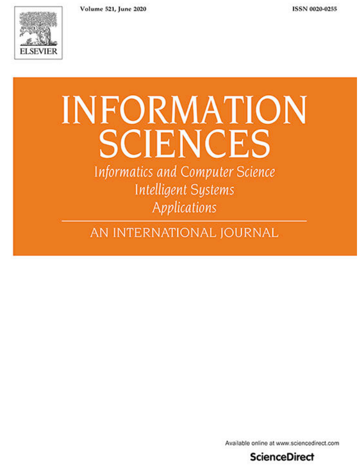
Journal Pre-proofs



Gradient-Based Optimizer: A New Metaheuristic Optimization Algorithm

Iman Ahmadianfar, Omid Bozorg-Haddad, Xuefeng Chu

PII:	S0020-0255(20)30624-1
DOI:	https://doi.org/10.1016/j.ins.2020.06.037
Reference:	INS 15600
To appear in:	*Information Sciences*
Received Date:	7 December 2019
Revised Date:	9 May 2020
Accepted Date:	14 June 2020

Please cite this article as: I. Ahmadianfar, O. Bozorg-Haddad, X. Chu, Gradient-Based Optimizer: A New Metaheuristic Optimization Algorithm, *Information Sciences* (2020), doi: https://doi.org/10.1016/j.ins.2020.06.037

Gradient-Based Optimizer: A New Metaheuristic Optimization Algorithm

Iman Ahmadianfar[1]*, Omid Bozorg-Haddad[2], Xuefeng Chu[3]

Assistant Professor, Dept. of Civil Engineering, Behbahan Khatam Alanbia Univ. of Technology, Behbahan, Iran (corresponding author). E-mail: i.ahmadianfar@bkatu.ac.ir, +989371588755.

[2] Professor, Dept. of Irrigation and Reclamation Engineering, Faculty of Agricultural Engineering and Technology, College of Agriculture and Natural Resources, Univ. of Tehran, Karaj, Tehran, Iran. E-mail: OBHaddad@ut.ac.ir.

[3] Professor, Department of Civil & Environmental Engineering, North Dakota State University, Department 2470, Fargo, ND, USA. xuefeng.chu@ndsu.edu.

Abstract

In this study, a novel metaheuristic optimization algorithm, gradient-based optimizer (GBO) is proposed. The GBO, inspired by the gradient-based Newton's method, uses two main operators: gradient search rule (GSR) and local escaping operator (LEO) and a set of vectors to explore the search space. The GSR employs the gradient-based method to enhance the exploration tendency and accelerate the convergence rate to achieve better positions in the search space. The LEO enables the proposed GBO to escape from local optima. The performance of the new algorithm was evaluated in two phases. 28 mathematical test functions were first used to evaluate various characteristics of the GBO, and then six engineering problems were optimized by the GBO. In the first phase, the GBO was compared with five existing optimization algorithms, indicating that the GBO yielded very promising results due to its enhanced capabilities of exploration, exploitation, convergence, and effective avoidance of local optima. The second phase also demonstrated the superior performance of the GBO in solving complex real-world engineering problems. Source codes of the GBO algorithm are publicly available at http://imanahmadianfar.com/codes/.

Keywords: Optimization; Gradient-based method; Metaheuristic algorithm; Constrained optimization problem.

1. Introduction

Many real-world applications in various science and engineering fields can be converted to optimization problems. However, the related problems are often highly non-convex, non-linear, and multimodal. Although a variety of optimization algorithms have been developed, they frequently fail to provide satisfactory results for such challenging problems, which emphasizes the need for new optimization methods. The metaheuristic algorithms (MAs) [1], which are known as global optimization techniques, have been successfully used to solve various complex and real optimization problems [2, 3]. The metaheuristic methods use some principles of physics, swarm intelligence, and biology [4].

In the last decades, different MAs have been developed and used. For example, the genetic algorithm (GA) was derived from the Darwin's theory of evolution [5]. The differential evolution (DE) algorithm employs the same operators (i.e., mutation and crossover) as those in the GA but with a different approach [6]. The DE algorithm uses the difference between two randomly selected vectors to generate a new vector. Particle swarm optimization (PSO) was inspired by the social behaviors of birds and fish for catching food [7]. The artificial bee colony (ABC) algorithm simulates the information sharing capability and food foraging behavior of honey bees [8]. The gravitational search algorithm (GSA) uses the laws of gravity and motion [9]. The bat algorithm (BA) simulates the echolocation behavior involved in bats [10]. The grey wolf optimizer (GWO) mimics the hunting behavior of grey wolves in nature [11]. The sine and cosine algorithm (SCA) uses a mathematical function on the basis of sine and cosine functions

[4]. The thermal exchange optimization (TEO) algorithm is based on the principle of the Newton's law of cooling [12]. Atom search optimization (ASO) simulates the motion of atoms in nature based on atom dynamics [13].

The aforementioned methods are categorized as population-based algorithms, which involve a set of solutions in the optimization process. The search engines of such optimization methods are based on different phenomena as described above. Many studies have demonstrated successful applications of these methods for a broad variety of real-world problems [2, 14, 15]. Generally, the population-based optimizers share common information despite their natures [16]. In these algorithms, the search engine implements two steps: exploration and exploitation [17]. Exploration involves exploring new positions far from the current position in the entire search area, while exploitation aims to explore the near-optimal positions. The utilization of exploration alone may lead to new positions with low accuracy. In contrast, the employment of exploitation alone increases the chance to get stuck in local optimal positions. Many studies emphasized the importance to balance the exploration and exploitation search processes in the metaheuristic algorithms [15]. Hence, creating a suitable balance between these two processes is crucial [18].

Most of the metaheuristic algorithms are managed to create a proper trade-off between exploration and exploitation. To do this, some studies have been conducted to enhance the efficiency of basic algorithms by using suitable setting of the control parameters or hybridization of optimization algorithms [19-21]. However, to date, creating a suitable balance between exploration and exploitation in the metaheuristic methods is a challenging and unsolved issue. On the other hand, based on the rule of No Free Lunch (NFL) [22], no metaheuristic algorithm can solve all problems, indicating that a specific algorithm may provide very good results for a set of problems, but the same method may have low efficiencies for a different set of problems.

NFL also implies that this field of research is highly dynamic, which leads to the development of many new metaheuristic optimization algorithms over years. This study attempts to fill the research gap by proposing a new metaheuristic algorithm with population-based characteristics.

Thus, the main objective of this study is to develop a novel gradient-based metaheuristic algorithm, namely gradient-based optimizer (GBO). The most popular gradient-based search methods include the Newton's method [23], Quasi-Newton method [24], Levenberg Marquardt (LM) algorithm [25], and the conjugate direction method [26]. These methods have been applied in many studies to solve different types of optimization problems. For example, Salajegheh and Salajegheh combined the Quasi-Newton method with the PSO algorithm to promote the performance and reliability of the basic PSO [27]. Ibtissem and Nouredine [28] introduced a hybrid of the DE algorithm and the conjugate gradient method to increase the local search ability in the basic DE. Shahidi et al. [29] developed a self-adaptive optimization algorithm employing the conjugate gradient as a local search method. Bandurski and Kwedlo [30] combined the conjugate gradient method with the DE algorithm to improve the local search of the basic DE algorithm. Parwani et al. [31] introduced a hybrid DE with a local optimization method, in which the conjugate gradient method was used for local search to increase the convergence speed. These studies demonstrated the important role of the gradient-based methods. Therefore, this study proposes the GBO algorithm with a search engine based on the Newton's method and employs a set of vectors to search the solution space, which involves two operators including the gradient search rule (GSR) and the local escaping operator (LEO). The performance of the GBO is evaluated by using 28 mathematical test functions and 6 real-world engineering optimization problems that have been examined in previous studies.

The remaining sections are organized as follows: A brief review of the Newton's method as a gradient-based optimization method is presented in Section 2 and the main structures of the GBO are explained in Section 3. Experimental results are detailed in Section 4 and the conclusions from this study are summarized in Section 5.

2. Methodology

2.1. Theoretical background

Generally, the optimization methods can be categorized into two groups: gradient-based (GB) methods such as the LM algorithm [25], gradient descent (GD) [32], and Newton's method [23], and modern non-gradient-based methods (i.e., metaheuristic algorithms (MAs)) such as genetic algorithms (GAs) [5], simulated annealing (SA) [33], water evaporation optimization (WEO) [34], teaching learning based optimization (TLBO) [35], self-defense mechanism of plants (SDMP) algorithm [36], henry gas solubility optimization (HGSO) [37], and Harris hawks optimization (HHO) [38]. The gradient-based methods have been broadly employed to solve optimization problems. To determine an optimal solution using the gradient-based methods, an extreme point, at which the gradient is equal to zero, must be identified. The gradient methods such as the conjugate direction [26] and Newton's method are based on this concept. In the gradient methods and most of other optimization methods, a search direction is selected and the searching process moves along this direction towards the optimal solution [29]. Exploring the search directions in these methods needs to determine the derivatives of the objective function together with the constraints. The two main disadvantages of this type of optimization are: (1) the convergence speed is very slow and (2) there is no guarantee to achieve the optimal solution [27].

In the second category, some initial points (i.e., initial population) are randomly generated. Each point has a search direction, which is determined by the information acquired from previous results. The optimization process is continued by updating the search directions until the convergence criterion is met. Such optimization techniques (i.e., MAs) have been widely utilized to optimize different engineering problems. The MAs provide great robustness to find the global optima, while the gradient-based methods tend to converge into local optima. However, the non-gradient-based methods require higher computational capacities, especially for the problems with high-dimensional search spaces. Hence, it will be very worthwhile to develop an optimization method that uses a gradient method to skip the unfeasible points and move towards the feasible area and also takes advantage of the capabilities of the population-based optimization methods. Thus, one of the unique features of this study is to combine the concept of the gradient-based methods with the population-based methods for creating a powerful and efficient algorithm to overcome the drawbacks of previous methods.

2.2. Newton's method

The Newton's method is a powerful method to numerically solve equations [24]. This method is a root-finding algorithm that employs the initial terms of the Taylor series. This method starts with a single point (x_0) and then uses the Taylor series assessed at point x_0 for estimating another point that is nearby to the solution. This procedure continues until the final solution is obtained. The Taylor series of function $f(x)$ can be expressed as:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)(x - x_0)^2}{2!} + \frac{f^{(3)}(x_0)(x - x_0)^3}{3!} + \dots \quad (1)$$

where $f'(x)$, $f''(x)$, and $f^{(3)}(x)$ respectively are the first-, second-, and third-order derivatives of $f(x)$ with respect to x. Assuming that the initial point is very close to the actual root, $(x - x_0)$ is small and the higher-order terms in the Taylor series will approach to zero. Therefore, truncating the series (Eq. 1) attains a linear approximation of $f(x)$ as follows:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) \tag{2}$$

To determine the root for $f(x)$, let $f(x)$ be zero and solve for x:

$$x = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{3}$$

Accordingly, given x_n, next approximation x_{n+1} can be expressed as:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4}$$

The Newtom's method implements an iterative process to eventuallly obtain the final solution.

2.3. Modification of Newton's method

In this study, a new variant of the Newton's method introduced by Weerakoon and Fernando [39] is used to formulate the proposed algorithm, which is defined as:

$$x_{n+1} = x_n - \frac{f(x_n)}{[f'(x_{n+1}) + f'(x_n)]/2} \tag{5}$$

where $f'(x_{n+1})$ is the first-order derivative of $f(x)$ with respect to x_{n+1}.

According to Özban [40], Eq. (5) can be expressd as:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'\left(\frac{[z_{n+1} + x_n]}{2}\right)}, \tag{6}$$

where

$$z_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (6\text{-}1)$$

So, the new variant of the Newton's method can be achieved by using the arithmetic mean of z_{n+1} and x_n.

2.4. Gradient-based optimizer

In the proposed GBO that combines the gradient and population-based methods, the search direction is specified by the Newton's method to explore the search domain utilizing a set of vectors and two main operators (i.e., gradient search rule and local escaping operators). Minimization of the objective function is considered in the optimization problems.

2.4.1. Initialization

An optimization problem involves a set of decision variables, constraints, and an objective function. The control parameters of the GBO include a parameter for transition from the exploration to exploitation (α) and a probability rate. The number of iterations and the population size are determined, depending on the problem complexity. In the proposed algorithm, each member of the population is called "vector". Accordingly, the GBO includes *N* vectors in a *D*-dimensional search space. Thus, a vector can be expressed as:

$$X_{n,d} = [X_{n,1}, X_{n,2}, \ldots, X_{n,D}], \quad n = 1, 2, \ldots, N, \quad d = 1, 2, \ldots, D$$

(7)

Usually, the initial vectors of the GBO are randomly generated in the *D*-dimensional search domain, which can be defined as:

$$X_n = X_{min} + rand(0,1) \times (X_{max} - X_{min}) \quad (8)$$

where X_{min} and X_{max} are the bounds of decision variable X, and $rand(0,1)$ is a random number in [0, 1].

