Computers & Industrial Engineering 152 (2021) 107050

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Golden eagle optimizer: A nature-inspired metaheuristic algorithm

Abdolkarim Mohammadi-Balani [a], Mahmoud Dehghan Nayeri [a,*], Adel Azar [a], Mohammadreza Taghizadeh-Yazdi [b]

[a] Department of Industrial Management, Faculty of Management and Economics, Tarbiat Modares University, Tehran, Iran
[b] Department of Industrial Management, Faculty of Management, University of Tehran, Tehran, Iran

ARTICLE INFO

Keywords:
Golden eagle optimizer
Multi-objective golden eagle optimizer
Nature-inspired computing
Swarm intelligence
Metaheuristic algorithm

ABSTRACT

This paper proposes a nature-inspired swarm-based metaheuristic for solving global optimization problems called Golden Eagle Optimizer (GEO). The core inspiration of GEO is the intelligence of golden eagles in tuning speed at different stages of their spiral trajectory for hunting. They show more propensity to cruise around and search for prey in the initial stages of hunting and more propensity to attack in the final stages. A golden eagle adjusts these two components to catch the best possible prey in feasible region the shortest possible time. This behavior is mathematically modeled to highlight exploration and exploitation for a global optimization method. The performance of the proposed algorithm is tested and confirmed using 33 benchmark test functions and a scalability test. Results were compared to that of six other well-known algorithms, which revealed GEO's superiority, which indicates that it can find the global optimum and avoid local optima effectively. The Multi-Objective Golden Eagle Optimizer (MOGEO) is also proposed to solve multi-objective problems. The performance of MOGEO is also tested and verified on ten multi-objective benchmark functions. Results were compared to that of two other multi-objective algorithms, which showed that it can approximate true Pareto optimal solutions better than the other two algorithms. The software (toolbox) and source code for GEO and MOGEO are also provided, which are publicly available.

1. Introduction

Optimization is the process of finding the state of decision/design variables that yields the best value for single or multiple objective functions. Analytical methods were the dominant approach to solve mathematical problems before the heuristic optimization era. In addition to the primary information on the objective function value and constraint violation, analytical methods rely on the information about the derivatives of the sole or constraint-penalized objective functions in the form of first- and second-order derivatives. This extra information enables them to find the exact optimum for linear or convex non-linear problems efficiently. However, this comes at the cost of vulnerability to local optima entrapment in more complex problems–that has many local optima–and unavailability for problems with stochastic or unknown search space (Mirjalili, 2015). The stochastic behavior and unknown search space are the prominent features of real-world problems. This led to the advent of metaheuristic algorithms. The notable characteristics of metaheuristic algorithms are that they are derivative-free and do not require limiting assumptions. Therefore, they can be readily utilized for solving different classes of problems (Rakotonirainy & van Vuuren, 2020).

Such flexibility, however, is not costless. It has been observed, and later addressed as the No Free Lunch (NFL) theorem (Wolpert & Macready, 1997), that the excellent performance of an optimization algorithm on a specific set of problems does not guarantee the same performance on other problems. NFL provides an avenue for researchers to develop novel metaheuristic algorithms. Relative simplicity in understanding and application, as well as good performance, have resulted in the popularity of metaheuristic algorithms (Massan, Wagan, & Shaikh, 2020). Numerous algorithms have been introduced recently to solve business and engineering problems effectively.

Metaheuristic methods can be classified through various approaches. One common approach suggests classifying these methods based on the source inspiration: (a) evolutionary, (b) human-based, (c) physics-

* Corresponding author at: Department of Industrial Management, Faculty of Management and Economics, Tarbiat Modares University, Adjacent Gisha Bridge, Jalal Al-Ahmad Highway, P.O. Box: 14115-111, Postal Code: 1411713116, Tehran, Iran.
E-mail addresses: a_mohammadi@modares.ac.ir (A. Mohammadi-Balani), mdnayeri@modares.ac.ir (M. Dehghan Nayeri), azara@modares.ac.ir (A. Azar), mrtaghizadeh@ut.ac.ir (M. Taghizadeh-Yazdi).

https://doi.org/10.1016/j.cie.2020.107050
Received 10 August 2020; Received in revised form 1 December 2020; Accepted 12 December 2020
Available online 17 December 2020

based, and (d), synthetic, and (e) swarm intelligence. Evolutionary algorithms are normally based on the natural selection law of biology. These methods evolve the initial population using evolutionary operators to improve the population's fitness and find the global optimum (Bozorg-Haddad, Solgi, & Loaiciga, 2017; Husseinzadeh Kashan, Tavakkoli-Moghaddam, & Gen, 2019). Selection, crossover, and mutation are the most common of such operators. Genetic Algorithm (Goldberg & Holland, 1988) and Differential Evolution (Das & Suganthan, 2011) are two popular evolutionary algorithms. The human-based approach encompasses any algorithm that is inspired specifically by humans' social behavior or concepts that have been developed by humans. Queuing Search Algorithm (QSA) (Zhang, Xiao, Gao, & Pan, 2018), Group Teaching Optimization Algorithm (GTOA) (Zhang & Jin, 2020), and Teaching-Learning-Based Optimization (TLBO) (Rao, Savsani, & Vakharia, 2011) are the examples of algorithms proposed in this area. Physics-based methods tend to perceive the landscape as a physical phenomenon and move the search agents using formulae borrowed from physical rules or theories. Some of the recent algorithms proposed under this approach are Atom Search Optimization (ASO) (Zhao, Wang, & Zhang, 2019), Henry Gas Solubility Optimization (HGSO) (Hashim, Houssein, Mabrouk, Al-Atabany, & Mirjalili, 2019), Water Cycle Algorithm (WCA) (Eskandar, Sadollah, Bahreininejad, & Hamdi, 2012), Electron Radar Search Algorithm (ERSA) (Rahmanzadeh & Pishvaee, 2019), Lightning Attachment Procedure Optimization (LAPO) (Nematollahi, Rahiminejad, & Vahidi, 2017), Optics Inspired Optimization (OIO) (Husseinzadeh Kashan, 2015), Gravitational Search Algorithm (GSA) (Rashedi, Nezamabadi-pour, & Saryazdi, 2009), Equilibrium Optimizer (EO) (Faramarzi, Heidarinejad, Stephens, & Mirjalili, 2020), Thermal Exchange Optimization (TEO) (Kaveh & Dadras, 2017), Multi-Verse Optimizer (MVO) (Mirjalili, Mirjalili, & Hatamlou, 2016), Electro-Search algorithm (ES) (Tabari & Ahmad, 2017), and Colliding Bodies Optimization (CBO) (Kaveh & Mahdavi, 2014). Synthetic methods are solely based on mathematical equations like trigonometry functions or well-known constants. These algorithms are not inspired by a specific natural phenomenon. Sine Cosine Algorithm (SCA) (Mirjalili, 2016), Golden Ratio Optimization Method (GROM) (Nematollahi, Rahiminejad, & Vahidi, 2020), and Stochastic Fractal Search (SFS) (Salimi, 2015) are among the algorithms proposed within this approach. Algorithms belonging to the swarm intelligence approach imitate the social behavior and communications within a group of species of animals, plants, or other living things (Mavrovouniotis, Li, & Yang, 2017; Piotrowski, Napiorkowski, Napiorkowski, & Rowinski, 2017). Searching for food, hunting, mating, and memorizing are the common social behaviors considered in this class. Because communication is an indispensable element of social behavior, swarm intelligence algorithms allow the search agents to enjoy the information produced by other search agents in the current previous iteration (Zahedi, Akbari, Shokouhifar, Safaei, & Jalali, 2016). This approach has gained increasing popularity in terms of both application and new algorithm development. Some of the recently proposed algorithms that can be categorized under this approach are Pathfinder algorithm (PFA) (Yapici & Cetinkaya, 2019), Harris Hawks Optimization (HHO) (Heidari et al., 2019), Squirrel Search Algorithm (SSA) (Jain, Singh, & Rani, 2019), Seagull Optimization Algorithm (SOA) (Dhiman & Kumar, 2019), Sailfish Optimizer (SFO) (Shadravan, Naji, & Bardsiri, 2019), Black Widow Optimization (BWO) (Hayyolalam & Pourhaji Kazem, 2020), Emperor Penguin Optimizer (EPO) (Dhiman & Kumar, 2018), Mouth Brooding Fish algorithm (MBF) (Jahani & Chizari, 2018), Grasshopper Optimization Algorithm (GOA) (Saremi, Mirjalili, & Lewis, 2017), Spotted Hyena Optimizer (SHO) (Dhiman & Kumar, 2017), and Selfish Herd Optimizer (SHO) (Fausto, Cuevas, Valdivia, & González, 2017).

Metaheuristic methods can also be classified according to the number of search agents they use (Mirjalili et al., 2017). Individualist methods use only one search agent in each iteration, while population-based methods use multiple search agents in each iteration. A population of search agents produces more information in each iteration and can better explore the problem's feasible region. However, this massively increases the number of objective function evaluations, which can be problematic for computationally intensive objective functions. Simulated annealing (SA) (Kirkpatrick, Gelatt, & Vecchi, 1983), Tabu Search (TS) (Glover, 1989), and hill-climbing (Davis, 1991) are among the well-known individualist metaheuristics.

Fig. 1. Golden eagle (Veldman, 2018).

This paper proposes a novel swarm-intelligence metaheuristic algorithm and its multi-objective version based on the golden eagles' hunting process. They are called Golden Eagle Optimizer (GEO) and Multi-Objective Golden Eagle Optimizer (MOGEO). GEO is founded on the intelligent adjustments on attack propensity and cruise propensity that golden eagles perform while searching for prey and hunting. MOGEO uses the same principles and is equipped with special tools to handle multi-objective problems. The remaining parts of this paper are organized as follows. Section 2 provides the fundamental inspiration and mathematical formulation of the GEO for single- and multi-objective problems. Section 3 presents the experimental results of applying the proposed algorithm on different classes of single-objective benchmark functions in addition to a convergence and scalability analysis. Section 4 presents the results of applying MOGEO on the benchmark functions for multi-objective optimization. Section 5 explores the application for real-world engineering optimization problems. The paper concludes in Section 6 by presenting final remarks and suggestions for future studies.

2. Golden Eagle Optimizer (GEO)

This section is dedicated to introducing in detail the proposed Golden Eagle Optimizer algorithm. First, the inspiration for the algorithm is presented, then the mathematical model is discussed.

2.1. Inspiration

The golden eagle (Fig. 1), scientifically known as *Aquila chrysaetos*, belongs to the *Accipitridae* family, which covers different species of birds of prey like eagles and hawks (Golden eagle, Wikipedia., 2020). With exceptional vision, high speed, and powerful talons, golden eagles are professional hunters that can catch preys of a broad range of sizes from insects to mid-sized mammals (Tack, Noon, Bowen, & Fedy, 2020). This bird can fly as fast as 190 km/h (Golden eagle, Wikipedia., 2020). The golden eagle is the most widely distributed member of the *Accipitridae* family. Despite many other types of eagles, it can be found all over the Earth's northern hemisphere (Tikkanen et al., 2018).

Golden eagles have always had a close relationship with humans. They held lofty and sacred positions in the beliefs since ancient and tribal humans and were considered a sign of positive events (Golden eagles in human culture, Wikipedia., 2020). Even today, more than ten countries have an eagle as the national emblem or on the national flag (Eagle, 2020). The tradition of hunting with eagles is also practiced

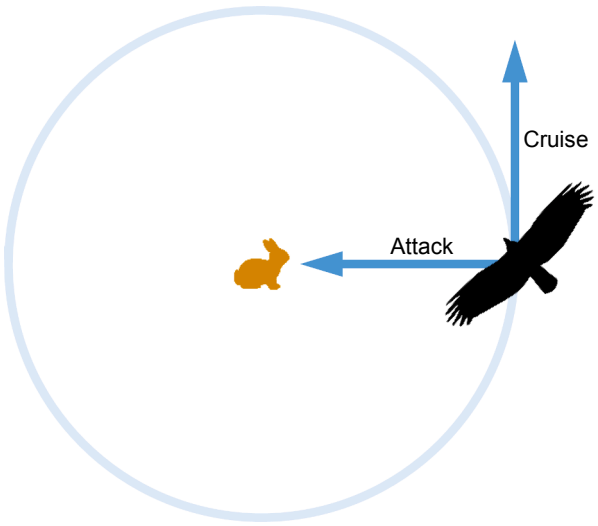

Fig. 2. Spiral motion of golden eagles.

throughout Kyrgyzstan and Kazakhstan. The golden eagle is the main bird of prey to be used there (Hunting with eagles, Wikipedia., 2020).

The unique feature of the golden eagle's cruising and hunting is that it takes place in a spiral trajectory, meaning that the prey is most of the time on one side of the eagle. This enables them to monitor the targeted prey and the nearby boulders and bushes for finding a proper angle of attack. In the meantime, they also survey other regions if they can find better food.

At each instance of the flight, the golden eagle's behavior is driven by two forces: the propensity to attack, and the propensity to cruise. Golden eagles know that if they attack hastily, they may catch small prey that does not even compensate for the energy consumed for hunting. On the other hand, if they engage in an endless search for bigger prey, they may run out of energy and catch nothing. Golden eagles intelligently create a balance between these two desires to snatch the best prey they can in a reasonable time and with a reasonable amount of energy. They switch from a low-attack-high-cruise profile to a high-attack-low-cruise profile smoothly. Each golden eagle starts the hunt by flying at high altitudes within its realm in large circles and searches for prey. Once prey is spotted, it starts moving on the perimeter of a hypothetical circle centered at the prey. The golden eagle memorizes the location of the prey but continues to circle it. The eagle gradually lowers its altitude and simultaneously gets closer to the prey, making the radius of the hypothetical circle around the prey smaller and smaller. At the same time, it also surveys the nearby regions for better alternatives. Sometimes golden eagles share the location of the best prey they found so far with other eagles. If the eagle does not spot better location/prey, it continues to circle around the current one in smaller circles and finally attacks the prey. Otherwise, if the eagle finds a better alternative, it flies on a new circle around the new prey and forgets the previous one. It is noteworthy that the final attacks are performed in a straight line.

With that said, the main characteristics of the hunting process of golden eagles can be summarized as follows.

- They follow a spiral trajectory for search and a straight path for the attack,
- They show more propensity to cruise in initial stages of hunting and smoothly transition to more propensity to attack in the final stages,
- They retain tendency for both cruise and attack in every moment of the flight,
- They look for other eagles' information on prey.

Cruise, attack, and the intelligent balance that the golden eagle creates between these two are the natural manifestation of exploration, exploitation, and the transition from the former to the latter. This paves the way for devising a metaheuristic algorithm. The next subsection mathematically models this behavior.

2.2. Mathematical model and optimization algorithm

This subsection describes the proposed mathematical formulation to mimic the movements of golden eagles that search for prey. The formulation for the spiral motion is presented, followed by its decomposition into attack and cruise vectors to emphasize exploitation and exploration, respectively.

2.2.1. The spiral motion of golden eagles

GEO is based on the spiral motion of golden eagles. As mentioned earlier, each golden eagle memorizes the best location it has visited so far. The eagle simultaneously has attraction toward attacking the prey and toward cruise to search for better food. Attack and cruise vectors in 2D space can be visualized as in Fig. 2.

In each iteration, each golden eagle i randomly selects the prey of another golden eagle f and circles around the best location visited so far by golden eagle f. The golden eagle i can also choose to circle its own memory; therefore, we have $f \in \{1, 2, \cdots, PopSize\}$.

