 https://doi.org/10.3311/PPci.20133 

**Periodica Polytechnica Civil Engineering,** 66(3), pp. 820–842, 2022

# A Physics-based Metaheuristic Algorithm Based on Doppler Effect Phenomenon and Mean Euclidian Distance Threshold

Ali Kaveh[1*], Seyed Milad Hosseini[1], Ataollah Zaerreza[1]

[1] School of Civil Engineering, Iran University of Science and Technology, Narmak, 16846-13114, Tehran, Iran
[*] Corresponding author, e-mail: alikaveh@iust.ac.ir

Received: 09 March 2022, Accepted: 10 April 2022, Published online: 09 May 2022

**Abstract**

Doppler Effect (DE) is a physical phenomenon observed by Doppler, an Austrian mathematician, in 1842. In recent years, the mathematical formulation of this phenomenon has been used to improve the frequency equation of the standard Bat Algorithm (BA) developed by Yang in 2010. In this paper, we use the mathematical formulation of DE with some idealized rules to update the observer velocity existing in the Doppler equation. Thus, a new physics-based Metaheuristic (MH) optimizer is developed. In the proposed algorithm, the observers' velocities as the algorithm's search agents are updated based on the DE equation. A new mechanism named Mean Euclidian Distance Threshold (MEDT) is introduced to enhance the quality of the observers. The proposed MEDT mechanism is also employed to avoid the locally optimum solutions and increase the convergence rate of the presented optimizer. Since the proposed algorithm simultaneously utilizes the DE equation and MEDT mechanism, it is called the Doppler Effect-Mean Euclidian Distance Threshold (DE-MEDT) metaheuristic algorithm. The proposed DE-MEDT algorithm's efficiency is evaluated by solving well-known unconstrained and constrained optimization problems. In the unconstrained optimization problems, 23 well-known optimization functions are used to assess the exploratory, exploitative, and convergence behaviors of the DE-MEDT algorithm.

**Keywords**

Doppler effect, optimization, metaheuristics, physics-based algorithm, engineering design problems

## 1 Introduction

Most real-world optimization problems in science and engineering applications are highly complex and have nonlinear limitations with non-convex search space. Since they have these challenging characteristics, it can be hard or even impossible to solve them using mathematical-based optimization methods. These methods are mostly deterministic-based and suffer from local optima entrapment. Moreover, the most popular of them, known as gradient-based algorithms, require gradient information to search near an initial starting point [1]. Many research items have recently revealed that these algorithms are not sufficiently efficient when dealing with complex problems. The competitive alternative solver, known as meta-heuristic (MH), does not have the handicaps of the gradient-based methods. They are free from requiring gradient information and have a high local optima avoidance ability [2]. Inspiring by a simple concept existing in natural phenomena and having easy implementation when optimizing the problems are the other reasons that show why MH algorithms have become considerably common in recent years [3].

In a general form, each MH algorithm can be either single solution-based or population-based. The number of candidate solutions improving during the optimization process determines the type of MH technique in terms of being single solution-based or population-based. In the former case, the optimization process starts with a single random solution. After that, it is iteratively improved in the cyclic body of the optimizer until satisfying a termination criterion, such as Maximum Number of Function Evaluations (MaxNFEs). In the latter case, the MH algorithm begins the optimization process with a set of randomly generated solutions, and the solutions are iteratively evolved until the termination criterion is met. Both of these types have their own advantages and disadvantages. For example, the advantages and disadvantages of individual-based metaheuristics are needing fewer function evaluations but suffering from unwanted premature convergence. On the contrary, population-based metaheuristics suffer from more function evaluations but benefit from high search ability to avoid local optima. In other

**Cite this article as:** Kaveh, A., Hosseini, S. M., Zaerreza, A. "A Physics-based Metaheuristic Algorithm Based on Doppler Effect Phenomenon and Mean Euclidian Distance Threshold", Periodica Polytechnica Civil Engineering, 66(3), pp. 820–842, 2022. https://doi.org/10.3311/PPci.20133

words, population-based metaheuristics have a higher search ability to avoid local optima compared to individual-based ones because more than one solution is involved during the optimization process. Furthermore, information obtained by the candidate solutions can be exchanged between themselves. This mechanism can help the candidate solutions to search different areas of the solution space more efficiently.

Based on the source of inspiration of the different MH methods, they can be roughly categorized into four main groups: Evolutionary Algorithms (EAs), Swarm Intelligence Algorithms (SIAs), Human-based Algorithms (HAs), and Physics-based algorithms (PAs).

EAs are inspired by biological evolution behaviors, such as crossover, mutation, and selection. GA is the most well-known EA that attempts to simulate the phenomenon of natural evolution. SIAs, as the second class of MH algorithms, are inspired by the social behavior of organisms living in a group, which can be swarm, herd, or flock. Particle Swarm Optimization (PSO) [4] is the most popular SIA simulating the social behavior of bird flocking. Ant Colony Optimization (ACO) [5] and Bat Algorithm (BA) [6] are the other popular examples of SIAs. The third classification of MH algorithms is composed of optimizers that mimic some human behaviors. For example, Teaching–Learning-Based Optimization (TLBO) [7] is one of the most well-known HAs proposed based on the effect of a teacher on the grade of the learners in a class. PAs, as the fourth class of MH algorithms, are inspired by physical laws. Examples of well-established and recently developed MHs that belong to this category are Ray Optimization (RO) [8], Colliding Bodies Optimization (CBO) [9], and Plasma Generation Optimization (PGO) [10].

Regardless of the inspiration source of various MH techniques, the searching steps of each MH algorithm are composed of two conflicting phases: exploration (diversification) and exploitation (intensification). In the exploration phase, the algorithm should explore deeply various regions of the solution space using its randomized operators. In contrast, in the exploitation phase, normally performed after the exploration phase, the metaheuristic attempts to search around solutions with higher fitness located inside the search space. The agents of a metaheuristic will trap into a local optimum without exploratory behavior. On the other hand, the lack of exploitative behavior decreases the metaheuristic performance in terms of finding better-quality solutions. In this case, the metaheuristic can never converge to the optimum solution. Thus, making a reasonable and fine balance between exploration and exploitation tendencies results in a well-organized metaheuristic.

As a challenging problem, some optimization problems require extreme exploratory behavior, and others may need extreme exploitative behavior to find the optimum solution. Making a proper trade-off between exploration and exploitation tendencies of the algorithms is another challenging issue. On the other hand, based on the No Free Launch (NFL) theorem [11], no unique MH algorithm can solve all types of optimization problems. It means that a specific MH method can provide promising results for a set of optimization problems. In contrast, the same method may not have enough efficiency for a different set of problems. Thus, this theorem encourages developing more efficient MH optimizers to solve the current problems better or test the performance of the existing MH optimizers in the new problems. For example, Kaveh et al. [12] introduced the Enhanced Shuffled Shepherd Optimization algorithm (ESSOA) for the optimal design of large-scale space structures. Kaveh and Zaerreza [13] introduced a new framework for reliability-based design optimization using ESSOA. The performance of this framework can be evaluated in different reliability problems presented by Movahedi Rad et al. [14], Lógó et al. [15], and Movahedi Rad and Khaleel Ibrahim [16].

In the present paper, a physics-based MH algorithm is developed to compete with other MH algorithms. The main idea behind the proposed algorithm is inspired by a physical phenomenon observed by Christian Andreas Doppler in 1842 [17]. According to the observed phenomenon, the Doppler Effect (DE), perceived frequency by the observer is determined based on its movement relative to the source. Compared to the frequency emitted by the source, the received frequency is higher when the observer approaches the source and is lower when the observer moves away from the source [18]. In recent years, DE has been used to improve BA by incorporating compensation of this effect in echoes of bats [19] or updating the frequency equation [20]. However, here, we inspire the formulation of Doppler in the sense of updating the velocities of observers as the search agents of the proposed algorithm. Accordingly, we propose a new mathematical model. Then, a new MH algorithm is designed based on the proposed mathematical framework to tackle different optimization problems. A new mechanism called Mean Euclidian Distance Threshold is developed to enhance the quality of the observers generated by the proposed algorithm. Since the proposed algorithm

integrates DE equation and MEDT mechanism simultaneously, it is named the Doppler Effect-Mean Euclidian Distance Threshold (DE-MEDT) optimization algorithm. Two collections of well-known constrained and unconstrained optimization problems are used to evaluate the DE-MEDT algorithm's performance. All obtained results indicate the superior performance of this algorithm compared to the considered MH optimization algorithms.

The rest of this paper is organized as follows. Section 2 provides a summarized review of the DE phenomenon in physics and metaheuristic. Section 3 presents the background inspiration and mathematical model of the DE-MEDT algorithm. The efficiency of the proposed algorithm in optimizing different benchmark test functions is evaluated in Section 4. Section 5 gives the result of the DE-MEDT to solve engineering design problems. The conclusion and potential research directions are finally presented in Section 6.

## 2 Overview of Doppler effect phenomenon in physics and metaheuristic

### 2.1 Historical background

In 1842, Christian Andreas Doppler, an Austrian mathematician, observed a phenomenon in the colored light of the binary stars and some other stars of the heavens [17]. Based on this phenomenon, the movement of the light source changes its apparent color. When a light source moves toward an observer, the light appears bluer. On the contrary, the light source appears redder when the light moves away from an observer. The observed event was known as the Doppler Effect (DE) or Doppler shift, and Doppler discovered it for the first time. In 1845, Buys Ballot tested this observation experimentally for sound waves [21]. He found out that the sound's pitch was higher than the emitted frequency when the sound source approached him, and it was lower than the emitted frequency when the sound source moved away from him. In today's world, the DE has many applications in human life. For example, the policeman can estimate the velocity of a vehicle using the DE.

DE elucidates that either the frequency of a light source or its wavelength depends on the velocity of the source relative to the observer [22]. As shown in Fig. 1, the light source movement compresses and stretches the waves in front of and back of the source, respectively.

### 2.2 Description and formulation

The primary formulation of the DE can be expressed as follows [22]:

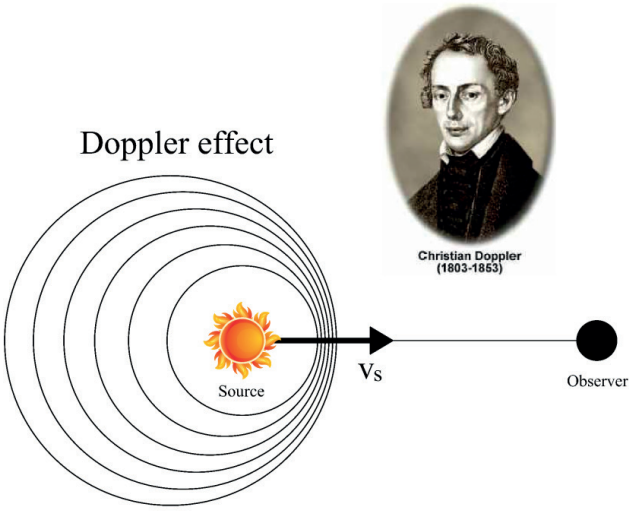



**Fig. 1** Compressing and stretching the waves in front of and back of the light source

$$f_o = f_s\left(\frac{v + v_o}{v + v_s}\right), \tag{1}$$

in which $f_o$ is the frequency perceived by the observer, $f_s$ is the frequency of the source, and $v$, $v_o$, and $v_s$ are respectively the velocity of the wave in a stationary medium, and the velocities of the observer and source with respect to this medium.

Based on the movement of the source and the observer, whether they move toward each other or move away from each other, two possible cases can be generally occurred (see Fig. 2). In the first case, observers A and B with the velocities of $V_O^A$ and $V_O^B$ move toward the source, respectively. According to this state, the perceived frequencies of observers A and B are respectively lower and higher than the emitted frequency by the source. In the latter case, when the observers C and D with the velocities of $V_O^C$ and $V_O^D$ move away from the source, observers C and D hear the sound with lower and higher frequencies than the source frequency, respectively.

The following equation states the mathematical relationship between the wavelength and frequency [23]:

$$v = f\lambda, \tag{2}$$

where $v$, $f$, and $\lambda$ respectively denote the propagation speed in the medium (m/s), the frequency (Hz), and wavelength (m). According to this equation, the frequency and wavelength are inversely proportional to each other so that an increase in frequency leads to decreasing in wavelength and vice versa [23]. Using Eq. (2), the mathematical relationship between the wavelength and frequency can be also obtained for both observer and source as follows:

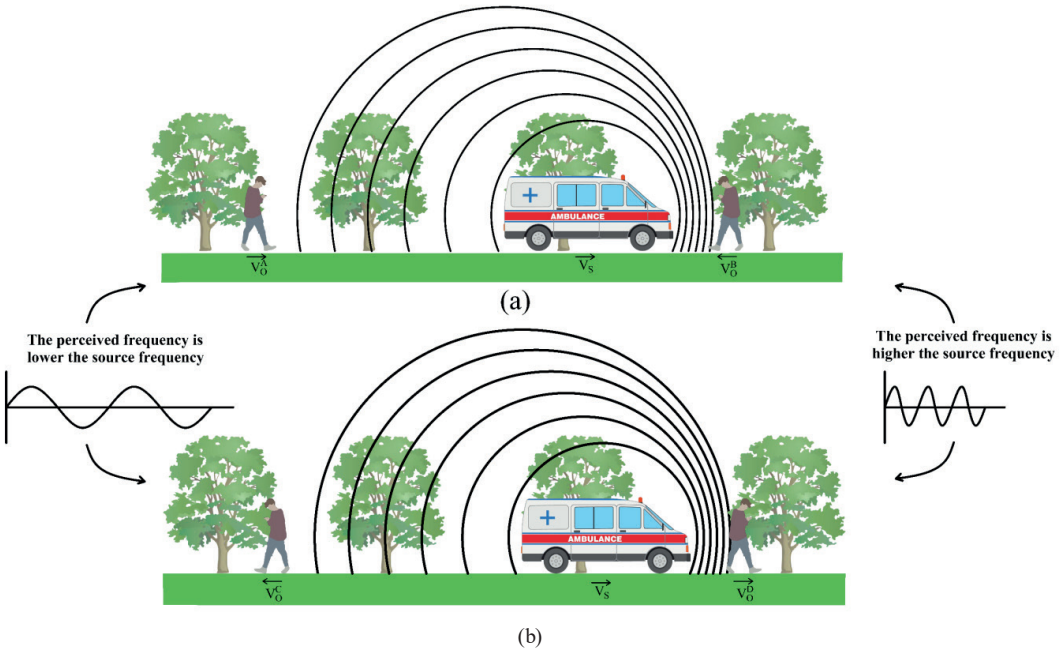



**Fig. 2** The possible movement of source and observer relative to each other

$$f_o = \frac{v}{\lambda_o}, \tag{3}$$

$$f_s = \frac{v}{\lambda_s}. \tag{4}$$

### 2.3 Overview of DE in metaheuristic

DE in metaheuristic has been only used to improve the Bat algorithm (BA). This algorithm is a population-based metaheuristic developed by Yang [6] in 2010. BA has been inspired by the echolocation behavior of bats in nature. Bats are fascinating birds. Although BA has been successfully applied to a wide range of optimization applications [24], many studies have focused on solving the basic BA's shortcomings, including local optima entrapment and unwanted premature convergence, especially when dealing with complex optimization problems [25]. One of the ways to improve the standard BA is to change the frequency equation existing in the position updating formulation of virtual bats [6]. Thus, many studies have been conducted based on changing the frequency equation of BA. Since DE has been formulated based on the changes in the frequency of the periodic event, researchers tried to incorporate DE in the frequency equation of BA. For example, Meng et al. [19] proposed Novel Bat Algorithm (NBA) and incorporated the formulation DE in the basic BA to enhance its performance. This research item or other relevant studies use the DE phenomenon to modify the frequency equation existing in the standard BA. However, the mathematical formulation of DE can be used differently and can be formulated with some idealized rules as a new optimization algorithm.