2.4.2. Gradient search rule (GSR)

In the gradient search rule, the movement of vectors is controlled to better search in the feasible domain and achieve better positions. With the aim of enhancing the exploration tendency and accelerating the convergence of the GBO, the GSR is proposed based on the concept of the GB method. However, this rule is extracted from the Newton's gradient-based method [23]. Given the fact that many optimization problems are not differentiable, a numerical gradient approach is employed as a substitute for the direct derivation of the function. Generally, the GB method begins a guessed initial solution and moves toward the next position along a gradient-specified direction. To derive the GSR based on Eq. (4), the first-order derivative must be calculated by utilizing the Taylor series. The Taylor series for functions $f(x+\Delta x)$ and $f(x-\Delta x)$ can be respectively expressed as:

$$f(x+\Delta x)=f(x)+f'(x_0)\Delta x+\frac{f''(x_0)\Delta x^2}{2!}+\frac{f^{(3)}(x_0)\Delta x^3}{3!}+\dots \tag{9}$$

$$f(x-\Delta x)=f(x)-f'(x_0)\Delta x+\frac{f''(x_0)\Delta x^2}{2!}-\frac{f^{(3)}(x_0)\Delta x^3}{3!}+\dots \tag{10}$$

From the truncated Eqs. (9) and (10), the first-order derivative is given by the following central differencing formula [41]:

$$f'(x)=\frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x} \tag{11}$$

Based on Eqs. (4) and (11), the new position (x_{n+1}) is then defined as:

$$x_{n+1} = x_n - \frac{2\Delta x \times f(x_n)}{f(x_n + \Delta x) - f(x_n - \Delta x)} \tag{12}$$

Since the *GSR* is considered as the main core of the proposed algorithm, some modifications are essential to handle the population-based search. Regarding Eq. (12), the neighboring positions of x_n are $x_n + \Delta x$ and $x_n - \Delta x$, which are depicted in Fig. 1. In the GBO algorithm, these neighboring positions are replaced with two other positions (vectors) in the population. Since $f(x)$ is a minimization problem, as shown in Fig. 1, position $x_n + \Delta x$ has a worse fitness than x_n, while $x_n - \Delta x$ is better than x_n. Accordingly, the GBO algorithm substitues position $x_n - \Delta x$ with x_{best}, which has a better position in the neighbohood of position x_n, while $x_n + \Delta x$ is replaced with x_{worst}, which is a worse position in the neighbohood of x_n. In addition, the proposed algorithm employs the position (x_n), instead of its fitness ($f(x_n)$) because the use of fitness of a position is more time-consuming in the computation. The proposed GSR is then formulated as follows:

$$GSR = randn \times \frac{2\Delta x \times x_n}{(x_{\text{worst}} - x_{best} + \varepsilon)} \tag{13}$$

where $randn$ is a normally distributed random number, and ε is a small number within the range of [0, 0.1]. x_{best} and x_{worst} are the best and worst solutions obtained during the optimization process. Eq. (13) can assisst the current solution to update its position. To improve the search capability of the proposed GBO and balance exploration (global) and exploitation (local), the *GSR* is modified by introducing a random parameter ρ_1 in Eq. 13, as detailed below.

Generally, an optimization algorithm should be capable of balancing the global exploration and local exploitation to explore the promising areas in the search domain and eventually converge to the global optimal solution. To achieve this goal, the GSR can be changed by utlizing an adaptive coefficient. In this study, ρ_1 is introduced as the most significant parameter in the GBO to balance the exploration and exploitation searching processes, and it can be expressed as:

$$\rho_1 = 2 \times rand \times \alpha - \alpha \tag{14}$$

$$\alpha = \left| \beta \times \sin\left(\frac{3\pi}{2} + sin\left(\beta \times \frac{3\pi}{2}\right)\right) \right| \tag{14-1}$$

$$\beta = \beta_{min} + (\beta_{max} - \beta_{min}) \times \left(1 - \left(\frac{m}{M}\right)^3\right)^2 \tag{14-2}$$

where β_{min} and β_{max} are 0.2 and 1.2, respectively, m is the number of iterations, and M is the total number of iterations. To balance the exploration and exploitation processes, parameter ρ_1 changes based on the sine function α. Fig. 2 depicts how the parameter α changes with the iteration number. The maximum iteration number is 1000. This parameter can be changed at each iteration. It has a large value at the early iterations to enhance the population diversity and then its value decreases as the iteration number increases to accelerate the convergence. The generated solutions should be capable of exploring the search space around their corresponding best solutions. In this regard, the parameter value increases for the iteration numbers ranging from 550 to 750, which assists the proposed algorithm to escape from any local optima because it can increase the diversity of population to search around the best solution ever obtained. Thus, Eq. (13) can be rewrriten as:

$$GSR = randn \times \rho_1 \times \frac{2\Delta x \times x_n}{(x_{\text{worst}} - x_{best} + \varepsilon)} \tag{15}$$

The proposed GSR helps the GBO to account for the random behavior during the optimization process, promoting exploration and escaping local optima. In Eq. (15), Δx is determined based on the difference between the best solution (x_{best}) and a randomly selected position (x_{r1}^{m}) (see Eqs. 16, 16-1, and 16-2). To ensure that Δx changes at each iteration, parameter δ is defined by Eq. (16-2). Additionaly, to improve exploration, a random number ($rand$) is added to Eq. (16-2).

$$\Delta x = rand(1{:}N) \times |step| \tag{16}$$

$$step = \frac{(x_{best} - x_{r1}^{m}) + \delta}{2} \tag{16-1}$$

$$\delta = 2 \times rand \times \left(\left|\frac{x_{r1}^{m} + x_{r2}^{m} + x_{r3}^{m} + x_{r4}^{m}}{4} - x_{n}^{m}\right|\right) \tag{16-2}$$

where $rand(1{:}N)$ is a random number with N dimensions, $r1$, $r2$, $r3$, and $r4$ ($r1 \neq r2 \neq r3 \neq r4 \neq n$) are different integers randomly chosen from [1, N], $step$ is a step size, which is determined by x_{best} and x_{r1}^{m}. Based on the proposed GSR, Eq. (12) can be rewritten as:

$$x_{n+1} = x_n - GSR \tag{17}$$

The direction of movement (DM) is also added to better exploit the nearby area of x_n. This term uses the best vector and moves the current vector (x_n) in the direction of ($x_{best} - x_n$). Therefore, this process creates a suitable local search tendency to promote the convergence speed of the GBO algorithm. The proposed DM is formulated as follows:

$$DM = rand \times \rho_2 \times (x_{best} - x_n) \tag{18}$$

where $rand$ is a random number in [0, 1], and ρ_2 is a random parameter, which assists each vector to have a different step size. In addition, this can be another component of the GBO that supports the exploration process. ρ_2 is given by:

$$\rho_2 = 2 \times rand \times \alpha - \alpha \tag{19}$$

Finally, based on the terms of the *GSR* and *DM*, Eqs. (20) and (21) can be used to update the position of current vector (x_n^m).

$$X1_n^m = x_n^m - GSR + DM \tag{20}$$

$$X1_n^m = x_n^m - randn \times \rho_1 \times \frac{2\Delta x \times x_n^m}{(x_{\text{worst}} - x_{best} + \varepsilon)} + rand \times \rho_2 \times (x_{best} - x_n^m) \tag{21}$$

where $X1_n^m$ is the new vector generated by updating x_n^m. Fig. 3 displays how the current position is updated. As shown in Fig. 3, the position $X1_n^m$ is created at a random point which is specified by the *GSR* and *DM* in the search space.

In this study, the Newton's method introduced by Özban [40] (Eq. (6)) is used to improve the *GSR*. Based on Eqs. (6) and (11), the *GSR* can also be expressed as:

$$x_{n+1} = x_n - \frac{2\Delta x \times f(x_n)}{f(y_n + \Delta x) - f(y_n - \Delta x)} \tag{22}$$

where

$$y_n = \frac{[z_{n+1} + x_n]}{2} \tag{22-1}$$

Eq. 22 is employed to update the position of the current solution with a formula different from Eq. 12. This equation uses the average of two vectors z_{n+1} and x_n, instead of x_n only. This new formula can assist the optimization algorithm by improving the search process in the solution space.

Similar to Eq. (15), to convert Eq. (22) to a population-based search method, z_{n+1} is first formulated as:

$$z_{n+1} = x_n - \frac{2\Delta x \times f(x_n)}{f(x_n + \Delta x) - f(x_n - \Delta x)} \tag{22-2}$$

Then, to change to a population-based algorithm, Eq. (22-2) can be rewritten as:

$$z_{n+1} = x_n - randn \times \frac{2\Delta x \times x_n}{(x_{\text{worst}} - x_{\text{best}} + \varepsilon)} \tag{22-3}$$

$y_n + \Delta x$ and $y_n - \Delta x$ in Eq. (22) are respectively given by:

$$y_n + \Delta x = \frac{[z_{n+1} + x_n]}{2} + \Delta x \tag{22-4}$$

$$y_n - \Delta x = \frac{[z_{n+1} + x_n]}{2} - \Delta x \tag{22-5}$$

In this research, to enhance the diversity and exploration and to create a robust population-based search method, Eqs. 22-4 and 22-5 are revised as (note that $y_n + \Delta x$ and $y_n - \Delta x$ are simplified as yp_n and yq_n):

$$yp_n = rand \times (\frac{[z_{n+1} + x_n]}{2} + rand \times \Delta x) \tag{22-6}$$

$$yq_n = rand \times (\frac{[z_{n+1} + x_n]}{2} - rand \times \Delta x) \tag{22-7}$$

where yp_n and yq_n are two positions created in regard to z_{n+1} and x_n, respectively.

Using the above equations, the GSR can be expressed as:

$$GSR = randn \times \rho_1 \times \frac{2\Delta x \times x_n}{(yp_n - yq_n + \varepsilon)} \tag{23}$$

With respect to the *GSR* and *DM*, Eqs. (24) and (25) are used to produce the position of $X1_n^m$.

$$X1_n^m = x_n^m - GSR + DM \tag{24}$$

$$X1_n^m = x_n^m - randn \times \rho_1 \times \frac{2\Delta x \times x_n^m}{(yp_n^m - yq_n^m + \varepsilon)} + rand \times \rho_2 \times (x_{\text{best}} - x_n^m) \tag{25}$$

By replacing the position of the best vector (x_{best}) with the current vector (x_n^m) in Eq. (25), the new vector ($X2_n^m$) can be generated as follows:

$$X2_n^m = x_{\text{best}} - randn \times \rho_1 \times \frac{2\Delta x \times x_n^m}{(yp_n^m - yq_n^m + \varepsilon)} + rand \times \rho_2 \times (x_{r1}^m - x_{r2}^m) \tag{26}$$

This search direction method emphasizes the exploitation process. The search method expressed by Eq. (26) is good for local search but is limited for global search, while the search method introduced in Eq. (25) is good for global search but is limited for local search. Therefore, the GBO takes advantage of both search methods (Eqs. (25) and (26)) to enhance both exploration and exploitation. Accordingly, based on the positions $X1_n^m$, $X2_n^m$, and the current position (X_n^m), the new solution at the next iteration (x_n^{m+1}) can be defined as:

$$x_n^{m+1} = r_a \times \left(r_b \times X1_n^m + (1 - r_b) \times X2_n^m\right) + (1 - r_a) \times X3_n^m$$

(27)

$$X3_n^m = X_n^m - \rho_1 \times (X2_n^m - X1_n^m) \tag{27-1}$$

where r_a and r_b are two random numbers in [0, 1].

Fig. 4 depicts how a vector updates its position with regard to $X1_n^m$, $X2_n^m$, and $X3_n^m$ in a 2D search space. According to Fig. 4 and Eq. (30), the position x_n^{m+1} would be at a random place determined by the positions $X1_n^m$, $X2_n^m$, and $X3_n^m$ in the search space. Indeed, these three positions specify the position x_n^{m+1}, and other vectors change their positions randomly around x_n^{m+1}.

2.4.3. Local escaping operator (LEO)

The LEO is introduced to promote the efficiency of the proposed GBO algorithm for solving complex problems. This operator can significantly change the position of the solution x_n^{m+1}. The LEO generates a solution with a superior performance (X_{LEO}^m) by using several solutions, which include the best position (x_{best}), the solutions $X1_n^m$ and $X2_n^m$, two random solutions x_{r1}^m and x_{r2}^m, and a new randomly generated solution (x_k^m). The solution X_{LEO}^m is generated by the following scheme:

$$
\begin{aligned}
&\boldsymbol{if}\ rand < pr \\
&\quad \boldsymbol{if}\ rand < 0.5 \\
&\quad X_{LEO}^m = X_n^{m+1} + f_1 \times \left(u_1 \times x_{\text{best}} - u_2 \times x_k^m\right) + f_2 \times \rho_1 \times \left(u_3 \times (X2_n^m - X1_n^m\right) + u_2 \times (x_{r1}^m - x_{r2}^m))/2 \\
&\quad X_n^{m+1} = X_{LEO}^m \\
&\quad \textbf{else} \\
&\quad X_{LEO}^m = x_{\text{best}} + f_1 \times \left(u_1 \times x_{\text{best}} - u_2 \times x_k^m\right) + f_2 \times \rho_1 \times \left(u_3 \times (X2_n^m - X1_n^m\right) + u_2 \times (x_{r1}^m - x_{r2}^m))/2 \\
&\quad X_n^{m+1} = X_{LEO}^m \\
&\quad \textbf{End} \\
&\textbf{End}
\end{aligned}
\tag{28}
$$

where f_1 is a uniform random number in the range of [-1,1], $f_2$is a random number from a normal distribution with mean of 0 and standard deviation of 1, pr is the probability, and u_1, u_2, and u_3 are three random numbers, which are defined as:

$$u_1 = \begin{cases} 2 \times rand & if\ \mu_1 < 0.5 \\ 1 & otherwise \end{cases} \tag{28-1}$$

$$u_2 = \begin{cases} rand & if\ \mu_1 < 0.5 \\ 1 & otherwise \end{cases} \tag{28-2}$$

$$u_3 = \begin{cases} rand & if\ \mu_1 < 0.5 \\ 1 & otherwise \end{cases} \tag{28-3}$$

where $rand$ is a random number in the range of [0, 1], and μ_1 is a number in the range of [0, 1]. The above equations can be simplified:

$$u_1 = L_1 \times 2 \times rand + (1 - L_1) \tag{28-4}$$

$$u_2 = L_1 \times rand + (1 - L_1) \tag{28-5}$$

$$u_3 = L_1 \times rand + (1 - L_1) \tag{28-6}$$

where L_1 is a binary parameter with a value of 0 or 1. If parameter μ_1 is less than 0.5, the value of L_1 is 1, otherwise, it is 0.

To determine the solution x_k^m in Eq. (28), the following scheme is suggested.

$$x_k^m = \begin{cases} x_{rand} & if\ \mu_2 < 0.5 \\ x_p^m & otherwise \end{cases} \tag{28-7}$$

$$x_{rand} = X_{min} + rand(0,1) \times (X_{max} - X_{min}) \tag{28-8}$$

where x_{rand} is a new solution, x_p^m is a randomly selected solution of the population ($p \in$ [1, 2,..., N]), and $\mu_2$is a random number in the range of [0, 1]. Eq. (28-7) can be simplified as:

$$x_k^m = L_2 \times x_p^m + (1 - L_2) \times x_{rand} \tag{28-9}$$

where L_2 is a binary parameter with a value of 0 or 1. If μ_2 is less than 0.5, the value of L_2 is 1, otherwise, it is 0. This random behavior in selecting the values of parameters u_1, u_2, and u_3 assists to increase the diversity of the population and escape from local optimal solutions. The pseudo code of the GBO algorithm is shown in Table 1.