2.2.2. Prey selection

In each iteration, each golden eagle must choose a prey to perform the cruise and attack operations. In GEO, the prey is modeled as the best solution found so far by the flock of golden eagles. Each golden eagle is capable of memorizing the best solution it has found so far. In each iteration, each search agent selects a target prey from the memory of the whole flock. Attack and cruise vectors for each golden eagle are then calculated relative to the selected prey. If the new position (calculated via attack and cruise vectors) is better than the previous position in the memory, then the memory is updated. The prey selection strategy plays an important role in GEO. Selection can take place in a basic way, where each golden eagle only selects the prey in its own memory. To make golden eagles better explore the landscape, we propose a random one-to-one mapping scheme, where each golden eagle randomly selects its prey in the current iteration from the memory of any other flock member. It is noteworthy that the selected prey is not necessarily the nearest or farthest prey. In this scheme, each prey in the memory is assigned or mapped to one and only one golden eagle. Then each golden eagle performs the attack and cruise operations on the selected prey. Fig. 3

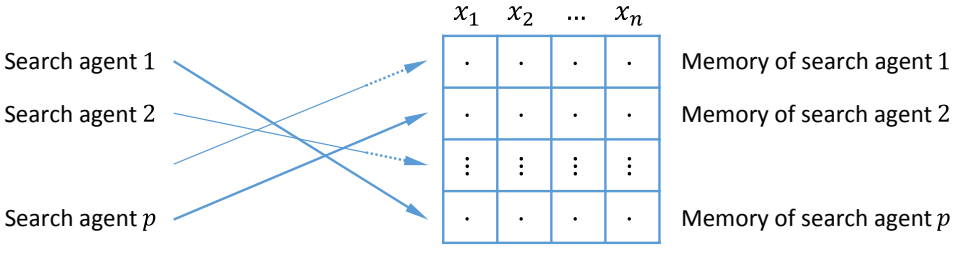

Fig. 3. One-to-one mapping in GEO prey selection.

shows that each search agent can only attack one of the positions in the memory that belong to another search agent.

2.2.3. Attack (exploitation)

The attack can be modeled via a vector starting from the current position of the golden eagle and ending in the location of the prey in the eagle's memory. The attack vector for golden eagle i can be calculated via Eq. (1).

$$\overrightarrow{A}_i = \overrightarrow{X}_f^* - \overrightarrow{X}_i \tag{1}$$

where $\overrightarrow{A}_i$ is the attack vector of eagle i, $\overrightarrow{X}_f^*$ is the best location (prey) visited so far by eagle f, and $\overrightarrow{X}_i$ is the current position of eagle i. Since the attack vector guides the population of golden eagles toward the best-visited locations, it highlights the exploitation phase in GEO.

2.2.4. Cruise (exploration)

The cruise vector is calculated based on the attack vector. The cruise vector is a tangent vector to the circle and perpendicular to the attack vector. The cruise can also be thought of as the linear speed of the golden eagle relative to the prey. The cruise vector in n-dimensions is located inside the tangent hyperplane to the circle; thus, to calculate the cruise vector, we have to first calculate the equation of the tangent hyperplane. The equation of a hyperplane in n-dimensions can be determined by an arbitrary point from that hyperplane and a perpendicular vector to that hyperplane, which is called the normal vector of the hyperplane. Eq. (2) displays the scalar form of the hyperplane equation in n-dimensional space.

$$h_1x_1 + h_2x_2 + \cdots + h_nx_n = d \Rightarrow \sum_{j=1}^{n} h_jx_j = d \tag{2}$$

where $\overrightarrow{H} = [h_1, h_2, \cdots, h_n]$ is the normal vector, $X = [x_1, x_2, \cdots, x_n]$ is the variables vector, $\overrightarrow{P} = [p_1, p_2, \cdots, p_n]$ is the arbitrary point on the hyperplane, and $d = \overrightarrow{H} \cdot \overrightarrow{P} = \sum_{i=1}^{n} h_jp_j$. If we consider $\overrightarrow{X}_i$ (the location of the eagle i) as the arbitrary point in the hyperplane and consider $\overrightarrow{A}_i$ (the attack vector) as the normal of the hyperplane, one can show the hyperplane to which $\overrightarrow{C}_i^t$ (the cruise vector for the golden eagle i in iteration t) belongs according to Eq. (3).

$$\sum_{j=1}^{n} a_jx_j = \sum_{j=1}^{n} a_j^t x_j^* \tag{3}$$

where $\overrightarrow{A}_i = [a_1, a_2, \cdots, a_n]$ is the attack vector, $X = [x_1, x_2, \cdots, x_n]$ is the decision/design variables vector, and $X^* = [x_1^*, x_2^*, \cdots x_n^*]$ is the location of the selected prey.

Now that the cruise hyperplane for eagle i in iteration t is calculated, it is time to find a cruise vector for this golden eagle within this hyperplane. A golden eagle can choose any destination point on the cruise hyperplane. To find a random vector on the cruise hyperplane, we have to first find a random destination point C on this hyperplane other than the one we already have (the current location of the golden eagle i). Note that the starting point of the cruise vector is the current location of the golden eagle i. Since hyperplanes are one dimension smaller than their ambient space, we cannot simply generate a random $1 \times n$ point. A simple random point in n-dimensional space is not guaranteed to be located on the cruise hyperplane. A new point located on the n-dimensional cruise hyperplane has $n-1$ degrees of freedom, meaning that $n-1$ dimensions can be chosen freely, but the hyperplane equation dictates the last dimension, as shown in Eq. (2). The last dimension must be chosen so as it satisfies the hyperplane equation; therefore, we have $n-1$ free variables and one fixed variable. We use the following procedure to find a random n-dimensional destination point C located on the cruise hyperplane for golden eagle i.

Step 1. Randomly choose one variable out of n variables as the fixed variable. We denote the index of the selected variable with k. Note that the fixed variable cannot be chosen from the variables whose corresponding element in the attack vector $\overrightarrow{A}_i$ is zero. The reason is that when the coefficient of a variable in Eq. (2) is equal to zero, the hyperplane is parallel to the axis of that variable, and that variable can take any value for a random combination of the other $n-1$ variables. For example, in the 3D plane $3x_1 + 2x_2 = 10$, if we choose $k = 3$ and choose random numbers for x_1 and x_2, say $\{x_1 = 2, x_2 = 5\}$, we cannot find a unique point. Instead, an infinite number of point on this plane is obtained, and all of them satisfy the plane equation $\{[2,5,1],[2,5,2],[2,5,3],\cdots\}$.

Step 2. Assign random values to all the variables except the k-th variable because the k-th variable is fixed.

Step 3. Find the value of the fixed variable using Eq. (4).

$$c_k = \frac{d - \sum_{j,j\neq k} a_j}{a_k} \tag{4}$$

where c_k is the k-th element of the destination point C, a_j is the j-th element of the attack vector $\overrightarrow{A}_i$, d is the right-hand side of the Eq. (2), a_k^t is the k-th element of the attack vector $\overrightarrow{A}_i$, and k is the index of the fixed variable. The random destination point on the cruise hyperplane is found. Eq. (5) displays the general representation of the destination point on the cruise hyperplane.

$$\overrightarrow{C}_i = \left(c_1 = \text{random}, c_2 = \text{random}, \cdots, c_k = \frac{d - \sum_{j,j\neq k} a_j}{a_k}, \cdots, c_n = \text{random}\right) \tag{5}$$

Now that the destination point is determined, the cruise vector can now be calculated for the golden eagle i in iteration t. The elements of the obtained destination point are random numbers between zero and one. It is noteworthy that the cruise vector attracts the population of golden eagles toward the areas other than the ones in the memory; therefore, it emphasizes the exploration phase of GEO.

2.2.5. Moving to new positions

The displacement of the golden eagles comprises of attack and vector. We define the step vector for golden eagle i in iteration t as Eq. (6).

$$\Delta x_i = \overrightarrow{r}_1 p_a \frac{\overrightarrow{A}_i}{\|\overrightarrow{A}_i\|} + \overrightarrow{r}_2 p_c \frac{\overrightarrow{C}_i}{\|\overrightarrow{C}_i\|} \tag{6}$$

where p_a^t is the attack coefficient in iteration t and p_c^t is the cruise coefficient in iteration t and adjust how golden eagles are affected by attack and cruise. $\overrightarrow{r}_1$ and $\overrightarrow{r}_2$ are random vectors whose elements lie in the interval $[0,1]$. p_a and p_c will be discussed later. $\|\overrightarrow{A}_i\|$ and $\|\overrightarrow{C}_i\|$ are the Euclidean norm of the attack and cruise vectors and are calculated using Eq. (7).

$$\|\overrightarrow{A}_i\| = \sqrt{\sum_{j=1}^{n} a_j^2}, \|\overrightarrow{C}_i\| = \sqrt{\sum_{j=1}^{n} c_j^2} \tag{7}$$

The position of the golden eagles in iteration $t+1$ is calculated simply by adding the step vector in iteration t to the positions in iteration t.

$$x^{t+1} = x^t + \Delta x_i^t \tag{8}$$

If the fitness of the new position of the golden eagle i is better than the position in its memory, the memory of this eagle is updated with the

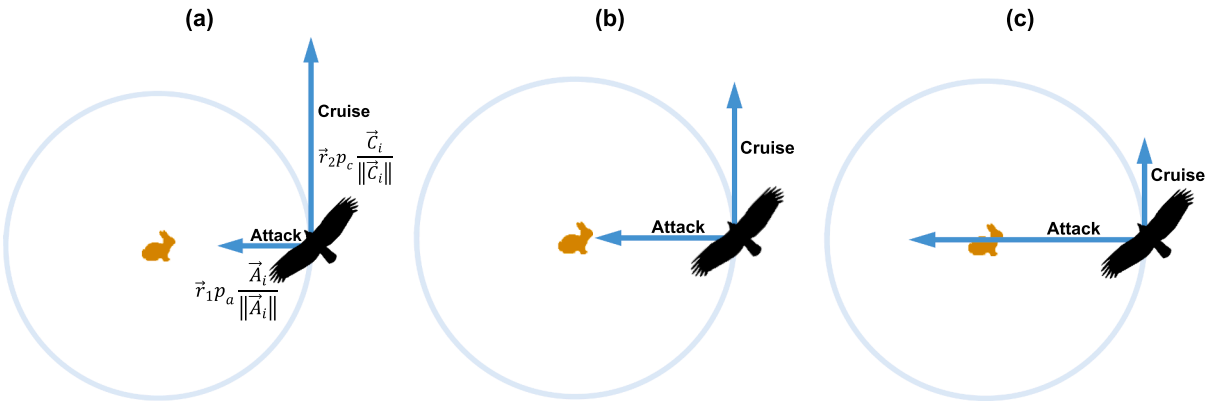

Fig. 4. Golden eagle's transition from exploratory behavior (intense cruise) to exploitative behavior (intense attack).

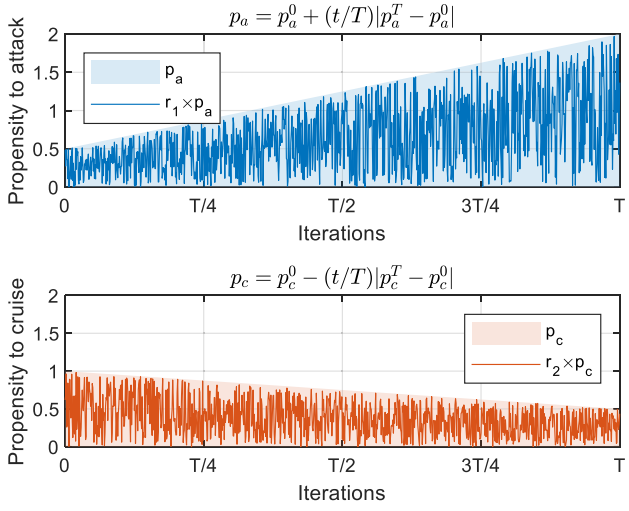

Fig. 5. p_a and p_c over the course of iterations.

new position. Otherwise, the memory remains intact, but the eagle will reside in the new position. In the new iteration, each golden eagle randomly chooses a golden eagle from the population to circle around its best-visited location, calculates attack vector, calculates cruise vector, and finally, the step vector and the new position for the next iteration. This loop is executed until any of the termination criteria are satisfied.

We mentioned that there are two coefficients in Eq. (6), namely attack coefficient p_a^t and cruise coefficient p_c^t, that control how the step vector is affected by attack and cruise vectors. The next subsection discusses how the values of these two coefficients are adjusted over the course of iterations.

2.2.6. *Transition from exploration to exploitation*

As mentioned earlier, golden eagles show a higher propensity to cruise in the initial stages of the hunting flight and show a higher propensity to attack in the final stages, which correspond to more exploration in initial iterations and more exploitation in the final iterations in the proposed optimizer. Fig. 4 shows how the attack and cruise change.

GEO uses p_a and p_c to shift from exploration to exploitation. The algorithm starts with low p_a and high p_c. As the iterations proceed, p_a is gradually increased while p_c is gradually decreased. The initial and final values of both parameters are defined by the user. Intermediate values can be calculated using the linear transition displayed in Eq. (9).

$$\begin{cases} p_a = p_a^0 + \dfrac{t}{T}|p_a^T - p_a^0| \\ p_c = p_c^0 - \dfrac{t}{T}|p_c^T - p_c^0| \end{cases} \tag{9}$$

where t indicates current iteration, T indicates maximum iterations, p_a^0 and p_a^T are the initial and final values for propensity to attack (p_a), respectively, and p_c^0 and p_c^T are the initial and final values for propensity to cruise (p_c), respectively. Our experiments, which will be discussed

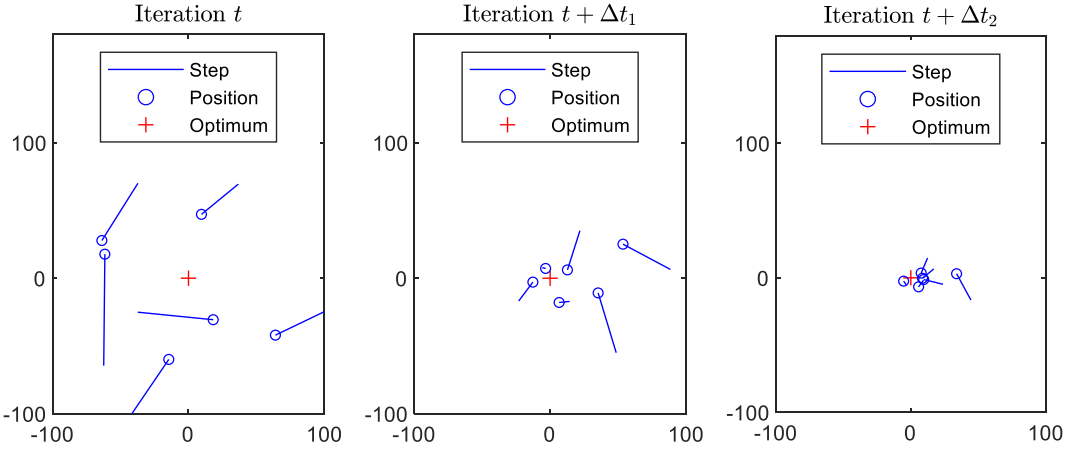

Fig. 6. Movement of search agents in 2D space.

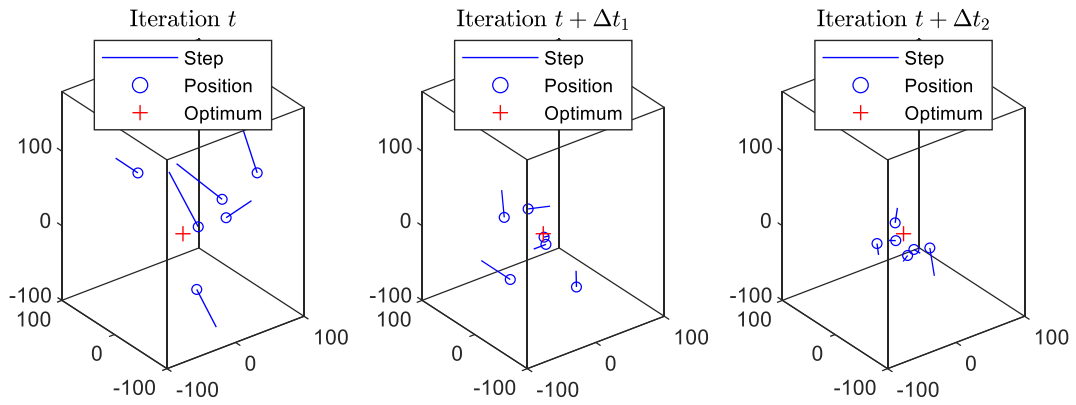

Fig. 7. Movement of search agents in 3D space.