## 3 Doppler effect-mean Euclidian distance threshold algorithm

The main objective of this section is to formulate the new physics-based metaheuristic algorithm based on the DE phenomenon and a new mechanism called Mean Euclidian Distance Threshold Algorithm (MEDT). Since both these concepts are simultaneously utilized in this paper, the proposed metaheuristic is abbreviated as DE-MEDT.

### 3.1 Inspiration

In DE-MEDT algorithm, the search agents are defined as observers, and the population size is fixed equal to the number of observers in the search space. Thus, the DE-MEDT algorithm is a population-based optimizer in which each candidate solution containing a number of optimization variables is considered as an observer. The observers update their positions based on the DE formulation as discussed in Section 2 and Eqs. (1)–(4). In our implementation, we use the velocities of the observer and source to simulate an efficient search mechanism. We virtually eliminate the effect of perceived frequency by the observer and emitted frequency by the source from the DE equation. For this purpose, the propagation speed in the numerator of Eqs. (3) and (4) is respectively replaced with the velocities of $v_o$ and $v_s$:

$$f_o = \frac{v_o}{\lambda_o}, \tag{5}$$

$$f_s = \frac{v_s}{\lambda_s}. \tag{6}$$

By substituting Eqs. (5) and (6) in Eq. (1) and manipulating it, the new velocity, $v_o^{new}$, as stepsize for updating the position of the observers is calculated as follows:

$$v_o^{new} = \frac{\lambda_o}{\lambda_s} \cdot v_s \cdot \left( \frac{v + v_o}{v + v_s} \right). \tag{7}$$

### 3.2 Mathematical model of the DE-MEDT optimization algorithm

In this subsection, the mathematical model of DE-MEDT algorithm is described in detail.

#### 3.2.1 Initialization step

In the initialization step, DE-MEDT randomly initializes a set of agents within the allowed range. In this regard, *nOs* positions are randomly generated with the population size equal to *nOs*. Each member of the population called an observer, $O_i$, is a solution containing *nd* design variables for an optimization problem. Mathematically speaking, the *j*th design variable of *i*th observer in the initialization phase, $O_{i,j}^0$, is randomly generated as:

$$\begin{aligned} &O_{i,j}^0 = O_{j,min} + rand \cdot \left( O_{j,max} - O_{j,min} \right); \\ &i = 1,2,\ldots,nOs, \quad j = 1,2,..,nd, \end{aligned} \tag{8}$$

in which *rand* is a random number from a uniform distribution in the interval [0,1]. This random number is generated separately for any observer and any optimization variable; $O_{j,max}$ and $O_{j,min}$ represent the maximum and minimum permissible values of the *j*th design variable, respectively. Each randomly-generated observer is then evaluated by the objective function of the optimization problem. When all observers are evaluated, the quality vector can be obtained as follows:

$$\boldsymbol{QV}^0 = \left[ fobj\left(O_1^0\right), fobj\left(O_2^0\right), \ldots, fobj\left(O_i^0\right), \ldots, fobj\left(O_{nOs}^0\right) \right], \tag{9}$$

where $\boldsymbol{QV}^0$ is the quality vector of observers in the initialization phase, and $fobj(O_i^0)$ is the objective function value of the *i*th observer in the initialization step. If the $\boldsymbol{QV}^0$ is sorted based on the value of the objective function in ascending order, the first and last elements of the $\boldsymbol{QV}^0$ vector will become the best and worst members of the initial population, respectively.

#### 3.2.2 Updating the mean position

The cyclic body of the DE-MEDT is started from this step. In this step, the mean position of the observers is obtained. Since each member of the population has *nd* design variables, the mean value of each design variable should be first determined. To do this, the mean value of the *j*th design variable is obtained from averaging of the *nOs* observers as follows:

$$Mean_j = \frac{1}{nOs} \sum_{i=1}^{nOs} O_{i,j}, \tag{10}$$

where $Mean_j$ is the mean value of the *j*th design variable. By obtaining the mean value of all design variables (j = 1,2,…,*nd*), the mean position of the algorithm agents, $MP_{[1:nOs]}$, will be determined:

$$MP_{[1:nOs]} = \left[ Mean_1, Mean_2, \ldots, Mean_j, \ldots, Mean_{nd} \right]. \tag{11}$$

#### 3.2.3 Updating the Euclidian distance

This step deals with calculating the Euclidian distance of each observer from the mean position of the observers. Using Eq. (10), the Euclidian distance of th observer in *j*th design variable is obtained as follows:

$$ED_i = \sqrt{\sum_{j=1}^{nd} \left( O_{i,j} - Mean_j \right)^2}. \tag{12}$$

#### 3.2.4 Determining the velocities

In this step, for each observer, the velocities of the observer, source, and propagation velocity of the medium are determined. To this end, for the respective observer (i.e., $O_i$), the agent with better quality is selected randomly from the sorted population based on the quality of observers. This selected agent is called the determinative agent ($X_{det}$) for the $O_i$ and is calculated as follows:

$$\begin{aligned} &\textbf{\textit{if}}\ O_i \neq O_1 \\ &X_{det} = randi\left(O_1, O_{i-1}\right) \\ &\textbf{\textit{elseif}}\ O_i = O_1 \\ &X_{det} = O_1 \\ &\textbf{\textit{end}} \end{aligned} \tag{13}$$

in which $randi(O_1, O_{i-1})$ returns a random observer from the sorted population of $O_1$ to $O_{i-1}$. Using Eq. (13), the velocities of the observer ($v_o$) and source ($v_s$), and the propagation velocity of the medium ($v$) are obtained by the following equations:

$$v_o = X_{det} - O_i, \tag{14}$$

$$v_s = X_{det} - O_{nOs}\,, \tag{15}$$

$$v = X_{det}\,. \tag{16}$$

### 3.2.5 Determining wavelength

This step determines the wavelengths emitted by the source and received by the observer. According to Eqs. (2)–(4), the frequency of a wave is inversely proportional to its wavelength so that the waves with a low frequency have a longer wavelength and vice versa. If the wavelength perceived by an observer ($\lambda_o$) smaller than or equal to the wavelength emitted by the source ($\lambda_s$), the $\lambda_o/\lambda_s$ will become lower than or equal to 1. This result means that the frequency perceived by an observer is more than the frequency emitted by the source ($\lambda_o > \lambda_s$). Using Eq. (7), for each observer in each iteration, we replace the $\lambda_o/\lambda_s$ with a number randomly generated in the [0,1] interval as:

$$\frac{\lambda_o}{\lambda_s} = rand\,. \tag{17}$$

Thus, using Eq. (17), Eq. (7) can be rewritten as below:

$$v_o^{new} = rand \cdot v_s \cdot \left(\frac{v + v_o}{v + v_s}\right). \tag{18}$$

### 3.2.6 Calculating the position of the observer

In this step, the new position of the th observer, $O_i^{new}$, is calculated as follows:

$$O_i^{new} = O_i + v_o^{new}\,, \tag{19}$$

in which $O_i$ is the position of *i*th observer in the current iteration, and *stepsize* is obtained based on Eq. (18). Fig. 3 schematically shows how the new position of the *i*ith observer is obtained.

### 3.2.7 Mean Euclidean distance threshold

MH algorithms should be equipped with a mechanism to escape from the local optima, especially when they are close to the optimum solutions. Moreover, balancing each metaheuristic's exploration and exploitation phases is an essential issue in finding the global optimum in the solution space. Accordingly, an efficient mechanism called Mean Euclidean Distance Threshold (MEDT) is proposed here to improve the quality of solutions generated by the proposed algorithm. The proposed MEDT mechanism makes a good trade-off between the exploration and exploitation phases of the proposed DE-MEDT algorithm and causes escape from local minima. MEDT comprises two definitions. The first definition is a radius determining how much the agents

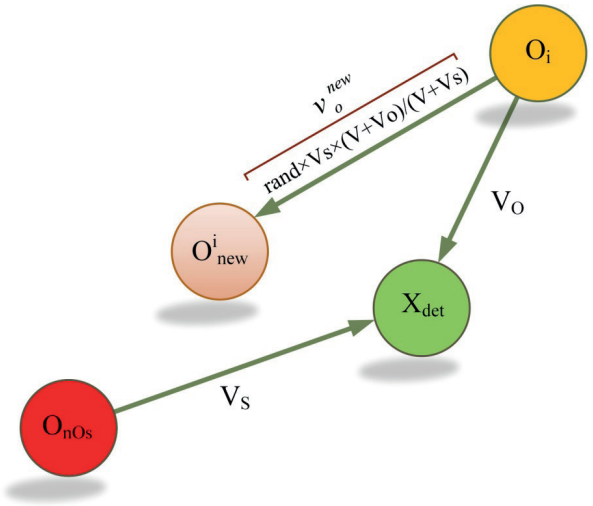



**Fig. 3** Position updating of the ith observer in DE-MEDT algorithm

are ideally close to each other in the search space of the current iteration. This radius is called the Scatter Radius Index (SRI) and is obtained by averaging all agents' Euclidian distances using Eq. (12) as follows:

$$SRI^{Iter} = \frac{1}{nOs}\sum_{i=1}^{nOs} ED_i\,, \tag{20}$$

where $SRI^{Iter}$ is the scatter radius in the current iteration, and $ED_i$ is the Euclidian distance of *i*th agent obtained by Eq. (12). If the $SRI^{Iter}$ is normalized to the search space of the optimization problem, the following equation can be obtained:

$$NSRI^{Iter} = \frac{SRI^{Iter}}{max\left(\mathrm{Var}_{[1:nd]}^{max} - \mathrm{Var}_{[1:nd]}^{min}\right)}\,, \tag{21}$$

where $\mathrm{Var}_{[1:nd]}^{max}$ and $\mathrm{Var}_{[1:nd]}^{min}$ are two vectors representing the maximum and minimum permissible values of the design variables. By defining the $NSRI^{Iter}$ in each iteration, we can ideally eliminate the effect of $SRI^{Iter}$ dependency on the search space of the optimization problem.

The second definition is a criterion that indicates the convergence of the solutions in the current iteration. This index is called the Convergence Index (*CI*). The value of $NSRI^{Iter}$ determines the value of the *CI* as follows:

$$CI = \begin{cases} \alpha \times NSRI^{Iter} & if\ NSRI^{Iter} < 1/\alpha \\ 1 & Otherwise \end{cases}\,, \tag{22}$$

in which $\alpha$ is a sensitive parameter that determines the convergence criterion of the algorithm. In this paper, $\alpha$ is fixed equal to 10 based on the sensitivity analysis performed in the next sections. The following scheme is executed to change *N*th design variable of the $O_i^{new}$ in Eq. (19) by using the proposed MEDT:

$$\begin{aligned} &\textbf{\textit{if}}\ rand < pa \times (1 - CI) \\ &N = randi(nd,1,1) \\ &O_{i,N}^{new} = O_{1,N}^{new} \cdot Uinfrnd \cdot SRI^{Iter} \\ &\textbf{\textit{end}} \end{aligned} \tag{23}$$

in which *rand* is a random number in the range of [0,1]; *CI* is the convergence index obtained by Eq. (22); *N* returns a value from the integer 1 to the number of design variables (*nd*) by *randi*(*nd*,1,1) operator; $O_{1,N}^{new}$ and $O_{i,N}^{new}$ represent the *N*th design variable of the first and *i*th newly generated observers, respectively, and *Uinfrnd* is a continuous uniform random number in the range of [–1,1]. It is worth mentioning that $O_1^{new}$ is not necessarily the best observer and is just the first newly generated observer in the current iteration. Since the convergence of the DE-MEDT algorithm to local or global optimum is unknown, the possibility of working the proposed MEDT depends on the value of $pa \times (1 - CI)$. It means that the maximum probability of occurrence of MEDT is equal to *pa* %. In this study, the value of *pa* is set equal to 0.5 according to the sensitivity analysis carried out in the following sections. As the iteration number increases, the value of $NSRI^{Iter}$ decreases. Thus, *CI* decreases and the probability of performing this mechanism based on Eq. (23) will increase. The proposed MEDT mechanism indicates the local search or intensification capability of the algorithm. Mathematically speaking, one dimension of the newly generated solution by Eq. (19) is randomly selected and intelligently changed around the best observer $O_1^{new}$ based on the value of $SRI^{Iter}$. Fig. 4 schematically indicates how the proposed mechanism works during the course of iterations.