3. Results and discussion

The performance of the GBO algorithm is extensively evaluated by using 28 mathematical functions, which have been broadly employed in previous studies [2, 15, 42]. These test functions can be categorized into four different types, comprising unimodal functions (f_1-f_6), multimodal functions (f_7-f_{14}), hybrid functions (f_{15}-f_{20}), and composite functions (f_{21}-f_{28}). A brief summary of all functions is shown in Tables 2-4 in Appendix A. Note that optimization of the hybrid and composite mathematical functions is more complicated and challenging than that of the unimodal and multimodal functions. Hence, it is more proper to evaluate the capabilities of the algorithms in solving complex real-world optimization problems.

3.1. Experimental setup

To test the performance of the GBO, it is compared with five metaheuristic algorithms including GWO, CS, ABC, WOA, and ISA. Each optimization algorithm is independently run 30 times for each test function. Table 5 shows the control parameters of all algorithms, which are recommended on the basis of the trial-and-error technique and/or estimated experimentally. The population size and the maximum number of iterations are respectively set to 50 and 500 for the unimodal and multimodal functions, and 50 and 1000 for the hybrid and composite functions. Tables 6 to 7 show the best, average, and standard deviation values of the objective function calculated for unimodal, multimodal, hybrid, and composite test functions over the 30 runs. In the following sections, the exploitation and exploration behaviors of the GBO are first investigated, and then its capability of avoidance of local optima and convergence behavior are tested.

3.2. Evaluation of the exploitation ability

The unimodal functions are usually used to evaluate the exploiation ability of the optimization algorithms. These test functions have only one global position and no local position, so the exploitation behavior and the convergence speed of the GBO algorithm can be assessed by using these functions. Table 6 shows the results of the GBO, GWO, CS, ABC, WOA, and ISA algorithms for the unimodal functions (f_1-f_6). It can be observed that the GBO provided more promising results than the GWO, CS, ABC, WOA, and ISA algorithms. In particular, the GBO was the best optimization algorithm to solve all unimodal functions in the terms of the best, average, and standard deviation values of the objective function for the 30 independent runs. The GBO had a good accuracy for the unimodal functions, indicating that this new algorithm proposed in this study has more promising exploitation capability that the other five optimization algorithms.

3.3. Evaluation of the exploration ability

The exploration ability of the GBO was evaluated by the multimodal test functions (f_7-f_{14}). These functions are known for having a large number of local optimal solutions so that the number of these solutions increases exponentially with increasing the problem dimensions. Hence, it is proper to evaluate the exploration capability of the optimization methods. Table 6 shows the results of the GBO algorithm and the GWO, CS, ABC, WOA, and ISA algorithms. As shown in Table 6, the GBO yielded much better results than the five other algorithms, except for functions f_{11}-f_{14}. The GBO was inferior to the WOA on functions f_{11}, f_{13}, and f_{14} and was outperfomred by the ISA, WOA, and ABC on function f_{12}. However, the ISA had the worst performance on functions f_7, f_8, f_{11}, f_{13}, and f_{14}, and the ABC provided the worst results for functions f_9 and f_{10}. These results indicate that the performances of the GBO and WOA methods are approximately equal in solving the multimodal functions. Note that the SD values for the

GBO are much better than those of the other algorithms, expect for function f_{14}. The results demonstrate that the good competency of the GBO to improve the exploration search in optimization.

3.4. Evaluation of the capability escaping from local optima

The hybrid and composite functions (f_{15} - f_{28}) were used to evaluate the ability of the GBO to escape from local optima in this research. These functions are known as the most challenging optimization problems because only an algorithm with a suitable balance between exploration and exploitation can escape from local optimal solutions. Table 7 shows the results of the six algorithms on the hybrid and composite functions.

For the hybrid functions (f_{15} - f_{20}) in Table 7, the average values of the objective function achieved for functions f_{15} - f_{17} and f_{19} over the 30 runs using the GBO are better than those achieved by the other algorithms, while the average values of the objective function obtained for functions f_{18} and f_{20} using the ISA and GWO, respectively, are better than those obtained by the GBO. In other words, the GBO is inferior to the ISA on function f_{18} and is outperformed by the GWO on function f_{20}. The best values of the objective function achieved for functions f_{15} - f_{16} and f_{19} - f_{20} over the 30 runs using the GBO are better than those obtained by the other algorithms. The GBO is outperformed by the ISA on the best values of the objective function for functions f_{17} - f_{18} over the 30 runs.

For the composite functions (f_{21} - f_{28}), Table 7 indicates that the GBO provided much better results on all composite function than the other algorithms. Since the composite functions are formed by some standard test functions, they have a randomly located global optimum and many deep local optima. From Table 7, it can be clearly observed that the GBO achieved more promising results than the GWO, CS, ABC, WOA, and ISA algorithms on functions f_{21} - f_{28}, due to the local escaping operator of the GBO that contributed to the exploration and assisted to escape from local optima effectively.

3.5. Ranking analysis

To select the prominent performance among the six algorithms, the Friedman and Quade tests [43] were performed. The Friedman test is a well-known, non-parametric test to determine the considerable difference in the efficiency between two or more samples. The null hypothesis in this test implies that there is equality of medians among the samples, while the alternative hypothesis explains the negation of the null hypothesis. The Quade test is used to for multiple comparisons. In contrast to the Friedman test, the Quade test is based on the assumption that some problems are more complex or important than others (in the Friedman test, all problems have an equal importance). Therefore, the computed rankings are scaled in regards to the differences specified between the samples [43].

Tables 8-9 show the rankings from the Friedman and Quade tests, including the individual, average, and final ranks for the average performances of the six algorithms on the unimodal, multimodal, hybrid and composite functions. The Friedman test results (Table 8) indicate that the GBO has the best rank on all test functions compared to the GWO, CS, ABC, WOA, and ISA algorithms, except for the multimodal functions. Note that the GBO was outperformed by the WOA algorithm on the multimodal functions. In the case of the Quade test, the GBO has the best rank on all test functions including the unimodal, multimodal, hybrid, and composite functions (Table 9). Table 10 illustrates the statistics and p-values of the Friedman and Quade tests. According to the p-values for the two tests, considerable differences can be observed among the six algorithms.

3.6. Evaluation of the convergence behavior

Generally, sudden changes in solutions would be expected at the early stages of the optimization [44], which can help an optimization algorithm to appropriately explore the search domain. Next, the changes in the solutions should be decreased to focus on the exploitation during the remaining stages of the optimization process. In this study, three metrics including the search history, trajectory variation, and convergence rate were used to evaluate the convergence behavior of the GBO algorithm.

In this regard, eight different test functions including f_2, f_4, f_8, f_{10}, f_{12}, f_{21}, f_{23}, and f_{25} with a dimension of 2, were selected. The GBO was applied to minimize these functions by utilizing five search agents (solutions) during 200 iterations. Fig. 5 illustrates the search history and the trajectory curves of the five solutions in their first dimension. It can be observed that the GBO successfully found the promising areas in the search domain and exploited the best position. The distribution density of the solutions in the search domain demonstrates how the GBO accounted for the exploration and exploitation. Obviously, the low distribution density demonstrates the exploration and the high distribution density illustrates the exploitation. Fig. 5 also indicates that the distribution of solutions is high in the area near to the global optimum and low in the areas far from the global optimum.

The trajectory graphs effectively display the exploration and exploitation behaviors of the optimization algorithms. Fig. 5 depict the trajectory curves of five solutions for the first dimension, indicating the high fluctuations at the early iterations. These variations are decreased with an increase in the number of iterations, and the positions of the solutions tend to move toward the global optimum at the later iterations. Apparently, the high fluctuations demonstrate the exploration search and the low fluctuations illustrate the exploitation search. It can be concluded from the trajectory curves that the GBO first implemented the exploration search and then the exploitation search.

The final aim of all optimization algorithms is to reach the global optimum accurately and rapidly. Thus, displaying this behavior is very important. The convergence curve is commonly used to evaluate the convergence efficiency of the optimization algorithms. As shown in Fig. 5, the convergence curves of functions f_2, f_4, f_8, f_{10}, f_{12}, f_{21}, and f_{25} are smooth and drop quickly, indicating that the GBO performed more efficiently in the exploitation than the exploration. In contrast, the convergence curve of function f_{23} is relatively rough and drops slowly, which demonstrates better performance of the GBO in the exploration than the exploitation. Thus, all convergence curves can precisely approximate the global optimum during the optimization process.

Figs. 6 and 7 depict the convergence curve variations of the six algorithms for different functions (unimodal and multi modal functions in Fig. 6, and hybrid and composite functions in Fig. 7). The y-axis shows the best-so-far objective function value explored, and the x-axis shows the number of function evaluations. From Figs. 6 and 7, the following conclusions can be reached:

1. In terms of the convergence speed, the ABC, CS, and ISA algorithms are the poorest to solve the unimodal and multimodal functions, followed by the GWO and WOA algorithms. The main cause for the weak performance of these algorithms is the imbalance between the exploration and exploitation searches. Note that the WOA algorithm has a suitable convergence speed on functions f_5, f_6, f_7, f_8, and f_9.

2. For the unimodal and multimodal functions, the GBO algorithm converged faster than the others, which can be attributed to the suitable balance between the exploration and exploitation searches in the GBO.

3. Except for the GBO algorithm, all other optimization algorithms have a slow convergence speed for solving the hybrid and composite functions. Note that the ISA algorithm had good performances on functions f_{15}, f_{18}, f_{20}, and f_{28}.

4. The variations of the convergence curves demonstrate that the GBO algorithm has a superior convergence speed to solve the test functions compared to the GWO, WOA, CS, ISA, and ABC algorithms.

4. Evaluation of the GBO algorithm on real-world engineering problems

Six engineering problems were optimized by utilizing the GBO, and the results were compared with those from the GWO, WOA, CS, ISA, and ABC algorithms. To achieve fair comparisons, the GBO and the other optimization algorithms were executed for 30 different runs. The population size and the maximum number of function evaluations were respectively 20 and 1000 for each problem.

4.1. Speed reducer problem

The main objective of the speed reducer problem is to minimize the weight of speed reducer (Fig. 8). This problem was explained in details in [45]. The mathematical formulas of the speed reducer problem are detailed in Appendix B.

Table 11 shows the statistical results of the GBO and the other algorithms. As shown in Table 11, the best value of the objective function is 2996.3481, which was achieved by the GBO, CS, and ISA algorithms. The optimal decision variables obtained by the GBO are listed in Table 12. The GBO has more suitable standard deviation values than the others. Note that the ISA and

CS algorithms have better performances than the GWO, ABC, and WOA algorithms. The results demonstrate that the proposed GBO can provide reliable and very comprising solutions compared with the other algorithms.

4.2. Three-bar truss problem

Minimizing the weight of three-bar truss is the main aim of the three-bar truss problem [46, 47]. The components of this problem are depicted in Fig. 9. The decision variables in this case are the cross-sectional area of the truss bars (x_{A1}, x_{A2}). The objective function and the constraints of this problem are detailed in Appendix B.

Table 13 shows the statistical results obtained by the GBO and the other algorithms. Clearly, the GBO provided more suitable results than the GWO, WOA, CS, ABC, and ISA algorithms. Table 13 indicates that the average value of the objective function obtained by the GBO (263.8959) over the 30 runs is better than those achieved by the others. In addition, the standard deviation of the objective function achieved by the GBO over the 30 independent runs is smaller than those of the other optimization algorithms. It should be noted that the ABC and ISA had higher efficiencies than the GWO, CS, and WOA algorithms. The optimal decision variables from the six algorithms are presented in Table 14, which again prove the superior ability of the GBO to solve complex engineering problems.

4.3. I-beam design problem

The performance of the GBO was assessed by using the I-beam design problem with four variables [47] (Fig. 10). Minimizing the vertical deflection of the I-beam is the main goal of this problem. The details on the objective function and the constraints of the problem can be found in Appendix B.

Table 15 shows the statistical results of the GBO algorithm and the other algorithms. It can be observed that the average of the objective function calculated by the GBO and CS algorithms are better than those of the other algorithms. In addition, the standard deviation for the proposed GBO (8.82E-18) is much smaller than those of the other optimization algorithms. The standard deviation for the CS (2.68E-12) is also better than those of the GWO, WOA, ABC, and ISA algorithms. The optimal variables of the problem are listed in Table 16. According to the results, it can be concluded that the GBO algorithm can provide a very competitive solution to this problem.

4.4. Cantilever beam problem

The cantilever beam problem is shown in Fig. 11. This problem includes five hollow blocks, so the number of variables is five [48]. The objective function and the constraints are detailed in Appendix B.

The statistical results of the objective function calculated by the six algorithms over 30 runs are litsed in Table 17. The proposed GBO provided the best values of the objective function and standard deviation. Table 18 shows the optimal values of the variables from the GBO and the other algorithms, demonstrating that the GBO effectively optimized this problem and yielded the best design.

4.5. Rolling element bearing design problem

In this problem, the objective function is to maximize the fatigue life. The fatigue life depends on the dynamic load-carrying capacity [49]. The problem has 10 decision variables and 9 constraints (See Fig. 12). The objective function and constraints of the problem are detailed in Appendix B.

The results of the objective function computed by the GBO and the five other algorithms over 30 runs are shown in Table 19. The proposed GBO yielded the best results on the average values of the objective function and standard deviation. The optimal values of the decision variables of all optimization algorithms are listed in Table 20. It can be observed from Table 20 that the proposed GBO more efficiently solved this problem than the other algorithms.

Since this problem is the most complicated one and has more decision variables and contraints than the other engineering design problems selected in this study, it was used to evaluate the computational efficiency of the proposed GBO. As aforementioned, this problem was run 30 times and the total computing time was calculated after these runs. Table 19 shows the computational times of all optimization algorithms, indicating that the GBO took shorter computational time than the ABC and CS algorithms, and longer time than the ISA, GWO, and WOA algorithms. This is due to the main formulas of the GBO used to update the positions of solutions based on the GSR and the DM. In addition, the proposed algorithm used two operators (i.e., GSR and LEO) to move toward the best solution. Therefore, it was expected that the GBO had an average performance in terms of the computational time compared to the other algorithms.