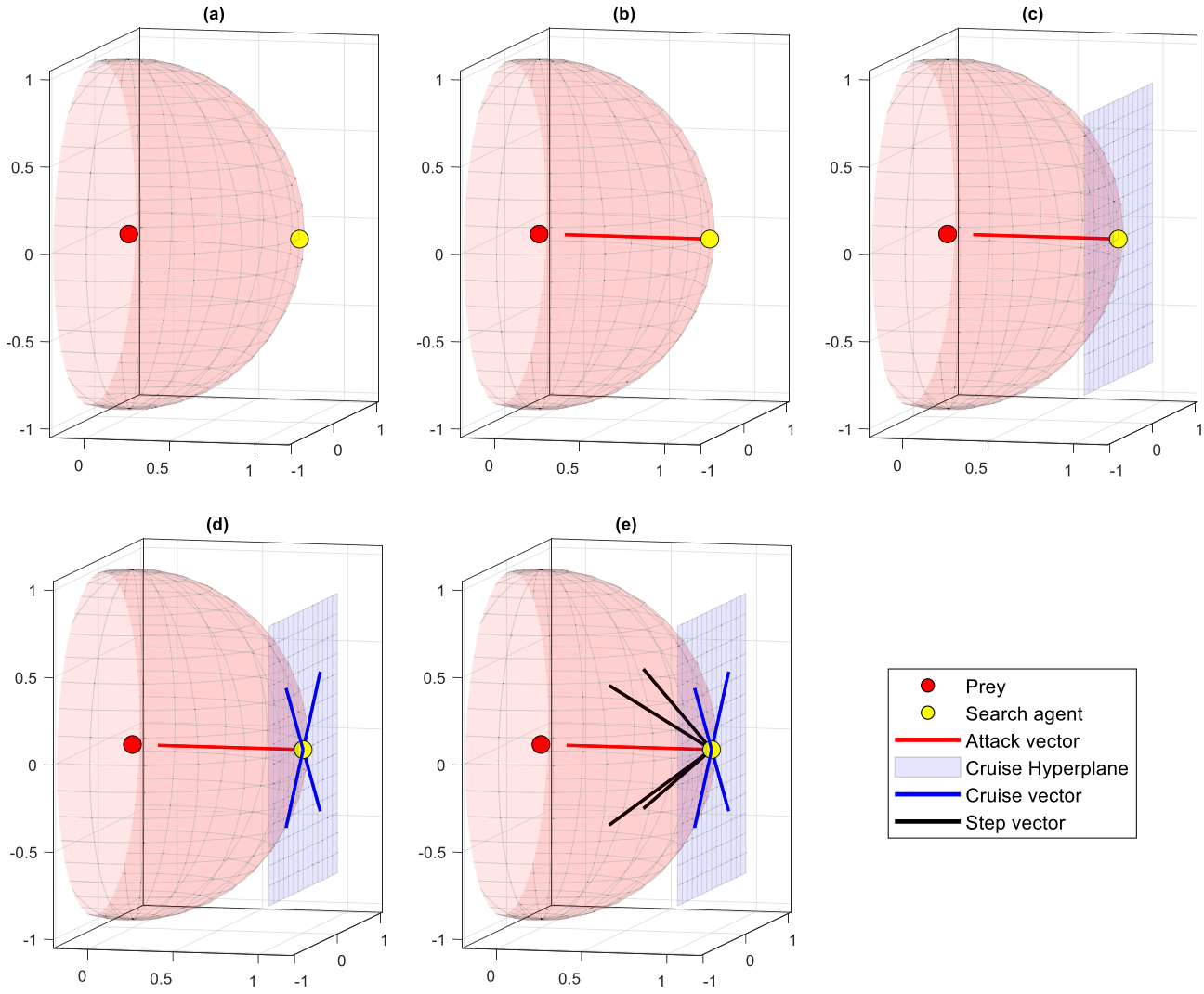

Fig. 8. Main steps of GEO: (a) the search agent selects a prey from the flock's memory, (b) attack vector is calculated, (c) cruise hyperplane is constructed, (d) a random cruise vector is constructed inside the cruise hyperplane, and (e) step vector is constructed from attack and cruise vectors.

later, show that $[p_a^0, p_a^T] = [0.5, 2]$ and $[p_c^0, p_c^T] = [1, 0.5]$ seem to be suitable parameters. This means that p_a is set to 0.5 in the first iteration and linearly drops to reach 2 in the last iteration. The same goes for p_C where it starts with 1 in the first iteration and is linearly lowered to reach 0.5 in the last iteration. It worths noting here that Eq. (9) linearly changes the parameters. However, they can be changed logarithmically or by means of any other function. Fig. 5 shows how p_a, $r_1 \times p_a$, p_c, and $r_2 \times p_c$ change over the course of iterations. Note that r_1 and r_2 are random numbers in the interval $[0, 1]$ in Eq. (6).

The movement of search agents in 2D and 3D spaces is displayed in Fig. 6 and Fig. 7, respectively. These figures show the position and step vector in different iterations, where t is one of the initial iterations, $t +$

Δt_1 belongs to midway, and $t + \Delta t_2$ is one of the final iterations. In other words, $t < t + \Delta t_1 < t + \Delta t_2$.

To sum up this subsection, a visual summary of the main steps of GEO is illustrated in Fig. 8. In each iteration, each search agent first selects prey from the flock's memory and constructs a hypothetical hypersphere (Fig. 8a). Next, the search agents construct the attack vector, which is a vector from the search agents to their selected prey (Fig. 8b). Then, each search agent constructs its cruise hyperplane, which is basically the tangent hyperplane to the hypothetical sphere at the search agent's position (Fig. 8c). Next, the cruise vector, which is a random vector inside the cruise hyperplane (Fig. 8d), is constructed. Finally, attack and cruise vectors are combined to form the step vector (Fig. 8e).

2.2.7. Single-objective golden eagle Optimizer (GEO)

According to the basic concepts and their corresponding mathematical modeling presented in Section 2.2, the pseudo-code of the single-objective implementation of GEO is presented in Algorithm 1.

Algorithm 1. Pseudo-code of GEO

```
Initialize the population of golden eagles
Evaluate fitness function
Initialize population memory
Initialize p_a and p_c
for each iteration t
    Update p_a and p_c (Eq. (9))
    for each golden eagle i
        Randomly select a prey from the population's memory
        Calculate attack vector A (Eq. (1))
        if attack vector's length is not equal to zero
            Calculate cruise vector C (Eqs. (2)–(5))
            Calculate step vector Δx (Eqs. (6)–(8))
            Update position (Eq. (8))
            Evaluate fitness function for the new position
            if fitness is better than the fitness of the position in eagle i's memory
                Replace the new position with the position in eagle i's memory
            end
        end
    end
end
```

2.2.8. Computational complexity of GEO

The computational complexity of the proposed GEO algorithm can be discussed for the two major parts of the algorithm:

(a) Initialization. The algorithm requires $\mathcal{O}(n_{\text{population}} \times n_{\text{dimensions}})$ time to initialize the position vector, the step vector, and memory for the search agents.
(b) Main loop. The main loop requires $\mathcal{O}(n_{\text{population}} \times n_{\text{dimensions}} \times n_{\text{iteration}})$ time to select prey, calculate attack and cruise vectors, and update the position of the search agents.

It can be concluded that the total time complexity of GEO is $\mathcal{O}(n_{\text{population}} \times n_{\text{dimensions}} \times n_{\text{iteration}})$. It is noteworthy that the space complexity of GEO is equal to $\mathcal{O}(n_{\text{population}} \times n_{\text{dimensions}})$ since it is the space that is occupied in the initialization and does not grow or shrink during iterations of the main loop.

2.3. Golden Eagle Optimizer for multi-objective problems

2.3.1. Multi-objective optimization

Multi-objective problems are relatively similar to single-objective problems in terms of problem definition. The only difference is that, as their name suggests, they contain multiple objective functions instead of a single objective function. This apparently negligible difference, however, creates challenges in terms of optimization procedure that cannot be addressed by algorithms that are designed to deal with single-objective optimization. That is where the need for optimization algorithms that can handle and solve multi-objective problems emerges. A general multi-objective problem can be defined as Eq. (10) (Cui, Geng, Zhu, & Han, 2017).

$$\begin{aligned} &\text{Minimize } F(\vec{x}) = \{f_1(x), f_2(x), \cdots, f_k(x)\} \\ &\text{Subjectto :} \\ &g_i(\vec{x}) \le 0, \quad i = 1, 2, \cdots, r \\ &h_i(\vec{x}) = 0, \quad i = r+1, r+2, \cdots, s \end{aligned} \tag{10}$$

where F is the set of objectives to be optimized, $\vec{x}$ is the vector of decision/design variables, g_i is the i-th inequality constraint, h_i is the i-th equality constraint, r is the number of inequality constraints, and s is the total number of constraints.

In the single-objective optimization, the solution $\vec{x}_1$ is better than $\vec{x}_2$ if $f\left(\vec{x}_1\right) < f\left(\vec{x}_2\right)$. However, in multi-objective optimization, such a definition cannot be used. Instead, the Pareto dominance concept is introduced to deal with multi-objective problems. It suggests that solution $\vec{x}_1$ dominates (is better than) $\vec{x}_2$ if for all of the objective functions we have $f\left(\vec{x}_1\right) < f\left(\vec{x}_2\right)$. The two solutions are called non-dominated if for at least one objective, but not all of them we have $f\left(\vec{x}_1\right) \nless f\left(\vec{x}_2\right)$. If such a relation holds for a solution compared to other solutions in the feasible region, that solution is called Pareto optimal. The ultimate goal in multi-objective optimization is to find the Pareto optimal solutions (Khoroshiltseva, Slanzi, & Poli, 2016). In contrast to single-objective problems, multi-objective problems do not have a single Pareto optimal solution. Instead, they have a set of non-dominated solutions as the Pareto optimal solutions. So the ultimate goal in these problems is shifted toward finding the Pareto optimal set of solutions, which is also called the Pareto front (Martín & Schütze, 2018).

2.3.2. Multi-objective golden eagle Optimizer (MOGEO)

The proposed algorithm is able to find the best location of food using different operators. However, it is not capable of finding the Pareto optimal solution to problems with multiple objectives. In particular, the drawbacks of GEO for handling multi-objective problems are as follows:

- In GEO, each golden eagle has its own separate memory of the best prey visited by itself so far. This means that GEO saves multiple individual best prey in each iteration. Saving multiple solutions are useful for multi-objective problems, but the saved solutions must be non-dominated, which is not guaranteed in GEO. Therefore, a mechanism should be introduced to only save the non-dominated solutions so far.
- In the prey selection stage of GEO, each golden eagle chooses another golden eagle arbitrarily to perform attack and cruise operators on its best prey stored in its memory. However, quality optimal Pareto fronts contain members that are uniformly distributed along the front. This implies that a criterion is needed so that golden eagles can prioritize some of the preys in the memory to the others with that criterion.
- In the prey selection stage of GEO, a one-to-one mapping occurs between golden eagles and preys in the memory. In other words, each prey in the memory is assigned to one and only one golden eagle. However, the Pareto front in a given iteration might have more or fewer members than the population size. Therefore, the one-to-one mapping between search agents and prey cannot be implemented in multi-objective problems.

With that said, Multi-objective Golden Eagle Optimizer (MOGEO) is built upon the concepts of single-objective optimization mentioned above plus three additional concepts: (a) external archive, (b) prey prioritization criterion, and (c) multi-objective prey selection.

External-archive-based algorithms are popular yet robust approaches in multi-objective optimization. Some well-known multi-

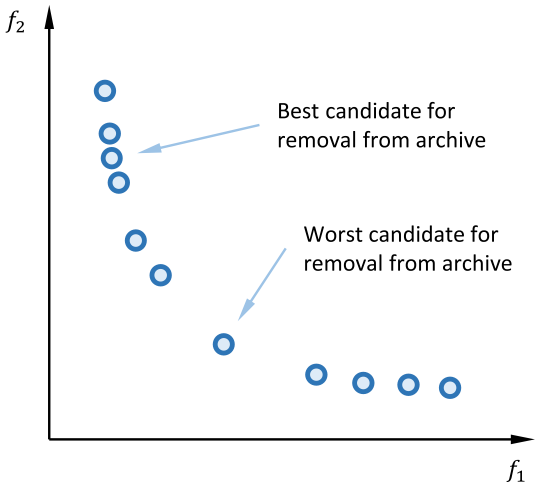

Fig. 9. Solutions located in the dense and sparse regions of the external archive.

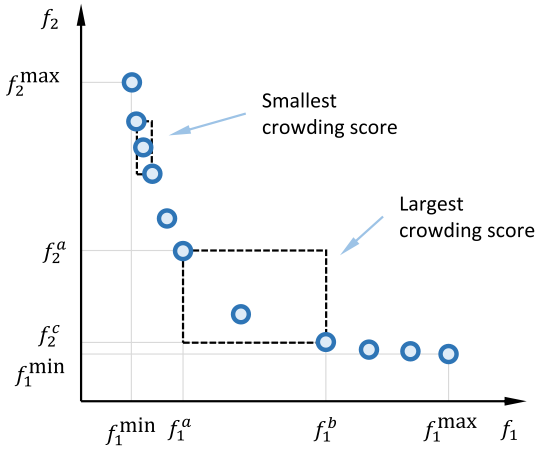

Fig. 10. Crowding distance for members in dense and sparse regions of the external archive.

objective algorithms like Multi-Objective Particle Swarm Optimization (MOPSO) (Coello Coello & Lechuga, 2002) or recent ones like Multi-Objective Grasshopper Optimization Algorithm (MOGOA) (Mirjalili, Mirjalili, Saremi, Faris, & Aljarah, 2018) or Multi-Objective Ant Lion Optimizer (MOALO) (Mirjalili, Jangir, & Saremi, 2017) utilize an archive-based approach.

The basic idea is to keep promising non-dominated solutions in an external archive and update it as the optimization algorithm proceeds. Search agents are steered toward the archive members and ultimately to the region where the optimal Pareto front exists (Cai, Qu, & Cheng, 2018; Mirjalili, Saremi, Mirjalili, & Coelho, 2016; Zhang, Gong, Sun, & Qu, 2018). Since GEO uses a dedicated memory to keep promising preys, the external archive approach can be easily implemented in GEO. The archive's capacity is limited. Therefore, a mechanism should be introduced for updating the external archive to keep the Pareto optimal solutions visited so far and avoid violating the maximum capacity limit of the archive.

When each of the search agents moves to a new position, it may face one of the three following conditions. If the new solution (position) is dominated by one or more of the current archive members, the new solution is discarded. If the new solution is non-dominated to the current members of the archive and the archive is not full, simply add the new position to the archive. If the new position is non-dominated compared to the current members of the archive, randomly select one of the

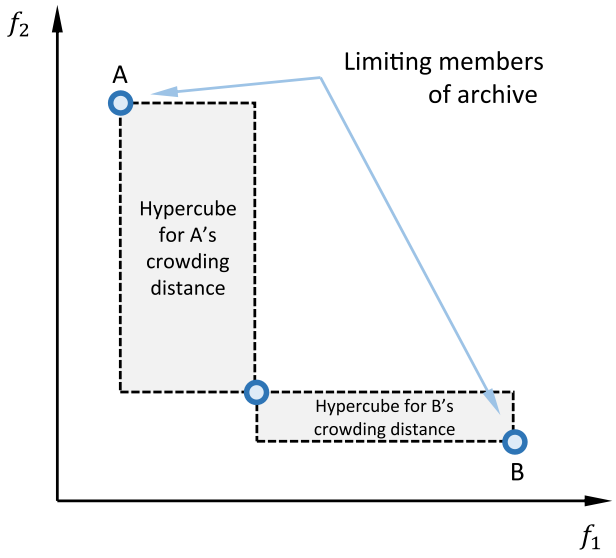

Fig. 11. Crowding score for limiting members of the archive.

archive members and substitute it with the new solution. One of the desirable characteristics of an ideal optimal Pareto front is the uniform dispersion of archive members along the front in the objective space. Therefore, the outgoing member should be selected from the dense regions of the archive in order to decrease the density in those regions (Ahmadi, Tiruta-Barna, Capitanescu, Benetto, & Marvuglia, 2016; Chen et al., 2019). Fig. 9 shows an example of archive members located in the dense and sparse regions of the archive.

