and velocity of each CP particle is calculated using Eqs. (9, 10):

\[
X_{j,new} = X_{j,old} + V_{j,new} \cdot \Delta t
\]

\[
V_{j,new} = \frac{X_{j,new} - X_{j,old}}{\Delta t}
\]

where \(k_a\) denotes the acceleration coefficient and \(k_v\) is the velocity coefficient for controlling the previous velocity, which are calculated using Eqs. (11, 12). \(rand_1\) and \(rand_2\) are two random numbers that are evenly distributed in the range \((0,1)\). Moreover, \(m_j\) is the mass of the \(j\)-th CP, which is equal to \(q_j\) and \(\Delta t\) represents the time step.

\[
k_a = 0.5 \left( 1 - \text{iter}/\text{iter}_\text{max} \right)
\]

\[
k_v = 0.5 \left( 1 + \text{iter}/\text{iter}_\text{max} \right)
\]

where \(\text{iter}\) denotes the actual number of iterations and \(\text{iter}_\text{max}\) is the maximum number of iterations. The steps of the CSS algorithm are shown in the form of a flowchart in Fig. 2.

### 2.2 The teaching–learning-based optimization (TLBO) algorithm

The TLBO algorithm [20] is inspired by the teacher’s influence on the performance of students in a class, and the output is considered in terms of results or grades. The teacher is generally considered to be a highly educated person who shares his knowledge with the students. The quality of a teacher affects the outcome for students. A good teacher teaches students in a way that they can get better grades. The TLBO process is divided into two parts: the first part includes the “teacher phase” and the second part includes the “learner phase.” The “teacher phase” means learning from the teacher, and the “learner phase” means learning through interaction between learners. In general, in this method, the teacher tries to spread the knowledge among the students, which in turn increases the level of knowledge of the whole class and helps the students to get good grades. Therefore, a teacher can increase the average grades of the class depending on his ability in the class. In optimization algorithms, the population is a different design variable. Like other nature-inspired algorithms, the TLBO algorithm is a population-based approach that uses a population of solutions to advance a global solution. According to this algorithm, the population is considered as a group of students or a class of students. Figure 3 shows the diagram of the TLBO algorithm.
Fig. 2 Flowchart of the charged search system (CSS) algorithm

1. Initialize the problem and define the algorithm parameters \((nCP, \text{cmcr}, \text{par}, \text{bw}, \text{and maxiter})\), randomly initialize charged particles (CP) with velocities equal to zero \((V)\), and evaluate them.

2. Memorize \(nCP/4\) of the best particles.

3. Determine the charge of magnitude \((q)\) vector, the separation distance \((r)\), and the resultant electrical force \((f)\) matrices using Eqs. (1) to (7).

4. Determine the new positions of the charged particles and update the velocity vector.

5. Regenerate the charged particles exited from the search space using the harmony search based handling approach.

6. Evaluate the new charged particles and replace the old ones with them and corresponding velocities.

7. Update the memory.

8. Check if \(iter \leq \text{maxiter}\) ?

   - If \(YES\), report the best memorized charged particle.
   - If \(NO\), repeat steps 2 to 8.

End
2.2.1 Teacher phase

In this phase, the best member of the community is selected as the teacher or instructor and directs the average population toward himself. This is similar to what a real-world teacher does. This step is formulated as follows:

\[
X_{\text{new}}^k = X_{\text{old}}^k + r(X_{\text{teacher}} - T_F \times M(j))
\]

(13)

\[
M(j) = \frac{\sum_{K=1}^{N} X^K(j)}{F^k} \quad \sum_{K=1}^{N} \frac{1}{F^K}
\]

(14)

where \(X^k(j)\) represents the \(j\)-th design variable, \(T_F\) is used as the training factor, \(r\) is a random number in the range \([0,1]\), \(M(j)\) indicates the average of the class, and \(F^k\) denotes the penalty fitness function.

2.2.2 Learning phase

In this phase, the people in the population (who are classmates) develop their knowledge by working together. This is similar to what really happens to friends and classmates. This step is formulated as follows:

Students \(p\) and \(q\) are randomly selected from the class so that they are unequal, thus:
where \( r \) is a random number in the range \([0,1]\), and \( X_p^j \) represents the \( j \)-th design for the \( p \)-th design vector.

### 2.3 Hybrid TLBO_CSS algorithm (HTC)

Nowadays, due to the advancement and complexity of optimization problems, researchers are trying to use more powerful algorithms to solve them. Therefore, the use of hybrid algorithms has attracted a lot of attention. The purpose of combining two or more algorithms is to improve the performance of each algorithm and reduce their disadvantages in a hybrid process. In this research, we intend to present a new and powerful algorithm for solving optimization problems by combining two novel algorithms, namely the TLBO and CSS algorithms that have been proposed in recent years. Both of these algorithms are population-based. One of the disadvantages of the TLBO algorithm is that it gets stuck in the local optimized point due to the lack of a system for measuring the distance between the student and the optimal point. Disadvantages of the CSS include high dependence on the number of iterations, large number of initial parameters and a large number of functions and complexity of physical and mathematical relationships, while one of the most important features of the TLBO algorithm is its independence on parameters because this algorithm has the least possible number of parameters, which can be considered as a special advantage of this algorithm. The purpose of combining these two algorithms is to propose a new algorithm that has a high convergence rate and has good control over exploration and operation compared to the CSS and TLBO algorithms. The main body of the HTC algorithm is composed of TLBO algorithm and two CSS operators, namely Newton and Coulomb, which are used to eliminate the weak points of this algorithm and upgrade the particles. This strengthens the operating parameter and makes the algorithm search better at local points. The flowchart of the HTC algorithm is shown in Fig. 4. The optimization steps of the HTC algorithm are as follows:

Step (1): Generation of the initial population and selection of the best particle or solution.
Step (2): Calculation of the average of each design variable.
Step (3): Selection of the best solution as the teacher and upgrading the particles in the teaching phase according to Eq. (13).
Step (4): Controlling the boundary conditions according to the upper and lower limits specified in the initial parameters of the algorithm.
Step (5): Assessment of the solution and comparison between the new solution and the older ones for choosing the best solution.

The learner phase:

Step (6): Selection of two solutions (students) randomly and production of new solutions according to Eqs. (15, 16).
Step (7): Assessment of the solution and comparison between the new solution and the older ones for choosing the best solution.

The CSS operator:

Step (8): Calculation of \( f, q, \) and \( r \) using the Coulomb’s rule.
Step (9): Calculation of new \( cp \) and new \( v \) using the Newton’s rule.
Step (10): Assessment of the solution and comparison between the new solution and the older ones for choosing the best solution.
Step (11): Check to see whether the termination condition can be applied or not.
Step (12): Reporting the best solution.

### 3 Problem definition

The examples discussed here are divided into two types which are benchmark functions (CEC2021, 2005) and the optimal design of truss structures. These problems are selected to show the reliability and efficiency of the proposed method.