4.6. Tension/compression spring design problem

Minimizing the weight of tension/compression spring is the main objective of this problem [50]. The schematic of this problem is depicted in Fig. 13. The problem has 3 decision variables and 4 constraints and the detailed formulas of this problem are shown in Appendix B.

The comparison of the results of the GBO and other algorithms in Table 21 indicates that the GBO provided a suitable design with the minimum objective function for this problem.

According to the results of the GBO, CS, and ABC, the standard deviation values for these algorithms are almost equal, and also the best and average values of the objective function for these algorithms only have small differences. Therefore, these three algorithms are better than others for this problem. The optimal values obtained by the six algorithms are shown in Table 22, indicating that the GBO provided more promising results than the other optimization algorithms.

5. Conclusions

A novel population-based algorithm, GBO was proposed in this study. The GBO algorithm was derived from the gradient-based search method and used the Newton's method to explore the better regions in the search space. Two operators (i.e., gradient-based rule (GSR) and local escaping operator (LEO)) were introduced and mathematically formulated in the GBO to facilitate both exploration and exploitation searches. The Newton's method was used as a search engine in the GSR to strengthen the exploration and exploitation processes and the LEO was employed to avoid the local optimal solutions in the GBO. The performance of the GBO was evaluated by using 28 unimodal, multimodal, hybrid, and composite test functions. The excellent performance of the GBO on the unimodal functions and convergence demonstrated its enhanced capability of exploitation and improved convergence speed, which can be attributed to the use of the local search term in the *GSR* and the local escaping operator. The superb ability of exploration of the GBO in the test of the multimodal functions can be attributed to the exploration term used in the GSR and the global search term employed in the LEO. Furthermore, the results of the hybrid and composite functions demonstrated that the GBO properly balanced exploration and exploitation by employing the adaptive parameters.

The GBO was compared with five well-known and recent metaheuristic algorithms, including the GWO, WOA, ISA, CS, and ABC algorithms. The Friedman and Quade tests were performed for comparing the efficiencies of the algorithms. The results indicated that the GBO algorithm yielded very promising results and outperformed the other algorithms in the majority of the mathematical test functions, demonstrating the ability of the GBO as an alternative optimization algorithm to solve different problems. Furthermore, this study also examined the performances of the GBO in optimizing six engineering problems and compared with the other algorithms, which indicated that the GBO was able to optimize the real-world problems with challenging and unknown search domains. Based on the results of this study, the following conclusions can be reached:

1) The exploration in the GBO is guaranteed by the global search term in the *GSR*.
2) The GBO uses the local search term in the *GSR* and local escaping operator (LEO) to promote the exploitation in the search domain.
3) The proposed GBO algorithm is able to find local areas around a promising solution.
4) The transition from exploration to exploitation is smoothly implemented by using the adaptive parameters in the GBO.
5) The local escaping operator effectively avoids being trapped in local optima and improves the convergence speed of the GBO algorithm.
6) The GBO uses the direction of movement term to guide solutions toward the promising regions.
7) Easy implementation of the GBO and few parameter settings are the main advantages of this algorithm.

For the future studies, the proposed GBO can be applied to solve water resources management, structural engineering, and other real-world engineering problems. Moreover, the binary version of the GBO can be developed to solve the problems with a discrete search space, and the multi-objective version of the GBO can be utilized to optimize multi-objective problems. Some aspects of the GBO can be further improved in the future. For instance, the third-order modifications of the Newton's method can be employed and used in the GBO.

References

[1] X.-S. Yang, Nature-inspired metaheuristic algorithms, Luniver press, 2010.
[2] I. Ahmadianfar, Z. Khajeh, S.-A. Asghari-Pari, X. Chu, Developing optimal policies for reservoir systems using a multi-strategy optimization algorithm, Applied Soft Computing, 80 (2019) 888-903.
[3] A. Kaveh, V. Mahdavi, Colliding bodies optimization: a novel meta-heuristic method, Computers & Structures, 139 (2014) 18-27.
[4] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowledge-Based Systems, 96 (2016) 120-133.
[5] J.H. Holland, Genetic algorithms, Scientific american, 267 (1992) 66-73.
[6] R. Storn, K. Price, Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces, ICSI Berkeley, 1995.
[7] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the sixth international symposium on micro machine and human science, New York, NY, 1995, pp. 39-43.
[8] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of global optimization, 39 (2007) 459-471.
[9] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Information sciences, 179 (2009) 2232-2248.

[10] X.-S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature inspired cooperative strategies for optimization (NICSO 2010), Springer, 2010, pp. 65-74.
[11] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Advances in engineering software, 69 (2014) 46-61.
[12] A. Kaveh, A. Dadras, A novel meta-heuristic optimization algorithm: thermal exchange optimization, Advances in Engineering Software, 110 (2017) 69-84.
[13] W. Zhao, L. Wang, Z. Zhang, A novel atom search optimization for dispersion coefficient estimation in groundwater, Future Generation Computer Systems, 91 (2019) 601-610.
[14] O. Castillo, P. Melin, E. Ontiveros, C. Peraza, P. Ochoa, F. Valdez, J. Soria, A high-speed interval type 2 fuzzy system approach for dynamic parameter adaptation in metaheuristics, Engineering Applications of Artificial Intelligence, 85 (2019) 666-680.
[15] W. Zhao, L. Wang, Z. Zhang, Atom search optimization and its application to solve a hydrogeologic parameter estimation problem, Knowledge-Based Systems, 163 (2019) 283-304.
[16] L. Rodríguez, O. Castillo, J. Soria, P. Melin, F. Valdez, C.I. Gonzalez, G.E. Martinez, J. Soto, A fuzzy hierarchical operator in the grey wolf optimizer algorithm, Applied Soft Computing, 57 (2017) 315-328.
[17] O. Olorunda, A.P. Engelbrecht, Measuring exploration/exploitation in particle swarms using swarm diversity, in: 2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence), IEEE, 2008, pp. 1128-1134.
[18] V.K. Patel, V.J. Savsani, Heat transfer search (HTS): a novel optimization algorithm, Information Sciences, 324 (2015) 217-246.
[19] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Transactions on Evolutionary Computation, 13 (2009) 526-553.
[20] A. Draa, S. Bouzoubia, I. Boukhalfa, A sinusoidal differential evolution algorithm for numerical optimisation, Applied Soft Computing, 27 (2015) 99-126.
[21] H. Garg, A hybrid GSA-GA algorithm for constrained optimization problems, Information Sciences, 478 (2019) 499-523.
[22] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE transactions on evolutionary computation, 1 (1997) 67-82.
[23] T.J. Ypma, Historical development of the Newton–Raphson method, SIAM review, 37 (1995) 531-551.
[24] M.S. Bazaraa, H.D. Sherali, C.M. Shetty, Nonlinear programming: theory and algorithms, John Wiley & Sons, 2013.
[25] J.J. Moré, The Levenberg-Marquardt algorithm: implementation and theory, in: Numerical analysis, Springer, 1978, pp. 105-116.
[26] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, NBS Washington, DC, 1952.
[27] F. Salajegheh, E. Salajegheh, PSOG: Enhanced particle swarm optimization by a unit vector of first and second order gradient directions, Swarm and Evolutionary Computation, 46 (2019) 28-51.
[28] C. Ibtissem, L. Nouredine, A hybrid method based on conjugate gradient trained neural network and differential evolution for non linear systems identification, in: 2013 International Conference on Electrical Engineering and Software Applications, IEEE, 2013, pp. 1-5.

[29] N. Shahidi, H. Esmaeilzadeh, M. Abdollahi, E. Ebrahimi, C. Lucas, Self-adaptive memetic algorithm: an adaptive conjugate gradient approach, in: IEEE Conference on Cybernetics and Intelligent Systems, 2004., IEEE, 2004, pp. 6-11.
[30] K. Bandurski, W. Kwedlo, A Lamarckian hybrid of differential evolution and conjugate gradients for neural network training, Neural processing letters, 32 (2010) 31-44.
[31] A.K. Parwani, P. Talukdar, P. Subbarao, A hybrid approach using CGM and DE algorithm for estimation of boundary heat flux in a parallel plate channel, Numerical Heat Transfer, Part A: Applications, 65 (2014) 461-481.
[32] S.O. Madgwick, A.J. Harrison, R. Vaidyanathan, Estimation of IMU and MARG orientation using a gradient descent algorithm, in: 2011 IEEE international conference on rehabilitation robotics, IEEE, 2011, pp. 1-7.
[33] E. Aarts, J. Korst, Simulated annealing and Boltzmann machines, (1988).
[34] A. Kaveh, T. Bakhshpoori, Water evaporation optimization: a novel physically inspired optimization algorithm, Computers & Structures, 167 (2016) 69-85.
[35] R.V. Rao, V.J. Savsani, D. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Computer-Aided Design, 43 (2011) 303-315.
[36] C. Caraveo, F. Valdez, O. Castillo, A new optimization meta-heuristic algorithm based on self-defense mechanism of the plants with three reproduction operators, Soft Computing, 22 (2018) 4907-4920.
[37] F.A. Hashim, E.H. Houssein, M.S. Mabrouk, W. Al-Atabany, S. Mirjalili, Henry gas solubility optimization: A novel physics-based algorithm, Future Generation Computer Systems, 101 (2019) 646-667.
[38] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: Algorithm and applications, Future generation computer systems, 97 (2019) 849-872.
[39] S. Weerakoon, T. Fernando, A variant of Newton's method with accelerated third-order convergence, Applied Mathematics Letters, 13 (2000) 87-93.
[40] A.Y. Özban, Some new variants of Newton's method, Applied Mathematics Letters, 17 (2004) 677-682.
[41] P. Patil, U. Verma, Numerical Computational Methods. Alpha Science International Ltd, in, Oxford UK, 2006.
[42] Q. Huang, K. Zhang, J. Song, Y. Zhang, J. Shi, Adaptive differential evolution with a Lagrange interpolation argument algorithm, Information Sciences, 472 (2019) 180-202.
[43] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation, 1 (2011) 3-18.
[44] F. Van den Bergh, A.P. Engelbrecht, A study of particle swarm optimization particle trajectories, Information sciences, 176 (2006) 937-971.
[45] E. Mezura-Montes, C.A.C. Coello, Useful infeasible solutions in engineering optimization with evolutionary algorithms, in: Mexican International Conference on Artificial Intelligence, Springer, 2005, pp. 652-662.
[46] M.-Y. Cheng, D. Prayogo, Symbiotic organisms search: a new metaheuristic optimization algorithm, Computers & Structures, 139 (2014) 98-112.
[47] A.H. Gandomi, X.-S. Yang, A.H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Engineering with computers, 29 (2013) 17-35.

[48] H. Chickermane, H. Gea, Structural optimization using a new local approximation method, International journal for numerical methods in engineering, 39 (1996) 829-846.
[49] S. Gupta, R. Tiwari, S.B. Nair, Multi-objective design optimisation of rolling bearings using genetic algorithms, Mechanism and Machine Theory, 42 (2007) 1418-1443.
[50] C.A.C. Coello, E.M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Advanced Engineering Informatics, 16 (2002) 193-203.

Appendix A

See Tables 2-4

Appendix B

I- Speed reducer problem

$$Minimize\ Fitness = 0.7854x_1x_2^2 \times \left(3.3333 \times x_3^2 + 14.9334 \times x_3 - 43.0934\right) - 1.508 \times x_1 \times \left(x_6^2 + x_7^2\right) + 7.4777 \times \left(x_6^3 + x_7^3\right) + 0.7854 \times \left(x_4x_6^2 + x_5x_7^2\right)$$

Subject to

$$g_1(x) = \frac{27}{(x_1x_2^2 \times x_3)} - 1 \le 0$$

$$g_2(x) = \frac{397.5}{(x_1x_2^2 \times x_3^2)} - 1 \le 0$$

$$g_3(x) = \frac{1.93x_4^3}{(x_2x_3 \times x_6^4)} - 1 \le 0$$

$$g_4(x) = \frac{1.93x_5^3}{(x_2x_3 \times x_7^4)} - 1 \le 0$$

$$g_5(x) = \frac{1}{(110 \times x_6^3)} \times \sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6} - 1 \le 0$$

$$g_6(x) = \frac{1}{(85 \times x_7^3)} \times \sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6} - 1 \le 0$$

$$g_7(x) = \frac{x_2x_3}{40} - 1 \le 0$$

$$g_8(x) = 5\frac{x_2}{x_1} - 1 \le 0$$

$$g_9(x) = \frac{x_1}{12x_2} - 1 \le 0$$

$$g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0$$

$$g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0$$

II- Three-bar truss problem

Minimize $Fitness(\vec{x}) = (2\sqrt{2}x_{A1} + x_{A2}) \times l,$

Subject to: $g_1(\vec{x}) = \frac{\sqrt{2}x_{A1} + x_{A2}}{\sqrt{2}x_{A1}^2 + 2x_{A1}x_{A2}}P - \sigma \le 0,$

$g_2(\vec{x}) = \frac{x_{A2}}{\sqrt{2}x_{A1}^2 + 2x_{A1}x_{A2}}P - \sigma \le 0,$

$g_3(\vec{x}) = \frac{1}{\sqrt{2}x_{A2} + x_{A1}}P - \sigma \le 0,$

$0 \le x_{A1}, x_{A2} \le 1, l = 100\ cm,\ P = 2\frac{kN}{cm^2}, \sigma = 2\frac{kN}{cm^2}$

III- I-beam design problem

$$Minimize\quad Fitness = \frac{5000}{\frac{1}{12}t_w(h-2t_f)^3 + \frac{1}{6}lt_f^3 + 2lt_f\left(\frac{h-t_f}{2}\right)^2}$$

Subject to: $g_1(x) = 2lt_f + t_w(h-2t_f)^3 \le 300$

$$g_2(x) = \frac{180000x_1}{t_w(h-2t_f)^3 + 2lt_f\left[4t_f^2 + 3h(h-2t_f)\right]} + \frac{15000x_2}{(h-2t_f)t_w^3 + 2t_fl^3} \le 6$$