A measure is needed to determine the density of the nearby area for each member of the archive. We propose the crowding score to be used as the density index in MOGEO. The crowding score is grounded on the idea of crowding distance (Deb, Agrawal, Pratap, Meyarivan, & Fast, 2000). The crowding distance of a solution in the Pareto front is defined as the distance between the two nearest solutions in its vicinity and can be calculated through Eq. (11).

$$C_i = \frac{1}{n}\sum_{j\in J}\frac{(f_{i+1,j} - f_{i,j}) - (f_{i,j} - f_{i-1,j})}{f_j^{\max} - f_j^{\min}} \tag{11}$$

where $f_{i-1,j}, f_{i,j}, f_{i+1,j}$ are three consecutive members when the archive is sorted according to the objective values of the j-th objective function. Fig. 10 shows the crowding hypercube for solutions located in the sparse regions of the archive are assigned larger crowding scores, while solutions in the dense regions have smaller crowding scores. It can be seen that the crowding distance is equal to half of the crowding hypercube's perimeter (see Fig. 11).

The only exceptional cases are limiting members, i.e., the members with the largest or smallest value in any of the objective functions. Regular members have two adjacent members, but limiting members have only one. The crowding score for limiting members is calculated similarly to Eq. (11) except that one of the terms in the numerator is discarded.

The crowding distance is calculated for all of the archive members. The outgoing member is selected using a roulette wheel where the probabilities are proportional to crowding distances. We want to select the outgoing member from the denser parts of the archive, so we should assign larger weights to the solutions in denser regions. This can be easily achieved by subtracting the crowding scores from 1 since the crowding distances calculated by Eq. (11) fall in the interval $[0, 1]$. The new scores that are used for the roulette wheel procedure are called sparsity scores (S_i), which can be calculated using Eq. (12).

$$S_i = 1 - C_i \tag{12}$$

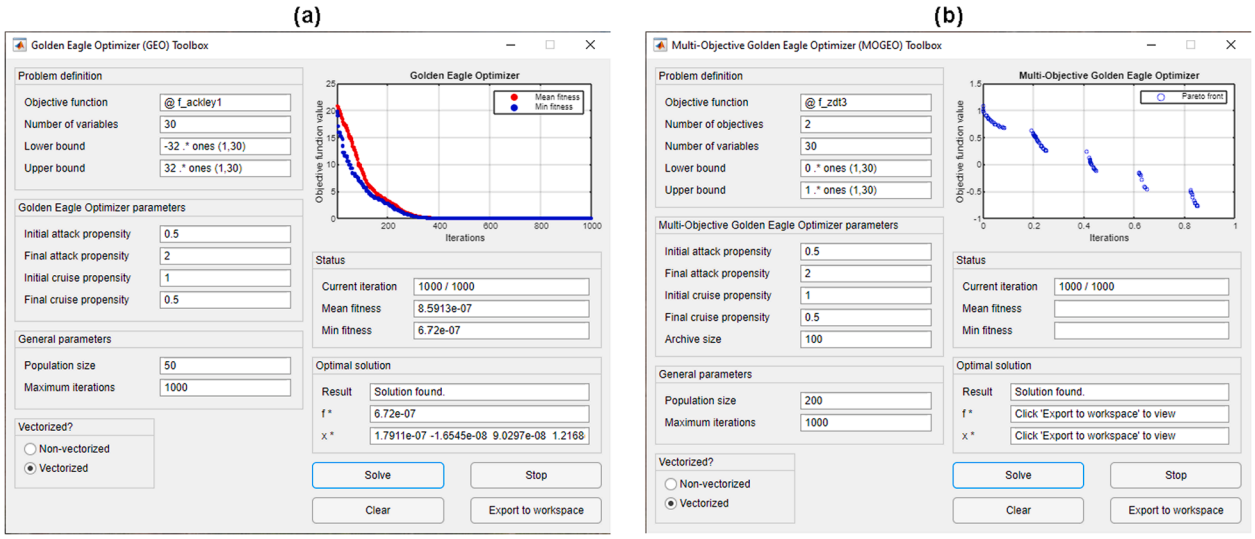

Fig. 12. The user interface of GEO (a) and MOGEO (b) toolboxes (can be downloaded from https://www.mathworks.com/matlabcentral/profile/authors/14675656).

The last important topic in MOGEO is the prey selection procedure. It is similar to the prey selection in GEO but with some modifications. In GEO, every search agent has its own memory to keep the best location visited so far. However, the memory in this sense cannot be used as the external archive in MOGEO because the archive keeps only non-dominated locations visited so far. This point leads to conditions where the number of archive members is less than, or in general, different from the population size. We propose the MOGEO prey selection procedure to be based on the roulette wheel, where the weights are the sparsity scores of the current archive members. This results in a higher probability of selection for members in the sparse regions of the front and less probability for archive members in the dense regions. The crowding scores are calculated according to Eq. (11). The pseudocode of MOGEO is presented in Algorithm 2.

Algorithm 2. Pseudo-code of MOGEO

```
Initialize the population of golden eagles
Evaluate the fitness function
Initialize population memory
Initialize p_a and p_c
for each iteration t
    Update p_a and p_c (Eq. (9))
    Calculate crowding distance for existing archive members
    for each golden eagle i
        Randomly select prey from the archive using the roulette wheel
            weighted by crowding distances
        Calculate attack vector A⃗ (Eq. (1))
        if the attack vector's length is not equal to zero
            Calculate cruise vector C⃗ (Eqs. (2)–(5))
            Calculate step vector Δx (Eqs. (6)–(8))
            Update position (Eq. (8))
            Evaluate fitness functions for the new position
            if the new position is non-dominated to the current archive
members
                if the external archive is not full
                    Add the new solution to the archive
                else
                    Calculate the sparsity distances (Eqs. (11)–(12))
                    Select the outgoing archive member using
                        roulette wheel weighted by sparsity distances
                    Replace the outgoing solution with the new one
                end
        end
    end
end
```

2.3.3. Computational complexity of MOGEO

The computational complexity of the proposed MOGEO algorithm can be discussed for the two major parts of the algorithm:

(a) Initialization. The algorithm requires $\mathcal{O}(n_{\text{population}} \times n_{\text{dimensions}})$ time to initialize the position vector, the step vector, and memory for the search agents.
(b) Main loop. The main loop requires $\mathcal{O}(n_{\text{population}} \times n_{\text{dimensions}} \times n_{\text{iteration}} \times n_{\text{objective}} \times n_{\text{archive}})$.

It can be concluded that the total time complexity of MOGEO is $\mathcal{O}(n_{\text{population}} \times n_{\text{dimensions}} \times n_{\text{iteration}} \times n_{\text{objective}} \times n_{\text{archive}})$.

2.4. Software (toolbox) and source code for GEO and MOGEO

To facilitate the implementation of GEO and MOGEO algorithms, separate open-source MATLAB toolboxes are developed for GEO and MOGEO. The user interfaces are shown in Fig. 12. Each toolbox is divided into two columns. Problem definition and solver parameters are defined in the left column and the algorithm's progress, and the final results are shown in the right column. By pressing the "Solve" button, the solver starts to optimize the problem. Both solvers show graphical and textual feedback about the solver's status in each iteration. GEO toolbox plots the mean fitness for each iteration as well as the best solution found so far. MOGEO toolbox plots the archive members' fitness values in each iteration. The toolboxes are able to evaluate the fitness function in a vectorized fashion, which is suitable for speeding up the optimization process. Both toolboxes allow the user to halt the solver anywhere in the middle of optimization. The results, whether the algorithm obtained or the user decided to halt, can be easily exported to the base workspace for post-optimization analysis. The plot can also be exported to many types of lossy and vector graphic formats. In addition, the source code for both GEO and MOGEO is also publicly available. Toolboxes and the source codes can be downloaded from https://www.mathworks.com/matlabcentral/profile/authors/14675656.

3. Single-objective optimization results for GEO

To verify the performance of the proposed algorithm, GEO is tested on 33 well-known benchmark problems. This section presents the results of these tests. The challenging benchmark problems in each class analyze different aspects of the proposed algorithm. First, an overview

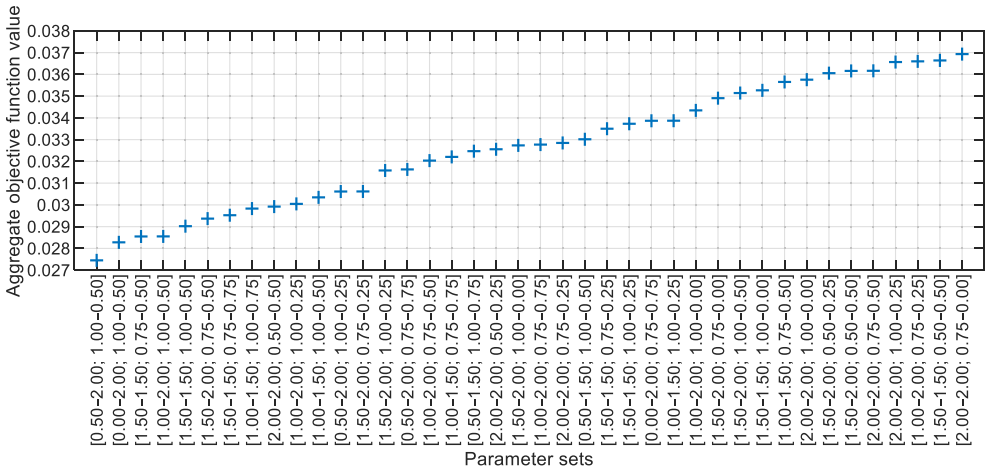

Fig. 13. Aggregate results for 20 of the best parameter sets for the GEO algorithm.

Table 1
Parameter settings for compared algorithms.

Algorithm	Parameter	Value
All algorithms	Population size	50
	Maximum iterations	1000
	Number of replications	30
GEO	p_a: Propensity to attack	$[0.5-2]$
	p_c: Propensity to cruise	$[1-0.5]$
GWO	C: Control parameter	$[2-0]$
	Number of leaders	3
GA	Elite fraction	0.05
	Selection method	Binary tournament
	Crossover method	Linear
	Crossover fraction	0.8
	Mutation method	Gaussian
CSA	fl: Flight length	2
	AP: Awareness probability	0.1
PSO	Neighboring ratio	0.25
	w: Inertia weight	0.8
	c_1, c_2: Acceleration weights	1.5
HS	Memory considering rate	0.95
	Pitch adjustment ratio	0.1
DA	b: Base coefficient	$[0.1-0]$
	r: Neighborhood radius	$[0.25-2.25]\times[ub-lb]$
	s: Separation coefficient	$2b$
	a: Alignment coefficient	$2b$
	c: Cohesion coefficient	$2b$
	f: Food attraction coefficient	2
	e: Enemy distraction coefficient	b

Table 2
Unimodal benchmark functions.

Name	Equation	D	Bounds	f^*
Beale	$f_1(x) = (1.5 - x_1 - x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	2	$[-4.5, 4.5]^D$	0
Matyas	$F_2(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	2	$[-10, 10]^D$	0
Three-hump camel	$F_3(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1}{6} + x_1x_2 + x_2^2$	2	$[-5, 5]^D$	0
Exponential	$F_4(x) = -e^{\left(-0.5\sum_{i=1}^{n} x_i^2\right)}$	30	$[-1, 1]^D$	0
Ridge	$F_5(x) = x_1 + 2\left(\sum_{i=2}^{n} x_i^2\right)^{0.1}$	30	$[-5, 5]^D$	−5
Sphere	$F_6(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^D$	0
Step	$F_7(x) = \sum_{i=1}^{n} (x_i + 0.5)^2$	30	$[-5.12, 5.12]^D$	0

of the experimental setup and compared algorithms are presented. Then, the details of the utilized benchmark functions of unimodal, multimodal, composite classes are presented. Next, the scalability analysis will be conducted to examine the performance of GEO in large problems.

3.1. Parameter setting

Before applying the proposed algorithm to the test functions, the parameters of GEO must be fine-tuned. The four parameters are initial attack propensity (p_a^0), final attack propensity (p_a^T), initial cruise propensity (p_c^0), and the final cruise propensity (p_c^T). GEO is applied to 15 of the test functions mentioned above, and the results are normalized and aggregated to construct a total measure to determine the best set of parameters. The values for the attack propensity are chosen from the set $\{0, 0.5, 1, 1.5, 2\}$, and the values for the cruise propensity are chosen from the set $\{0, 0.25, 0.5, 0.75, 1\}$. Every possible pair of attack propensity values that are non-decreasing are chosen. A similar approach was used to choose the values for the cruise propensity, except that the values must be non-increasing. A total of 225 parameter sets are obtained for the analysis. Each parameter set was used to run GEO 30 times on each problem. Fig. 13 displays the aggregate objective function values for the top 40 parameters set. It can be concluded that the best values for initial and final attack propensity are $\left[p_a^0 - p_a^T\right] = [0.5-2]$, and the best values for the initial and final cruise propensity are $\left[p_c^0 - p_c^T\right] = [1-0.5]$. Therefor, all of the experiments in this paper are performed using this set of parameters.

3.2. Experimental setup and compared algorithms

In order to verify the capabilities of GEO, its performance is compared to those of other well-known algorithms in the literature, namely, Grey Wolf Optimizer (GWO) (Mirjalili, Mirjalili, & Lewis, 2014), Genetic Algorithm (GA) (Goldberg & Holland, 1988), Crow Search Algorithm (CSA) (Askarzadeh, 2016), Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995), Harmony Search (HS) (Geem, Kim, & Loganathan, 2001), and Dragonfly Algorithm (DA) (Mirjalili, 2016). All of the algorithms were coded in MATLAB 9.6 (R2019a). To keep the comparisons fair and consistent, we used general and solver-specific parameters as reported in Table 1. Metaheuristic algorithms use random initial generation and random numbers in the intermediate calculations, which may affect the quality of the solutions. Each algorithm is implemented multiple times on each benchmark problem, so as to avoid these effects. As depicted in Table 1, we used 30 independent replications for all of the problems and solvers.

Table 3
Multimodal benchmark functions.

Name	Equation	D	Bounds	f^*
Drop wave	$F_8(x) = -\frac{1+\cos\left(12\sqrt{x_1^2+x_2^2}\right)}{0.5\left(x_1^2+x_2^2\right)+2}$	2	$[-5.2, 5.2]^D$	-1
Egg holder	$F_9(x) = -(x_2+47)\sin\left(\sqrt{\left\lvert x_2+\frac{x_1}{2}+47\right\rvert}\right) - x_1\sin\left(\sqrt{\lvert x_1-x_2-47\rvert}\right)$	2	$[-512, 512]^D$	-959.6407
Himmelblau	$F_{10}(x) = \left(x_1^2+x_2-11\right)^2+\left(x_1+x_2^2-7\right)^2$	2	$[-5, 5]^D$	0
Levi 13	$F_{11}(x) = \sin^2(3\pi x_1)+(x-1)^2\left(1+\sin^2(3\pi x_2)\right)+(x_2-1)^2\left(1+\sin^2(2\pi x_2)\right)$	2	$[-10, 10]^D$	0
Ackley 1	$F_{12}(x) = -20e^{\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)} - e^{\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)} + 20 + e$	30	$[-32, 32]^D$	0
Griewank	$F_{13}(x) = 1+\sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	30	$[-600, 600]^D$	0
Happy cat	$F_{14}(x) = \sqrt[8]{\left(\lVert\mathbf{x}\rVert^2-n\right)^2}+\frac{1}{n}\left(\frac{1}{2}\lVert\mathbf{x}\rVert^2+\sum_{i=1}^{n}x_i\right)+\frac{1}{2}$	30	$[-2, 2]^D$	0
Michalewicz	$F_{15}(x) = -\sum_{i=1}^{n}\sin(x_i)\left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{20}$	10	$[0, \pi]^D$	-9.6602
Penalized 1	$F_{16}(x) = \frac{\pi}{n}\left[10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}\left(\left(y_i-1\right)^2\left(1+10\sin^2\left(\pi y_{i+1}\right)\right)\right)+\left(y_n-1\right)^2\right]+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i = 1+\frac{1}{4}(x_i+1)$ $u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i-a)^m & x_i < a \end{cases}$	30	$[-50, 50]^D$	0
Penalized 2	$F_{17}(x) = 0.1\left[\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}\left(\left(x_i-1\right)^2\left(1+\sin^2(3\pi x_{i+1})\right)\right)+\left(x_n-1\right)^2\left(1+\sin^2(2\pi x_n)\right)\right]+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	$[-50, 50]^D$	0
Periodic	$F_{18}(x) = 1+\sum_{i=1}^{n}\sin^2(x_i)-0.1e^{\left(\sum_{i=1}^{n}x_i^2\right)}$	30	$[-50, 50]^D$	0.9
Qing	$F_{19}(x) = \sum_{i=1}^{n}\left(x_i^2-i\right)^2$	30	$[-500, 500]^D$	0
Rastrigin	$F_{20}(x) = 10n+\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)\right)$	30	$[-5.12, 5.12]^D$	0
Rosenbrock	$F_{21}(x) = \sum_{i=1}^{n}\left(100\left(x_{i+1}-x_i^2\right)^2+\left(1-x_i\right)^2\right)$	30	$[-5, 10]^D$	0
Salomon	$F_{22}(x) = 1-\cos\left(2\pi\sqrt{\sum_{i=1}^{n}x_i^2}\right)+0.1\sqrt{\sum_{i=1}^{n}x_i^2}$	30	$[-100, 100]^D$	0
Yang 4	$F_{23}(x) = \left(\sum_{i=1}^{n}\sin^2(x_i)\right)e^{\left(-\sum_{i=1}^{n}\sin^2\sqrt{\lvert x_i\rvert}\right)}$	30	$[-10, 10]^D$	-1

3.3. Benchmark functions

In order to numerically prove the theoretical claims mentioned in the previous sections and to test the performance of the proposed algorithm, a wide range of experiments are conducted. The benchmark functions can be grouped into three classes. Unimodal benchmark functions have only one optimum and are suitable for testing the exploitation ability of optimization algorithms. Table 2 shows the seven fixed-dimension and scalable unimodal benchmark functions used in this study (F_1 to F_7). Multimodal benchmark functions have many local optima that can trap the algorithms; therefore, they can test the exploration ability of algorithms. Table 3 displays the 16 fixed-dimension and scalable multimodal benchmark functions on which GEO is tested (F_8 to F_{23}). The last class is the composite functions that are more challenging than the previous two classes. Composite functions can aptly represent the landscapes that metaheuristic algorithms may face in real-world mathematical problems. Composite functions are basically the shifted, rotated, biased, and hybridized version of the well-known unimodal and multimodal functions. The ten composite benchmark functions introduced in the CEC2017 competition are utilized in this study, the details of which are reported in Table 4 (F_{24} to F_{33}). Further details of CEC2017 composition functions can be found at (Awad, Ali, Suganthan, Liang, & Qu, 2017).