An optimization problem without constraints can be formulated as follows:

\[
\text{Minimize } f(X) \quad X = \{x_1, x_2, \ldots, x_j, \ldots, x_n\} \in \mathbb{R}^d
\]  

where \( f(X) \) represents the objective function and \( n \) is the number of variables. The set of specified values are denoted by \( \mathbb{R}^d \) and the design variables \( X \) can only take values from this set.
3.1 The CEC2005 Benchmark functions

The CEC2005 benchmark functions are among the first benchmark functions proposed by other researchers to test the performance of algorithms. Therefore, to evaluate the performance of the proposed algorithm (HTC) in a regular way, the unimodal and multi-modal CEC2005 benchmark mathematical functions are used. These functions play an important role in the development of proposed search algorithms as well as evaluation of algorithmic ideas. The CEC2005 benchmark functions include five unimodal functions and 20 multi-modal functions. The multi-medal

---

Fig. 4 Flowchart of the HTC algorithm
Table 1 Comparison between the results of the HTC algorithm and the other ones when solving CEC2005 benchmark functions (D = 10)

|       | HTC  | TLBO | CSS  | PSO  | HTC  | TLBO | CSS  | PSO  |
|-------|------|------|------|------|------|------|------|------|
|       | F1   |      |      |      | F2   |      |      |      |
| Average | 0.00 | 0.00 | 0.89 | 0.00 | F2   |      |      |      |
| Median | 0.00 | 0.00 | 0.06 | 0.00 | Median | 0.00 | 0.00 | 27.39 |
| Best   | 0.00 | 0.00 | 0.01 | 0.00 | Best   | 0.00 | 0.00 | 2.38  |
| Average iteration | 263.00 | 388.10 | 481.00 | 263.00 | Average iteration | 612.90 | 835.80 | 474.00 |
| Std    | 0.00 | 1.5E−28 | 2.60E+00 | 2.56E−28 | Std | 1.36E−28 | 6.19E−28 | 50.37 |
| F3     | 66,581.63 | 129,722.12 | 586,039.80 | 290,690.13 | F4   | 9.59E−29 | 2.47E−28 | 101.49 |
| Median | 48,529.04 | 127,590.73 | 427,176.15 | 226,526.35 | Median | 1.10E−28 | 1.84E−28 | 63.90 |
| Best   | 9040.36 | 25,976.33 | 40,834.13 | 138,489.60 | Best   | 0.00 | 0.00 | 19.37 |
| Average iteration | 821.90 | 779.80 | 512.90 | 986.00 | Average iteration | 846.05 | 978.50 | 397.30 |
| Std    | 54,069.38 | 8.08E+04 | 6.95E+05 | 1.70E+05 | Std | 7.43E−29 | 2.05E−28 | 102.50 |
| F6     | 1.57 | 1.21 | 975.92 | 4.31 | F8   | Average | 20.30 | 20.41 | 20.48 | 20.42 |
| Average | 0.00 | 0.01 | 190.51 | 4.70 | Median | 0.00 | 0.01 | 20.41 | 20.48 |
| Best   | 5.49E−05 | 1.99E−04 | 7.55 | 2.19 | Best   | 5.49E−05 | 1.99E−04 | 20.29 | 20.34 |
| Average iteration | 993.05 | 982.40 | 512.90 | 986.00 | Average iteration | 502.50 | 540.90 | 356.80 |
| Std    | 1.99 | 1.92 | 1572.22 | 1.27 | Std | 5.49E−05 | 1.99E−04 | 20.29 | 20.34 |
| F9     | 5.642 | 6.6452 | 4.7646 | 6.4762 | F10  | Average | 6.43 | 10.75 | 7.44 | 26.31 |
| Median | 4.9748 | 6.4672 | 5.0546 | 6.4672 | Median | 7.12 | 10.29 | 6.68 | 25.87 |
| Best   | 1.9899 | 2.9849 | 2.0225 | 2.985  | Best | 1.99 | 4.97 | 3.98 | 4.97 |
| Average iteration | 860.55 | 795.4 | 446.9 | 425  | Average iteration | 994.75 | 993.50 | 435.50 | 640.60 |
| Std    | 2.3298 | 2.6633 | 2.0799 | 2.3097 | Std | 1.92 | 4.77 | 3.08 | 15.51 |
| F11    | 3.1405 | 4.6616 | 5.6285 | 5.7893 | F14  | Average | 2.90 | 3.06 | 3.55 | 3.44 |
| Median | 3.1418 | 4.3565 | 5.0923 | 6.0235 | Median | 2.88 | 3.10 | 3.52 | 3.48 |
| Best   | 0.17691 | 2.6416 | 2.727 | 4.2156 | Best | 2.52 | 2.65 | 2.96 | 2.61 |
| Average iteration | 957.6 | 942.7 | 263.4 | 928  | Average iteration | 764.20 | 784.50 | 266.40 | 511.50 |
| Std    | 1.2541 | 1.3662 | 2.0625 | 0.86  | Std | 0.24 | 0.29 | 0.34 | 0.50 |
| F15    | 171.778 | 284.3507 | 437.3551 | 413.0044 | F18  | Average | 701.53 | 764.47 | 957.45 | 841.55 |
| Median | 90.7625 | 247.4351 | 414.0578 | 432.7043 | Median | 800.00 | 800.16 | 993.17 | 950.76 |
| Best   | 0.00 | 84.0378 | 88.4934 | 112.8063 | Best | 300.00 | 472.64 | 702.19 | 429.92 |
| Average iteration | 871.55 | 889.2 | 426.8 | 868.4  | Average iteration | 879.95 | 965.60 | 501.80 | 956.40 |
| Std    | 161.111 | 141.2261 | 231.714 | 149.1189 | Std | 244.98 | 181.09 | 114.51 | 228.12 |
| F19    | 621.703 | 784.6694 | 942.6356 | 942.4802 | F20  | Average | 632.04 | 732.13 | 944.68 | 980.84 |
| Median | 800 | 800.0033 | 1002.302 | 975.6657 | Median | 800.00 | 800.05 | 999.16 | 996.38 |
| Best   | 300 | 300 | 706.2557 | 990.6  | Best | 300.00 | 426.17 | 712.98 | 800.00 |
| Average iteration | 822.25 | 913.9 | 379.1 | 990.6  | Average iteration | 916.95 | 979.80 | 352.40 | 957.70 |
| Std    | 254.315 | 246.4278 | 127.9968 | 111.7707 | Std | 239.20 | 152.69 | 114.03 | 74.11 |
functions themselves are divided into seven basic functions, two expanded functions and hybrid functions. The types of these functions are presented in Table 15, Appendix A.1, and the search space diagrams of some of these functions are shown in Figs. 35, 36, 37, 38, 39, 40, 41 and 42, Appendix A.2.

### 3.2 The CEC2021 benchmark functions

Nowadays, due to the development of different types of meta-heuristic algorithms in solving engineering problems, the need to use new and efficient methods to evaluate and compare these algorithms with each other led researchers to use benchmark functions called CEC. The latest version of CEC functions was released in 2021, in which a number of specific real-time optimization problems are examined under the presence or absence of shift, rotation, and bias operators, the main purpose of which is to examine all possible cases of the mentioned operators in all benchmark functions. A summarized list of all CEC2021 functions is given in Table 16, Appendix B.1, and the 3D diagram and search space contours for each of these functions are shown in Figs. 43, 44, 45, 46, 47, 48, 49 and 50, Appendix B.2.