The variables are subject to:

$10 \le h \le 80,$

$10 \le l \le 50,$

$0.9 \le t_w \le 5,$

$0.9 \ge t_f \le 5,$

IV- Cantilever beam problem

$Minimize\quad Fitness = 0.0624 \times (x_1 + x_2 + x_3 + x_4 + x_5)$

Subject to:

$$g(x) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \le 0$$

Variable ranges

$0.01 \le x_1, x_2, x_3, x_4, x_5 \le 100$

V- Rolling element bearing design problem

$$Maximize\ \ Z = \begin{cases} f_c \times Z^{2/3} \times {D_b}^{1.8} & if\ D_b \leq 25.4 \\ 3.647 \times f_c \times Z^{2/3} \times {D_b}^{1.4} & otherwise \end{cases}$$

Subject to:

$$g_1(\vec{x}) = \frac{\varphi_0}{2sin^{-1}(\frac{D_b}{D_m})} - Z + 1 \leq 1,$$

$$g_2(\vec{x}) = 2D_b - K_{Dmin}(D - d) \geq 0,$$

$$g_3(\vec{x}) = K_{Dmax}(D - d) - 2D_b \geq 0,$$

$$g_4(\vec{x}) = \zeta B_w - D_b \leq 0,$$

$$g_5(\vec{x}) = D_m - 0.5 \times (D + d) \geq 0,$$

$$g_6(\vec{x}) = (0.5 + e) \times (D + d) - D_m \geq 0,$$

$$g_7(\vec{x}) = 0.5\big(D - D_m - D_b\big) - \varepsilon D_b \geq 0,$$

$$g_8(\vec{x}) = f_i \geq 0.515,$$

$$g_9(\vec{x}) = f_0 \geq 0.515,$$

where

$$f_c = 37.91\left[1 + \left(1.04\left(\frac{1+\gamma}{1-\gamma}\right)^{1.72}\left(\frac{f_i(2f_0 - 1)}{f_0(2f_i - 1)}\right)^{0.41}\right)^{\frac{10}{3}}\right]^{-0.3} \times \left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{\frac{1}{3}}}\right] \times \left[\frac{2f_i}{2f_i - 1}\right]^{0.41}$$

$$x = \left[\left\{\frac{(D-d)}{2} - 3\left(\frac{T}{4}\right)\right\}^2 + \left\{\frac{D}{2} - \frac{T}{4} - D_b\right\}^2 - \left\{\frac{d}{2} + \frac{T}{4}\right\}^2\right]$$

$$y = 2\left\{\frac{(D-d)}{2} - 3\left(\frac{T}{4}\right)\right\} \times \left\{\frac{D}{2} - \frac{T}{4} - D_b\right\}$$

$$\varphi_0 = 2\pi - cos^{-1}(\frac{x}{y})$$

$$B_w = 30,\ D = 160,\ d = 90,\ r_i\ = r_0 = 11.033$$

$$\gamma = \frac{D_b}{D_m}, f_i = \frac{r_i}{D_b}, f_i = \frac{r_0}{D_b},\ T = D - d - 2D_b$$

$$0.15(D-d) \leq D_b \leq 0.45(D-d), 4 \leq Z \leq 50,\ 0.515 \leq f_i, f_0 \leq 0.60$$

$$0.4 \leq K_{Dmin} \leq 0.5, 0.6 \leq K_{Dmax} \leq 0.7, 0.3 \leq \varepsilon \leq 0.4, 0.02 \leq e \leq 0.1$$

$0.6 \leq \zeta \leq 0.85$

VI- Tension/compression spring design problem

Consider $\vec{x} = [m, D_c, d_w]$

Minimize $Z(\vec{x}) = (d_w + 2)D_c\mathrm{m}^2$

Subject to: $g_1(\vec{x}) = 1 - \frac{{D_c}^3 d_w}{71785m^4} \leq 0,$

$$g_2(\vec{x}) = \frac{4{D_c}^2 - md_w}{12566(D_c m^3 - m^4)} + \frac{1}{5108m^2} \leq 0,$$