3.4. Qualitative results

This section explores a set of qualitative measures for the performance of GEO. Qualitative measures are commonly reported for new algorithms. The most important qualitative measures of single-objective optimization for GEO are presented in Fig. 14. It is noteworthy that this figure contains the qualitative measures for two unimodal functions, two multimodal functions, and three composite functions. The first column shows the landscape of the benchmark function. The second column displays the search history, which is basically the points that have been visited by all of GEO search agents to find the optimum. It is evident that GEO can search the entire landscape, but it puts more emphasis on exploring promising areas. The third column shows the trajectory of the first search agent along the x_1 axis (first decision variable). Plots in this column show that the search agents undergo drastic changes in their position in initial iterations of the optimization process while reducing the changes in later iterations to slow down and converge to the optimum. This behavior can guarantee the convergence of GEO (Qi, Zhu, & Zhang, 2017). The fourth column displays the mean fitness of the population over the course of iterations. It can be seen that the large values of the mean fitness and its rapid changes in initial iterations, followed by a reduction in value and diminishing changes implies the transition from high exploration in initial iterations toward high exploitation during the final iterations. This corresponds to the transition of golden eagles from intense cruise to intense attack. The last column depicts the convergence curve for the selected benchmark functions, which is the best position visited by GEO over the course of iterations. It shows how well GEO improves the fitness to finally converge toward the optimum. It is seen that in unimodal functions, the convergence curve is continuously improving. However, this might not be the case for multimodal and composite functions, where GEO is exposed to many local minima and may not visit better positions for some iterations.

3.5. Quantitative results

Although the qualitative measures proved the exploration and exploitation capability of GEO, they cannot fully reflect how well it can solve optimization problems. This section uses statistical measures to quantify the performance of GEO in different classes of benchmark functions. The arithmetic mean and the standard deviation obtained from 30 independent runs are used as statistical measures for revealing GEO's performance. The arithmetic mean shows how GEO performs on average, while the standard deviation shows how stable this algorithm

Table 4
Composite benchmark functions of CEC2017 competition.

	Equation	D	Bounds	f^*
CF1	$F_{24}(x) = \begin{cases} f_1 : \text{Shifted and rotated Rosenbrock's function} \\ f_2 : \text{Shifted and rotated High Conditioned Elliptic function} \\ f_3 : \text{Shifted and rotated Rastrigin's function} \end{cases}$ $\sigma = [10, 20, 30]$ $\lambda = [1, 10^{-6}, 1]$ $bias = [0, 100, 200]$	30	$[-100, 100]^D$	2100
CF2	$F_{25}(x) = \begin{cases} f_1 : \text{Shifted and rotated Rastrigin's function} \\ f_2 : \text{Shifted and rotated Griewank's function} \\ f_3 : \text{Shifted and rotated Modified Schwefel's function} \end{cases}$ $\sigma = [10, 20, 30]$ $\lambda = [1, 10, 1]$ $bias = [0, 100, 200]$	30	$[-100, 100]^D$	2200
CF3	$F_{26}(x) = \begin{cases} f_1 : \text{Shifted and rotated Rosenbrock's function} \\ f_2 : \text{Shifted and rotated Ackley's function} \\ f_3 : \text{Shifted and rotated Modified Schwefel's function} \\ f_4 : \text{Shifted and rotated Rastrigin's function} \end{cases}$ $\sigma = [10, 20, 30, 40]$ $\lambda = [1, 10, 1, 1]$ $bias = [0, 100, 200, 300]$	30	$[-100, 100]^D$	2300
CF4	$F_{27}(x) = \begin{cases} f_1 : \text{Shifted and rotated Ackley'sfunction} \\ f_2 : \text{Shifted and rotated High Conditioned Elliptic function} \\ f_3 : \text{Shifted and rotated Girewank's function} \\ f_4 : \text{Shifted and rotated Rastrigin's function} \end{cases}$ $\sigma = [10, 20, 30, 40]$ $\lambda = [1, 10^{-6}, 10, 1]$ $bias = [0, 100, 200, 300]$	30	$[-100, 100]^D$	2400
CF5	$F_{28}(x) = \begin{cases} f_1 : \text{Shifted and rotated Rastrigin's function} \\ f_2 : \text{Shifted and rotated Happy Cat function} \\ f_3 : \text{Shifted and rotated Ackley's function} \\ f_4 : \text{Shifted and rotated Discus function} \\ f_5 : \text{Shifted and rotated Rosenbrock's function} \end{cases}$ $\sigma = [10, 20, 30, 40, 50]$ $\lambda = [10, 1, 10, 10^{-6}, 1]$ $bias = [0, 100, 200, 300, 400]$	30	$[-100, 100]^D$	2500
CF6	$F_{29}(x) = \begin{cases} f_1 : \text{Shifted and rotated Expanded Schaffer's function} \\ f_2 : \text{Shifted and rotated Modified Schwefel's function} \\ f_3 : \text{Shifted and rotated Griewank's function} \\ f_4 : \text{Shifted and rotated Rosenbrock's function} \\ f_5 : \text{Shifted and rotated Rastrigin's function} \end{cases}$ $\sigma = [10, 20, 20, 30, 40]$ $\lambda = [10^{-26}, 10, 10^{-6}, 10, 5 \times 10^{-4}]$ $bias = [0, 100, 200, 300, 400]$	30	$[-100, 100]^D$	2600
CF7	$F_{30}(x) = \begin{cases} f_1 : \text{Shifted and rotated HGBat function} \\ f_2 : \text{Shifted and rotated Rastrigin's function} \\ f_3 : \text{Shifted and rotated Modified Schwefel's function} \\ f_4 : \text{Shifted and rotated Bent-Cigar function} \\ f_5 : \text{Shifted and rotated High Conditioned Elliptic function} \\ f_6 : \text{Shifted and rotated Expanded Schaffer's function} \end{cases}$ $\sigma = [10, 20, 30, 40, 50, 60]$ $\lambda = [10, 10, 2.5, 10^{-26}, 10^{-6}, 5 \times 10^{-4}]$ $bias = [0, 100, 200, 300, 400, 500]$	30	$[-100, 100]^D$	2700
CF8	$F_{31}(x) = \begin{cases} f_1 : \text{Shifted and rotated Ackley's function} \\ f_2 : \text{Shifted and rotated Griewank's function} \\ f_3 : \text{Shifted and rotated Discus function} \\ f_4 : \text{Shifted and rotated Rosenbrock's function} \\ f_5 : \text{Shifted and rotated Happy Cat function} \\ f_6 : \text{Shifted and rotated Expanded Schaffer's function} \end{cases}$ $\sigma = [10, 20, 30, 40, 50, 60]$ $\lambda = [10, 10, 10^{-6}, 1, 1, 5 \times 10^{-4}]$ $bias = [0, 100, 200, 300, 400, 500]$	30	$[-100, 100]^D$	2800
CF9	$F_{32}(x) = \begin{cases} f_1 : \text{Hybrid function 5 in CEC2017 competition} \\ f_2 : \text{Hybrid function 8 in CEC2017 competition} \\ f_3 : \text{Hybrid function 9 in CEC2017 competition} \end{cases}$ $\sigma = [10, 30, 50]$ $\lambda = [1, 1, 1]$ $bias = [0, 100, 200]$	30	$[-100, 100]^D$	2900
CF10	$F_{33}(x) = \begin{cases} f_1 : \text{Hybrid function 5 in CEC2017 competition} \\ f_2 : \text{Hybrid function 6 in CEC2017 competition} \\ f_3 : \text{Hybrid function 7 in CEC2017 competition} \end{cases}$ $\sigma = [10, 30, 50]$ $\lambda = [1, 1, 1]$ $bias = [0, 100, 200]$	30	$[-100, 100]^D$	3000

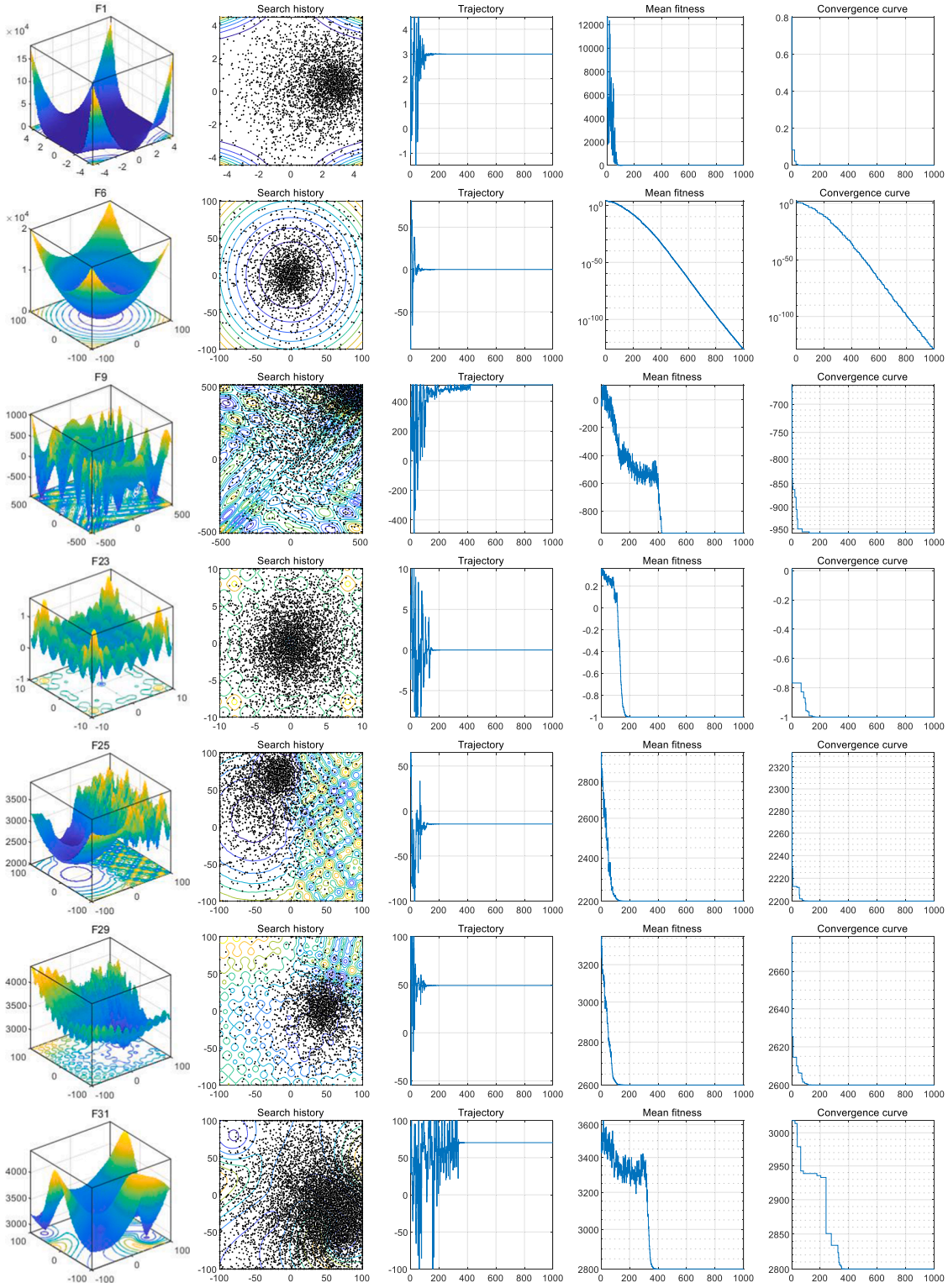

Fig. 14. Qualitative results including landscape, search history, the trajectory of the first agent in the first variable, mean fitness, and convergence curve.

is. Fixed-dimension unimodal and multimodal functions must be run with a fixed amount of decision variables. However, scalable unimodal and multimodal functions can be run with an arbitrary number of decision variables. All of the scalable functions were utilized with 30 dimensions.

The results of unimodal, multimodal, and composite benchmark functions are tabulated in Table 5, Table 6, and Table 7, respectively. In all of these tables, the best average performances are highlighted with the bold font for each benchmark function. Table 5 shows that GEO outperforms other algorithms in half of the unimodal functions and competitive results in other unimodal functions. This depicts the good ability of GEO to use the best solutions to guide the search toward promising areas of the search region. The results of the standard deviation prove GEO's stability. Table 6 reveals that GEO outperforms other algorithms in 13 out of 16 multimodal functions. This certifies the ability of GEO to explore different regions within the search region to find

Table 5
Results of unimodal benchmark functions.

		GEO	GWO	GA	CSA	PSO	HS	DA
F_1	Mean	**0.00E+00**	1.02E−08	6.64E−12	2.49E−24	2.39E−29	2.03E−02	3.55E−03
	Std	0.00E+00	8.77E−09	1.69E−11	4.24E−24	7.54E−29	2.38E−02	1.83E−02
F_2	Mean	1.99E−94	**3.67E−320**	9.64E−14	1.40E−25	5.85E−33	4.27E−03	4.21E−06
	Std	5.15E−94	0.00E+00	2.71E−13	1.91E−25	2.44E−32	4.13E−03	1.12E−05
F_3	Mean	6.28E−126	**0.00E+00**	7.28E−14	3.81E−25	7.63E−45	2.33E−05	4.72E−07
	Std	1.73E−125	0.00E+00	1.66E−13	5.92E−25	3.94E−44	4.91E−05	1.48E−06
F_4	Mean	**−1.00E+00**	**−1.00E+00**	−1.00E+00	−1.00E+00	−5.92E−01	−9.93E−01	−9.56E−01
	Std	3.24E−16	0.00E+00	2.14E−06	4.95E−07	1.72E−01	9.17E−03	4.57E−02
F_5	Mean	−4.91E+00	**−5.00E+00**	−4.02E+00	−4.21E+00	−2.30E+00	−1.59E+00	−3.10E+00
	Std	7.41E−03	1.21E−07	4.41E−02	4.04E−02	2.10E−01	7.41E−02	4.32E−01
F_6	Mean	4.56E−12	**8.01E−77**	1.15E−01	1.75E−02	9.68E+03	2.71E+02	8.99E+02
	Std	3.02E−12	2.11E−76	6.64E−02	8.24E−03	5.85E+03	3.40E+02	6.98E+02
F_7	Mean	**3.22E−14**	3.07E−01	3.79E−04	7.83E−05	2.68E+01	1.05E+00	3.30E+00
	Std	4.17E−14	2.47E−01	2.60E−04	3.70E−05	1.33E+01	3.08E+00	1.94E+00

Table 6
Results of multimodal benchmark functions.