### 3.3 Optimization of truss structures

A structural optimization problem can be formulated as follows:

Minimize \[ f(X) \]

Subject to \[ g_i(X) \leq 0 \quad i = 1, 2, \ldots, m \]

\[ X = \{x_1, x_2, \ldots, x_j, \ldots, x_n\} \in \mathbb{R}^d \]

in which \( g(X) \) is the behavioral constraint and \( m \) is the number of constraints. In size optimization problems, usually the main goal is to minimize the weight of truss structures under design constraints. The design variables are selected as cross sections of elements, which are usually chosen from discrete sets. Therefore, the optimization problem can be set as follows:

Minimize \[ f(X) = W(A_i) = \sum_{j=1}^{Ne} \rho_j A_j L_i \]

Subject to \[ g_{Si} = \frac{\sigma(A_j)}{\sigma_{all}} - 1 \leq 0 \quad i = 1, 2, \ldots, N_e \]

\[ g_{Di} = \frac{\Delta(A_i)}{\Delta_{all}} - 1 \leq 0 \quad j = 1, 2, \ldots, N_n \]

\[ A_i \in A = \{A_{e1}, A_{e2}, \ldots, A_{ep}\} \]

where \( W \) denotes the weight of the structure. \( A_i, \rho_i, L_i \) represent the cross section, density of material and length of the \( i \)-th member, respectively, and \( \sigma_i \) and \( \sigma_{all} \) are the stress of the \( i \)-th member and the allowable axial stress, respectively.
Also, $\Delta_j$ and $\Delta_{all}$ denote the nodal displacements of the \( i \)-th member and the allowable displacement, respectively. Moreover, $N_e$ and $N_n$ are the number of members and nodes in the structure, respectively, and $A_e$ represents the available profile list. A number of constraint control techniques have been proposed to solve constrained optimization problems.

Table 2  Comparison between the results of the HTC algorithm and the other ones when solving CEC2005 benchmark functions ($D = 30$)

|   | HTC   | TLBO  | CSS   | PSO   | HTC   | TLBO  | CSS   | PSO   |
|---|-------|-------|-------|-------|-------|-------|-------|-------|
| F2 Average | 0.01026 | 0.038834 | 6461.296 | 4063.426 | F6 Average | 125.7671 | 97.5034 | 194.048.4 | 1219.351 |
| Median | 0.00731 | 0.010426 | 6852.951 | 3854.883 | Median | 75.0689 | 80.4414 | 134.687.9 | 166.6365 |
| Best | 0.00117 | 0.004286 | 4021.257 | 1909.98 | Best | 1.3628 | 13.7128 | 72,810.28 | 10.8807 |
| Average iteration | 999.5 | 999.8 | 562.3 | 982.1 | Average iteration | 999.3 | 1000 | 564.2 | 1000 |
| Std | 0.01134 | 0.052623 | 1805.261 | 2021.205 | Std | 149.6651 | 67.1779 | 128,404.1 | 2901.793 |
| F1 Average | 13.0348 | 13.1503 | 13.4433 | 13.2713 | F24 Average | 383.7042 | 297.8036 | 964.8459 | 277.9445 |
| Median | 13.1743 | 13.1686 | 13.4486 | 13.2019 | Median | 200 | 200 | 965.2395 | 200 |
| Best | 12.1333 | 12.6381 | 13.066 | 12.9122 | Best | 200 | 200 | 952.7911 | 200 |
| Average iteration | 686.5 | 711.5 | 307.4 | 456.4 | Average iteration | 722.6 | 817.6 | 686.5 | 722 |
| Std | 0.37824 | 0.25082 | 0.2271 | 0.23973 | Std | 394.3861 | 244.3578 | 5.9753 | 246.4821 |

Fig. 5  Convergence history diagram of function 1 with 10 variables

Fig. 6  Convergence history diagram of function 2 with 10 variables

Fig. 7  Convergence history diagram of function 3 with 10 variables

Fig. 8  Convergence history diagram of function 4 with 10 variables
In this study, the penalty function is used to deal with the constrained search space:

\[ \tilde{f}(X) = \begin{cases} f(X) & \text{if } X \in \mathbb{R}^d \\ f(X) + \sum_i \max(g_i(X), 0) & \text{otherwise} \end{cases} \]  

(20)

where, \( f_i(X) \) is the modified function. Moreover, \( \mathbb{R}^d \) represents the practical search space.

4 Numerical examples

In this study, the benchmark functions CEC_2005 and CEC_2005 are solved to evaluate the performance of the HTC algorithm. Moreover, four trusses with 72, 120, 244 and 942 members are investigated individually to evaluate the performance of this algorithm in optimizing the weight of structures. All calculations are performed using MATLAB software in Microsoft Windows 10 with a personal computer with 8 GB of RAM. Furthermore, the stiffness method is used to analyze the structures. Finally, the results of the new method are compared with other studies.

4.1 The CEC2005 benchmark functions

In this study, the CEC2005 benchmark functions are used to evaluate the performance of the HTC algorithm. Each of these functions is examined with ten variables and 40 particles under 1000 analyses as 20 independent runs. Moreover, to challenge the proposed algorithm and evaluate its performance when a larger number of variables is applied, some functions with 30 variables and 60 particles under 1000 analyses as ten independent runs are examined. The
results show that the HTC algorithm is capable of responding to more than 76% of CEC2005 functions. Furthermore, the proposed algorithm is compared with the PSO, TLBO and CSS algorithms, and the results of a number of functions in this comparison are presented in Tables 1 and 2.

In the first group of benchmark functions (Unimodal), the proposed algorithm is able to respond to 80% of these functions. In functions 1, 2, 3, 4 and 6, the best and median values of the HTC algorithm are better than the ones of the other algorithms. Also, the average value of this algorithm in functions 1, 2, 3 and 4 is better than the PSO, TLBO and CSS algorithms. The standard deviation (Std) values in functions 1, 2, 3, and 4 show that the responses of the HTC algorithm are less scattered than the responses of the PSO, TLBO and CSS algorithms.

The investigation of the HTC algorithm’s performance in solving the second group of benchmark functions (Best modal) shows that this algorithm has a powerful performance in responding to more than 71% of functions and has a better minimum (best) value than the PSO, TLBO, and CSS algorithms. Moreover, the median value in functions 6, 8, 9, and 11 and the average value in functions 8, 10, and 11 are better than the ones of the other algorithms.

In the third and fourth groups (expanded and hybrid composition modals) of the benchmark functions, the HTC algorithm is able to solve, respectively, more than 50 and 64% of the functions and it has a more suitable and powerful performance compared to the PSO, TLBO, and CSS algorithms. In function 14, which belongs to the third group of functions (Expanded modal), the proposed algorithm has
better results than the other algorithms in minimum, median and average values. Moreover, in functions 15, 18, 19, 20, 21, 23, and 24, which are all in the fourth group of functions (Hybrid composition modal), the proposed algorithm has better results than the other algorithms in minimum, median and average values as well. In general, the results show that the proposed algorithm has a powerful performance in solving mathematical functions, and the ability of this algorithm is not limited to solving practical and structural problems.

In the following, the diagrams of the convergence history of the CEC2005 functions are shown in Figs. 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 and 21. These diagrams are drawn on a semi-logarithmic scale using MATLAB software.

Fig. 17 Convergence history diagram of function 19 with 10 variables

Fig. 18 Convergence history diagram of function 20 with 10 variables

Fig. 19 Convergence history diagram of function 21 with 10 variables

Fig. 20 Convergence history diagram of function 23 with 10 variables

The performance results of the proposed algorithm in solving the CEC2005 functions with 30 variables are given in Table 2. Moreover, the diagrams of these functions are provided in Figs. 22, 23, 24 and 25.