$$g_3(\vec{x}) = 1 - \frac{140.45m}{{D_c}^2 d_w} \leq 0,$$

$$g_4(\vec{x}) = \frac{m + D_c}{1.5} - 1 \leq 0,$$

$0.05 \leq m \leq 2.00,$

$0.25 \leq D_c \leq 1.30,$

$2.00 \leq d_w \leq 15.00,$

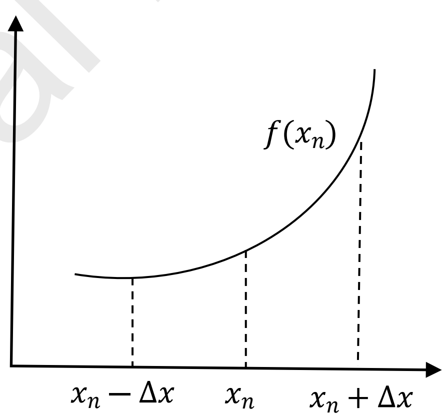


Fig. 1. Gradient estimation using x_n and its neighboring positions

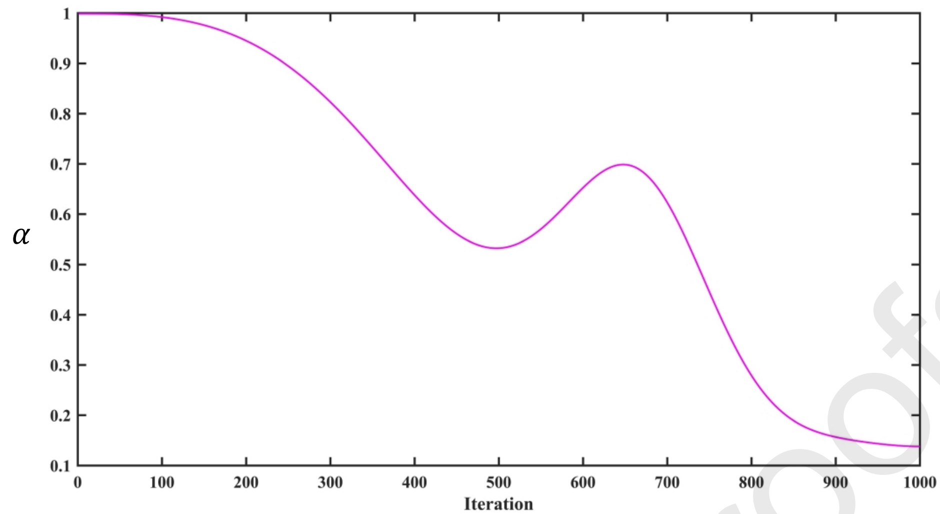


Fig. 2. Variation of α parameter over course of iterations

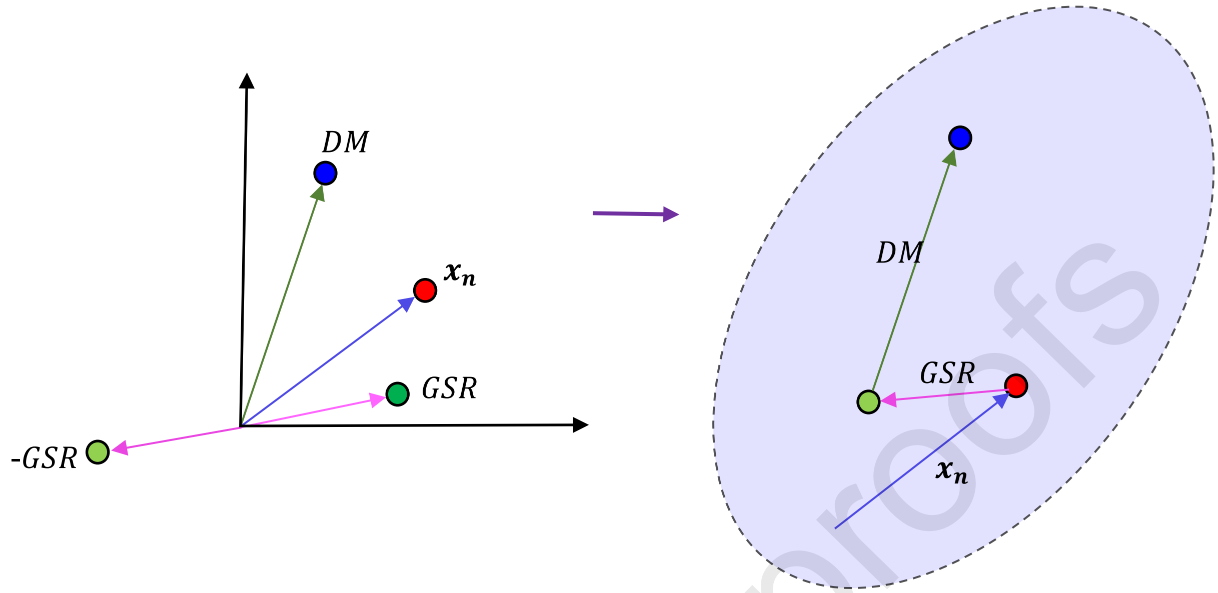


Fig. 3. Updating the current position x_n

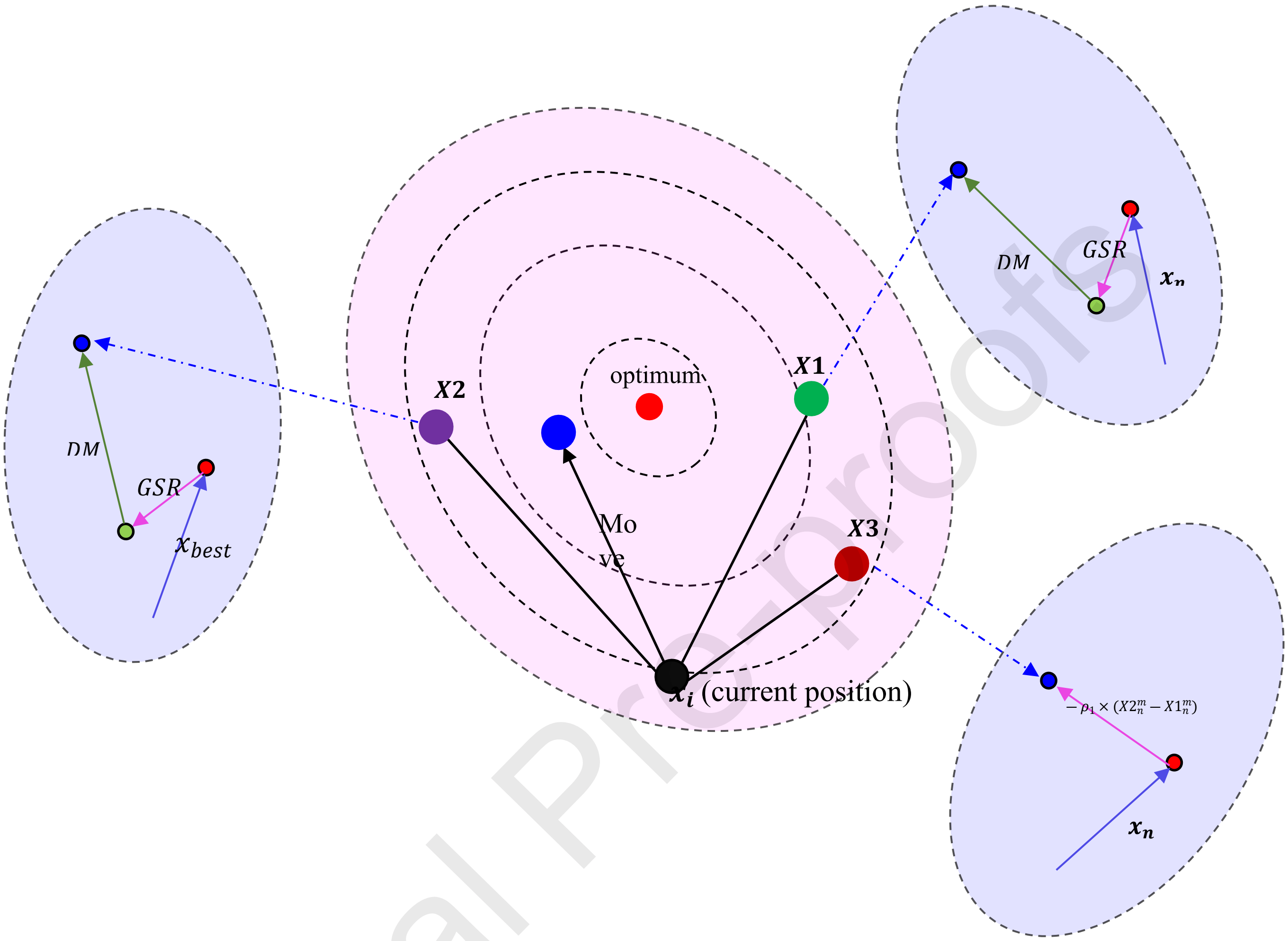


Fig. 4. Sketch map of the GBO algorithm

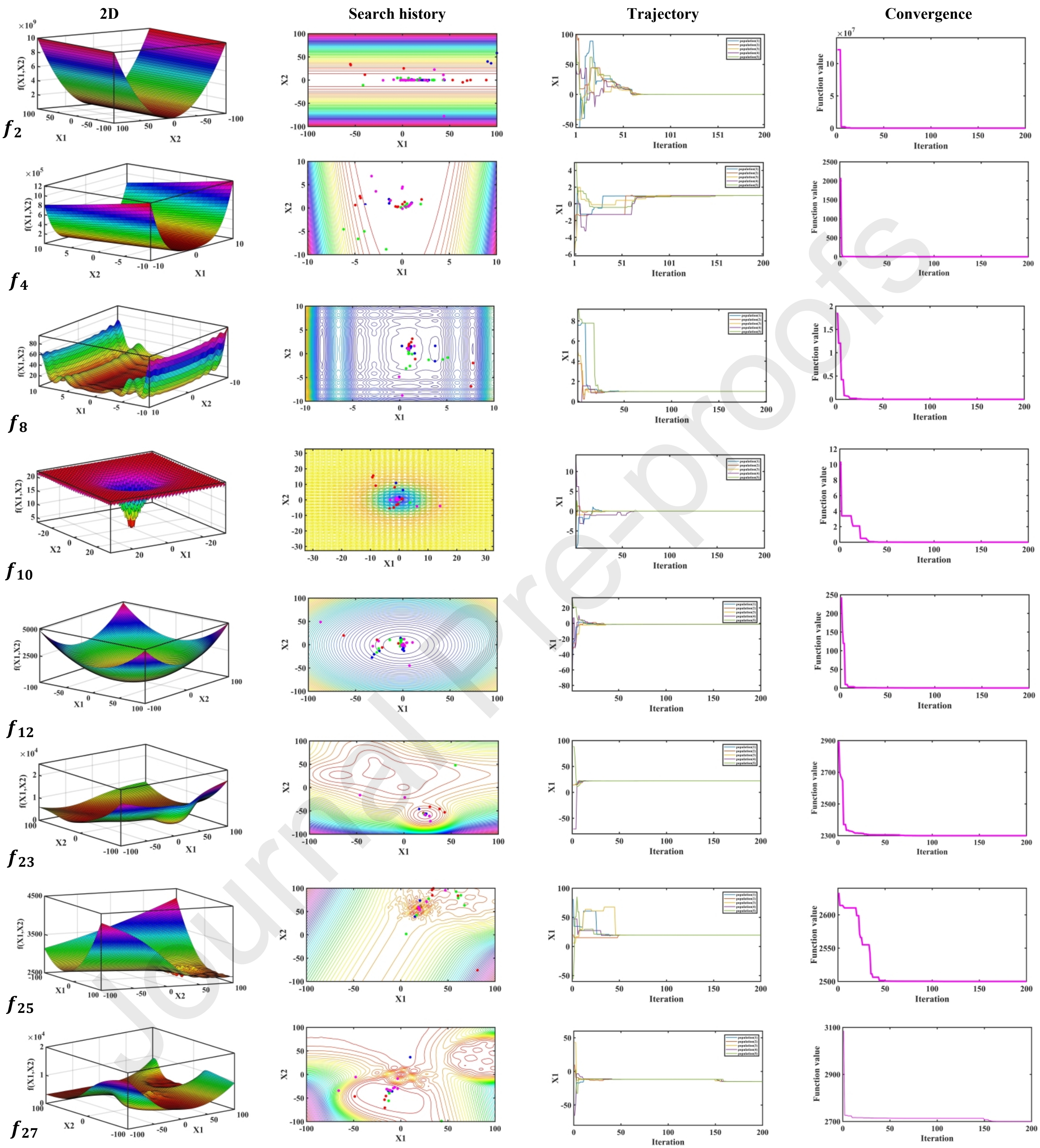


Fig. 5. 2D representation, search history, trajectory, and convergence curve of eight functions

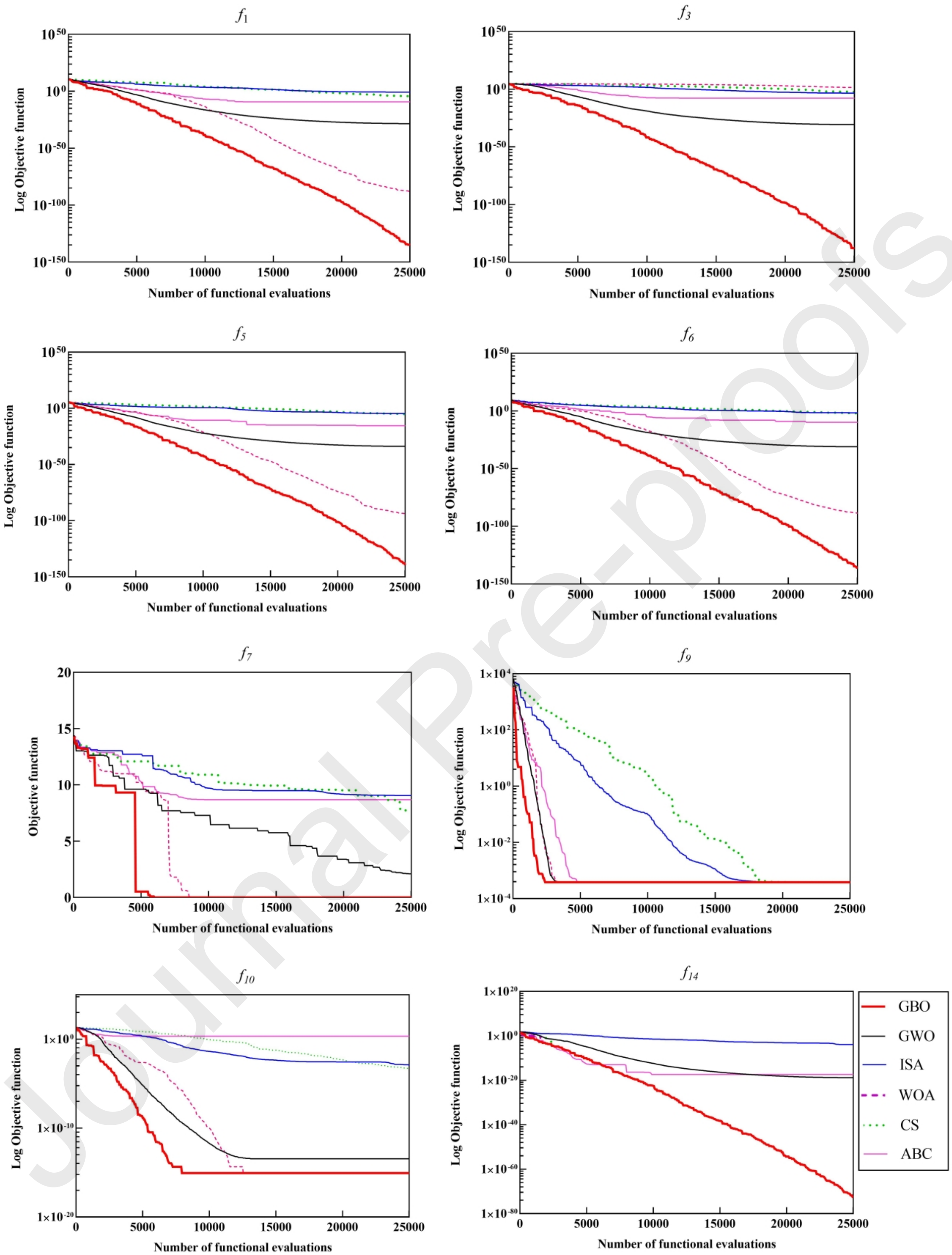


Fig. 6. Convergence curves of the GBO and the other algorithms achieved in some of unimodal and multimodal functions

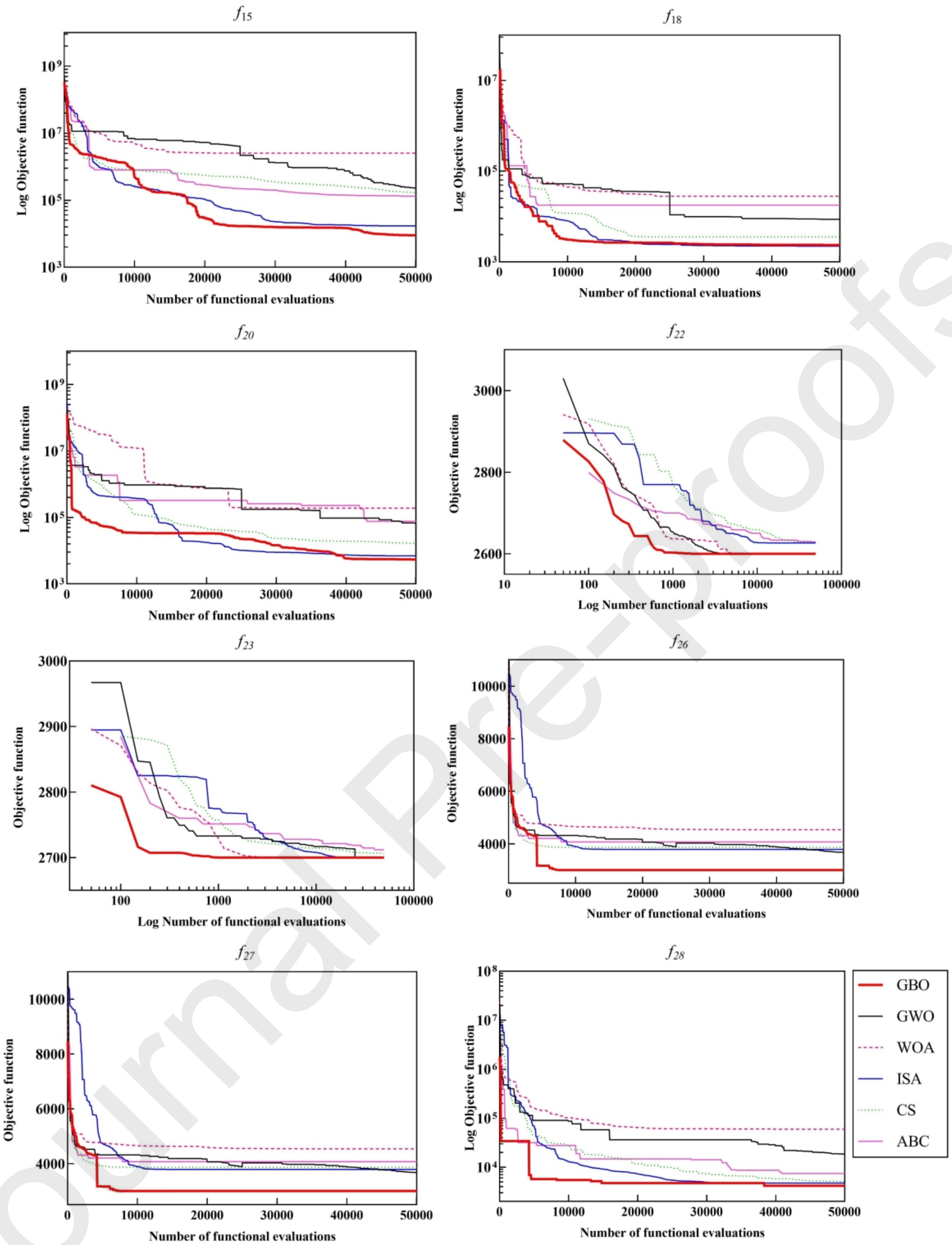


Fig. 7. Convergence curves of the GBO and the other algorithms achieved in some of hybrid and composite functions