### 3.2.8 Checking the boundary condition limitation

After generating each new solution by the DE-MEDT algorithm, the design variables of the solution should be checked to be in the permissible range. In DE-MEDT, if the *j*th design variable of the newly generated observer ($O_{i,j}^{new}$) lies out the allowed boundary, $O_{i,j}^{new}$ will be clipped on the closer boundary, whether it is upper bound or lower bound. For example, let us consider that the *j*th design variable has the lower and upper bounds equal to 0 and 1, respectively. If the DE-MEDT generates a value for the *j*th variable equal to -0.2, this value will be replaced by 0 due to being closer to the lower bound. On the contrary, if the algorithm generates a value for the *j*th design value equal to 1.1, it will be replaced by 1 due to being closer to the upper bound. Following this strategy makes that the newly generated solutions are being in the permissible range. Mathematically speaking, the following scheme is employed to check the boundary condition of the *j*th design variable:

$$O_{i,j}^{new} = max\left(O_{j,min}, O_{i,j}^{new}\right), \tag{24}$$

$$O_{i,j}^{new} = min\left(O_{j,max}, O_{i,j}^{new}\right), \tag{25}$$

where $O_{j,min}$ and $O_{j,max}$ are the lower bound and upper bound of the *j*th design variable.

### 3.2.9 Evaluating and sorting the agents

In this step, the newly generated observers are evaluated after checking the boundary condition of the design variables. Like the initialization step, each observer generated by the DE-MEDT algorithm in the previous steps is then evaluated by the objective function of the optimization problem. After evaluation of all observers in the current iteration, we can form the quality vector of the observer in the current iteration, $QV^{iter}$, as follows:

$$QV^{iter} = \begin{bmatrix} fobj\left(O_1^{new}\right), fobj\left(O_2^{new}\right), \ldots, \\ fobj\left(O_i^{new}\right), \ldots, fobj\left(O_{nOs}^{new}\right) \end{bmatrix}. \tag{26}$$

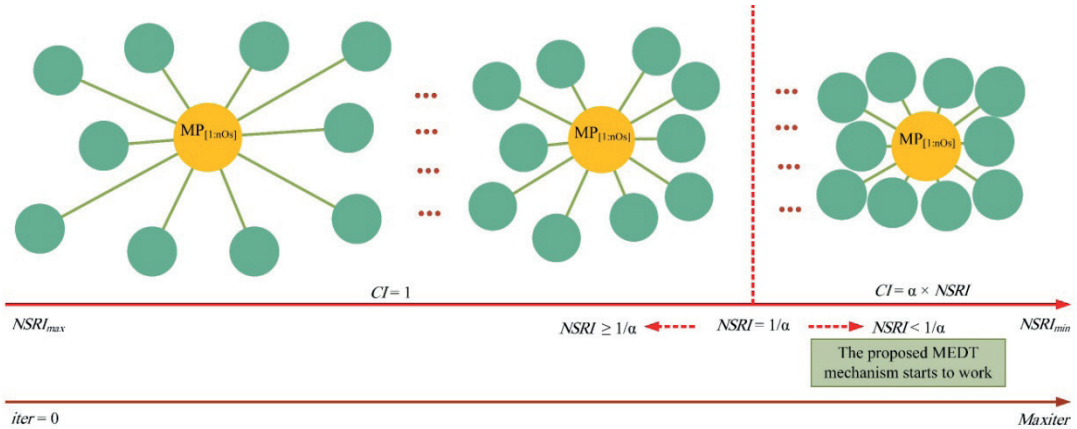



**Fig. 4** Schematic demonstration of working the proposed MEDT during the optimization process

Then, the greedy strategy between the population of observers in the current iteration and those in the previous iteration is carried out. Based on this strategy, the newly generated observers and those created in the previous iteration are merged. Then, the best *nOs* observers with better objective function values than other observers are selected from the merged population. The selected observers are considered as the current population, and they will contribute in the next iteration for the same selection. Applying this strategy makes the proposed DE-MEDT algorithm consider the higher quality agents by comparing the current and previous iteration solutions. In this regard, this technique can help the intensification ability of the DE-MEDT. It should be noted that in the first iteration of the algorithm, the previous population is generated in the initialization step. Then, the population of the initialization phase is merged with the current population to select the best *nOs* agents.

#### 3.2.10 Checking termination criterion of the DE-MEDT algorithm

As the last step of the proposed algorithm, the termination criterion of the DE-MEDT algorithm is checked. The algorithm terminates and reports the best solution if the termination criterion is satisfied. Otherwise, if the termination condition is not met, the algorithm goes to Step 2 (Section 3.2.2.) for a new loop of the DE-MEDT. Like other population-based optimizers, two common termination conditions can be considered as stopping criteria of the DE-MEDT: the maximum number of iterations (*MaxIter*) and the maximum number of function evaluations (*MaxNFEs*).

### 3.3 Pseudo-code and flowchart of DE-MEDT algorithm

The pseudo-code and flowchart of the proposed DE-MEDT are presented in Algorithm 1 and Fig. 5, respectively. As it can be seen, the proposed algorithm can be easily implemented in programming languages.

### 3.4 Computational complexity of the DE-MEDT algorithm

The computational complexity is a key metric for evaluating the run time of an algorithm taken to execute. For calculating the computational complexity of the DE-MEDT algorithm, four main factors are considered: the initialization process, objective function, sorting, and position updating of the observers. In the initialization process, the $N$ observers are randomly initialized. In this regard, the computational complexity of the initialization process is obtained equal to $O(N)$. The second process deals with the computational complexity of the objective function. Since it depends on the optimization problem, we do

**Algorithm 1** Pseudo-code of the DE-MEDT algorithm

Initialize the DE-MEDT algorithm parameters: *pa*, *nOs*, and *MaxIter*.
Initialize the observer's positions randomly using Eq.(8).
Evaluate observers, sort them, and form the initial quality vector using Eq. (9).
**while** (*Iter* < *MaxIter*) **do**
  Calculate the mean position of the observers ($MP_{[1:nOs]}$) using Eq. (10)–(11).
  Calculate the Euclidean distance of each observer from the mean position of the observers ($ED_i$) using Eq. (12).
  **for** (each observer, $O_i$) **do**
    Determine the determinative agent, $X_{det}$, from the sorted population of the observers using Eq.(13).
    Calculate the velocities of the observer and source, and propagation velocity of the medium using Eq. (14)–(16).
    Determine the wavelength emitted by the source and received by the observer using Eq. (17).
    Calculate the *i*th observer position using Eq. (19).
    Determine scatter radius index of the current iteration ($SRI^{Iter}$) using Eq. (20).
    Normalize the $SRI^{Iter}$ to the search space of the optimization problem using Eq. (21).
    Determine the convergence index (*CI*) using Eq. (22).
    **if** *rand* < *pa* × (1 – *CI*) **then**
      Update the th observer position by the proposed MEDT mechanism using Eq. (23).
    **end if**
    Check the boundary condition limitation of the *i*th observer using Eq. (24)–(25).
    Evaluate the new position of the *i*th observer using the objective function of the optimization problem.
  **end for**
  Form the quality vector for the newly generated observers using Eq. (26).
  Apply greedy strategy between the observers generated in the current iteration and those generated in the previous iteration.
  *Iter* = *Iter* + 1
**end while**
Report the best observer found by the DE-MEDT.

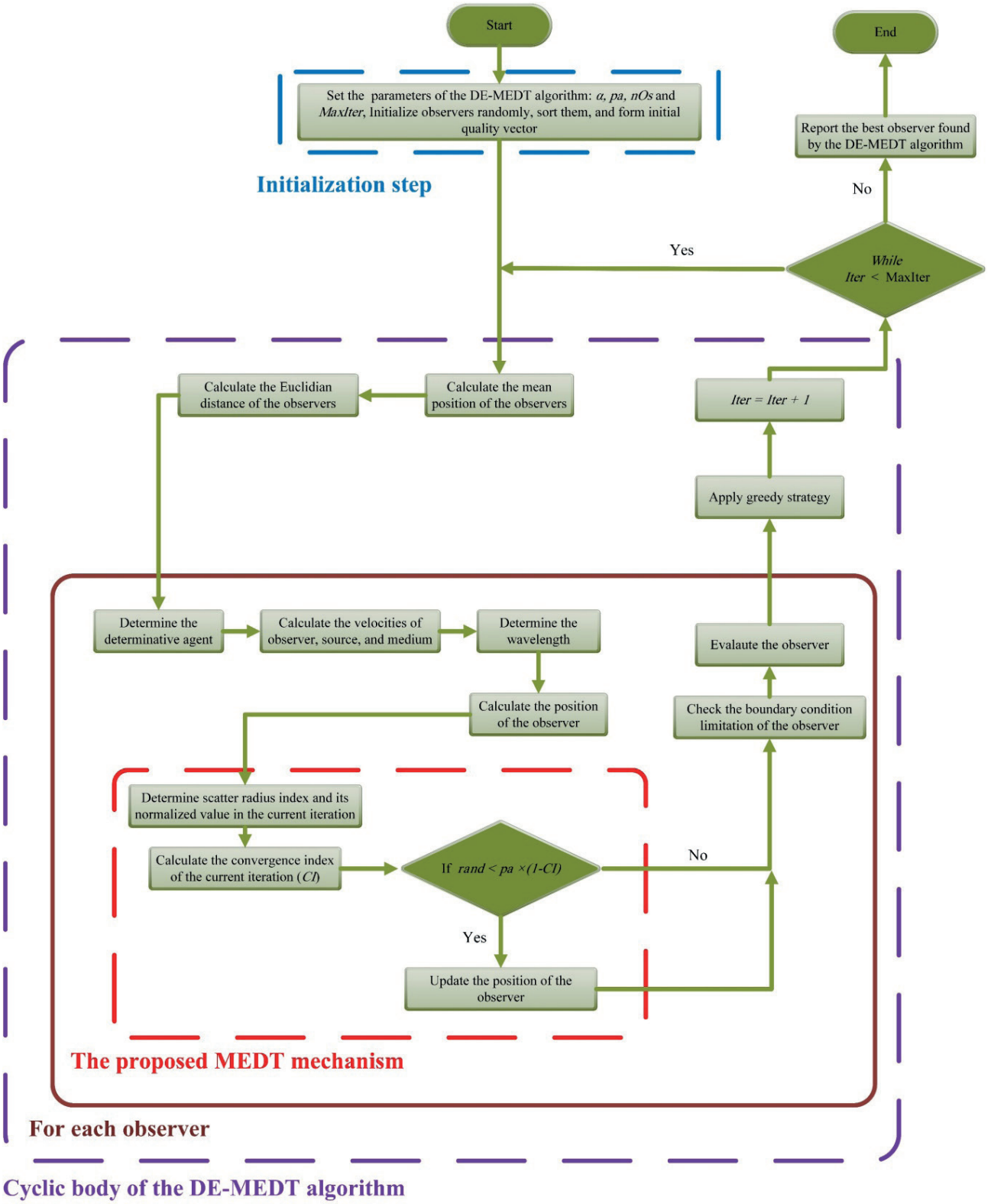



**Fig. 5** The flowchart of the DE-MEDT algorithm

not explain this complexity here. The computational complexity of sorting is $O(NlogN)$. The last factor is related to calculating the computational complexity of the observers' positions updated iteratively in the main loop of the DE-MEDT. This complexity is equal to $O(M \times N)$, where $M$ indicates the iterations. From these calculations, we can get the complexity of the whole algorithm as follows: $O\left(N \times \left(1 + M + logN + MlogN\right)\right)$.

## 4 Mathematical benchmark functions

In this section, the performance of the proposed DE-MEDT algorithm is examined through a set of 23 commonly used unimodal and multimodal benchmark functions. These benchmarks include three families of mathematical functions: unimodal functions ($F_1$–$F_7$), multimodal functions ($F_8$–$F_{13}$), and fixed dimension multimodal functions ($F_{14}$–$F_{23}$). Table 1 gives the mathematical description of these commonly used benchmark functions. In this table, *Dim* represents the dimension of the functions, *Range* is the definition domain of the function, and $f_{min}$ refers to the optimal value of the function.