4.2 The CEC_2021 benchmark functions

In this study, the HTC hybrid algorithm is analyzed using CEC2021 under cases with 10 and 20 variables with 200,000 and 1,000,000 runs, respectively, and it is evaluated in basic, shift, and shift and rotation cases. Moreover, to test the performance of this algorithm, the CEC2021 functions are solved under completely equal conditions by the teaching-and learning-based optimization (TLBO) algorithm, which
is the main basis of the HTC algorithm. The results, which include the values of minimum, median, average, maximum, and standard deviation (Std), are compared with each other, which are given in Tables 3 and 4.

The results of the comparison between the performance of the HTC algorithm and the teaching- and learning-based optimization (TLBO) algorithm in solving the 2021 benchmark functions in 10- and 20-variable cases show that in general, in 10-variable case, this algorithm reaches about 70% of the desired solutions in solving these functions. This value is 73% for the 20-variable case. In part, by comparing the minimum values in solving the benchmark functions with 10 variables in basic, shift, and shift and rotation cases, the HTC algorithm obtains the desired solutions in 80%, 70%, and 70% of the cases, respectively. These values are 70%, 80%, and 60% in the 20-variable case, respectively. Comparison of the median
values in the 10-variable and basic cases shows that the HTC algorithm has better results in functions 1, 2, 5, 6, 8, and 9, which in the cases of shift and shift and rotation are functions 1, 2, 4, 5, 6, 8, and 9, respectively. For the 20-variable case, these functions are 1, 2, 3, 4, 5, 6, 8, 9, and 10 in the basic case, 1, 2, 4, 6, and 10 in the shift case, and 1, 2, 4, 5, 6, 7, 8, and 10 in the shift and rotation cases. Other results including the standard deviation and average values are given in Tables 3 and 4.

### 4.3 Structural problems

In this section, different types of trusses are examined and analyzed to evaluate the performance of the HCT algorithm in structural problems. These trusses are divided into two categories with discrete and continuous variables. The 72, 120, and 942-member trusses are analyzed, respectively, with 16, 7, and 59 continuous design variables, and the
A 244-member truss is analyzed with 32 discrete design variables. In the end, the results of analyzing these trusses by
the current algorithm are compared with other algorithms.

### 4.3.1 72-bar truss

In the first example, a 72-bar space truss, as shown in
Fig. 26, is analyzed. The density of the material is 0.1 lb/in³,
and the modulus of elasticity is 10000 ksi. The applied loads
are: \( P_x = 5 \) kips, \( P_y = 5 \) kips and \( P_z = -5 \) kips. Moreover,
the truss members are divided into 16 groups, which are
given in Table 5. The results of comparing the HTC algo-
rithm with the other algorithms are given in Table 6, and the
convergence history of optimum answer is shown in Fig. 27.

The 72-bar truss is considered as one of the bench-
mark structural problems used to study the performance of

---

**Table 4** Comparison of the results of the TLBO and HTC algorithms for CEC2021 functions (\( D = 20 \))

|        | F1  | F2  | F3  | F4  | F5  |
|--------|-----|-----|-----|-----|-----|
|        | HTC | TLBO| HTC | TLBO| HTC | TLBO| HTC | TLBO| HTC | TLBO|
| Basic  |     |     |     |     |     |     |     |     |     |     |
| Minimum| 0.00| 0.00| 0.00| 0.12| 0.00| 0.00| 0.00| 0.00| 0.00| 0.00|
| Median | 0.00| 0.00| 0.19| 0.22| 2.73| 3.14| 0.00| 0.00| 0.00| 0.00|
| Average| 0.00| 0.00| 0.23| 0.32| 4.74| 3.73| 0.11| 0.08| 0.08| 0.08|
| Maximum| 0.00| 0.00| 1.77| 1.89| 39.32| 12.93| 1.21| 0.59| 2.30| 2.30|
| Std    | 0.00| 0.00| 0.30| 0.43| 7.58| 4.01| 0.31| 0.17| 0.42| 0.42|
| Shift  |     |     |     |     |     |     |     |     |     |     |
| Minimum| 0.00| 0.00| 6.89|123.69|30.34|36.20| 1.74| 1.88|303.25|136.72|
| Median | 0.00| 0.00|475.48|489.31|47.93|47.32| 3.64| 4.15|613.94|478.17|
| Average| 0.00| 0.00|535.75|500.95|47.44|48.07| 3.73| 4.19|609.66|531.27|
| Maximum| 0.00| 0.00|1219.33|1253.62|62.14|65.53| 6.33| 7.57|1078.54|1043.77|
| Std    | 0.00| 0.00|326.73|249.75| 7.46| 7.24| 1.26| 1.45|219.98|237.79|
| Shift and rotation |     |     |     |     |     |     |     |     |     |     |
| Minimum| 0.00| 0.01|125.55|125.27|29.61|31.94| 1.45| 1.47|2400.90|2081.77|
| Median | 11.64|30.66|390.95|473.22|46.61|45.67| 2.94| 3.67|7510.78|11,472.66|
| Average| 55.39|119.74|475.51|528.07|47.35|46.30| 3.53| 3.71|12,489.84|13,732.86|
| Maximum| 679.08|660.37|1341.17|1254.99|64.07|65.53| 6.33| 7.57|51,940.00|40,540.89|
| Std    | 130.60|178.19|306.55|256.91| 9.25| 8.48| 2.00| 1.42|12,144.87|8958.06|

---

244-member truss is analyzed with 32 discrete design var-
iables. In the end, the results of analyzing these trusses by
the current algorithm are compared with other algorithms.
meta-heuristic algorithms. In this study, the performance of HTC algorithm was investigated with this truss structure. The results showed that not only the proposed algorithm (HTC) has better performance than the CSS and TLBO algorithms, but also it performs better than some of the other algorithms listed in Table 6. According to the results of the previous studies presented in this table, the best performance is related to the BB-BC algorithm, which has optimized the weight of the structure up to 379.85 lb, while the HTC algorithm has reduced the weight to 379.6349 lb. Moreover, the results obtained from investigating the standard deviation (Std) values showed that the proposed algorithm has a stable performance and has less dispersion of results compared to the other methods.

### Table 5: Member groups of the 72-bar truss

| Number of group | Members               |
|-----------------|-----------------------|
| 1               | A1–A4                 |
| 2               | A5–A12                |
| 3               | A13–A16               |
| 4               | A17–A18               |
| 5               | A19–A22               |
| 6               | A23–A30               |
| 7               | A31–A34               |
| 8               | A35–A36               |
| 9               | A37–A40               |
| 10              | A11–A48               |
| 11              | A49–A52               |
| 12              | A53–A54               |
| 13              | A55–A58               |
| 14              | A59–A66               |
| 15              | A67–A70               |
| 16              | A71–A72               |