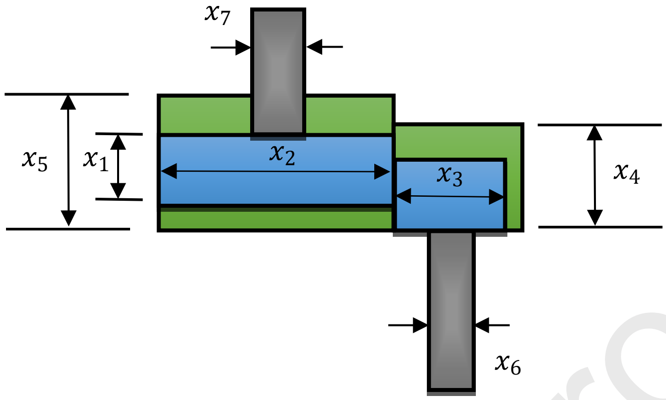


Fig. 8. Speed reducer problem

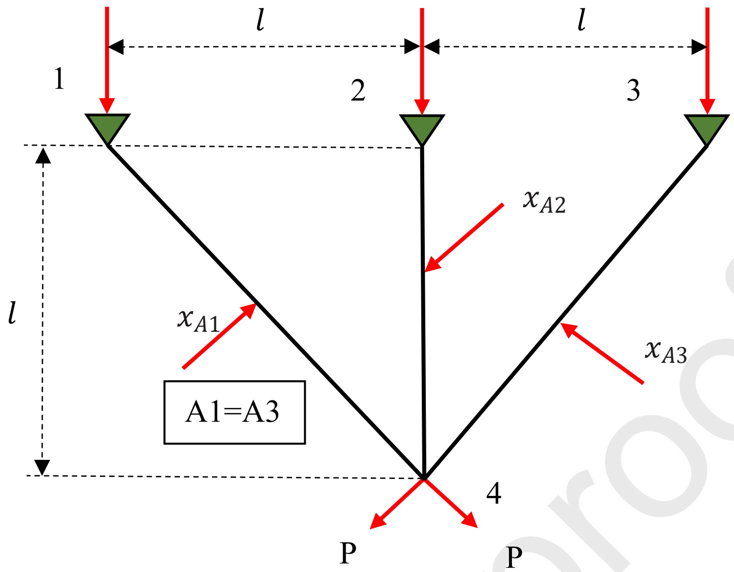


Fig. 9. Three-bar truss problem

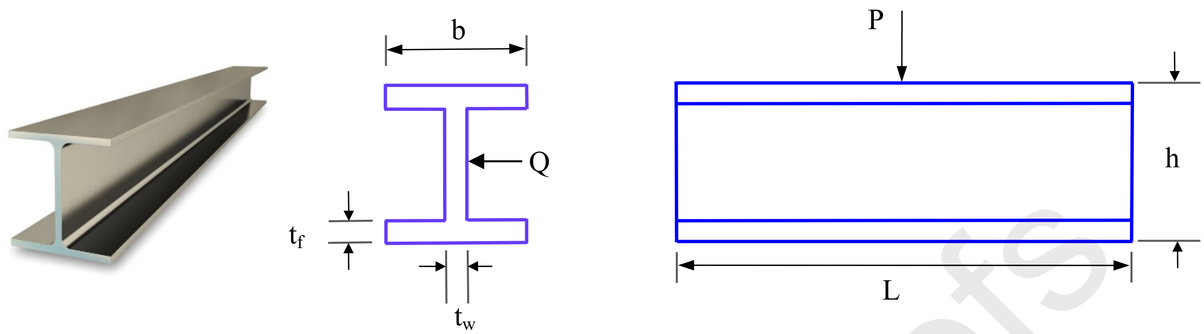


Fig. 10. I-beam design problem

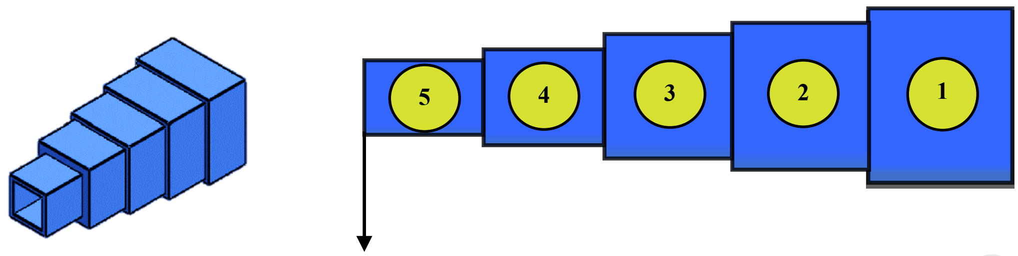


Fig. 11. Cantilever beam problem

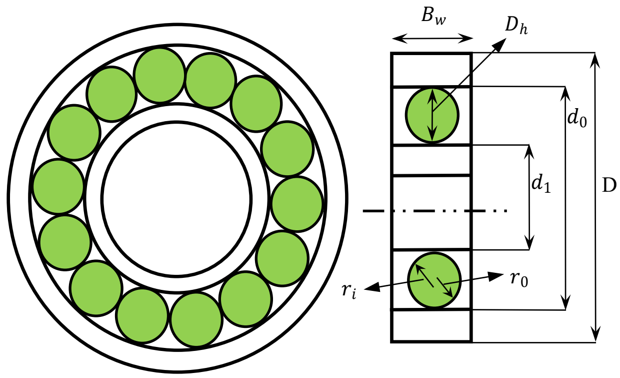


Fig. 12. Shape of Rolling element bearing design problem

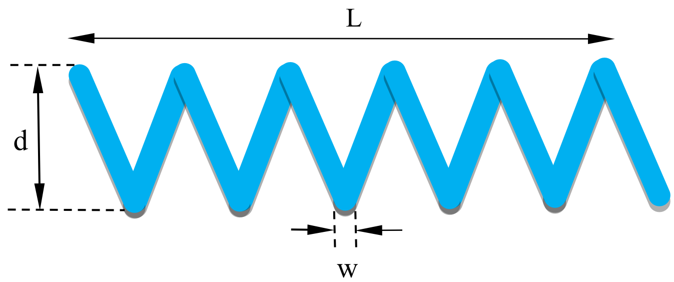


Fig. 13. Shape of Tension/compression spring problem

Table 1 Pseudo code of the GBO algorithm

Step 1. Initialization

Assign values for parameters pr, ε, and M
Generate an initial population $X_0 = [x_{0,1}, x_{0,2}, ..., x_{0,D}]$
Evaluate the objective function value $f(X_0),\ n = 1, ..., N$
Specify the best and worst solutions $x_{best}^m$and x_{worst}^m

Step 2. Main loop

```
While (m<M)
    for n = 1 : N
        for i = 1 : D
            Select randomly r1 ≠ r2 ≠ r3 ≠ r4 ≠ n in the range of [1, N]
            Calculate the position x_{n,i}^{m+1} using Eq. 27
        end for
        Local escaping operator
        if rand < pr
            Calculate the position x_{LEO}^m using Eq. 28
            X_n^{m+1} = x_{LEO}^m
        end
      Update the positions x_{best}^m and x_{worst}^m
    end for
  m=m+1
end
```

Step 3. return x_{best}^m

Table 2. Unimodal test functions.

Function	D	Range	f_{min}
$f_1(x) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2$	30	[-100, 100]	0
$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert^{i+1}$	30	[-100, 100]	0
$f_3(x) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^4$	30	[-100, 100]	0
$\mathrm{f}_4(\mathrm{x}) = \sum_{\mathrm{i}=1}^{\mathrm{D}-1} [100(\mathrm{x}_{\mathrm{i}+1} - \mathrm{x}_{\mathrm{i}}^2)^2 + (\mathrm{x}_{\mathrm{i}} - 1)^2]$	30	[-100, 100]	0
$f_5(x) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2$	30	[-100, 100]	0
$f_6(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$	30	[-100, 100]	0

Table 3. Multimodal test functions.

Function	D	Range	f_{min}
$f_7(x) = g(x_1, x_2) + g(x_2, x_3) + ... + g(x_{D-1}, x_D) + g(x_D, x_1)$ $g(x,y) = 0.5 + \frac{(sin^2(\sqrt{x^2+y^2}) - 0.5)}{(1+0.001(x^2+y^2))^2}$	30	[-100, 100]	0
$f_8(x) = sin^2(\pi w_1) + \sum_{i=1}^{D-1}(w_i - 1)^2[1 + 10sin^2(\pi w_i + 1)] + (w_D - 1)^2[1 + sin^2(2\pi w_D)]$ where $w_i = 1 + \frac{x_i - 1}{4}$	30	[-100, 100]	0
$f_9(x) = 418.9829 \times D - \sum_{i=1}^{D} g(z_i), \quad z_i = x_i + 4.209687462275036e+002$ $g(z_i) = \begin{cases} z_i \sin\left(\lvert z_i\rvert^{\frac{1}{2}}\right) & if \lvert z_i\rvert \le 500 \\ (500 - mod(z_i, 500))\sin\left(\sqrt{\lvert 500 - mod(z_i, 500)\rvert}\right) - \frac{(z_i - 500)^2}{10000D} & if z_i > 500 \\ (mod(\lvert z_i\rvert, 500) - 500)\sin\left(\sqrt{\lvert mod(\lvert z_i\rvert, 500) - 500\rvert}\right) - \frac{(z_i + 500)^2}{10000D} & if z_i < -500 \end{cases}$	30	[-100, 100]	0
$f_{10}(x) = 20 - 20 \times \exp\left(-0.2\sqrt{\frac{1}{D}(\sum_{i=1}^{D} x_i^2)}\right) - \exp\left(\frac{1}{D}\sqrt{\sum_{i=1}^{D} cox(2\pi x_i)}\right) + e$	30	[-32, 32]	0
$f_{11}(x) = \sum_{i=1}^{D}(\sum_{k=0}^{kmax}[a^k \cos(2\pi b^k(x_i + 0.5))])$ - $D\sum_{k=0}^{kmax}[a^k \cos(2\pi b^k.0.5)]$ $a = 0.5, b = 3, kmax = 20$	30	[-100, 100]	0
$f_{12}(x) = \lvert\sum_{i=1}^{D} x_i^2 - D\rvert^{\frac{1}{4}} + (0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i)/D + 0.5$	30	[-100, 100]	0
$f_{13}(x) = \lvert(\sum_{i=1}^{D} x_i^2)^2 - (\sum_{i=1}^{D} x_i)^2\rvert^{\frac{1}{2}} + (0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i)/D + 0.5$	30	[-100, 100]	0
$f_{14}(x) = \sum_{i=2}^{D}\lvert x_i \sin(x_i) + 0.1x_i\rvert$	30	[0, 100]	0

Table 4. Hybrid and composite mathematical benchmark functions

Function	Name	*D*	Range	f_{min}
$f_{15}(x)$	Hybrid Function 1 (*N*=3)	30	[-100, 100]	1700
$f_{16}(x)$	Hybrid Function 2 (*N*=3)	30	[-100, 100]	1800
$f_{17}(x)$	Hybrid Function 3 (*N*=4)	30	[-100, 100]	1900
$f_{18}(x)$	Hybrid Function 4 (*N*=4)	30	[-100, 100]	2000
$f_{19}(x)$	Hybrid Function 5 (*N*=5)	30	[-100, 100]	2100
$f_{20}(x)$	Hybrid Function 6 (*N*=5)	30	[-100, 100]	2200
$f_{21}(x)$	Composite Function 1 (*N*=5)	30	[-100, 100]	2300
$f_{22}(x)$	Composite Function 2 (*N*=3)	30	[-100, 100]	2400
$f_{23}(x)$	Composite Function 3 (*N*=3)	30	[-100, 100]	2500
$f_{24}(x)$	Composite Function 4 (*N*=5)	30	[-100, 100]	2600
$f_{25}(x)$	Composite Function 5 (*N*=5)	30	[-100, 100]	2700
$f_{26}(x)$	Composite Function 6 (*N*=5)	30	[-100, 100]	2800
$f_{27}(x)$	Composite Function 7 (*N*=3)	30	[-100, 100]	2900
$f_{28}(x)$	Composite Function 8 (*N*=3)	30	[-100, 100]	3000

Table 5. Control parameters of six algorithms

Algorithms	Parameters
GWO	a parameter that reduces linearly from 2 to 0 (Default)
WOA	a parameter that reduces linearly from 2 to 0 (Default)
	a parameter that reduces linearly from -1 to -2 (Default)
CS	discovery rate of alien eggs/solutions = 0.25
ABC	acceleration coefficient reduces exponentially from 2 to 0
ISA	Scale factor = 0.001
GBO	$\beta_{min} = 0.2$, $\beta_{max} = 1.2$
	$pr = 0.5$

Table 6. Results of the unimodal and multimodal test functions

Algorithm	Criteria	Unimodal functions					
		f_1	f_2	f_3	f_4	f_5	f_6
GBO	Best	1.26E-135	2.33E-206	1.50E-138	1.98E+01	3.92E-140	1.35E-136
	Average	**1.46E-125**	**3.29E-193**	**2.40E-128**	**2.16E+01**	**8.86E-131**	**9.61E-129**
	SD	7.96E-125	0.00E+00	1.21E-127	8.03E-01	4.07E-130	4.92E-128
GWO	Best	4.33E-29	2.79E-108	2.25E-31	2.52E+01	1.61E-34	1.12E-31
	Average	3.87E-27	4.17E-97	5.78E-29	2.68E+01	5.60E-33	5.14E-30
	SD	7.73E-27	1.87E-96	1.48E-28	7.53E-01	5.84E-33	8.14E-30
CS	Best	4.44E-05	1.46E-06	5.38E-03	2.96E+01	6.67E-06	1.22E-02
	Average	2.52E-02	1.81E+01	9.00E-01	1.39E+02	5.16E-04	1.88E-01
	SD	1.17E-01	8.44E+01	1.70E+00	2.37E+02	7.63E-04	3.04E-01
ABC	Best	6.25E-10	1.90E-76	2.11E-08	3.97E+01	4.06E-16	1.48E-10
	Average	1.77E-02	3.76E-54	1.32E+00	6.93E+01	1.56E-08	6.39E+00
	SD	6.49E-02	2.06E-53	2.68E+00	5.50E+01	7.60E-08	3.50E+01
WOA	Best	9.43E-89	9.17E-141	2.88E+01	2.69E+01	2.63E-94	2.90E-89
	Average	6.75E-80	1.56E-110	5.52E+03	2.75E+01	2.86E-84	1.30E-81
	SD	2.45E-79	7.86E-110	3.85E+03	4.12E-01	1.11E-83	5.59E-81
ISA	Best	2.94E+00	6.76E-09	4.16E-04	2.35E+01	2.54E-05	2.80E-02
	Average	9.87E+01	1.61E-01	1.54E-02	7.56E+01	1.50E-01	6.14E+01
	SD	1.92E+02	6.00E-01	2.88E-02	5.18E+01	7.42E-01	1.65E+02

Algorithm	Criteria	Multimodal functions							
		f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
GBO	Best	0.00E+00	4.60E-09	3.82E-04	8.88E-16	1.35E-13	3.46E-01	4.06E-01	2.95E-73
	Average	**0.00E+00**	**2.96E-07**	**3.82E-04**	**8.88E-16**	1.97E-13	5.31E-01	4.24E-01	6.45E-69
	SD	0.00E+00	8.45E-07	**0.00E+00**	0.00E+00	3.62E-14	1.72E-01	1.17E-02	1.99E-68
GWO	Best	2.11E+00	6.36E-01	3.82E-04	3.64E-14	2.27E+01	4.41E-01	3.43E-01	1.64E-19
	Average	5.91E+00	1.01E+00	3.82E-04	4.46E-14	2.91E+01	6.39E-01	4.65E-01	4.27E-04
	SD	2.20E+00	1.59E-01	8.72E-13	4.19E-15	3.34E+00	9.60E-02	7.24E-02	5.72E-04
CS	Best	7.74E+00	6.28E-01	3.82E-04	4.69E-04	8.53E-14	4.42E-01	3.18E-01	2.60E-05
	Average	9.86E+00	2.41E+00	4.12E-04	3.73E-03	6.23E-02	5.93E-01	4.54E-01	2.64E-02
	SD	8.36E-01	2.27E+00	4.54E-05	3.44E-03	9.52E-02	8.40E-02	1.47E-01	2.69E-02
ABC	Best	8.69E+00	4.49E-01	3.82E-04	2.22E+00	3.79E+00	2.64E-01	2.25E-01	4.20E-18
	Average	1.05E+01	3.84E+00	9.84E+01	4.90E+00	9.29E+00	5.19E-01	5.78E-01	1.74E-03
	SD	9.07E-01	3.98E+00	1.66E+02	1.51E+00	3.89E+00	1.84E-01	2.71E-01	9.45E-03
WOA	Best	0.00E+00	6.99E-02	3.82E-04	8.88E-16	7.11E-15	2.60E-01	1.21E-01	0.00E+00
	Average	3.00E+00	5.12E-01	3.82E-04	3.73E-15	**1.92E-14**	5.24E-01	**3.84E-01**	**0.00E+00**
	SD	4.43E+00	3.58E-01	5.55E-13	2.70E-15	6.62E-14	1.88E-01	9.68E-02	0.00E+00
ISA	Best	9.06E+00	4.72E+00	3.83E-04	1.33E-03	3.46E+01	3.31E-01	2.54E-01	1.27E-04
	Average	1.09E+01	3.42E+01	1.87E-03	9.27E-01	3.89E+01	**4.63E-01**	6.42E-01	6.15E-01
	SD	8.96E-01	2.25E+01	5.08E-03	8.13E-01	1.71E+00	9.80E-02	2.96E-01	1.22E+00