**Table 1** Description of 23 commonly used mathematical benchmark functions

| Function ID | Function Equation | *Dim* | *Range* | $f_{min}$ |
|---|---|---|---|---|
| | Unimodal functions | | | |
| F1 | $f_1(x)=\sum_{i=1}^{n}x_i^2$ | 30 | [–100,100] | 0 |
| F2 | $f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$ | 30 | [–10,10] | 0 |
| F3 | $f_3(x)=\sum_{i=1}^{n}\left(\sum_{j-1}^{i}x_j\right)^2$ | 30 | [–100,100] | 0 |
| F4 | $f_4(x)=max_i\{\lvert x_i\rvert,1\le i\le n\}$ | 30 | [–100,100] | 0 |
| F5 | $f_5(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$ | 30 | [–30,30] | 0 |
| F6 | $f_6(x)=\sum_{i=1}^{n}\left(\lfloor x_i+0.5\rfloor\right)^2$ | 30 | [–100,100] | 0 |
| F7 | $f_7(x)=\sum_{i=1}^{n}ix_i^4+random[0,1)$ | 30 | [–1.28,1.28] | 0 |
| | Multimodal functions | | | |
| F8 | $f_8(x)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$ | 30 | [–500,500] | |
| F9 | $f_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$ | 30 | [–5.12,5.12] | 0 |
| F10 | $f_{10}(x)=-20exp\left\{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i}\right\}-exp\left\{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right\}+20+e$ | 30 | [–32,32] | 0 |
| F11 | $f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$ | 30 | [–600,600] | 0 |
| F12 | $f_{12}(x)=\frac{\pi}{n}\left\{10\sin(ay_1)+\sum_{i=1}^{n-1}(y_i-1)^2\left[1+10sin^2(\pi y_{i+1})\right]+(y_n-1)^2+\sum_{i=1}^{n}\mu(x_i,10,100,4)\right\}$ | 30 | [–50,50] | 0 |
| F13 | $f_{13}(x)=0.1\left\{\begin{array}{l}sin^2(3\pi x_i)+\sum_{i=1}^{n}(x_i-1)^2\left[1+sin^2(3\pi x_i+1)\right]+\\(x_n-1)^2\left[1+sin^2(2\pi x_n)\right]+\sum_{i=1}^{n}\mu(x_i,5,100,4)\end{array}\right\}$ | 30 | [–50,50] | 0 |

**Continuation of Table 1**

| Function ID | Function Equation | *Dim* | *Range* | $f_{min}$ |
|---|---|---|---|---|
| | Fixed-dimension multimodal functions | | | |
| F14 | $f_{14}(x)=\left(\frac{1}{5000}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i+a_{ij}\right)^6}\right)^{-1}$ | 2 | | 1 |
| F15 | $f_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1\left(b_i^2+b_i x_2\right)}{b_i^2+b_i x_3+x_4}\right]^2$ | 4 | | 0.00030 |
| F16 | $f_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$ | 2 | | -1.0316 |
| F17 | $f_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$ | 2 | | 0.398 |
| F18 | $f_{18}(x)=\left[1+\left(x_1+x_2+1\right)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]$ $\times\left[30+\left(2x_1-3x_2\right)^2\times\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$ | 2 | | 3 |
| F19 | $f_{19}(x)=-\sum_{i=1}^{4}c_i exp\left(-\sum_{j=1}^{3}a_{ij}\left(x_j-p_{ij}\right)^2\right)$ | 3 | | -3.86 |
| F20 | $f_{20}(x)=-\sum_{i=1}^{4}c_i exp\left(-\sum_{j=1}^{6}a_{ij}\left(x_j-p_{ij}\right)^2\right)$ | 6 | | -3.32 |
| F21 | $f_{21}(x)=-\sum_{i=1}^{5}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$ | 4 | | -10.1532 |
| F22 | $f_{21}(x)=-\sum_{i=1}^{7}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$ | 4 | | -10.4028 |
| F23 | $f_{21}(x)=-\sum_{i=1}^{10}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$ | 4 | | -10.5363 |

### 4.1 Qualitative results of the DE-MEDT algorithm

Fig. 6 shows the qualitative results of some commonly used mathematical benchmark functions using the proposed DE-MEDT algorithm. The qualitative results shown in this figure include search history, trajectory of the first individual, the average fitness of all agents, and convergence curve.

The search history displays the position of all agents from the first to the last iteration. Trajectory of the first individual shows how the position of the first dimension in the respective function is changed during the optimization task. The average fitness of all agents is obtained by averaging the objective function values of all search agents in each iteration. It indicates how the average fitness of all algorithm individuals is changed in the whole optimization process. The convergence history shows the relationship between the values of the objective function and the number of iterations. Moreover, it reveals the trend of the algorithm from explorative behavior to exploitative behavior.

From observing the history position of algorithm individuals, first, we can see that individuals of the DE-MEDT explore a significant part of the search space. This observation reveals that the proposed algorithm has a strong capability of searching and indicates that the DE-MEDT algorithm can avoid falling into a local optimum. On the other hand, we can see that the algorithm can simultaneously find most of the positions around the neighborhood of the optimum solution. Both of these observations prove a good balance between the exploration and exploitation tendencies of the algorithm.

A close examination of the trajectory of the first individual demonstrates the sudden fluctuation of the position in the initial stages of the search process. As it can be seen, the range of this fluctuation coverages about 100% of the solution space. This behavior shows the exploration tendency of the DE-MEDT. As the iteration number increases, the fluctuation converges to a value and tends to be more stable. It means that the search mechanism of the algorithm is changed from the exploration to the exploitation phase. In some functions, such as F7 and F8, the fluctuation tends to converge and then diverge. This observation means the DE-MEDT algorithm can jump out of a local minima entrapment.

By examining the average of all fitness from Fig. 6, we can understand that the proposed algorithm tends to converge so fast by monitoring the average of all individuals. When the number of iterations increases, the downward trend slows down and goes with variations. However, the overall average gradually decreases, indicating the high searching abilities of the DE-MEDT.

The last qualitative result examined in the DE-MEDT algorithm is the convergence curve. This curve is usually employed to assess the convergence performance of algorithms. Fig. 6 shows the convergence curves of the DE-MEDT in different mathematical benchmark functions. As seen from these curves, the proposed method has an accelerated reducing pattern, especially in the early stage of the optimization task.

### 4.2 Comparison of DE-MEDT algorithm with other optimizers

In this section, the exploration and exploitation tendencies of the DE-MEDT algorithm are evaluated using a set of 23 well-known unimodal, multimodal, and fixed-dimension multimodal benchmark functions. Typically, unimodal functions ($F_1$–$F_7$) are used to evaluate the exploitation capability of the algorithm due to having only one global best solution. In contrast, the multimodal functions ($F_8$–$F_{13}$) and fixed-dimension multimodal function ($F_{14}$–$F_{23}$) are suitable for assessing the exploratory behavior of the algorithm.

Table 2 compares the statistical results (i.e., average (AVG) and standard deviation (SD) values) of the proposed DE-MEDT algorithm with other well-established and advanced metaheuristic algorithms, including Bat Algorithm (BA), a hybrid algorithm (PSOGSA) with the combination of Particle Swarm Optimization (PSO) and Gravitational Search Algorithm (GSA) [26], Novel Bat Algorithm (NBA) [19], High Exploration PSO (HEPSO) [27], a hybrid metaheuristic based on Firefly and PSO algorithms (HFPSO) [28], and Fractional Lévy flight Bat Algorithm (FLFBA) [29]. For a fair comparison, in all algorithms, the value of the Maximum Number of Function Evaluations (*MaxNFEs*) is considered equal to 5,000 × *Dim*. To achieve statistically meaningful results for comparison, each algorithm is run 30 times independently with a population size equal to 30. The algorithm-specific parameters of the involved metaheuristics are set based on the recommendation of their source paper. For the proposed DE-MEDT, the values of $\alpha$ and *pa* are taken 10 and 0.5, respectively according to the sensivity analysis carried out in Section 4.3.

According to the statistical results reported in Table 2, the DE-MEDT found the minimum average in 18 functions. After the DE-MEDT, NBA and HFPSO obtained the minimum average in 5 and 4 functions, respectively. A close examination of the results for unimodal functions indicates that the DE-MEDT stands in the first place, reflecting the good exploitation behavior of the proposed DE-MEDT algorithm. For multimodal functions, the comparison of DE-MEDT with six other MH algorithms revealed that DE-MEDT is the best optimizer and provides competitive results. Thus, the results acquired for multimodal functions illustrated an excellent exploration ability of the DE-MEDT algorithm.

### 4.3 Parameter sensitivity analysis of DE-MEDT algorithm

This section analyzes the sensitivity of the algorithm-specific parameters of the proposed algorithm, which are $\alpha$ and *pa*. This analysis is performed to determine which values of the parameters $\alpha$ and *pa* can give better results. For this purpose, three test functions from each collection of the unimodal and multimodal benchmark functions (i.e., F1, F5, F6, F8, F10, and F12) are taken for sensitivity analysis of these parameters. In this regard, the values of each parameter are defined as follows: $\alpha = [5, 10, 50, 100]$ and $pa = [0, 0.25, 0.5, 0.75, 1]$. Since $\alpha$ and *pa*, respectively, have 4 and 5 values, there are 20 combinations of the design. For each design, the *MaxNFEs* and *nOs* are fixed equal to 5,000 × *Dim* and 30, respectively. Furthermore, each design is evaluated by averaging the objective function value acquired from 30 independent runs. Table 3 illustrates the average results of these functions at each designed combination. From the obtained results, it can be observed that the values of $\alpha$ and *pa*, respectively, equal to 10 and 0.5, reports better results due to achieving the first rank among the other combinations.

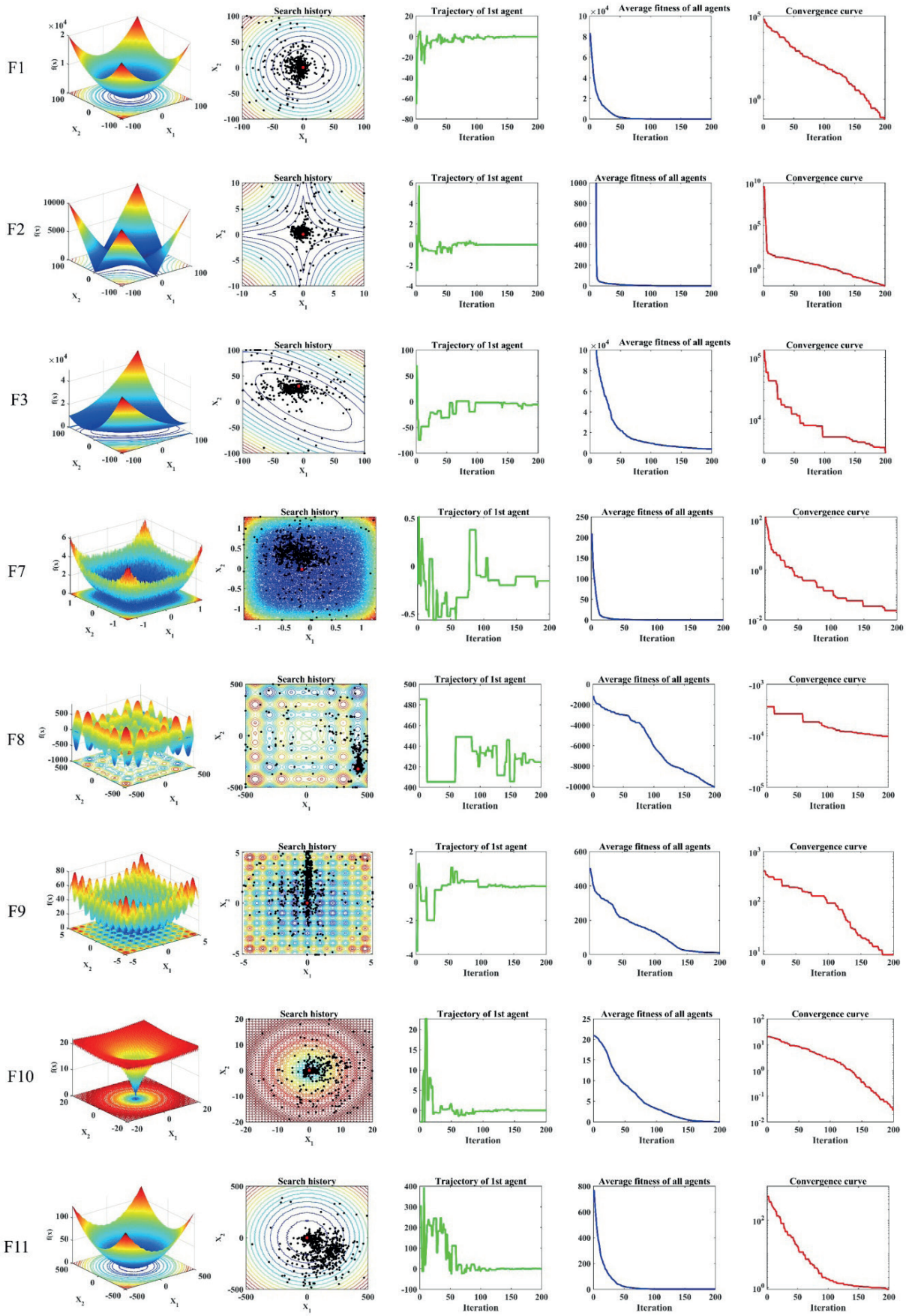



**Fig. 6** Qualitative results for eight commonly used benchmark functions

**Table 2** Comparison of the results from DE-MEDT and other optimization methods for the 23 commonly used mathematical benchmark functions