### Table 6: Optimum values of cross-sectional areas and weight of the 72-bar truss

| Area (in²) | GWO [35] | GSA | ACO [36] | PSO [37] | BB-BC [38] | RO [39] | CSS [40] | HTC |
|------------|----------|-----|----------|----------|------------|---------|----------|-----|
| A1         | 1.911    | 1.874 | 1.948    | 1.7427   | 1.5877     | 1.8365  | 1.9039   |
| A2         | 0.521    | 0.493 | 0.508    | 0.5185   | 0.5059     | 0.5021  | 0.442    | 0.5093 |
| A3         | 0.1      | 0.103 | 0.101    | 0.1      | 0.1        | 0.1     | 0.111    | 0.10009 |
| A4         | 0.1      | 0.109 | 0.102    | 0.1      | 0.1        | 0.1004  | 0.111    | 0.1    |
| A5         | 1.343    | 1.349 | 1.303    | 1.3079   | 1.2476     | 1.2522  | 0.994    | 1.2702 |
| A6         | 0.499    | 0.477 | 0.511    | 0.5193   | 0.5269     | 0.5033  | 0.563    | 0.51104 |
| A7         | 0.102    | 0.1   | 0.101    | 0.1      | 0.1        | 0.1002  | 0.111    | 0.1    |
| A8         | 0.1      | 0.1   | 0.1      | 0.1      | 0.1        | 0.1002  | 0.111    | 0.1    |
| A9         | 0.575    | 0.55  | 0.561    | 0.5142   | 0.5209     | 0.573   | 0.563    | 0.5238 |
| A10        | 0.504    | 0.531 | 0.492    | 0.5464   | 0.5172     | 0.5499  | 0.563    | 0.51657 |
| A11        | 0.1      | 0.103 | 0.1      | 0.1      | 0.1        | 0.1004  | 0.111    | 0.10002 |
| A12        | 0.104    | 0.107 | 0.107    | 0.1095   | 0.1005     | 0.1001  | 0.111    | 0.10004 |
| A13        | 0.155    | 0.155 | 0.156    | 0.1615   | 0.1565     | 0.1576  | 0.196    | 0.15657 |
| A14        | 0.549    | 0.619 | 0.550    | 0.5092   | 0.5509     | 0.5222  | 0.563    | 0.54229 |
| A15        | 0.391    | 0.329 | 0.390    | 0.4967   | 0.3922     | 0.4356  | 0.442    | 0.4123 |
| A16        | 0.533    | 0.57  | 0.592    | 0.5619   | 0.5922     | 0.5972  | 0.766    | 0.57705 |
| Weight (lb) | 379.95   | 382.31 | 380.24   | 381.91   | 379.85     | 380.458 | 393.05   | 379.6349 |
| W avg. (lb) | N/A      | N/A   | 383.16   | N/A      | 382.08     | 382.5538 | N/A      | 379.6556 |
| W std (lb)  | N/A      | N/A   | 3.66     | N/A      | 1.912      | 1.2211  | N/A      | 0.014371 |
| N analysis  | 28,000   | 10,040 | 18,500   | 150,000  | 19,621     | 19,084  | 5370     | 24,000 |

---

Fig. 26 Geometry, boundary conditions, and structural configuration of the 72-bar truss
4.3.2 120-bar truss

In the second example, a 120-bar truss with a dome geometry, as shown in Fig. 28, is analyzed. To evaluate the HTC algorithm in the 120-bar truss, the cross section of the members is examined as design variables assuming a fixed topology, under different constraints. The allowable compressive and tensile stresses are calculated based on the AISCASD standard as follows:

\[
\begin{align*}
\sigma^+ &= 0.6F_y, \quad \text{for} \quad \sigma_i \geq 0 \\
\sigma^- &= \quad \text{for} \quad \sigma_i < 0
\end{align*}
\]

(21)

in which \(\sigma_i\) is calculated based on the slenderness coefficient.

Table 7 Vertical loadings on the 120-bar truss structure nodes

| Node | Force (kips) |
|------|--------------|
| 1    | −13.49       |
| 2–14 | −6.744       |
| 15–37| −2.248       |

Fig. 27 Convergence history diagram of the 72-bar truss related to the HTC algorithm

Fig. 28 Geometry, boundary conditions, and structural configuration of the 120-bar truss
### Table 8: Optimum values of cross-sectional areas of the 120-bar truss (Case 1)

| Area (in²) | GWO     | GSA     | HS       | PSO     | HPSACO  | PSOPC   | HTC     |
|------------|---------|---------|----------|---------|---------|---------|---------|
| A1         | 3.119   | 3.126   | 3.295    | 3.147   | 3.311   | 3.235   | 3.1229  |
| A2         | 3.355   | 3.354   | 3.396    | 6.376   | 3.438   | 3.37    | 3.3538  |
| A3         | 4.076   | 5       | 3.874    | 5.957   | 4.147   | 4.116   | 4.1121  |
| A4         | 2.78    | 2.7841  | 2.571    | 4.806   | 2.831   | 2.784   | 2.7822  |
| A5         | 0.776   | 0.775   | 1.15     | 0.775   | 0.777   | 0.777   | 0.775   |
| A6         | 3.3     | 3.3013  | 3.331    | 13.79   | 3.474   | 3.343   | 3.3005  |
| A7         | 2.445   | 2.44    | 2.784    | 2.452   | 2.551   | 2.454   | 2.4459  |

Weight (Ib.) | 19,470  | 20,177  | 19,618   | 32,432  | 19,491  | 19,618  | **19,454.79** |

W avg. (Ib) | –       | –       | –        | –       | –       | –       | **19,454.79** |

W std (Ib)  | –       | –       | –        | –       | –       | –       | **9.02E−08**  |

N analysis  | 14,000  | 10,040  | 35,000   | 125,000 | 10,000  | 125,000 | **12,000** |

### Table 9: Optimum values of cross-sectional areas of the 120-bar truss (Case 2)

| Area (in²) | GWO       | GSA     | PSO     | RO      | PSOPC   | HTC     |
|------------|-----------|---------|---------|---------|---------|---------|
| A1         | 3.027     | 3.0257  | 12.802  | 3.03    | 3.04    | 3.02421 |
| A2         | 14.824    | 14.6877 | 11.765  | 14.806  | 13.149  | 14.7702 |
| A3         | 5.139     | 5.1805  | 5.654   | 5.44    | 5.646   | 5.04746 |
| A4         | 3.118     | 3.135   | 6.333   | 3.124   | 3.143   | 3.13594 |
| A5         | 8.323     | 8.3998  | 6.963   | 8.021   | 8.759   | 8.50037 |
| A6         | 3.84      | 3.3572  | 6.492   | 3.614   | 3.758   | 3.30729 |
| A7         | 2.497     | 2.4941  | 4.988   | 2.487   | 2.502   | 2.4972  |

Weight (Ib.) | 33,271   | 33,254  | 51,986  | 33,317.8| 33,481  | **33,251.31** |

W avg. (Ib) | –        | –       | –       | –       | –       | –       | **33,252.54** |

W std (Ib)  | –        | –       | –       | –       | –       | –       | **0.85**    |

N analysis  | 14,000   | 10,040  | 125,000 | –       | 125,000 | **12,000** |

**Fig. 29** Convergence history diagram of the 120-bar truss related to the HTC algorithm (Case 1)

**Fig. 30** Convergence history diagram of the 120-bar truss related to the HTC algorithm (Case 2)
\[
\sigma_i = \begin{cases} 
\left(1 - \frac{\lambda_i^2}{2c_i^2}\right)F_y \left(\frac{5}{3} + \frac{3\lambda_i}{8c_i} - \frac{\lambda_i^3}{8c_i^3}\right) & \text{for } \lambda_i < C_c \\
\frac{12E_i}{23c_i^4} & \text{for } \lambda_i \geq C_c 
\end{cases}
\]

The modulus of elasticity is 30450 ksi, the density of the material is 0.288 lb/in³, and the yield stress of steel is considered to be 58.0 ksi. The radius of gyration \(r_i\) is expressed in terms of the cross section as \(r_i = aAb_i\), where \(a\) and \(b\) are constant coefficients that are selected depending on the type of the cross section (pipe, L and T sections). In this example, the pipe section \((a=0.4993\text{ and } b=0.6777)\) is considered for the members. All members of this dome are divided into seven groups, which are shown in Fig. 28. All nodes, except the ones representing the supports, are subjected to vertical loading, as given in Table 7. The minimum and maximum cross-sectional areas of all members are 0.775 in² and 20 in², respectively. In this example, two of the four cases of constraints are considered. In the first case, only the stress constraints are considered, and the displacement constraints are ignored, while in the second case, along with the stress constraints, displacement constraints of ±0.1969 in for all nodes in the x and y directions are also considered. The results of comparing the HTC algorithm with the other algorithms for the two mentioned cases are given in Tables 8 and 9, and the convergence history of optimum answer is shown in Figs. 29 and 30, respectively.