Table 7. Results of the hybrid and composite test functions

Algorithm	Criteria	Hybrid functions					
		f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
GBO	Best	8935.40	1882.27	1908.72	2359.86	5380.12	2222.16
	Average	**54281.78**	3842.48	**1913.11**	3057.48	**25618.08**	2653.37
	SD	42432.48	2399.27	3.92	744.79	18468.09	204.78
GWO	Best	231111.70	5408.63	1912.26	8718.14	66706.84	2250.33
	Average	1779929.97	7749192.43	1945.42	25284.54	865855.49	**2581.81**
	SD	1644067.84	17864871.76	26.45	14344.57	1222558.84	145.41
CS	Best	168986.27	2070.91	1909.39	3577.19	16508.82	2364.87
	Average	1638591.37	8614.09	1931.73	94953.78	405641.76	3114.17
	SD	1608329.34	8165.00	30.62	309592.19	577986.74	364.57
ABC	Best	233476.61	2363.39	1910.94	17929.12	76522.99	2458.79
	Average	658957.42	4103.94	1914.55	32002.21	200710.29	2717.80
	SD	285149.46	1715.37	7.60	7762.82	68430.95	352.07
WOA	Best	2520022.97	9512.03	1919.07	28141.42	189834.25	2476.51
	Average	11178976.28	93612.11	1964.90	76381.26	3876550.62	3084.20
	SD	7349962.08	94864.91	34.80	48244.50	4182086.86	252.11
ISA	Best	20437.35	1902.304	1906.027	2234.03	6951.55	2371.626
	Average	85225.05	**3020.392**	1916.644	**2864.17**	28591.60	2659.081
	SD	51554.55	1325.382	19.50	747.64	23138.94	178.69

Algorithm	Criteria	Composite functions							
		f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}	f_{28}
GBO	Best	2500.00	2600.00	2700.00	2700.24	2900.00	3000.00	3130.97	4164.96
	Average	**2500.00**	**2600.00**	**2700.00**	**2700.51**	**2900.00**	**3034.06**	**3450.88**	**5322.75**
	SD	0.00	0.00	0.00	0.17	0.00	186.56	643.65	892.78
GWO	Best	2621.18	2600.01	2700.00	2700.33	3123.36	3675.15	7501.77	13904.34
	Average	2636.05	2600.02	2711.44	2737.11	3379.48	4045.94	308528.23	53939.59
	SD	11.16	0.01	4.47	48.71	111.60	346.48	600531.00	29108.60
CS	Best	2615.24	2628.98	2705.81	2700.51	3105.55	3861.53	5335.67	5041.06
	Average	2627.92	2648.17	2715.34	2705.09	3634.37	4481.44	9069.06	7161.88
	SD	33.34	8.01	7.43	18.32	326.63	364.02	2236.96	1256.38
ABC	Best	2615.27	2628.99	2712.05	2700.49	3123.23	4068.54	10795.11	7427.97
	Average	2615.66	2633.06	2714.83	2700.72	3138.99	4440.74	20057.04	9638.17
	SD	1.82	2.35	1.82	0.22	9.48	246.10	5239.48	1593.26
WOA	Best	2633.57	2601.39	2700.00	2700.27	3128.00	4534.27	11837.45	59603.38
	Average	2665.19	2609.03	2717.65	2710.48	3705.99	5499.60	7750877.69	241296.57
	SD	16.51	5.05	20.14	30.38	417.89	604.76	5275879.13	156977.31
ISA	Best	2615.244	2626.773	2700	2700.182	3101.926	3783.245	3946.881	4712.596
	Average	2615.245	2631.823	2712.202	2703.647	3156.223	4623.885	5458.514	6365.478
	SD	0.002225	4.186077	4.675988	18.21034	113.0097	730.5406	1915.738	1120.53

Table 8. Friedman ranks for the unimodal, multimodal, hybrid, and composite test functions

Algorithms	Unimodal functions						Average Rank	Rank
	f_1	f_2	f_3	f_4	f_5	f_6		
GBO	1	1	1	1	1	1	**1.00**	**1**
GWO	3	3	2	2	3	3	2.67	2
CS	5	5	4	5	5	4	4.67	5
ABC	4	4	3	4	4	5	4.00	4
WOA	2	2	6	3	2	2	2.83	3
ISA	6	6	5	6	6	6	5.83	6

Algorithms	Multimodal functions									
	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}		
GBO	1	1	2	1	1	1	4	1.5	**1.56**	**1**
GWO	3	3	2	3	6	6	5	3	3.88	4
CS	4	4	4	4	3	5	2.5	4	3.81	3
ABC	5	5	5	6	4	4	6	5	5.00	6
WOA	2	2	2	2	2	3	1	1.5	1.94	2
ISA	6	6	6	5	5	2	2.5	6	4.81	5

Algorithms	Hybrid functions							
	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}		
GBO	1	2	2	2	1	2	**1.67**	**1**
GWO	5	6	5	3	5	1	4.17	4
CS	4	4	4	6	4	6	4.67	5
ABC	3	3	3	4	3	4	3.33	3
WOA	6	5	6	5	6	5	5.50	6
ISA	2	1	3	1	2	3	2.00	2

Algorithms	Composite functions									
	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}	f_{28}		
GBO	1	1	1	1	1	1	1	1	**1.00**	**1**
GWO	5	2	2	6	4	2	5	5	3.88	4
CS	4	6	5	4	5	4	3	3	4.25	5
ABC	3	5	4	2	2	3	4	4	3.38	3
WOA	6	3	6	5	6	6	6	6	5.50	6
ISA	2	4	3	3	3	5	2	2	3.00	2

Table 9. Quade ranks for the unimodal, multimodal, hybrid, and composite test functions

Algorithms	Unimodal functions						Average Rank	Rank
	f_1	f_2	f_3	f_4	f_5	f_6		
GBO	4	2	6	5	1	3	**1.00**	**1**
GWO	12	6	12	10	3	9	2.48	2
CS	20	12	24	30	5	12	4.90	6
ABC	16	8	30	20	4	15	4.43	4
WOA	8	4	36	15	2	6	3.38	3
ISA	24	10	18	25	6	18	4.81	5

Algorithms	Multimodal functions									
	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}		
GBO	5	6	16	4	14	4	4	6	**1.64**	**1**
GWO	15	18	16	12	35	6	8	9	3.31	3
CS	20	24	32	16	21	5	6	15	3.86	4
ABC	25	30	48	24	28	2	10	12	4.97	5
WOA	10	12	16	8	7	3	2	3	1.69	2
ISA	30	36	40	20	42	1	12	18	5.53	6

Algorithms	Hybrid functions							
	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}		
GBO	6	10	1	6	4	4	**1.48**	**1**
GWO	30	30	5	9	20	2	4.57	5
CS	24	20	4	18	16	12	4.48	4
ABC	18	15	2	12	12	8	3.19	3
WOA	36	25	6	15	24	10	5.52	6
ISA	12	5	3	3	8	6	1.76	2

Algorithms	Composite functions									
	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}	f_{28}		
GBO	4	3	1	2	5	6	8	7	**1.00**	**1**
GWO	20	6	2	12	20	12	40	35	4.08	5
CS	16	18	5	8	25	24	24	21	3.92	4
ABC	12	15	4	4	10	18	32	28	3.42	3
WOA	24	9	6	10	30	36	48	42	5.69	6
ISA	8	12	3	6	15	30	16	14	2.89	2

Table 10. Statistics and *p*-values calculated by the Friedman and Quade tests for the unimodal, multimodal, hybrid, and composite functions

	Average ranking	
	Friedman	Quade
	Unimodal functions	
Statistic	22.47	6.05
p-value	4.25E-04	0.0008
	Multimodal functions	
Statistic	22.89	11.41
p-value	3.53E-04	0.00
	Hybrid functions	
Statistic	20.85	8.11
p-value	8.62E-04	0.0001
	Composite functions	
Statistic	25.64	8.30
p-value	1.05E-04	0.00

Table 11. Comparison of statistical results for the speed reducer problem

	GBO	CS	ABC	GWO	ISA	WOA
Best	2996.3481	2996.3481	2996.6383	2998.0976	2996.3481	3006.8794
Mean	**2996.3481**	2996.3485	2996.8856	3003.0686	**2996.3481**	3032.2744
SD	2.4736E-11	4.3014E-04	1.6207E-01	3.1431E+00	2.1806E-07	2.6815E+01

Table 12. Best solutions achieved by the six algorithms for the speed reducer problem

Algorithm	x_1	x_2	x_3	x_4	x_5	x_6	x_7
GBO	3.4999	0.70	17.00	7.30	7.80	3.3502	5.2866
GWO	3.5000	0.70	17.00	7.38	7.81	3.3504	5.2867
WOA	3.5000	0.70	17.00	8.03	7.91	3.3600	5.2850
CS	3.4999	0.70	17.00	7.30	7.80	3.3502	5.2866
ISA	3.4999	0.70	17.00	7.30	7.80	3.3502	5.2866
ABC	3.5001	0.70	17.00	7.30	7.80	3.3504	5.2868

Table 13. Comparison of statistical results for the three-bar truss problem

	GBO	CS	ABC	GWO	ISA	WOA
Best	263.8958	263.8958	263.8962	263.8960	263.8958	263.8974
Mean	**263.8959**	264.5276	263.9016	263.9002	263.8970	265.0745
SD	1.600E-04	3.459E+00	4.678E-03	4.437E-03	3.720E-03	2.005E+00

Table 14. Best solutions achieved by the six algorithms for the three-bar truss problem

Algorithm	x_1	x_2
GBO	0.788693	0.408197
GWO	0.788587	0.408499
WOA	0.787221	0.412378
CS	0.788619	0.408407
ISA	0.788720	0.408123
ABC	0.788304	0.409301

Table 15. Comparison of statistical results for the I-beam design problem

	GBO	CS	ABC	GWO	ISA	WOA
Best	0.013074	0.013074	0.013074	0.013074	0.013074	0.013074
Mean	**0.013074**	**0.013074**	0.013075	0.013081	0.013082	0.014118
SD	8.8219E-18	2.6848E-12	4.8552E-07	6.8688E-06	3.0950E-05	1.3340E-03

Table 16. Best solutions achieved by the six algorithms for the I-beam design problem

Algorithm	h	l	t_w	t_f
GBO	50.00	80.00	0.90	2.3217
GWO	49.99	80.00	0.90	2.3179
WOA	50.00	80.00	0.90	2.3217
CS	50.00	80.00	0.90	2.3217
ISA	50.00	80.00	0.90	2.3217
ABC	50.00	80.00	0.90	2.3217

Table 17. Comparison of statistical results for the cantilever beam problem

	GBO	CS	ABC	GWO	ISA	WOA
Best	1.339957	1.340332	1.340043	1.339970	1.339958	1.360547
Mean	**1.339970**	1.342806	1.340257	1.340070	1.340265	1.532439
SD	1.80E-05	1.70E-03	1.37E-04	8.68E-05	2.57E-04	1.35E-01

Table 18. Best solutions achieved by the six algorithms for the cantilever beam problem

Algorithm	x_1	x_2	x_3	x_4	x_5
GBO	6.0124	5.3129	4.4941	3.5036	2.1506
GWO	6.0189	5.3173	4.4922	3.5014	2.1440
WOA	5.5638	6.0809	4.6858	3.2263	2.2467
CS	6.0351	5.2763	4.4575	3.5818	2.1287
ISA	6.0246	5.2958	4.4790	3.5146	2.1600
ABC	5.9638	5.3312	4.5122	3.4744	2.1952

Table 19. Comparison of statistical results for the rolling element bearing design problem

	GBO	CS	ABC	GWO	ISA	WOA
Best	85245.0611	85245.0611	85244.8594	85156.2032	85245.0611	85012.7716
Mean	**85245.0611**	**85245.0611**	85238.5919	84668.3988	85245.0610	77318.5161
SD	5.96E-11	1.33E-08	5.94E+00	7.42E+02	4.24E-04	1.19E+04
CT*	26.47	34.83	46.83	14.41	21.25	15.79

* Computational Time (Second)

Table 20. Best solutions achieved by the six algorithms for the rolling element bearing design problem

Variables	GBO	CS	ABC	GWO	ISA	WOA
D_b	21.87500	21.87500	21.87498	21.86758	21.87500	21.86062
D_m	125.0000	125.0000	125.0000	125.01048	125.0000	125.0230
f_i	0.51500	0.51500	0.51500	0.51501	0.51500	0.51500
f_0	0.51500	0.51500	0.51500	0.51511	0.51500	0.51500
Z	11.28817	11.28817	11.28343	11.29305	11.28817	11.11528
K_{Dmin}	0.41484	0.40000	0.47491	0.41803	0.50000	0.50000
K_{Dmax}	0.62866	0.70000	0.65702	0.63379	0.69996	0.70000
ε	0.30000	0.30000	0.30000	0.30003	0.30000	0.30000
e	0.02033	0.02000	0.05815	0.02030	0.02010	0.06039
ζ	0.67206	0.60000	0.60320	0.60126	0.60023	0.60000

Table 21. Comparison of statistical results for the tension/compression problem

	GBO	CS	ABC	GWO	ISA	WOA
Best	0.012667	0.012672	0.012668	0.012669	0.012686	0.012695
Mean	**0.012696**	0.012713	0.012701	0.013037	0.014115	0.013323
SD	3.358E-05	3.54E-05	3.61E-05	1.254E-03	1.434E-03	5.805E-04

Table 22. Best solutions achieved by the six algorithms for the tension/compression problem

Optimization algorithm	Optimal decision variables		
	m	D_c	d_w
GBO	0.05203	0.36509	10.81456
CS	0.0512	0.3447	12.0328
ABC	0.05155	0.35350	11.48213
GWO	0.0514	0.3503	11.6764
ISA	0.0528	0.3830	9.8978
WOA	0.0514	0.3513	11.6284

Iman Ahmadianfar: Conceptualization, Methodology, Software, Writing-Original draft preparation, Visualization, Investigation, **Omid Bozorg-Haddad:** Supervision, Reviewing and Editing, Investigation, **xuefeng chu:** Supervision, Reviewing and Editing, Investigation.

Highlights

• The Gradient-Based Optimizer (GBO) inspired by the gradient-based Newton's search method.

• The GBO uses two main operators: gradient search rule (GSR) and local escaping operator (LEO).

• The exploration and exploitation capabilities of GBO is assessed by 28 mathematical functions.

• The results on the mathematical test functions demonstrate the high competitiveness of GBO algorithm.

• The results on engineering optimization problems prove the suitable performance of the GBO to optimize practical problems.