| | | BA | PSOGSA | NBA | HEPSO | HFPSO | FLFBA | DE-MEDT |
|---|---|---|---|---|---|---|---|---|
| F1 | AVG | 3.29E-03 | 1.67E+03 | 9.33E-62 | 1.00E-03 | 7.73E-23 | 2.60E-16 | **1.63E-148** |
| | SD | 3.84E-03 | 3.79E+03 | 5.10E-61 | 8.65E-04 | 6.87E-23 | 6.65E-16 | 4.94E-148 |
| F2 | AVG | 3.51E+02 | 2.00E+00 | 2.16E-36 | 2.11E-04 | 3.73E-12 | 3.63E-11 | **1.00E-92** |
| | SD | 1.67E+03 | 4.07E+00 | 1.14E-35 | 1.11E-03 | 2.49E-12 | 7.15E-11 | 1.83E-92 |
| F3 | AVG | 4.08E-03 | 8.06E+03 | **1.73E-35** | 3.88E+03 | 1.59E-06 | 7.54E-07 | 5.88E-05 |
| | SD | 3.96E-03 | 8.43E+03 | 9.35E-35 | 1.28E+03 | 1.43E-06 | 2.55E-06 | 1.69E-04 |
| F4 | AVG | 3.42E+01 | 4.24E+01 | 2.80E-01 | 1.46E+01 | 2.56E-05 | 2.29E-01 | **6.40E-28** |
| | SD | 1.14E+01 | 3.86E+01 | 1.50E+00 | 4.16E+00 | 1.60E-05 | 2.43E-01 | 2.44E-27 |
| F5 | AVG | 7.48E+01 | 4.87E+01 | **2.00E+01** | 6.85E+01 | 2.71E+01 | 1.76E+01 | 2.09E+01 |
| | SD | 1.32E+02 | 1.02E+02 | 1.68E+00 | 2.72E+01 | 2.31E+01 | 2.82E+00 | 1.12E+00 |
| F6 | AVG | 3.02E-03 | 2.33E+03 | 1.48E-01 | 1.05E-03 | 1.64E-22 | 1.65E-13 | **2.77E-32** |
| | SD | 2.27E-03 | 5.04E+03 | 2.28E-01 | 8.05E-04 | 1.64E-22 | 3.04E-13 | 5.97E-32 |
| F7 | AVG | 1.50E-02 | 1.61E-02 | 1.19E-03 | **2.80E-04** | 2.77E-03 | 2.77E-02 | 2.21E-03 |
| | SD | 9.11E-03 | 6.79E-03 | 1.03E-03 | 2.53E-04 | 1.11E-03 | 8.79E-03 | 9.55E-04 |
| F8 | AVG | -7.34E+03 | -7.40E+03 | -7.49E+03 | -9.30E+03 | -7.43E+03 | -9.63E+03 | **-1.26E+04** |
| | SD | 7.88E+02 | 8.06E+02 | 2.31E+03 | 2.61E+02 | 8.52E+02 | 6.45E+02 | 3.68E-12 |
| F9 | AVG | 1.96E+02 | 1.22E+02 | 3.64E+01 | 7.52E+00 | 4.53E+01 | 1.44E+01 | **0.00E+00** |
| | SD | 1.02E+02 | 3.83E+01 | 3.47E+01 | 2.09E+00 | 2.04E+01 | 4.34E+00 | 0.00E+00 |
| F10 | AVG | 1.89E+01 | 5.19E+00 | **4.91E-15** | 2.07E-01 | 5.77E-12 | 2.68E+00 | 5.51E-15 |
| | SD | 1.16E+00 | 8.12E+00 | 1.23E-15 | 1.27E-01 | 2.96E-12 | 4.01E+00 | 1.66E-15 |
| F11 | AVG | 7.19E+01 | 2.11E+01 | 6.64E-03 | 1.14E-02 | 1.39E-02 | 1.43E-02 | **0.00E+00** |
| | SD | 6.10E+01 | 3.88E+01 | 2.25E-02 | 2.25E-02 | 1.53E-02 | 1.87E-02 | 0.00E+00 |
| F12 | AVG | 2.39E+01 | 8.53E+06 | 8.64E-02 | 1.13E-05 | 3.46E-03 | 4.49E-02 | **1.60E-32** |
| | SD | 1.06E+01 | 4.67E+07 | 4.21E-01 | 3.23E-05 | 1.89E-02 | 7.55E-02 | 1.83E-34 |
| F13 | AVG | 6.02E+01 | 1.86E-20 | 8.15E-02 | 4.57E-04 | 1.04E-21 | 2.43E-02 | **1.69E-32** |
| | SD | 1.50E+01 | 3.15E-21 | 1.13E-01 | 5.18E-04 | 5.46E-21 | 6.10E-02 | 2.07E-33 |
| F14 | AVG | 6.89E+00 | 5.14E+00 | 7.08E+00 | **9.98E-01** | 2.71E+00 | 1.13E+01 | 1.10E+00 |
| | SD | 6.37E+00 | 4.55E+00 | 5.00E+00 | 7.31E-06 | 1.98E+00 | 7.18E+00 | 3.03E-01 |
| F15 | AVG | 6.78E-03 | 6.30E-03 | 2.42E-03 | 8.76E-04 | 2.14E-03 | 5.75E-03 | **3.07E-04** |
| | SD | 1.23E-02 | 8.58E-03 | 6.09E-03 | 3.70E-04 | 4.99E-03 | 1.22E-02 | 1.12E-18 |
| F16 | AVG | -1.03E+00 | **-1.03E+00** | **-1.03E+00** | -1.03E+00 | **-1.03E+00** | -9.77E-01 | **-1.03E+00** |
| | SD | 4.90E-04 | 5.76E-16 | 4.81E-16 | 4.55E-07 | 4.70E-16 | 2.07E-01 | 6.78E-16 |
| F17 | AVG | 3.98E-01 | **3.98E-01** | 3.98E-01 | 3.98E-01 | **3.98E-01** | 3.98E-01 | **3.98E-01** |
| | SD | 1.83E-04 | 0.00E+00 | 6.14E-16 | 3.53E-05 | 0.00E+00 | 2.43E-15 | 0.00E+00 |
| F18 | AVG | 5.73E+00 | **3.00E+00** | 5.70E+00 | 3.04E+00 | **3.00E+00** | 3.90E+00 | **3.00E+00** |
| | SD | 1.48E+01 | 2.59E-15 | 1.48E+01 | 7.33E-02 | 2.60E-15 | 4.93E+00 | 1.33E-15 |
| F19 | AVG | -3.85E+00 | **-3.86E+00** | **-3.86E+00** | -3.86E+00 | **-3.86E+00** | -3.84E+00 | **-3.86E+00** |
| | SD | 8.00E-03 | 2.48E-15 | 2.15E-15 | 6.99E-07 | 2.41E-15 | 1.41E-01 | 2.71E-15 |
| F20 | AVG | -3.04E+00 | -3.26E+00 | -3.28E+00 | -3.29E+00 | -3.28E+00 | -3.27E+00 | **-3.30E+00** |
| | SD | 8.95E-02 | 6.72E-02 | 5.83E-02 | 5.07E-02 | 5.83E-02 | 5.99E-02 | 4.39E-02 |
| F21 | AVG | -4.45E+00 | -5.15E+00 | -7.22E+00 | -8.40E+00 | -6.39E+00 | -5.33E+00 | **-9.14E+00** |
| | SD | 2.61E+00 | 3.21E+00 | 3.30E+00 | 2.77E+00 | 3.27E+00 | 3.52E+00 | 1.18E+00 |
| F22 | AVG | -6.02E+00 | -6.97E+00 | -8.87E+00 | -9.07E+00 | -7.31E+00 | -4.01E+00 | **-1.04E+01** |
| | SD | 3.23E+00 | 3.36E+00 | 2.62E+00 | 2.56E+00 | 3.29E+00 | 2.01E+00 | 8.73E-16 |
| F23 | AVG | -4.58E+00 | -6.60E+00 | -7.38E+00 | -9.12E+00 | -7.98E+00 | -3.74E+00 | **-1.05E+01** |
| | SD | 3.24E+00 | 3.80E+00 | 3.58E+00 | 2.75E+00 | 3.50E+00 | 2.48E+00 | 1.81E-15 |

**Table 3** Sensitivity analysis for the control parameters of DE-MEDT optimizer with different test functions.

| $\alpha$ | $pa$ | F1 | | F5 | | F6 | | F8 | | F10 | | F12 | | Mean Rank | Final Ranking |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | AVG | Rank | AVG | Rank | AVG | Rank | AVG | Rank | AVG | Rank | AVG | Rank | AVG | Rank |
| | 0 | 2.58E-108 | 19 | 2.62E+01 | 12 | 2.22E-30 | 12 | -1.01E+04 | 18 | 7.16E-15 | 16 | 3.46E-02 | 20 | 16.17 | 19 |
| | 0.25 | 2.98E-128 | 16 | 3.20E+01 | 18 | 2.13E-32 | 1 | -1.20E+04 | 16 | 6.57E-15 | 14 | 1.81E-32 | 5 | 11.67 | 15 |
| 5 | 0.5 | 2.43E-146 | 11 | 3.43E+01 | 20 | 3.55E-32 | 4 | -1.25E+04 | 8 | 5.86E-15 | 11 | 1.68E-32 | 4 | 9.67 | 10 |
| | 0.75 | 1.62E-164 | 8 | 3.03E+01 | 16 | 3.41E-20 | 15 | -1.26E+04 | 2 | 5.15E-15 | 4 | 7.99E-22 | 10 | 9.17 | 8 |
| | 1 | 5.40E-179 | 4 | 2.28E+01 | 3 | 1.19E-01 | 20 | -1.26E+04 | 6 | 5.03E-15 | 2 | 8.87E-04 | 15 | 8.33 | 4 |
| | 0 | 2.98E-110 | 18 | 2.36E+01 | 5 | 5.87E-32 | 8 | -1.01E+04 | 17 | 7.64E-15 | 19 | 1.04E-02 | 17 | 14.00 | 17 |
| | 0.25 | 1.28E-128 | 15 | 2.06E+01 | 1 | 3.71E-32 | 5 | -1.23E+04 | 12 | 6.45E-15 | 12 | 1.63E-32 | 2 | 7.83 | 2 |
| 10 | 0.5 | 1.63E-148 | 9 | 2.09E+01 | 2 | 2.77E-32 | 2 | -1.26E+04 | 1 | 5.51E-15 | 6 | 1.60E-32 | 1 | 3.50 | 1 |
| | 0.75 | 4.68E-165 | 5 | 2.75E+01 | 13 | 2.98E-20 | 14 | -1.26E+04 | 3 | 5.39E-15 | 5 | 3.84E-22 | 9 | 8.17 | 3 |
| | 1 | 3.07E-180 | 2 | 2.59E+01 | 11 | 8.94E-02 | 19 | -1.26E+04 | 5 | 5.03E-15 | 3 | 6.94E-04 | 14 | 9.00 | 7 |
| | 0 | 1.42E-111 | 17 | 2.29E+01 | 4 | 5.67E-32 | 7 | -9.86E+03 | 20 | 7.40E-15 | 18 | 2.42E-02 | 19 | 14.17 | 18 |
| | 0.25 | 2.46E-130 | 13 | 2.37E+01 | 6 | 5.51E-32 | 6 | -1.22E+04 | 14 | 6.57E-15 | 15 | 1.67E-32 | 3 | 9.50 | 9 |
| 50 | 0.5 | 3.47E-148 | 10 | 2.37E+01 | 7 | 1.02E-31 | 9 | -1.25E+04 | 10 | 5.74E-15 | 9 | 1.93E-32 | 6 | 8.50 | 6 |
| | 0.75 | 5.60E-165 | 6 | 3.23E+01 | 19 | 3.72E-20 | 16 | -1.26E+04 | 7 | 5.63E-15 | 7 | 1.11E-22 | 8 | 10.50 | 12 |
| | 1 | 2.06E-179 | 3 | 2.83E+01 | 15 | 1.47E-02 | 18 | -1.23E+04 | 13 | 4.68E-15 | 1 | 1.86E-05 | 13 | 10.50 | 13 |
| | 0 | 7.13E-107 | 20 | 2.78E+01 | 14 | 6.74E-31 | 11 | -1.00E+04 | 19 | 7.64E-15 | 20 | 1.38E-02 | 18 | 17.00 | 20 |
| | 0.25 | 7.27E-129 | 14 | 2.45E+01 | 10 | 2.96E-32 | 3 | -1.22E+04 | 15 | 7.28E-15 | 17 | 3.46E-03 | 16 | 12.50 | 16 |
| 100 | 0.5 | 3.98E-146 | 12 | 2.43E+01 | 9 | 1.38E-31 | 10 | -1.25E+04 | 11 | 6.45E-15 | 13 | 6.84E-32 | 7 | 10.33 | 11 |
| | 0.75 | 1.36E-164 | 7 | 3.04E+01 | 17 | 9.00E-21 | 13 | -1.25E+04 | 9 | 5.74E-15 | 10 | 1.24E-21 | 11 | 11.17 | 14 |
| | 1 | 1.17E-180 | 1 | 2.41E+01 | 8 | 3.29E-03 | 17 | -1.26E+04 | 4 | 5.63E-15 | 8 | 5.01E-06 | 12 | 8.33 | 5 |

## 5 Engineering design problems

In many real-world problems, it is well-known that there are many restrictions. One main difference between global benchmark cases, examined in the previous section, and engineering design problems is considering constraints for design variables and their effects on finding the optimum value for their objective functions. For this purpose, DE-MEDT algorithm is further evaluated through three constrained engineering design problems. The problems include a 72-bar spatial truss, a 200-bar planar truss, and a 3-bay 24-story frame. The optimization results acquired by DE-MEDT are compared with those of other well-known optimizers presented in the literature. Like the previous section, each engineering problem is implemented 30 times independently to achieve statistically meaningful results. Moreover, the number of observers is considered equal to 30 for all examined problems.

### 5.1 A 72-bar spatial truss design problem

The first engineering design problem is a 72-bar spatial truss structure, one of the most well-known problems in the area of structural optimization. The schematic representation of the truss structure is shown in Fig. 7. This figure also exhibits the total number of structure nodes and

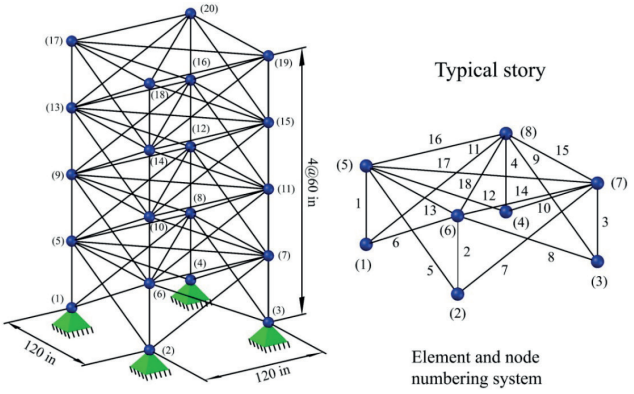



**Fig. 7** The 72-bar spatial truss design problem

the numbering of nodes and elements in each typical story. The 72 members are categorized into 16 element groups using structural symmetry; hence, this problem includes 16 sizing variables. The elasticity modulus and material density are $E$ = 10,000 ksi and $\rho$ = 0.1 lb/in$^3$, respectively. All nodes in three directions are subjected to the displacement limitation of ± 0.25 in. The cross-sectional area of structural elements can vary between 0.1 in.$^2$ and 3.0 in.$^2$. The truss structure must be designed for two independent load cases, as listed in Table 4.