According to the results of the performance of the HTC algorithm in Tables 8 and 9, it has a better performance in optimizing and solving the 120-bar truss problem in both considered cases compared to the other previous algorithms presented in these tables above. Moreover, the values of the standard deviation also indicates the stability and non-scattering of the data in the presented algorithm.

**Table 10** Information of the loading and displacement in the joints of the 244-bar truss

| Joint Number | Loading (KN) | Displacement constraint (mm) |
|--------------|--------------|------------------------------|
|              | \(X\)        | \(Z\)                        |
|              | \(X\)        | \(Z\)                        |
| 1            | −10          | −30                          |
| 2            | 10           | −30                          |
| 17           | 35           | −90                          |
| 24           | 175          | −45                          |
| 25           | 175          | −45                          |

**Table 11** Optimum cross section of each group

| Member | Area  | Member | Area  | Member | Area  | Member | Area  | Member | Area  |
|--------|-------|--------|-------|--------|-------|--------|-------|--------|-------|
| A_1    | 0.775 | A_9    | 0.775 | A_17   | 8.300 | A_25   | 1.400 |
| A_2    | 0.775 | A_10   | 1.400 | A_18   | 2.300 | A_26   | 5.900 |
| A_3    | 0.775 | A_11   | 1.400 | A_19   | 2.000 | A_27   | 2.300 |
| A_4    | 0.775 | A_12   | 0.775 | A_20   | 5.900 | A_28   | 0.775 |
| A_5    | 2.000 | A_13   | 4.700 | A_21   | 0.775 | A_29   | 0.775 |
| A_6    | 1.100 | A_14   | 3.200 | A_22   | 1.100 | A_30   | 2.300 |
| A_7    | 2.000 | A_15   | 6.200 | A_23   | 0.775 | A_31   | 5.300 |
| A_8    | 1.100 | A_16   | 2.600 | A_24   | 0.775 | A_32   | 5.300 |

**Fig. 31** Geometry, boundary conditions, and structural configuration of the 244-bar truss
4.3.3 244-bar truss

In the third example, the 244-bar truss shown in Fig. 31 is analyzed. Its members are divided into 32 groups. The modulus of elasticity of 30,450 ksi and the density of the material is 0.1 lb/in$^3$. The allowable tensile stress is 20.30 ksi and for the allowable compressive stress, the buckling relations of the AISCASD (1989) standard have been used, as in the 120-member truss problem. The displacement constraints are as follows: ± 1.77 in for nodes 1 and 2, ± 1.18 in for nodes 17, 24, and 25 in the x direction. These nodes are under a constraint of ± 0.59 in the z direction as well. The loading condition is applied as in the study of Ulker, which is presented in Table 10. The minimum and maximum cross-sectional areas for all members are 0.775 in$^2$ and 18.2 in$^2$, respectively. Moreover, Table 11 provides the optimum cross-section area for each member group of the 244-bar truss. The results of comparing the HTC algorithm with the other algorithms are given in Table 12, and the convergence history of the HTC algorithm is shown in Fig. 32, respectively.

According to the results of the HTC algorithm in solving the 244-bar truss, the most optimized weight obtained from this algorithm is 4808.093 lb, which has much better compared to the results of the HPSACO and PSOPC algorithms.

Moreover, the value of the standard deviation, which indicates the stability of the algorithm and the amount of data scattering, is 0.001102 for the HTC algorithm, which is not provided for the HPSACO and PSOPC algorithms.

4.3.4 942-bar truss

The last example is a 26-story space truss that consists of 942 members. The geometry of the structure is shown in Fig. 33. Due to the symmetry, the space truss is divided into 59 groups. The material density is 0.1 lb/in$^3$ and the modulus of elasticity is 10000 ksi. The structure is subjected to the following horizontal and vertical loads: (1) The vertical loads in the z direction applied to each node are 3, 6, and 9 kips in the first, second and third parts, respectively, (2) The lateral loads in the y direction applied to all nodes are 1 kips, and (3) the lateral loads in the x direction applied to each node are 1.5 and 1 kips in the left and right side of the structure, respectively. The areas of the allowable sections in this example are in the range of 1 to 200 in$^2$. Table 13 provides the optimum cross-section area for each member group of the 942-bar truss. Moreover, the results of comparing the HTC algorithm with the other algorithms are given in Table 14, and the convergence history is shown in Fig. 34, respectively.

According to the results of HTC algorithm in solving the 942-bar truss in Table 14, the most optimized weight obtained from HTC algorithm is 141175.5 lb, which is much better compared to the results of the ES, GNMS, GWO, SA algorithms.

### Table 12 Performance comparison for the 244-bar tower truss

| Method   | HPSACO [42]   | PSOPC [43]   | HTC   |
|----------|---------------|--------------|-------|
| Weight (lb) | 5324.2        | 5847.9       | 4808.093 |
| Average weight (lb) | N/A           | N/A          | 4808.094 |
| Std      | N/A           | N/A          | 0.001102 |
| Analysis | 10,775        | 150,000      | 18,000 |

### Table 13 Optimum cross section of each group

| Members | Area | Members | Area | Members | Area |
|---------|------|---------|------|---------|------|
| A_1     | 1.356| A_{21}  | 31.629| A_{41}  | 1.617|
| A_2     | 1.012| A_{22}  | 3.369 | A_{42}  | 2.842|
| A_3     | 2.913| A_{23}  | 16.397| A_{43}  | 81.107|
| A_4     | 1.929| A_{24}  | 26.392| A_{44}  | 3.438|
| A_5     | 1.002| A_{25}  | 38.431| A_{45}  | 1.643|
| A_6     | 15.479| A_{26} | 1.933 | A_{46}  | 4.082|
| A_7     | 3.141| A_{27}  | 12.431| A_{47}  | 1.1402|
| A_8     | 6.792| A_{28}  | 16.620| A_{48}  | 2.142|
| A_9     | 19.471| A_{29} | 14.612| A_{49}  | 95.504|
| A_{10}  | 3.316| A_{30}  | 16.631| A_{50}  | 3.813|
| A_{11}  | 5.279| A_{31}  | 38.178| A_{51}  | 1.008|
| A_{12}  | 5.572| A_{32}  | 3.597 | A_{52}  | 4.289|
| A_{13}  | 16.222| A_{33} | 3.406 | A_{53}  | 7.366|
| A_{14}  | 2.262| A_{34}  | 2.649 | A_{54}  | 4.277|
| A_{15}  | 4.053| A_{35}  | 1.006 | A_{55}  | 41.037|
| A_{16}  | 1.000| A_{36}  | 1.509 | A_{56}  | 1.002|
| A_{17}  | 21.735| A_{37} | 59.510| A_{57}  | 65.102|
| A_{18}  | 2.841| A_{38}  | 3.468 | A_{58}  | 2.617|
| A_{19}  | 8.559| A_{39}  | 1.925 | A_{59}  | 1.008|
| A_{20}  | 1.021| A_{40}  | 3.743 |        |      |

![Fig. 32 Convergence history diagram of the 244-bar truss related to the HTC algorithm](image-url)
and CGFA algorithms. After HTC algorithm, ES algorithm is in the second place, which has a much higher analysis cost than HTC algorithm.