		GEO	GWO	GA	CSA	PSO	HS	DA
F_8	Mean	**−1.00E+00**	−9.98E−01	−1.00E+00	−1.00E+00	−1.00E+00	−9.50E−01	−9.83E−01
	Std	0.00E+00	1.14E−02	4.22E−12	0.00E+00	1.26E−04	4.20E−02	2.82E−02
F_9	Mean	**−9.60E+02**	−8.92E+02	−9.60E+02	**−9.60E+02**	−9.56E+02	−9.42E+02	−9.28E+02
	Std	5.68E−13	8.22E+01	2.25E−04	5.68E−13	1.18E+01	3.01E+01	4.52E+01
F_{10}	Mean	**0.00E+00**	4.77E−05	8.88E−13	9.27E−24	5.26E−32	4.88E−02	5.53E−04
	Std	0.00E+00	2.54E−04	1.39E−12	9.04E−24	1.97E−31	6.19E−02	1.12E−03
F_{11}	Mean	**1.35E−31**	3.06E−08	2.73E−12	2.74E−23	**1.35E−31**	1.24E−02	9.99E−04
	Std	6.57E−47	2.78E−08	1.10E−11	5.07E−23	6.57E−47	2.94E−02	2.70E−03
F_{12}	Mean	1.98E−01	**1.01E−15**	2.48E+00	3.31E+00	1.59E+01	4.69E+00	7.03E+00
	Std	5.24E−01	6.38E−16	7.38E−01	5.94E−01	2.06E+00	3.39E+00	2.62E+00
F_{13}	Mean	**5.01E−03**	3.88E+00	2.42E−01	1.83E−01	1.04E+02	3.34E+00	1.87E+01
	Std	5.53E−03	2.94E+00	9.10E−02	5.18E−02	4.24E+01	4.10E+00	1.07E+01
F_{14}	Mean	**2.29E−01**	5.28E−01	4.65E−01	5.53E−01	5.39E−01	4.38E−01	7.13E−01
	Std	5.13E−02	1.09E−01	1.17E−01	1.19E−01	6.39E−02	1.64E−01	1.05E−01
F_{15}	Mean	**−9.50E+00**	−7.70E+00	−9.19E+00	−8.58E+00	−6.04E+00	−5.04E+00	−6.03E+00
	Std	1.94E−01	1.11E+00	3.19E−01	7.66E−01	3.85E−01	6.90E−01	7.38E−01
F_{16}	Mean	**2.08E−02**	2.58E−02	2.17E+00	3.21E+00	1.41E+07	3.06E+04	2.62E+01
	Std	4.15E−02	1.21E−02	1.10E+00	1.25E+00	7.86E+06	1.19E+05	5.82E+01
F_{17}	Mean	**7.93E−03**	3.30E−01	3.88E−02	1.11E−01	4.88E+07	1.12E+02	8.40E+04
	Std	7.24E−03	1.68E−01	3.59E−02	1.14E−01	1.91E+07	2.76E+02	2.71E+05
F_{18}	Mean	**1.00E+00**	2.23E+00	1.07E+00	1.01E+00	6.67E+00	1.04E+00	4.35E+00
	Std	1.01E−04	1.84E+00	2.20E−02	3.01E−03	6.04E−01	8.78E−02	8.68E−01
F_{19}	Mean	**2.54E−01**	8.72E+02	1.25E+02	6.80E+01	1.30E+10	1.30E+07	9.93E+07
	Std	3.79E−01	4.86E+02	7.14E+01	2.98E+01	5.41E+09	3.96E+07	2.82E+08
F_{20}	Mean	1.09E+01	**2.03E+00**	2.41E+01	2.39E+01	2.74E+02	2.10E+01	1.52E+02
	Std	3.82E+00	4.64E+00	5.05E+00	6.97E+00	1.50E+01	3.63E+01	4.47E+01
F_{21}	Mean	**4.17E+00**	2.63E+01	4.52E+01	1.02E+02	3.05E+04	2.18E+03	5.49E+03
	Std	1.28E+01	6.35E−01	6.08E+01	7.13E+01	1.11E+04	4.05E+03	5.39E+03
F_{22}	Mean	4.03E−01	**1.73E−01**	6.68E−01	8.36E−01	1.25E+01	2.87E+00	4.06E+00
	Std	6.57E−02	4.42E−02	9.39E−02	1.34E−01	1.36E+00	2.17E+00	1.72E+00
F_{23}	Mean	**2.22E−20**	9.39E−17	2.96E−15	3.88E−16	1.83E−10	2.41E−13	2.81E−12
	Std	7.01E−20	2.90E−17	1.42E−15	2.15E−16	9.82E−11	3.91E−13	3.91E−12

better solutions. The standard deviations reported in this table reveal that GEO yields highly stable results in the majority of the multimodal benchmark functions, compared to other algorithms. Table 7 provides the results for composite functions from the CEC2017 competition test suite. The contents of this table demonstrate that according to the average fitnesses obtained, GEO is able to outperform the other algorithms in eight of the ten composite functions available in the test suite. In addition, the standard deviations confirm the stability of the obtained solutions by GEO since it has the lowest standard deviation for most of the composite test functions. Fig. 15 provides the comparative box plots for the results of composite functions.

3.6. Convergence analysis

The effectiveness of GEO was verified in the previous subsection. However, the convergence analysis can better reveal the explorative and exploitative behavior of GEO. Fig. 16 shows the convergence curve for GEO and other algorithms for six functions (F_1 and F_7 from unimodal functions, F_{10} and F_{19} from multimodal functions, and F_{27} and F_{32} from composite functions). It can be concluded that GEO converges a little later than other algorithms in initial iterations, but can often compensate with better final values for the objective function.

Table 7
Results of composite benchmark functions.

		GEO	GWO	GA	CSA	PSO	HS	DA
F_{24}	Mean	**2.34E+03**	2.40E+03	2.44E+03	2.43E+03	2.55E+03	2.71E+03	2.59E+03
	Std	7.67E+00	4.95E+01	2.68E+01	2.92E+01	1.02E+01	2.68E+01	6.06E+01
F_{25}	Mean	**2.30E+03**	5.27E+03	2.31E+03	2.49E+03	3.05E+03	9.91E+03	8.19E+03
	Std	1.55E+00	2.23E+03	2.38E+00	8.56E+02	2.51E+02	5.42E+02	1.88E+03
F_{26}	Mean	**2.69E+03**	2.77E+03	2.87E+03	2.93E+03	2.89E+03	3.43E+03	3.07E+03
	Std	1.59E+01	5.40E+01	6.56E+01	9.81E+01	1.64E+01	5.66E+01	1.15E+02
F_{27}	Mean	**2.85E+03**	2.98E+03	3.02E+03	3.12E+03	3.05E+03	3.90E+03	3.22E+03
	Std	7.19E+00	7.30E+01	4.35E+01	1.27E+02	1.32E+01	1.25E+02	8.50E+01
F_{28}	Mean	**2.93E+03**	3.00E+03	2.95E+03	2.94E+03	3.38E+03	6.26E+03	3.27E+03
	Std	1.34E+01	5.77E+01	1.66E+01	2.19E+01	2.22E+02	1.05E+03	2.41E+02
F_{29}	Mean	**4.00E+03**	4.85E+03	6.09E+03	5.34E+03	6.42E+03	1.09E+04	7.30E+03
	Std	1.10E+03	4.68E+02	1.54E+03	1.47E+03	1.68E+02	1.15E+03	1.13E+03
F_{30}	Mean	3.26E+03	**3.25E+03**	3.38E+03	3.33E+03	3.26E+03	4.03E+03	3.40E+03
	Std	1.50E+01	1.59E+01	5.48E+01	6.93E+01	2.15E+01	2.14E+02	8.64E+01
F_{31}	Mean	**3.27E+03**	3.42E+03	3.29E+03	3.30E+03	3.59E+03	1.01E+04	3.90E+03
	Std	1.44E+01	9.23E+01	1.92E+01	2.32E+01	1.22E+02	1.72E+03	3.46E+02
F_{32}	Mean	**3.70E+03**	3.83E+03	4.25E+03	4.29E+03	4.60E+03	6.25E+03	4.92E+03
	Std	9.48E+01	1.97E+02	2.36E+02	2.13E+02	1.74E+02	4.03E+02	4.81E+02
F_{33}	Mean	1.47E+06	7.25E+06	**8.56E+05**	1.42E+06	4.97E+06	7.33E+08	3.16E+07
	Std	5.59E+05	5.10E+06	2.95E+05	1.38E+06	8.81E+06	3.44E+08	3.27E+07

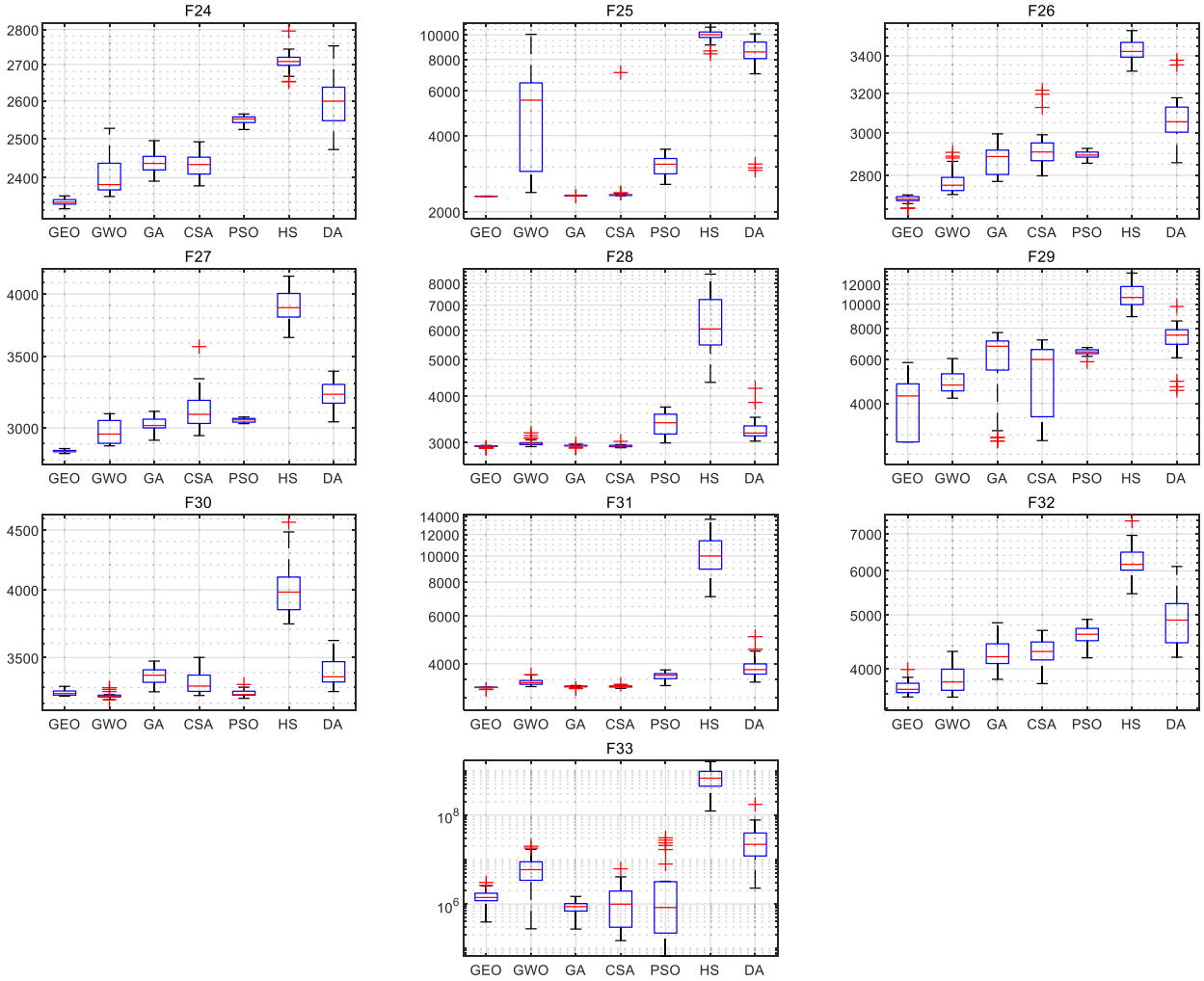

Fig. 15. Boxplots of the results of CEC2017 composite functions.

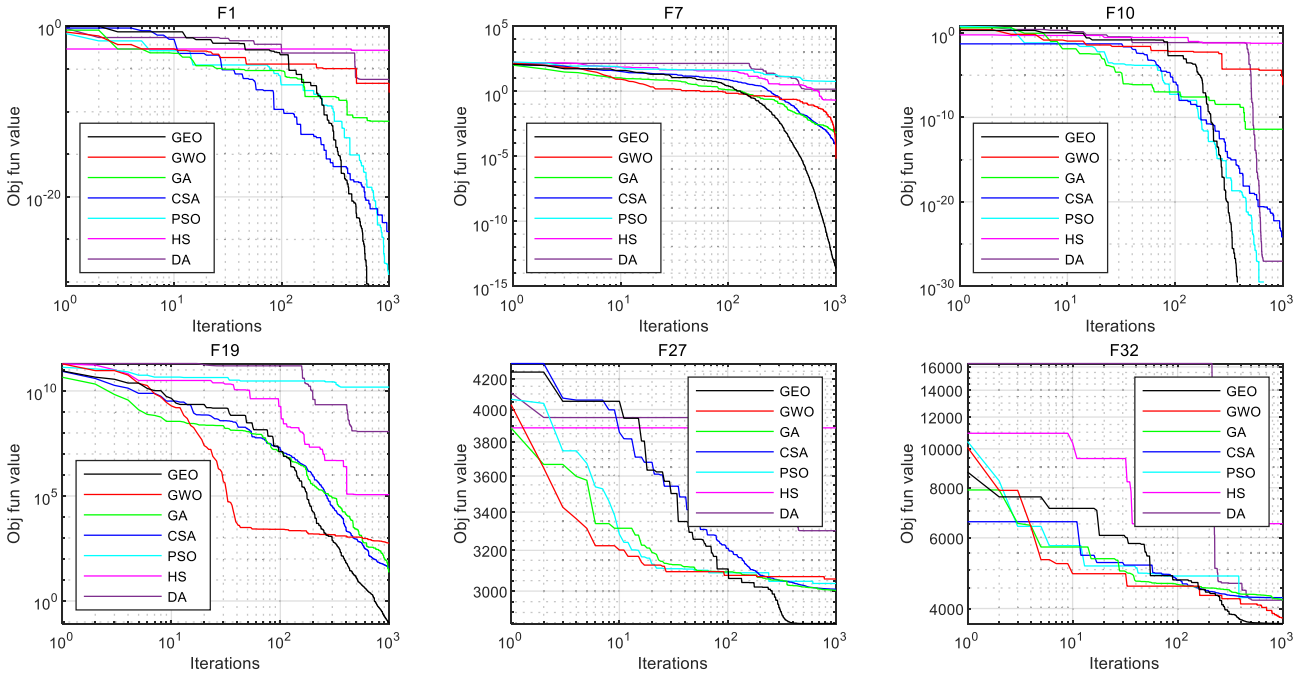

Fig. 16. Convergence curve of GEO and compared algorithms.