**Table 4** Load cases for the 72-bar spatial truss design problem

| Case | Node | $F_x$ (kips) | $F_y$ (kips) | $F_z$ (kips) |
|---|---|---|---|---|
| 1 | 17 | 5.0 | 5.0 | -5.0 |
| 2 | 17 | 0.0 | 0.0 | -5.0 |
| | 18 | 0.0 | 0.0 | -5.0 |
| | 19 | 0.0 | 0.0 | -5.0 |
| | 20 | 0.0 | 0.0 | -5.0 |

Table 5 [30–34] presents the optimization results gained by the proposed DE-MEDT algorithm in comparison with previous studies conducted by other researchers. It can be seen that DE-MEDT acquires the best weight (379.62 lb) and the best average weight (379.68 lb) among all considered algorithms. Fig. 8 provides the constrained boundaries of the problem evaluated at the best optimum design by the DE-MEDT algorithm. It can be seen that all design constraints have been satisfied.

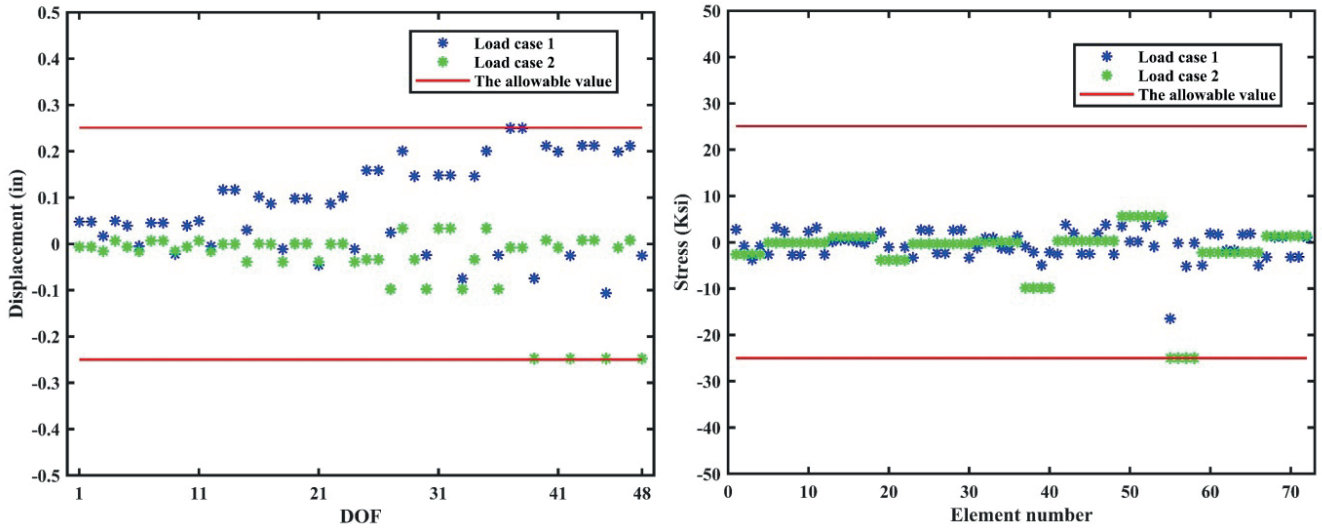
### 5.2 A 200-bar planar truss design problem

The second engineering design problem is a 200-bar planar truss, one of the challenging benchmark problems in size optimization of truss structures. Fig. 9 shows the schematic view of this truss structure. Due to structural symmetry, the 200 bars of the truss are categorized into 29 element groups; hence, this problem includes 29 sizing variables. Information about which members belong to which element groups is given in Fig. 9. The elasticity modulus and material density are $E$ = 30,000 ksi and $\rho$ = 0.283 lb/in$^3$, respectively. Although no displacement constraints are considered for the problem, it is designed optimally subjected to three independent loading conditions as follows: (1) 1.0 kip applied in the positive x-direction at nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, and 71; (2) 10.0 kips applied in the negative Y-direction at nodes 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 26, 28, 29, 30, 31, 32, 33, 34, 36, 38, 40, 42, 43, 44, 45, 46, 47, 48, 50, 52, 54, 56, 57, 58, 59, 60, 61, 62, 64, 66, 68, 70, 71, 72, 73, 74 and 75; (3) loading conditions (1) and (2) applied together. The lower bound of cross-sectional area in structure elements is 0.1 in.$^2$.

Table 6 [31, 32, 34–37]compares the optimization results achieved by the DE-MEDT optimization method and those found by other techniques like SAHS, TLBO, ADEA,

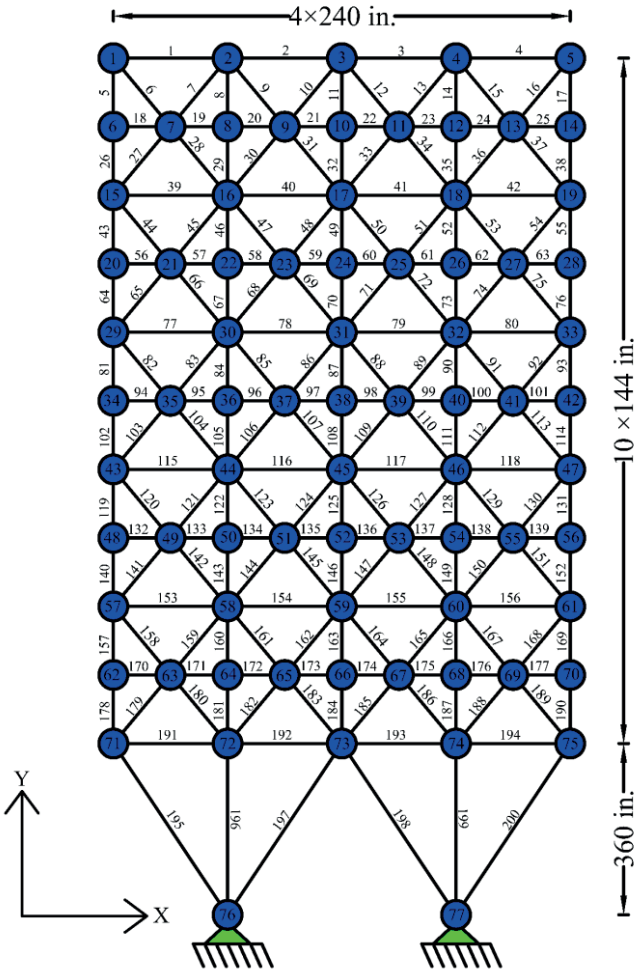
**Table 5** Comparison of the results from DE-MEDT and other optimization methods for the 72-bar spatial truss design problem

| Element group | | Optimal values of cross-sectional areas with continuous design variables (in$^2$) | | | | | |
|---|---|---|---|---|---|---|---|
| | | Kaveh and Talatahari [30] | Degertekin [31] | Degertekin and Hayalioglu [32] | Jafari and Salajeghheh [33] | Kaveh and Zakian [34] | Present work |
| | HBB-BC | SAHS | TLBO | PSOC | IGWO | DE-MEDT | |
| 1 | $A_1$–$A_4$ | 1.9042 | 1.860 | 1.9064 | 1.8733 | 1.8585 | 1.8941 |
| 2 | $A_5$–$A_{12}$ | 0.5162 | 0.521 | 0.50612 | 0.5135 | 0.5021 | 0.5167 |
| 3 | $A_{13}$–$A_{16}$ | 0.1000 | 0.100 | 0.100 | 0.1000 | 0.1002 | 0.1000 |
| 4 | $A_{17}$–$A_{18}$ | 0.1000 | 0.100 | 0.100 | 0.1000 | 0.1000 | 0.1000 |
| 5 | $A_{19}$–$A_{22}$ | 1.2582 | 1.271 | 1.2617 | 1.2766 | 1.3011 | 1.2668 |
| 6 | $A_{23}$–$A_{30}$ | 0.5035 | 0.509 | 0.5111 | 0.5133 | 0.5151 | 0.5111 |
| 7 | $A_{31}$–$A_{34}$ | 0.1000 | 0.100 | 0.100 | 0.1000 | 0.1000 | 0.1000 |
| 8 | $A_{35}$–$A_{36}$ | 0.1000 | 0.100 | 0.100 | 0.1000 | 0.1001 | 0.1000 |
| 9 | $A_{37}$–$A_{40}$ | 0.5178 | 0.485 | 0.5317 | 0.5187 | 0.5311 | 0.5213 |
| 10 | $A_{41}$–$A_{48}$ | 0.5214 | 0.501 | 0.51591 | 0.5132 | 0.5122 | 0.5155 |
| 11 | $A_{49}$–$A_{52}$ | 0.1000 | 0.100 | 0.100 | 0.1000 | 0.1008 | 0.1000 |
| 12 | $A_{53}$–$A_{54}$ | 0.1007 | 0.100 | 0.100 | 0.1000 | 0.1030 | 0.1000 |
| 13 | $A_{55}$–$A_{58}$ | 0.1566 | 0.168 | 0.1562 | 0.1562 | 0.1560 | 0.1566 |
| 14 | $A_{59}$–$A_{66}$ | 0.5421 | 0.584 | 0.54927 | 0.5596 | 0.5472 | 0.5432 |
| 15 | $A_{67}$–$A_{70}$ | 0.4132 | 0.433 | 0.40966 | 0.3890 | 0.4202 | 0.4115 |
| 16 | $A_{71}$–$A_{72}$ | 0.5756 | 0.520 | 0.56976 | 0.5679 | 0.5793 | 0.5663 |
| Best weight (lb) | | 379.66 | 380.62 | 379.63 | 379.68 | 379.7615 | 379.62 |
| Average weight (lb) | | 381.85 | 382.42 | 380.20 | 380.48 | 380.6811 | 379.68 |
| Worst weight (lb) | | N/A | 383.89 | 380.83 | N/A | N/A | 380.40 |
| SD (lb) | | 1.201 | 1.38 | 0.41 | 0.58 | 0.73 | 0.14 |
| NSAs | | 13,200 | 13,742 | 19,709 | 8,050 | 11,960 | 17,040 |




(a) (b)

**Fig. 8** Constraint boundaries of the 72-bar spatial truss evaluated at the optimized design by the DE-MEDT algorithm: (a) Displacement and (b) Stress




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

**Fig. 9** The 200-bar planar truss design problem

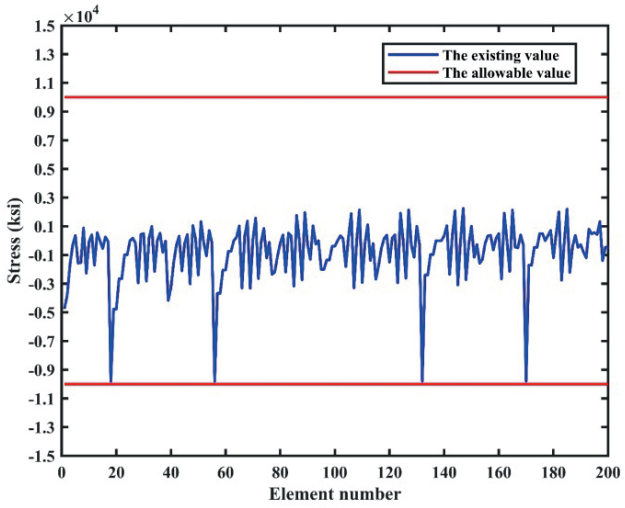
**Table 6** Comparison of the results from DE-MEDT and other optimization methods for the 200-bar planar truss design problem

| Element group | Optimal values of cross-sectional areas with continuous design variables ($in^2$) | | | | | | |
|---|---|---|---|---|---|---|---|
| | Degertekin [31] | Degertekin and Hayalioglu [32] | Bureerat and Pholdee [35] | Kaveh and Zakian [34] | Kim and Byun [36] | Pierezan et al. [37] | Present work |
| | SAHS | TLBO | ADEA | IGWO | CNNT-PSO | MCOA | DE-MEDT |
| 1 | 0.154 | 0.146 | 0.1020 | 0.1024 | 0.1482 | 0.1390 | 0.1483 |
| 2 | 0.941 | 0.941 | 1.1193 | 0.9654 | 0.9405 | 0.9355 | 0.9772 |
| 3 | 0.100 | 0.100 | 0.1000 | 0.1391 | 0.1000 | 0.1000 | 0.1000 |
| 4 | 0.100 | 0.101 | 0.1223 | 0.1741 | 0.1000 | 0.1000 | 0.1020 |
| 5 | 1.942 | 1.941 | 1.9622 | 1.9613 | 1.9408 | 1.9355 | 1.9797 |
| 6 | 0.301 | 0.296 | 0.2693 | 0.2899 | 0.2975 | 0.2909 | 0.2952 |
| 7 | 0.100 | 0.100 | 0.1719 | 0.1294 | 0.1000 | 0.1000 | 0.1010 |
| 8 | 3.108 | 3.121 | 3.0690 | 3.1511 | 3.1067 | 3.0816 | 3.1089 |
| 9 | 0.100 | 0.100 | 0.1004 | 0.1251 | 0.1000 | 0.1000 | 0.1000 |
| 10 | 4.106 | 4.173 | 4.1509 | 4.0627 | 4.1067 | 4.0816 | 4.0990 |
| 11 | 0.409 | 0.401 | 0.4317 | 0.4131 | 0.4057 | 0.3967 | 0.4087 |
| 12 | 0.191 | 0.181 | 0.2122 | 0.4043 | 0.1897 | 0.2959 | 0.2215 |
| 13 | 5.428 | 5.423 | 5.3974 | 5.3357 | 5.4343 | 5.3854 | 5.4287 |
| 14 | 0.100 | 0.100 | 0.1102 | 0.2632 | 0.1000 | 0.1000 | 0.1313 |
| 15 | 6.427 | 6.422 | 6.3959 | 6.3226 | 6.4340 | 6.3853 | 6.4323 |
| 16 | 0.581 | 0.571 | 0.6141 | 0.7972 | 0.5745 | 0.6332 | 0.5315 |
| 17 | 0.151 | 0.156 | 0.2666 | 0.1791 | 0.1366 | 0.1842 | 0.1416 |
| 18 | 7.973 | 7.958 | 7.9408 | 8.1268 | 7.9803 | 8.0396 | 7.9617 |
| 19 | 0.100 | 0.100 | 0.1471 | 0.1141 | 0.1000 | 0.1000 | 0.1000 |
| 20 | 8.974 | 8.958 | 8.9445 | 9.1337 | 8.9802 | 9.0395 | 8.9584 |
| 21 | 0.719 | 0.720 | 0.8141 | 0.8000 | 0.71089 | 0.7460 | 0.6954 |
| 22 | 0.422 | 0.478 | 1.1050 | 0.2487 | 0.4659 | 0.1306 | 0.6196 |
| 23 | 10.892 | 10.897 | 11.2893 | 11.2008 | 10.9110 | 10.9114 | 10.9674 |
| 24 | 0.100 | 0.100 | 0.1004 | 0.1136 | 0.1000 | 0.1000 | 0.1023 |
| 25 | 11.887 | 11.897 | 12.2891 | 12.1703 | 11.9112 | 11.9114 | 11.9635 |
| 26 | 1.040 | 1.080 | 1.4742 | 0.9947 | 1.0712 | 0.8627 | 1.1796 |
| 27 | 6.646 | 6.462 | 5.3417 | 6.3377 | 6.5030 | 6.9169 | 6.0056 |
| 28 | 10.804 | 10.799 | 9.8931 | 10.5338 | 10.7210 | 10.9674 | 10.4521 |
| 29 | 13.870 | 13.922 | 14.9127 | 14.0917 | 13.9310 | 13.6742 | 14.1025 |
| Best weight (lb) | 25,491.9 | 25,488.15 | 25800.5708 | 25,771.78 | 25,453.0957 | 25,450.18 | 25,350.28 |
| Average weight (lb) | 25,610.2 | 25,533.14 | 26438.1058 | 26,699.19 | 25,459.1089 | 25,522.07 | 25,516.21 |
| Worst weight (lb) | 25,799.3 | 25,563.05 | 26851.1460 | N/A | 25,466.0958 | 25,557.53 | 25,650.79 |
| SD (lb) | 141.85 | 27.44 | 2,288.7184 | 410.40 | 3.1544 | 47.62 | 87.40 |
| NSAs | 19,670 | 28,059 | 20,000 | 23,760 | 1500000 | 27,720 | 30,000 |