5 Concluding remarks

Nowadays, most of the algorithms proposed by researchers, including the teaching- and learning-based optimization (TLBO) and the charged system search (CSS) algorithms, are powerful tools for solving mathematical and structural optimization problems. However, in some cases, the complexity of problems has a direct impact on the performance of these types of algorithms, which causes the algorithm to not have a powerful performance in solving complex problems. The purpose of this study was to introduce a powerful algorithm (the hybrid teaching–learning-based optimization and charged system search (HTC) algorithm) with high ability in exploration and exploitation using a combination of the two mentioned algorithms, namely the TLBO and CSS algorithms. The proposed algorithm is based on the TLBO algorithm and the CSS algorithm is used to increase exploration and exploitation. As a result, it increases the possibility of exploring a global optimization in the search space and ensures the convergence of the algorithm. Comparing the results with other modern methods for solving benchmark functions and truss design problems showed that the HTC algorithm is a powerful tool for finding
good solutions to various problems. Therefore, the use of this 
algorithm in solving other design and mathematical problems 
is suggested.

### Appendix A

#### CEC2005 functions

See Table 15.

| Function | Type | Initialization range | Search bound | Minimum answer |
|----------|------|----------------------|--------------|----------------|
| C1: Shifted sphere function | Unimodal | [− 100, 100] | [− 100, 100] | − 450 |
| C2: Shifted Schwefel’s Problem 1.2 | | [− 100, 100] | [− 100, 100] | − 450 |
| C3: Shifted rotated high conditioned elliptic function | | [− 100, 100] | [− 100, 100] | − 450 |
| C4: Shifted Schwefel’s problem 1.2 with noise in fitness | | [− 100, 100] | [− 100, 100] | − 450 |
| C5: Schwefel’s problem 2.6 with global optimum on bounds | | [− 100, 100] | [− 100, 100] | − 310 |
| C6: Shifted Rosenbrock’s function | Basic modal | [− − 100, 100] | [− 100, 100] | 390 |
| C7: Shifted rotated Griewank’s function without bounds | | [0,600] | [− 600,600] | − 180 |
| C8: Shifted rotated Ackley’s function with global optimum on bounds | | [− 32,32] | [− 32,32] | − 140 |
| C9: Shifted Rastrigin’s function | | [− 5,5] | [− 5,5] | 330 |
| C10: Shifted rotated Rastrigin’s function | | [− 5,5] | [− 5,5] | 330 |
| C11: Shifted rotated Weierstrass function | | [− 0.5,0.5] | [− 0.5,0.5] | 90 |
| C12: Schwefel’s Problem 2.13 | | [− 100,100] | [− 100,100] | 460 |
| C13: Expanded extended Griewank’s plus Rosenbrock’s function (CECSCEC2) | Expanded modal | [− 3,1] | [− 3,1] | 130 |
| C14: Shifted rotated expanded Scaffer’s CEC6 | | [− 100,100] | [− 100,100] | 300 |
| C15: Hybrid composition function | Hybrid composition modal | [− 5,5] | [− 5,5] | 120 |
| C16: Rotated Hybrid Composition Function | | [− 5,5] | [− 5,5] | 120 |
| C17: Rotated hybrid composition function with noise in fitness | | [− 5,5] | [− 5,5] | 120 |
| C18: Rotated hybrid composition function | | [− 5,5] | [− 5,5] | 10 |
| C19: Rotated hybrid composition function with a narrow basin for the global optimum | | [− 5,5] | [− 5,5] | 10 |
| C20: Rotated hybrid composition function with the global optimum on the bounds | | [− 5,5] | [− 5,5] | 10 |
| C21: Rotated hybrid composition function | | [− 5,5] | [− 5,5] | 360 |
| C22: Rotated hybrid composition function with high condition number matrix | | [− 5,5] | [− 5,5] | 360 |
| C23: Non-continuous rotated hybrid composition function | | [− 5,5] | [− 5,5] | 360 |
| C24: Rotated hybrid composition function | | [− 5,5] | [− 5,5] | 260 |
| C25: Rotated hybrid composition function without bounds | | [− 2,5] | [− 5,5] | 260 |
Search space diagrams of the CEC2005 functions

See Figs. 35, 36, 37, 38, 39, 40, 41 and 42.

Fig. 35 Search space diagram of CEC2005 function 2

Fig. 36 Search space diagram of CEC2005 function 4

Fig. 37 Search space diagram of CEC2005 function 6

Fig. 38 Search space diagram of CEC2005 function 9

Fig. 39 Search space diagram of CEC2005 function 8

Fig. 40 Search space diagram of CEC2005 function 13
Appendix B

CEC2021 functions

See Table 16.

Table 16 Types of CEC2021 function

| Type            | No | Functions                                                                 | \( F_1^* \) |
|-----------------|----|---------------------------------------------------------------------------|-------------|
| Unimodal function | 1  | Shifted and Rotated Bent Cigar Function (CEC 2017 F1)                    | 100         |
| Basic function  | 2  | Shifted and Rotated Schwefel’s Function (CEC 2014 F11)                  | 1100        |
|                 | 3  | Shifted and Rotated Lunacek bi-Rastrigin Function (CEC 2017 F7)         | 700         |
|                 | 4  | Expanded Rosenbrock’s plus Griewangk’s Function (CEC2017F19)             | 1900        |
| Hybrid function | 5  | Hybrid function 1 \((N=3)\) (CEC 2014 F17)                               | 1700        |
|                 | 6  | Hybrid function 2 \((N=4)\) (CEC 2017 F16)                               | 1600        |
|                 | 7  | Hybrid function 3 \((N=5)\) (CEC 2014 F21)                               | 2100        |
| Composition function | 8 | Composition function 1 \((N=3)\) (CEC 2017 F22)                          | 2200        |
|                 | 9  | Composition function 2 \((N=4)\) (CEC 2017 F24)                          | 2400        |
|                 | 10 | Composition function 3 \((N=5)\) (CEC 2017 F25)                          | 2500        |

Search range: \([-100, 100]\)^T
Three-dimensional diagram and search space contours for each of CEC2021 functions

See Figs. 43, 44, 45, 46, 47, 48, 49 and 50.

**Fig. 43** Three-dimensional diagram and contours of the first CEC2021 function

**Fig. 44** Three-dimensional diagram and contours of the second CEC2021 function

**Fig. 45** Three-dimensional diagram and contours of the third CEC2021 function
Fig. 46 Three-dimensional diagram and contours of the fourth CEC2021 function

Fig. 47 Three-dimensional diagram and contours of the sixth CEC2021 function

Fig. 48 Three-dimensional diagram and contours of the eighth CEC2021 function
Data availability  We do not analysis or generate any datasets, because our work proceeds within a theoretical and mathematical approach.