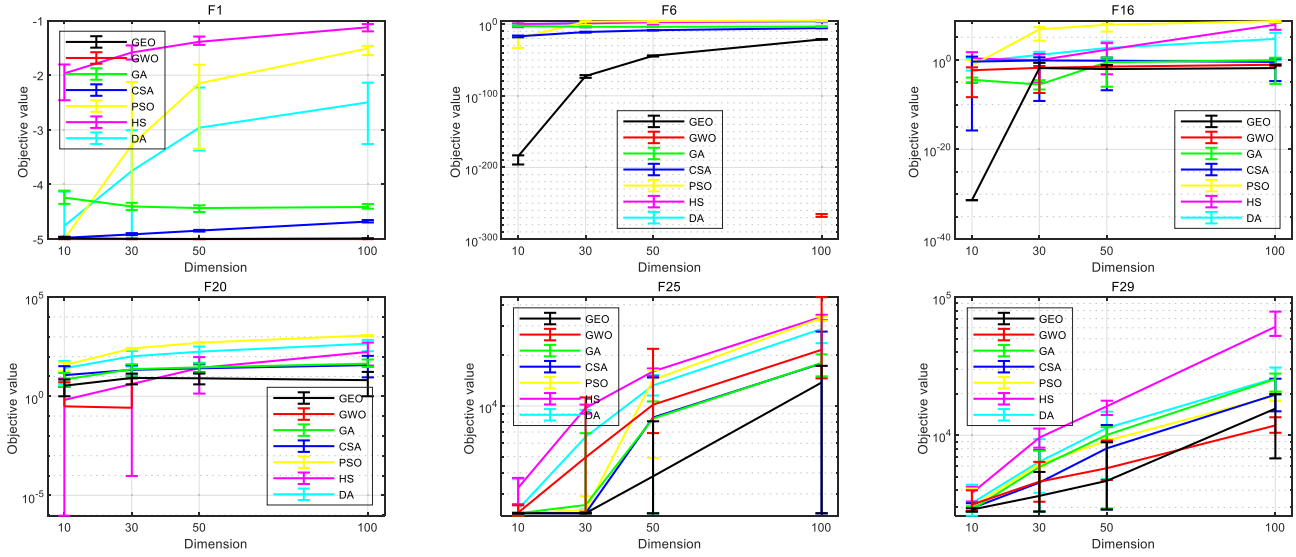

Fig. 17. Results of scalability analysis for objective value.

3.7. Scalability analysis

This subsection presents the results of the scalability analysis conducted to see how GEO is scalable for problems with a large number of decision variables. Six benchmark functions from different classes were considered for scalability analysis (F_1 and F_6 from unimodal functions, F_{16} and F_{20} form multimodal functions, and F_{25} and F_{29} from composite functions). In addition to GEO, all the other algorithms previously compared in this study also participate in this analysis for comparison. The experiment is carried out for 10D, 30D, 50D, and 100D since the CEC2017 test suite only supports these numbers of dimensions. The best objective value obtained and the computation time of each algorithm was recorded for 30 independent runs on each benchmark function. Fig. 17 displays the results for the best objective value obtained in the form of error bars. The center points show the arithmetic mean of the 30 independent runs, while the upper and lower bars show the minimum and maximum objective values obtained. Results confirm GEO's almost consistent performance as the dimensions rise. Fig. 18 shows the same statistics for computation times. It is observed that in terms of computation time, GEO belongs to the midpack and can retain its relative computation time compared to other algorithms. This implies that the temporal performance of GEO is consistent relative to that of other algorithms when we transit from small to large problems. It worths noting that the plots in Figs. 17 and 18 have a logarithmic scale along the y-axis to better demonstrate the differences in small values. However, since logarithmically scaled plots cannot show the values exactly equal to zero. In this experiment, the maximum number of function evaluations of 10^6 was used, similar to the CEC2017 competition (Awad et al., 2017), and a population size of $2 \times D$ was used for all of the algorithms and all of the benchmark functions for the scalability analysis.

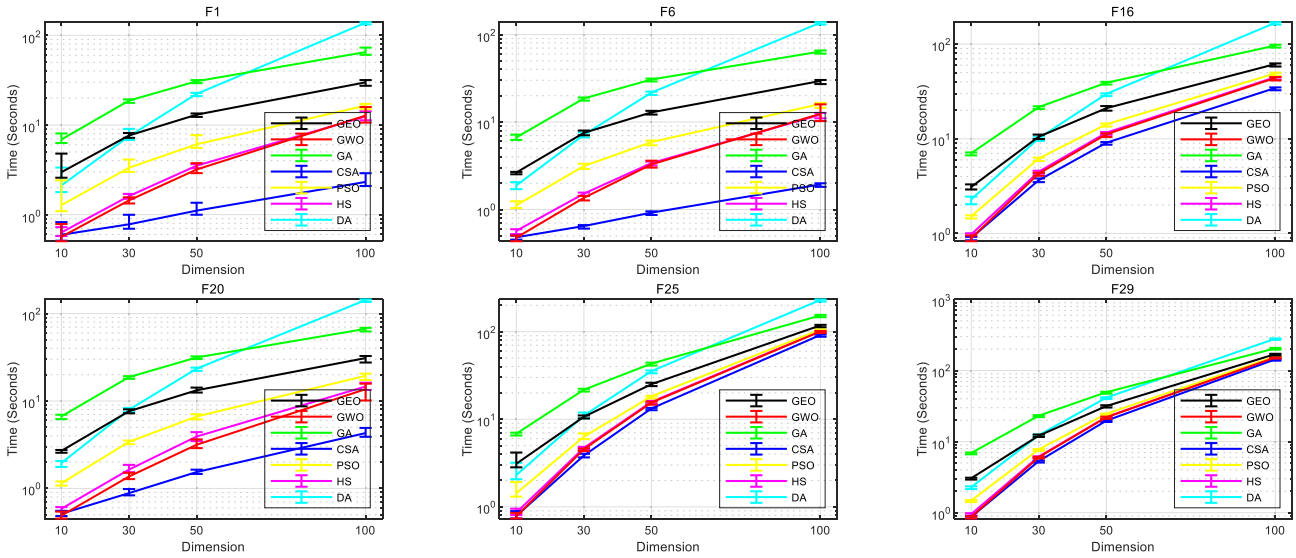

Fig. 18. Results of scalability analysis for computation time.

Table 8
Parameter setting for multi-objective benchmark functions.

Algorithm	Parameter	Value
All algorithms	Population size	200
MOGEO	Archive size	100
MOGWO	Archive size	100
	Number of grids	20
	Grid multiplier	10
NSGA-II	–	–
MOPSO	Archive size	100
	Number of grids	20
	Grid multiplier	10
MOSSA	Archive size	100
	Number of grids	20
	Grid multiplier	10
	Number of leader salps	Population size / 2

4. Multi-objective optimization results for MOGEO

This section provides the results of applying MOGEO to multi-objective benchmark functions. The parameters specific to multi-objective optimization are set according to Table 8. For parameters that the algorithms share with their single-objective version, the parameters introduced in Table 1 are used. Since both MOGEO, MOGWO, MOPSO, and MOSSA are archive-based solvers, a similar archive size is used for both of them. However, MOGEO does not use the grid mechanism and does not need parameters like the number of grids and grid multiplier. CEC2009 (Zhang, Zhou, Zhao, Suganthan, Liu, & Tiwari, 2009) and DTLZ (Deb, Thiele, Laumanns, & Zitzler, 2005) test suites, which are among the most challenging test suites for multi-objective problems, are utilized to test the performance of MOGEO. Details of the mathematical formulation of CEC2009 and DTLZ benchmark functions are presented in Table 9 and Table 10, respectively. In consistence with previous experiments, the results of MOGEO are compared to that of four well-known multi-objective algorithms, namely Multi-Objective Grey Wolf Optimizer (MOGWO) (Mirjalili et al., 2016), Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2000), Multi-Objective Particle Swarm Optimization (MOPSO) (Coello Coello & Lechuga, 2002), and Multi-Objective Salp Swarm Algorithm (MOSSA) (Mirjalili et al., 2017).

Since the solution to the multi-objective problems are a set of solutions rather than a single solution, the comparison of the Pareto fronts becomes an issue. Inverse Generational Distance (IGD) (Sierra & Coello Coello, 2005; Van Veldhuizen & Lamont, 1998) provides a way to quantify the obtained Pareto front by mapping the whole Pareto front to a single value that can be used for comparing the quality of the obtained Pareto fronts. It measures the average distance between each member of the true Pareto front to the nearest member of the obtained Pareto front.

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{13}$$

where d_i is the Euclidean distance between the i-th member of the true Pareto front and the nearest member of the obtained Pareto front and n is the total number of members of the true Pareto front.

Table 11 presents the arithmetic mean, and the standard deviation of the IGD score calculated for each of the 30 independent runs of each algorithm on each of the multi-objective benchmark functions. It reveals that MOGEO outperformed the other algorithms in eight of the multi-objective benchmark functions. MOGEO was also able to provide more stable results in three of the problems in the test suite. This confirms that MOGEO can successfully handle multi-objective optimization problems. MOGEO's higher rate of converging to the optimal Pareto front can be attributed to the fact the search agents always choose prey from the external archive that stores the Pareto front obtained so far. The archive update mechanism, when triggered, usually drop a member from the most densely populated areas of the archive, which helps MOGEO converge to more uniformly distributed fronts. The good exploration mechanism of GEO, which also benefits MOGEO and was numerically proved in the previous section, helps MOGEO avoid local fronts to converge to the true Pareto front. Fig. 19 displays the best Pareto front (out of the 30 independent runs) by the tested algorithms in terms of IGD.

MOGEO is also tested on the DTLZ test suite, which is another challenging multi-objective test suite. A notable feature of this test suite is its scalability in the number of objectives. In other words, this test suite can be used with any number of objective functions. In this study, we focus on problems with two and three objectives. Table 12 displays the arithmetic mean and the standard deviation of IGD scores for 30 independent runs of MOGEO on the problems of the DTLZ test suite with two and three objective functions. It is revealed that MOGEO is able to outperform the other algorithms in one problem out of seven bi-objective problems, and two out of seven tri-objective problems. MOGEO has provided competitive results in other problems. Figs. 20 and 21 display the best optimal Pareto front achieved by the algorithms, according to IGD scores.

Table 9
Multi-objective benchmark functions from the CEC2009 test suite.

Name	Equation	D
UF1	$\begin{cases} f_1(x) = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right]^2 \\ f_2(x) = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right]^2 \end{cases}$ $J_1 = \{j \mid j \text{is odd and} 2 \le j \le n\}, J_2 = \{j \mid j \text{is even and} 2 \le j \le n\}$	30
UF2	$\begin{cases} f_1(x) = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} y_j^2 \\ f_2(x) = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} y_j^2 \end{cases}$ $J_1 = \{j \mid j \text{is odd and} 2 \le j \le n\}, J_2 = \{j \mid j \text{is even and} 2 \le j \le n\}$ $y_j = \begin{cases} x_j - \left[0.3x_1^2\cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right]\cos\left(6\pi x_1 + \frac{j\pi}{n}\right) & j \in J_1 \\ x_j - \left[0.3x_1^2\cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right]\cos\left(6\pi x_1 + \frac{j\pi}{n}\right) & j \in J_2 \end{cases}$	30
UF3	$\begin{cases} f_1(x) = x_1 + \frac{2}{\lvert J_1\rvert}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right) \\ f_2(x) = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right) \end{cases}$ $J_1 = \{j \mid j \text{is odd and} 2 \le j \le n\}, J_2 = \{j \mid j \text{is even and} 2 \le j \le n\}$ $y_j = x_j - x_1^{0.5\left(1 + \frac{3(j-2)}{n-2}\right)}, j = 2, \cdots, n$	30
UF4	$\begin{cases} f_1(x) = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} h\left(y_j\right) \\ f_2(x) = 1 - x_1^2 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} h\left(y_j\right) \end{cases}$ $J_1 = \{j \mid j \text{is odd and} 2 \le j \le n\}, J_2 = \{j \mid j \text{is even and} 2 \le j \le n\}$ $y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \cdots, n, h(t) = \frac{\lvert t\rvert}{1 + e^{2\lvert t\rvert}}$	30
UF5	$\begin{cases} f_1(x) = x_1 + \left(\frac{1}{2N} + \varepsilon\right)\lvert\sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} h\left(y_j\right) \\ f_2(x) = 1 - x_1 + \left(\frac{1}{2N} + \varepsilon\right)\lvert\sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} h\left(y_j\right) \end{cases}$ $J_1 = \{j \mid j \text{is odd and} 2 \le j \le n\}, J_2 = \{j \mid j \text{is even and} 2 \le j \le n\}$ $y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \cdots, n, h(t) = 2t^2 - \cos(4\pi t) + 1, N \text{ is an integer}, \varepsilon > 0$	30
UF6	$\begin{cases} f_1(x) = x_1 + \max\left\{0, 2\left(\frac{1}{2N}\right)\sin(2N\pi x_1)\right\} + \frac{2}{\lvert J_1\rvert}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right) \\ f_2(x) = 1 - x_1 + \max\left\{0, 2\left(\frac{1}{2N}\right)\sin(2N\pi x_1)\right\} + \frac{2}{\lvert J_2\rvert}\left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right) \end{cases}$ $J_1 = \{j \mid j \text{is odd and} 2 \le j \le n\}, J_2 = \{j \mid j \text{is even and} 2 \le j \le n\}$ $y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \cdots, n$	30
UF7	$\begin{cases} f_1(x) = \sqrt[5]{x_1} + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} y_j^2 \\ f_2(x) = 1 - \sqrt[5]{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} y_j^2 \end{cases}$ $J_1 = \{j \mid j \text{is odd and} 2 \le j \le n\}, J_2 = \{j \mid j \text{is even and} 2 \le j \le n\}$ $y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \cdots, n$	30
UF8	$\begin{cases} f_1(x) = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right)^2 \\ f_2(x) = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left(x_j - 2x_2\sin\left(2\pi x_2 + \frac{j\pi}{n}\right)\right)^2 \\ f_3(x) = \sin(0.5x_1\pi) + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right)^2 \end{cases}$ $J_1 = \{j \mid 3 \le j \le n, \text{and} j-1 \text{is a mulitplication of} 3\},$ $J_2 = \{j \mid 3 \le j \le n, \text{and} j-2 \text{is a mulitplication of} 3\},$ $J_3 = \{j \mid 3 \le j \le n, \text{and} j \text{is a mulitplication of} 3\}$	30
UF9		30

(*continued on next page*)

Table 9 (*continued*)

Name	Equation	D
UF10	$\begin{cases} f_1(x) = 0.5\left[\max\left\{0,(1+\varepsilon)\left(1-4(2x_1-1)^2\right)\right\}+2x_1\right]x_2+\frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left(x_j-2x_2\sin\left(2\pi x_1+\frac{j\pi}{n}\right)\right)^2 \\ f_2(x) = 0.5\left[\max\left\{0,(1+\varepsilon)\left(1-4(2x_1-1)^2\right)\right\}-2x_1+2\right]x_2+\frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left(x_j-2x_2\sin\left(2\pi x_2+\frac{j\pi}{n}\right)\right)^2 \\ f_3(x) = 1-x_2+\frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}\left(x_j-2x_2\sin\left(2\pi x_1+\frac{j\pi}{n}\right)\right)^2 \end{cases}$ $J_1 = \{j\mid 3\le j\le n, \text{and} j-1 \text{is a mulitplication of} 3\}$, $J_2 = \{j\mid 3\le j\le n, \text{and} j-2 \text{is a mulitplication of} 3\}, J_3 = \{j\mid 3\le j\le n, \text{and} j \text{is a mulitplication of} 3\}$ $\begin{cases} f_1(x) = \cos(0.5x_1\pi)\cos(0.5x_2\pi)+\frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left[4y_j^2-\cos\left(8\pi y_j\right)+1\right] \\ f_2(x) = \cos(0.5x_1\pi)\cos(0.5x_2\pi)+\frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left[4y_j^2-\cos\left(8\pi y_j\right)+1\right] \\ f_3(x) = \sin(0.5x_1\pi)+\frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}\left[4y_j^2-\cos\left(8\pi y_j\right)+1\right] \end{cases}$ $J_1 = \{j\mid 3\le j\le n, \text{and} j-1 \text{is a mulitplication of} 3\}$, $J_2 = \{j\mid 3\le j\le n, \text{and} j-2 \text{is a mulitplication of} 3\}$, $J_3 = \{j\mid 3\le j\le n, \text{and} j \text{is a multiplication of} 3\}$, $y_j = x_j-2x_2\sin\left(2\pi x_1+\frac{j\pi}{n}\right), j=3,\cdots,n$	10

5. Engineering benchmark tests

In order to test how the proposed GEO can solve real-world engineering problems, the proposed GEO is applied to five well-known engineering benchmark problems in this section. The nonlinear nature of many engineering optimization problems makes metaheuristic algorithms a compelling candidate for solving these problems. In this study, we solve the following engineering benchmark problems: three-bar truss design, cantilever beam design, tension/compression spring design, and welded beam design. In all of the tests, the results of GEO is compared to that of other metaheuristic methods that are already used in previous sections.