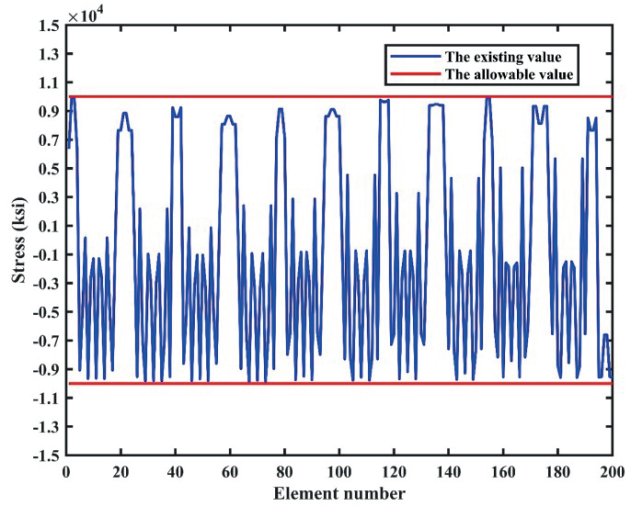
IGWO, CNNT-PSO, and MCOA. Based on the obtained results, it can be observed that the proposed DE-MEDT algorithm outperforms other considered methods in terms of solution accuracy. Fig. 10 displays the stress constraint boundaries of the truss structure obtained at the best optimum design, indicating that the design constraint of the problem has not been violated.

### 5.3 A 3-bay 24-story frame design problem

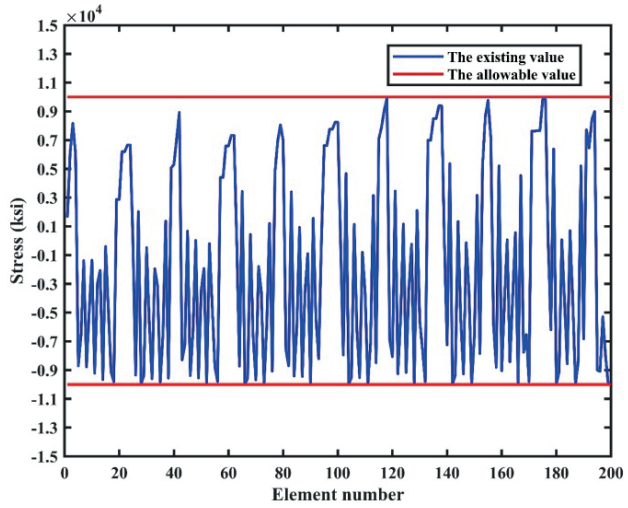
The last design problem is a 3-bay 24-story frame, as shown in Fig. 11. The frame structure has 168 members (96 columns and 72 beams) and is a well-known engineering benchmark in structural optimization with discrete design variables. Davison and Adams [38] first proposed this benchmark problem in which the modulus of elasticity




(a)




(b)




(c)

**Fig. 10** Stress constraint boundaries of the 200-bar planar truss evaluated at the optimized design by the DE-MEDT: (a) Case 1; (b) Case 2 and (c) Case 3

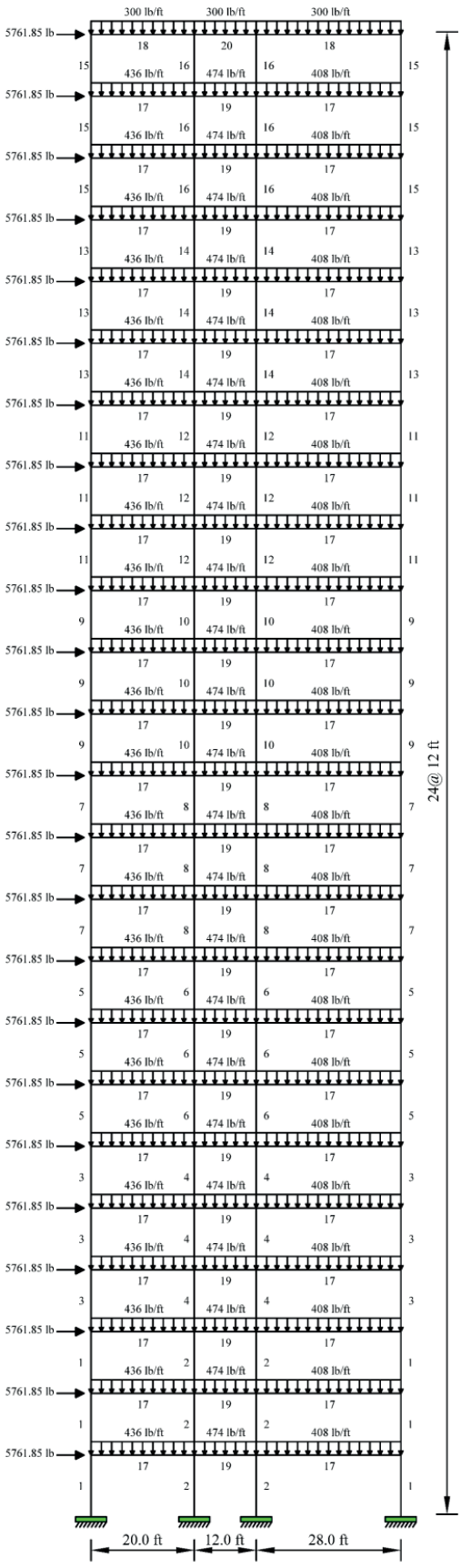



**Fig. 11** The 3-bay 24-story frame design problem

and yield stress of the steel material are considered as $E$ = 29,732ksi and $F_y$ = 33.4 ksi, respectively. All 168 members are categorized into 20 distinct groups (16 column groups and 4 beam groups). Information about which members of the frame belong to which element groups is shown in Fig. 11. The strength and displacement constraints are according to the AISC-LRFD requirements.

Table 7 [39–42] provides the comparisons between the results of the weight optimization process obtained by the proposed DE-MEDT and those obtained by Eagle Strategy with Differential Evolution (ES-DE) [39], Accelerated WEO [40], Improved Black Hole (IBH) [41], Self-Adaptive Multi-Population-based Jaya (SAMP-Jaya) [42], and Improved Shuffled based Jaya (IS-Jaya) [42]. This table shows that DE-MEDT found the best weight (201,402) against other considered methods.

The constraints of the problem, including the inter-story drift and stress ratio of all structural elements, are evaluated at the best optimum design and displayed in Fig. 12. From this figure, it can be observed that all design constraints of the problem are satisfied.

## 6 Conclusions

This paper introduced a physically inspired population-based MH algorithm named Doppler Effect-Mean Euclidian Distance Threshold (DE-MEDT) Optimization Algorithm. The mathematical formulation of the DE with some idealized rules acted as the stepsize of the proposed DE-MEDT to update the observers' positions. The quality of the observers was then enhanced based on the proposed Mean Euclidian Distance Threshold (MEDT). This new efficient mechanism was also presented to improve

**Table 7** Comparison of the results from DE-MEDT and other optimization methods for the 3-bay 24-story frame design problem

| Element group | Optimal cross-sectional areas with discrete design variables (W shape sections) | | | | | |
|---|---|---|---|---|---|---|
| | Talatahari et al. [39] | Kaveh and Bakhshpoori [40] | Gholizadeh et al. [41] | Kaveh et al. [42] | | Present work |
| | ES-DE | AWEO | IBH | SAMP-Jaya | IS-Jaya | DE-MEDT |
| 1 | W14 × 145 | W14 × 159 | W14 × 132 | W14 × 159 | W14 × 145 | W14 × 159 |
| 2 | W14 × 99 | W14 × 132 | W14 × 99 | W14 × 109 | W14 × 132 | W14 × 109 |
| 3 | W14 × 109 | W14 × 99 | W14 × 109 | W14 × 132 | W14 × 99 | W14 × 99 |
| 4 | W14 × 132 | W14 × 109 | W14 × 109 | W14 × 90 | W14 × 90 | W14 × 82 |
| 5 | W14 × 99 | W14 × 68 | W14 × 109 | W14 × 61 | W14 × 61 | W14 × 68 |
| 6 | W14 × 109 | W14 × 38 | W14 × 99 | W14 × 38 | W14 × 53 | W14 × 43 |
| 7 | W14 × 145 | W14 × 30 | W14 × 90 | W14 × 34 | W14 × 38 | W14 × 34 |
| 8 | W14 × 68 | W14 × 22 | W14 × 90 | W14 × 22 | W14 × 22 | W14 × 22 |
| 9 | W14 × 109 | W14 × 90 | W14 × 68 | W14 × 90 | W14 × 99 | W14 × 90 |
| 10 | W14 × 68 | W14 × 99 | W14 × 74 | W14 × 109 | W14 × 99 | W14 × 109 |
| 11 | W14 × 48 | W14 × 99 | W14 × 53 | W14 × 90 | W14 × 99 | W14 × 99 |
| 12 | W14 × 68 | W14 × 74 | W14 × 53 | W14 × 82 | W14 × 82 | W14 × 90 |
| 13 | W14 × 38 | W14 × 68 | W14 × 30 | W14 × 74 | W14 × 74 | W14 × 74 |
| 14 | W14 × 61 | W14 × 61 | W14 × 38 | W14 × 61 | W14 × 53 | W14 × 61 |
| 15 | W14 × 30 | W14 × 34 | W14 × 22 | W14 × 34 | W14 × 30 | W14 × 34 |
| 16 | W14 × 22 | W14 × 22 | W14 × 22 | W14 × 22 | W14 × 22 | W14 × 22 |
| 17 | W30 × 90 | W30 × 90 | W30 × 90 | W30 × 90 | W30 × 90 | W30 × 90 |
| 18 | W21 × 55 | W8 × 18 | W6 × 16 | W8 × 18 | W6 × 15 | W6 × 15 |
| 19 | W21 × 48 | W24 × 55 | W24 × 55 | W24 × 55 | W24 × 55 | W24 × 55 |
| 20 | W10 × 45 | W6 × 8.5 | W6 × 9 | W6 × 8.5 | W6 × 8.5 | W6 × 8.5 |
| Best weight (lb) | 212,479.17 | 202,194.02 | 208,719 | 202,410.034 | 201,618.034 | 201,402 |
| Average weight (lb) | N/A | 203,412.88 | 215,226 | 207979.482 | 203,169.205 | 203,321 |
| Worst weight (lb) | N/A | N/A | N/A | 230,940.004 | 209,586.084 | 213,240 |
| SD (lb) | N/A | N/A | N/A | 7,031.549 | 1,877.119 | 2,447 |
| NSAs | 12,500 | 11,300 | 10,000 | 16,600 | 14,100 | 17,550 |