References

1. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA
2. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12:702–713
3. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359
4. Salehan A, Deldari A (2021) Corona virus optimization (CVO): a novel optimization algorithm inspired from the Corona virus pandemic. J Supercomput https://doi.org/10.1007/s11227-021-04100-z
5. Li Z, Tam V (2020) A novel meta-heuristic optimization algorithm inspired by the spread of viruses. arXiv Prepr arXiv200606282
6. Eberhart R, Kennedy J (1995) Particle swarm optimization. In: Proceedings of the IEE international conference on neural networks. Citeseer, pp 1942–1948
7. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1:28–39
8. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39:459–471
9. Chu SC, Tsai PW, Pan JS (2006) Cat swarm optimization. In: Yang Q, Webb G (eds) PRICAI 2006: Trends in artificial intelligence. Lecture Notes in Computer Science, vol 4099. Springer, Berlin, Heidelberg
10. Mirjalili SZ, Mirjalili S, Saremi S (2018) Grasshopper optimization algorithm for multi-objective optimization problems. Appl Intell 48:805–820
11. Hayyolalam V, Kazem AAP (2020) Black widow optimization algorithm: a novel meta-heuristic approach for solving engineering optimization problems. Eng Appl Artif Intell 87:103249
12. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213:267–289
13. Hatamlou A (2013) Black hole: a new heuristic optimization approach for data clustering. Inf Sci (Ny) 222:175–184
14. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci (Ny) 179:2232–2248
15. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(880):671–680
16. Erol OK, Eksin I (2006) A new optimization method: big bang–big crunch. Adv Eng Softw 37:106–111
17. Lam AYS, Li VOK (2010) Chemical reaction optimization for cognitive radio spectrum allocation. In: 2010 IEEE global telecommunications conference GLOBECOM 2010. IEEE, pp 1–5
18. Kaveh A, Khayatazad M (2012) A new meta-heuristic method: ray optimization. Comput Struct 112–113:283–294. https://doi.org/10.1016/j.compstruct.2012.09.003
19. Kaveh A, Ghazaan MI (2017) A new meta-heuristic algorithm: vibrating particles system. Sci Iran Trans A Civ Eng 24:551
20. Rao RV, Savsani VJ, Vakharia DP (2011) Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Des 43:303–315
21. Zhang Y, Jin Z (2020) Group teaching optimization algorithm: a novel method for solving global optimization problems. Expert Syst Appl 148:113246
22. Ahmadi S-A (2017) Human behavior-based optimization: a novel metaheuristic approach to solve complex optimization problems. Neural Comput App 28:233–244
23. Maheri MR, Askarian M, Shojaee S (2016) Size and topology optimization of trusses using hybrid genetic-particle swarm algorithms. Iran J Sci Technol Trans Civ Eng 40:179–193
24. Talatahari S, Goodarzimehr V, Taghizadieh N (2020) Hybrid teaching-learning-based optimization and harmony search for optimum design of space trusses. J Optim Ind Eng 13:177–194
25. Talatahari S, Goodarzimehr V, Shojaee S (2021) Symbiotic organisms search and harmony search algorithms for discrete optimization of structures. Iran Univ Sci Technol 11:177–194
26. Omidinasab F, Goodarzimehr V (2020) A hybrid particle swarm optimization and genetic algorithm for truss structures with discrete variables. J Appl Comput Mech 6:593–604
27. Kaveh A, Talatahari S (2012) A hybrid CSS and PSO algorithm for optimal design of structures. Struct Eng Mech 42:783–797. https://doi.org/10.12989/jscm.2012.42.6.783
28. Awad R (2021) Sizing optimization of truss structures using the political optimizer (PO) algorithm. Structures 33:4871–4894. https://doi.org/10.1016/j.istruc.2021.07.027
29. Artar M, Carbas S (2021) Discrete sizing design of steel truss bridges through teaching-learning-based and biogeography-based optimization algorithms involving dynamic constraints. Structures 34:3533–3547. https://doi.org/10.1016/j.istruc.2021.09.101
30. Talatahari S, Azizi M (2020) Optimum design of building structures using tribe-interior search algorithm. Structures 28:1616–1633. https://doi.org/10.1016/j.istruc.2020.09.075
31. Jafari M, Salajegheh E, Salajegheh J (2021) Optimal design of truss structures using a hybrid method based on particle swarm optimizer and cultural algorithm. Structures. Elsevier, pp 391–405
32. Baykasoğlu A, Baykasoğlu C (2021) Weighted superposition attraction-repulsion (WSAR) algorithm for truss optimization with multiple frequency constraints. Structures 30:253–264. https://doi.org/10.1016/j.istruc.2021.01.017
33. Jawad FJK, Mahmood M, Wang D (2021) Heuristic dragonfly algorithm for optimal design of truss structures with discrete variables. Structures 29:843–862. https://doi.org/10.1016/j.istruc.2020.11.071
34. Jawad FJK, Ozturk C, Dansheng W, (2021) Sizing and layout optimization of truss structures with artificial bee colony algorithm. Structures. Elsevier, pp 546–559
35. Kaveh A, Zakian P (2018) Improved GWO algorithm for optimal design of truss structures. Eng Comput 34:685–707. https://doi.org/10.1007/s00366-017-0567-1
36. Kaveh A, Talatahari S (2009) A particle swarm ant colony optimization for truss structures with discrete variables. J Constr Steel Res 65:1558–1568. https://doi.org/10.1016/j.jcsr.2009.04.021
37. Perez REL, Behdinan K (2007) Particle swarm approach for structural design optimization. Comput Struct 85:1579–1588
38. Kaveh A, Talatahari S (2009) Size optimization of space trusses using Big Bang-Big Crunch algorithm. Comput Struct 87:1129–1140. https://doi.org/10.1016/j.compstruc.2009.04.011
39. Kaveh A, Khayatazad M (2013) Ray optimization for size and shape optimization of truss structures. Comput Struct 117:82–94. https://doi.org/10.1016/j.compstruc.2012.12.010
40. Kaveh A, Talatahari S (2010) Optimal design of double layer barrel vaults using improved magnetic charged system search. Asian J Civil Eng (building and housing) 15:135–154
41. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
42. Kaveh A, Talatahari S (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 87:267–283. https://doi.org/10.1016/j.compstruc.2009.01.003
43. Lu YC, Jan JC, Hung S-L, Hung GH (2013) Enhancing particle swarm optimization algorithm using two new strategies for optimizing design of truss structures. Eng Optim 45:1251–1271
44. Hasancıoğlu O, Erbatur F (2002) On efficient use of simulated annealing in complex structural optimization problems. Acta Mech 157:27–50
45. Hasancıoğlu O (2008) Adaptive evolution strategies in structural optimization: enhancing their computational performance with applications to large-scale structures. Comput Struct 86:119–132. https://doi.org/10.1016/j.compstruc.2007.05.012
46. Rahami H, Kaveh A, Aslani M, Najian Asl R (2011) A hybrid modified genetic-nelder mead simplex algorithm for large-scale truss optimization. Iran Univ Sci Technol 1:29–46
47. Kaveh A, Moghanni RM, Javadi SM (2019) Optimum design of large steel skeletal structures using chaotic firefly optimization algorithm based on the Gaussian map. Struct Multidiscip Optim 60:879–894

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.