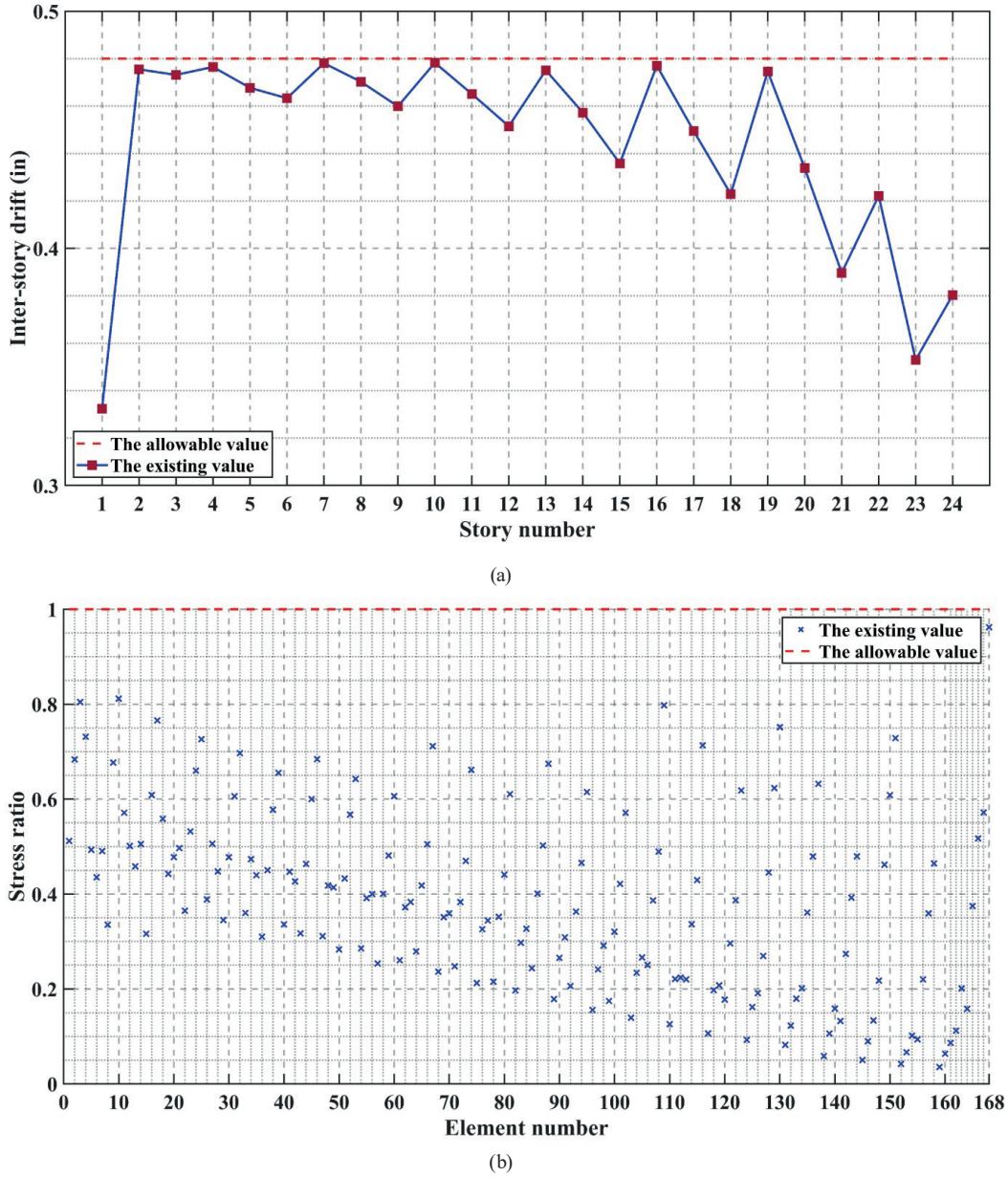



**Fig. 12** Constraint boundaries of the 3-bay 24-story frame evaluated at the optimized design by DE-MEDT: (a) Inter-story drift, (b) Stress ratio

the local optima avoidance capability and convergence speed of the DE-MEDT algorithm. Easy implementation and requiring few control parameters are the main advantages of the proposed algorithm. Qualitative analysis of DE-MEDT was conducted using four criteria, including search history, trajectory of the first individual, the average fitness of all agents, and convergence curve. Two well-known collections of constrained and unconstrained optimization problems were considered for evaluating the effectiveness of DE-MEDT against its other counterparts.

Twenty-three classical unimodal and multimodal test functions were utilized in the first collection. In solving this set of benchmark functions, six well-established and recently developed MH algorithms abbreviated as BA, PSOGSA, NBA, HEPSO, HFPSO, and FLFBA were considered for comparison with the proposed DE-MEDT optimizer. The results of numerical experiments yielded wondrous results, showing that DE-MEDT has excellent exploration and exploitation abilities.

Three constrained engineering design problems were considered in the second collection, including a 72-bar spatial truss, a 200-bar planar truss, and a 3-bay 24-story frame. Based on the obtained results from engineering design problems, the DE-MEDT optimization method showed better performance than other state-of-art MH optimization algorithms developed in the literature.

## References

[1] Kirsch, U. "Structural Optimization: Fundamentals and Applications", Springer, Verlag, Berlin, Heidelberg, Germany, 1993. https://doi.org/10.1007/978-3-642-84845-2

[2] Kaveh, A. "Advances in Metaheuristic Algorithms for Optimal Design of Structures", 3rd ed., Springer, Cham, Switzerland, 2021. https://doi.org/10.1007/978-3-319-05549-7

[3] Kaveh, A., Bakhshpoori, T. "Metaheuristics: Outlines, MATLAB Codes and Examples", Springer, Cham, Switzerland, 2019. https://doi.org/10.1007/978-3-030-04067-3

[4] Kennedy, J., Eberhart, R. "Particle swarm optimization", In: Proceedings of ICNN'95 - International Conference on Neural Networks, Perth, WA, Australia, 1995, pp. 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

[5] Dorigo, M., Di Caro, G. "Ant colony optimization: a new meta-heuristic", In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 1999, pp. 1470–1477. https://doi.org/10.1109/CEC.1999.782657

[6] Yang, X.-S. "A new metaheuristic bat-inspired algorithm", In: González, J. R., Pelta, D. A., Cruz, C., Terrazas, G., Krasnogor, N. (eds.) Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, Berlin, Germany, 2010, pp. 65–74.

[7] Rao, R. V., Savsani, V. J., Vakharia, D. P. "Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems", Computer-Aided Design, 43(3), pp. 303–315, 2011. https://doi.org/10.1016/j.cad.2010.12.015

[8] Kaveh, A., Khayatazad, M. "A new meta-heuristic method: Ray Optimization", Computers & Structures, 112–113, pp. 283–294, 2012. https://doi.org/10.1016/j.compstruc.2012.09.003

[9] Kaveh, A., Mahdavi, V. R. "Colliding bodies optimization: A novel meta-heuristic method", Computers & Structures, 139, pp. 18–27, 2014. https://doi.org/10.1016/j.compstruc.2014.04.005

[10] Kaveh, A., Akbari, H., Hosseini, S. M. "Plasma generation optimization: a new physically-based metaheuristic algorithm for solving constrained optimization problems", Engineering Computations, 38(4), pp. 1554–1606, 2021. https://doi.org/10.1108/EC-05-2020-0235

[11] Wolpert, D. H., Macready, W. G. "No free lunch theorems for optimization", IEEE Transactions on Evolutionary Computation, 1(1), pp. 67–82, 1997. https://doi.org/10.1109/4235.585893

[12] Kaveh, A., Zaerreza, A., Hosseini, S. M. "An enhanced shuffled Shepherd Optimization Algorithm for optimal design of large-scale space structures", Engineering with Computers, 2021. https://doi.org/10.1007/s00366-021-01292-z

[13] Kaveh, A., Zaerreza, A. "A new framework for reliability-based design optimization using metaheuristic algorithms", Structures, 38, pp. 1210–1225, 2022. https://doi.org/10.1016/j.istruc.2022.02.069

[14] Movahedi Rad, M., Habashneh, M., Lógó, J. "Elasto-Plastic limit analysis of reliability based geometrically nonlinear bi-directional evolutionary topology optimization", Structures, 34, pp. 1720–1733, 2021. https://doi.org/10.1016/j.istruc.2021.08.105

[15] Lógó, J., Movahedi Rad, M., Knabel, J., Tauzowski, P. "Reliability based design of frames with limited residual strain energy capacity", Periodica Polytechnica Civil Engineering, 55(1), pp. 13–20, 2011. https://doi.org/10.3311/pp.ci.2011-1.02

[16] Movahedi Rad, M., Khaleel Ibrahim, S. "Optimal Plastic Analysis and Design of Pile Foundations Under Reliable Conditions", Periodica Polytechnica Civil Engineering, 65(3), pp. 761–767, 2021. https://doi.org/10.3311/PPci.17402

[17] Doppler, C. "Ueber das farbige Licht der Doppelsterne und einiger anderer Gestirne des Himmels: Versuch einer das Bradley'sche Aberrations-Theorem als integrirenden Theil in sich schliessenden allgemeineren Theorie" (About the colored light of the double stars and some other celestial bodies), K. Böhm Gesellschaft der Wissenschaften, Prague, Czechia, 1903. (in German)

[18] Young, H. D., Freedman, R. A., Sandin, T., Ford, A. L. "University physics", Addison-Wesley, Boston, MA, USA, 1996.

[19] Meng, X.-B., Gao, X. Z., Liu, Y., Zhang, H. "A novel bat algorithm with habitat selection and Doppler effect in echoes for optimization", Expert Systems with Applications, 42, pp. 6350–6364, 2015. https://doi.org/10.1016/j.eswa.2015.04.026

[20] Mihoubi, M., Rahmoun, A., Lorenz, P., Lasla, N. "An effective Bat algorithm for node localization in distributed wireless sensor network", Security and Privacy, 1(1), e7, 2018. https://doi.org/10.1002/spy2.7

[21] Buijs Ballot, C. H. D. "Akustische Versuche auf der Niederlandische Eisenbahn, nebst gelegentliche Bemerkungen zur Theorie des Herrn Prof. Dobbler" (Acoustic experiments on the Dutch railways, along with occasional comments on Professor Dobbler's theory), Annalen der Physik und Chemie, 66, pp. 321–351, 1845. (in German) https://doi.org/10.1002/andp.18451421102

[22] Chen, V.C. "The Micro-Doppler Effect in Radar", Artech House, Boston, MA, USA, 2019.

[23] Rosen, J., Gothard, L. Q. "Encyclopedia of physical science", Facts On File, New York, NY, USA, 2010.

[24] Yang, X.-S., He, X. "Bat algorithm: literature review and applications", International Journal of Bio-Inspired Computation, 5(3), pp. 141–149, 2013. https://doi.org/10.1504/IJBIC.2013.055093

[25] Liu, Q., Wu, L., Xiao, W., Wang, F., Zhang, L. "A novel hybrid bat algorithm for solving continuous optimization problems", Applied Soft Computing, 73, pp. 67–82, 2018. https://doi.org/10.1016/j.asoc.2018.08.012

[26] Mirjalili, S., Hashim, S. Z. M. "A new hybrid PSOGSA algorithm for function optimization", In: 2010 International Conference on Computer and Information Application, Tianjin, China, 2010, pp. 374–377. https://doi.org/10.1109/ICCIA.2010.6141614

[27] Mahmoodabadi, M. J., Salahshoor Mottaghi, Z., Bagheri, A. "HEPSO: High exploration particle swarm optimization", Information Sciences, 273, pp. 101–111, 2014. https://doi.org/10.1016/j.ins.2014.02.150

[28] Aydilek, İ. B. "A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems", Applied Soft Computing, 66, pp. 232–249, 2018. https://doi.org/10.1016/j.asoc.2018.02.025

[29] Boudjemaa, R., Oliva, D., Ouaar, F. "Fractional Lévy flight bat algorithm for global optimisation", International Journal of Bio-Inspired Computation, 15(2), pp. 100–112, 2020. https://doi.org/10.1504/IJBIC.2020.106441

[30] Kaveh, A., Talatahari, S. "Size optimization of space trusses using Big Bang–Big Crunch algorithm", Computers & Structures, 87, pp. 1129–1140, 2009. https://doi.org/10.1016/j.compstruc.2009.04.011

[31] Degertekin, S. O. "Improved harmony search algorithms for sizing optimization of truss structures", Computers & Structures, 92-93, pp. 229–241, 2012. https://doi.org/10.1016/j.compstruc.2011.10.022

[32] Degertekin, S. O., Hayalioglu, M. S. "Sizing truss structures using teaching-learning-based optimization", Computers & Structures, 119, pp. 177–188, 2013. https://doi.org/10.1016/j.compstruc.2012.12.011

[33] Jafari, M., Salajegheh, E., Salajegheh, J. "Optimal design of truss structures using a hybrid method based on particle swarm optimizer and cultural algorithm", Structures, 32, pp. 391–405, 2021. https://doi.org/10.1016/j.istruc.2021.03.017

[34] Kaveh, A., Zakian, P. "Improved GWO algorithm for optimal design of truss structures", Engineering with Computers, 34, pp. 685–707, 2018. https://doi.org/10.1007/s00366-017-0567-1

[35] Bureerat, S., Pholdee, N. "Optimal Truss Sizing Using an Adaptive Differential Evolution Algorithm", Journal of Computing in Civil Engineering, 30(2), 04015019, 2016. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000487

[36] Kim, T.-H., Byun, J.-I. "Truss Sizing Optimization with a Diversity-Enhanced Cyclic Neighborhood Network Topology Particle Swarm Optimizer", Mathematics, 8(7), 1087, 2020. https://doi.org/10.3390/math8071087

[37] Pierezan, J., dos Santos Coelho, L., Cocco Mariani, V., Hochsteiner de Vasconcelos Segundo, E., Prayogo, D. "Chaotic coyote algorithm applied to truss optimization problems", Computers & Structures, 242, 106353, 2021. https://doi.org/10.1016/j.compstruc.2020.106353

[38] Davison, J. H., Adams, P. F. "Stability of braced and unbraced frames", Journal of the Structural Division, 100(2), pp. 319–334, 1974. https://doi.org/10.1061/JSDEAG.0003710

[39] Talatahari, S., Gandomi, A. H., Yang, X.-S., Deb, S. "Optimum design of frame structures using the Eagle Strategy with Differential Evolution", Engineering Structures, 91, pp. 16–25, 2015. https://doi.org/10.1016/j.engstruct.2015.02.026

[40] Kaveh, A., Bakhshpoori, T. "An accelerated water evaporation optimization formulation for discrete optimization of skeletal structures", Computers & Structures, 177, pp. 218–228, 2016. https://doi.org/10.1016/j.compstruc.2016.08.006

[41] Gholizadeh, S., Razavi, N., Shojaei, E. "Improved black hole and multiverse algorithms for discrete sizing optimization of planar structures", Engineering Optimization, 51(10), pp. 1645–1667, 2019. https://doi.org/10.1080/0305215X.2018.1540697

[42] Kaveh, A., Hosseini, S. M., Zaerreza, A. "Improved Shuffled Jaya algorithm for sizing optimization of skeletal structures with discrete variables", Structures, 29, pp. 107–128, 2021. https://doi.org/10.1016/j.istruc.2020.11.008