5.1. Constraint handling method

The distinguishing feature of the benchmark problems of this section is that they contain constraints. Therefore, the constraints should be handled properly so that the obtained results do not significantly violate the constraints. Constraint handling is one of the challenges in optimization problems, and various methods have been proposed to overcome this challenge. We use the penalty function approach in this study. The penalty function can be defined as (14) (Yang & Karamanoglu, 2013).

$$F\left(x,m_i,v_j\right)=f(x)+\sum_{i=1}^{M}m_i\varphi_i^2+\sum_{j=1}^{N}v_j\omega_j^2 \tag{14}$$

where $f(x)$ is the original objective function, M is the number of inequality constraints, m_i is the penalty factor for inequality constraints, and φ_i is the amount of constraint violation for the i-th inequality constraint, N is the number of equality constraints, v_j is the penalty factor for equality constraints, and ω_j is the amount of constraint violation for the j-th equality constraint. The advantage of using the penalty function is that it transforms the constrained problem into an unconstrained problem. Important notice for implementing penalty function is to assign suitable values for penalty factors (m_i and v_j). We use 10^{15} for both of the penalty factors, which is suitable in this regard (Yang, 2014).

5.2. Three-bar truss design

This engineering problem seeks to find the area of bars 1 (x_1) and 3 (x_2) that minimizes the total weight of the truss. The structure of the three-bar design problem is presented in Fig. 22, and the mathematical formulation is shown in Eq. (15).

$$\begin{aligned} &\text{Minimize } f(\vec{x})=\left(2\sqrt{2}x_1+x_2\right)\times l \\ &\text{Subjectto :} \\ &g_1(\vec{x})=\frac{\sqrt{2}x_1+x_2}{\sqrt{2}x_1^2+2x_1x_2}P-\sigma\le 0 \\ &g_2(\vec{x})=\frac{x_2}{\sqrt{2}x_1^2+2x_1x_2}P-\sigma\le 0 \\ &g_3(\vec{x})=\frac{1}{\sqrt{2}x_2+x_1}P-\sigma\le 0 \end{aligned} \tag{15}$$

Where

$$l=100\ \text{cm},\ P=2\ \text{KN/cm}^2,\ \sigma=2\ \text{KN/cm}^2,\ 0\le x_1,\ x_2\le 1$$

Optimal values of decision variables (x_i), constraint violation (g_i), and the optimal objective function values (f) obtained by applying GEO and other algorithms on the three-bar truss design problem are tabulated in Table 13. It reveals that the proposed GEO can outperform GWO, GA, PSO, HS, and DA while showing competitive results compared to CSA. It can also be witnessed that the first constraint (g_1) is active in the optimal solution, and GEO is among the algorithms that have the smallest constraint violation. This confirms that the proposed algorithm can perform quite well in constrained problems.

Table 13 only showed the best obtained results. To see which algorithms have similar performance, we need to take into account the results of all of the 30 runs of the algorithms on the problem. A Kruskal-Wallis test is performed here to see whether the mean objective function obtained by algorithms are significantly different. Table 14 shows that the null hypothesis is rejected. In other words, one or more of the algorithms has significantly different performance compared to the others. To find out which algorithms perform statistically similar, a multiple comparisons (post hoc) test is performed. Fig. 23 shows the confidence intervals of the Tukey-Kramer test, which reveals that GEO has statistically similar performance compared to GWO.

5.3. Cantilever beam design

This problem considers finding the height of five attached hollow blocks (h_1 to h_5) in the form of a cantilever beam so that the total weight of the structure is minimized. The structure of the cantilever beam is

Table 10
Multi-objective benchmark functions from the DTLZ test suite.

Name	Equation	D	Number of objectives
DTLZ 1	$\begin{cases} f_1(x) = \frac{1}{2}x_1x_2\cdots x_{M-1}(1+g(x_M)) \\ f_2(x) = \frac{1}{2}x_1x_2\cdots(1-x_{M-1})(1+g(x_M)) \\ \vdots \\ f_{M-1}(x) = \frac{1}{2}x_1(1-x_2)(1+g(x_M)) \\ f_M(x) = \frac{1}{2}(1-x_1)(1+g(x_M)) \end{cases}$ $g(x_M) = 100\left[\lvert x_M\rvert + \sum_{x_i\in x_M}(x_i-0.5)^2 - \cos(20\pi(x_i-0.5))\right]$ $0 \le x \le 1$, for $i = 1,2,\cdots,n$	3	2, 3
DTLZ 2	$\begin{cases} f_1(x) = (1+g(x_M))\cos\left(x_1\frac{\pi}{2}\right)\cdots\cos\left(x_{M-2}\frac{\pi}{2}\right)\cos\left(x_{M-1}\frac{\pi}{2}\right) \\ f_2(x) = (1+g(x_M))\cos\left(x_1\frac{\pi}{2}\right)\cdots\cos\left(x_{M-2}\frac{\pi}{2}\right)\sin\left(x_{M-1}\frac{\pi}{2}\right) \\ f_3(x) = (1+g(x_M))\cos\left(x_1\frac{\pi}{2}\right)\cdots\sin\left(x_{M-2}\frac{\pi}{2}\right) \\ \vdots \\ f_M(x) = (1+g(x_M))\sin\left(x_1\frac{\pi}{2}\right) \end{cases}$ $g(x_M) = \sum_{x_i\in x_M}(x_i-0.5)^2$ $0 \le x \le 1$, for$i = 1,2,\cdots,n$	10	2, 3
DTLZ 3	$\begin{cases} f_1(x) = (1+g(x_M))\cos\left(x_1\frac{\pi}{2}\right)\cdots\cos\left(x_{M-2}\frac{\pi}{2}\right)\cos\left(x_{M-1}\frac{\pi}{2}\right) \\ f_2(x) = (1+g(x_M))\cos\left(x_1\frac{\pi}{2}\right)\cdots\cos\left(x_{M-2}\frac{\pi}{2}\right)\sin\left(x_{M-1}\frac{\pi}{2}\right) \\ f_3(x) = (1+g(x_M))\cos\left(x_1\frac{\pi}{2}\right)\cdots\sin\left(x_{M-2}\frac{\pi}{2}\right) \\ \vdots \\ f_M(x) = (1+g(x_M))\sin\left(x_1\frac{\pi}{2}\right) \end{cases}$ $g(x_M) = 100\left[\lvert x_M\rvert + \sum_{x_i\in x_M}(x_i-0.5)^2 - \cos(20\pi(x_i-0.5))\right]$ $0 \le x \le 1$, for$i = 1,2,\cdots,n$	3	2, 3
DTLZ 4	$\begin{cases} f_1(x) = (1+g(x_M))\cos\left(x_1^\alpha\frac{\pi}{2}\right)\cdots\cos\left(x_{M-2}^\alpha\frac{\pi}{2}\right)\cos\left(x_{M-1}^\alpha\frac{\pi}{2}\right) \\ f_2(x) = (1+g(x_M))\cos\left(x_1^\alpha\frac{\pi}{2}\right)\cdots\cos\left(x_{M-2}^\alpha\frac{\pi}{2}\right)\sin\left(x_{M-1}^\alpha\frac{\pi}{2}\right) \\ f_3(x) = (1+g(x_M))\cos\left(x_1^\alpha\frac{\pi}{2}\right)\cdots\sin\left(x_{M-2}^\alpha\frac{\pi}{2}\right) \\ \vdots \\ f_M(x) = (1+g(x_M))\sin\left(x_1^\alpha\frac{\pi}{2}\right) \end{cases}$ $g(x_M) = \sum_{x_i\in x_M}(x_i-0.5)^2$ $\alpha = 100$ $0 \le x \le 1$, for$i = 1,2,\cdots,n$	30	2, 3
DTLZ 5	$\begin{cases} f_1(x) = (1+g(x_M))\cos\left(\theta_1\frac{\pi}{2}\right)\cdots\cos\left(\theta_{M-2}\frac{\pi}{2}\right)\cos\left(\theta_{M-1}\frac{\pi}{2}\right) \\ f_2(x) = (1+g(x_M))\cos\left(\theta_1\frac{\pi}{2}\right)\cdots\cos\left(\theta_{M-2}\frac{\pi}{2}\right)\sin\left(\theta_{M-1}\frac{\pi}{2}\right) \\ f_3(x) = (1+g(x_M))\cos\left(\theta_1\frac{\pi}{2}\right)\cdots\sin\left(\theta_{M-2}\frac{\pi}{2}\right) \\ \vdots \\ f_M(x) = (1+g(x_M))\sin\left(\theta_1\frac{\pi}{2}\right) \end{cases}$ $g(x_M) = \sum_{x_i\in x_M}(x_i-0.5)^2$ $\theta_i = \frac{\pi}{4(1+g(x_M))}(1+2g(x_M)x_i)$, for$i = 2,3,\cdots,(M-1)$ $0 \le x \le 1$, for$i = 1,2,\cdots,n$	30	2, 3
DTLZ 6	$\begin{cases} f_1(x) = (1+g(x_M))\cos\left(\theta_1\frac{\pi}{2}\right)\cdots\cos\left(\theta_{M-2}\frac{\pi}{2}\right)\cos\left(\theta_{M-1}\frac{\pi}{2}\right) \\ f_2(x) = (1+g(x_M))\cos\left(\theta_1\frac{\pi}{2}\right)\cdots\cos\left(\theta_{M-2}\frac{\pi}{2}\right)\sin\left(\theta_{M-1}\frac{\pi}{2}\right) \\ f_3(x) = (1+g(x_M))\cos\left(\theta_1\frac{\pi}{2}\right)\cdots\sin\left(\theta_{M-2}\frac{\pi}{2}\right) \\ \vdots \\ f_M(x) = (1+g(x_M))\sin\left(\theta_1\frac{\pi}{2}\right) \end{cases}$ $g(x_M) = \sum_{x_i\in x_M}x_i^{0.1}$ $\theta_i = \frac{\pi}{4(1+g(x_M))}(1+2g(x_M)x_i)$, for$i = 2,3,\cdots,(M-1)$ $0 \le x \le 1$, for$i = 1,2,\cdots,n$	10	2, 3
DTLZ 7		30	2, 3

(*continued on next page*)

Table 10 (*continued*)

Name	Equation	*D*	Number of objectives
	$\begin{cases} f_1(x_1) = x_1 \\ f_2(x_2) = x_2 \\ \vdots \\ f_{M-1}(x_{M-1}) = x_{M-1} \\ f_M(x) = (1 + g(x_M))h(f_1, f_2, \cdots, f_{M-1}, g) \end{cases}$ $g(x_M) = 1 + \frac{9}{\lvert x_m \rvert} \sum_{x_i \in x_M} x_i$ $h(f_1, f_2, \cdots, f_{M-1}, g) = M - \sum_{i=1}^{M-1} \left[\frac{f_i}{1+g} (1 + \sin(3\pi f_i)) \right]$ $0 \le x \le 1$, for $i = 1, 2, \cdots, n$		

Table 11
Results of IGD scores for CEC 2009 multi-objective benchmark functions.

		MOGEO	MOGWO	NSGA-II	MOPSO	MOSSA
UF1	Mean	**0.004**	0.0057	0.0066	0.0052	0.0058
	Std	0.0004	0.0005	0.0018	0.0007	**0.0003**
UF2	Mean	**0.0024**	0.0036	0.0039	0.0032	0.0036
	Std	0.0004	0.0005	0.0007	**0.0002**	0.0005
UF3	Mean	**0.009**	0.0171	0.0148	0.0204	0.0134
	Std	0.0013	0.003	0.0016	**0.0003**	0.0045
UF4	Mean	**0.0024**	0.0049	0.0063	0.006	0.0047
	Std	**0.0001**	0.0004	0.0004	0.0002	0.0005
UF5	Mean	0.1915	0.1994	**0.1462**	0.1944	0.1558
	Std	0.061	0.0868	0.0628	0.1051	**0.0284**
UF6	Mean	0.0173	0.021	0.0219	0.0248	**0.0104**
	Std	0.0056	0.0074	0.006	0.007	**0.0028**
UF7	Mean	**0.0021**	0.0064	0.0169	0.0112	0.0042
	Std	**0.0001**	0.0065	0.0079	0.0072	0.0003
UF8	Mean	**0.0121**	0.052	0.0169	0.0142	0.0207
	Std	0.0027	0.0188	0.0012	**0.0005**	0.0032
UF9	Mean	**0.0137**	0.0176	0.0221	0.0176	0.0291
	Std	0.0035	0.0019	0.005	**0.0016**	0.0087
UF10	Mean	**0.0219**	0.0918	0.0828	0.0486	0.0585
	Std	**0.0067**	0.1208	0.0365	0.0166	0.0304

presented in Fig. 24, and the mathematical programming formulation is shown in Eq. (16).

$$\text{Consider } \vec{x} = [x_1, x_2, x_3, x_4, x_5] = [h_1, h_2, h_3, h_4, h_5]$$

$$\text{Minimize } f(\vec{x}) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5)$$

$$\text{Subject to :}$$

$$g_1(\vec{x}) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \le 0 \tag{16}$$

Where

$$0 \le x_i \le 100$$

Table 15 displays the best results obtained from GEO and other competing algorithms. This table shows that GEO outperforms GWO, GA, PSO, and HS while providing competitive results compared to CSA and DA in terms of optimal objective value and constraint violation.

Table 16 shows that the null hypothesis of the Kruskal-Wallis test is rejected for the cantilever beam design problem. Fig. 25 displays the results of the multiple comparisons test and reveals that GEO has statistically similar performance to GA in this problem.

5.4. Tension/compression spring design

This problem considers minimizing the total weight of a tension/compression spring, considering diameter (d), mean coil diameter (D), and the number of active coils (P) as the three design variables. The structure of the tension/compression spring is shown in Fig. 26, and the mathematical formulation of this problem is presented in Eq. (17).

$$\text{Consider } \vec{x} = [x_1 x_2, x_3] = [d, D, P]$$

$$\text{Minimize } f(\vec{x}) = (x_3 + 2)x_2 x_1^2$$

$$\text{Subject to :}$$

$$g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \le 0$$

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1 x_2}{12566(x_1^3 - x_1^4)} - \frac{1}{5108 x_1^2} \le 0 \tag{17}$$

$$g_3(\vec{x}) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0$$

$$g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} \le 0$$

Where

$$0.05 \le x_1 \le 2, \quad 0.25 \le x_2 \le 1.3, \quad 2 \le x_3 \le 15$$

Table 17 tabulates the obtained values for design variables (x_j), constraint violations (g_i), and the objective function for GEO and other algorithms. It is evident in this table that the proposed GEO outperforms GWO, PSO, HS, and DA while providing competitive results in comparison to GA, and CSA.

Table 18 shows that the null hypothesis of the Kruskal-Wallis test is rejected for the tension/compression spring design problem. Fig. 27 displays the results of the multiple comparisons test and shows that no other algorithm perform statistically similar to GEO.

5.5. Welded beam design

The objective of this problem is to find optimal values for the thickness of weld (h), length (l), height (t), and thickness of the bar (b) that minimizes the total cost of manufacturing a welded beam. The structure of the considered design is presented in Fig. 28, and the corresponding mathematical formulation is shown in Eq. (18).

Consider $\vec{x} = [x_1, x_2, x_3 x_4] = [h, l, t, b]$

Minimize $f(\vec{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2)$

Subject to :

$$
\begin{aligned}
&g_1(\vec{x}) = \tau(\vec{x}) - \tau_{\max} \le 0 \\
&g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{\max} \le 0 \\
&g_3(\vec{x}) = \delta(\vec{x}) - \delta_{\max} \le 0 \\
&g_4(\vec{x}) = x_1 - x_4 \le 0 \\
&g_5(\vec{x}) = P - P(\vec{x}) \le 0 \\
&g_6(\vec{x}) = 0.125 - x_1 \le 0 \\
&g_7(\vec{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2) - 5 \le 0
\end{aligned}
\tag{18}
$$

Where

$$
\tau(\vec{x}) = \sqrt{(\tau')^2 + (\tau'')^2 + \frac{l\tau'\tau''}{\sqrt{0.25\left(l^2 + (h+t)^2\right)}}}, \quad \tau' = \frac{6000}{\sqrt{2}hl}, \quad \tau'' = \frac{6000(14 + 0.5l)\sqrt{0.25\left(l^2 + (h+t)^2\right)}}{2\left[0.707hl\left(\frac{l^2}{12} + 0.25(h+t)^2\right)\right]}
$$

$$
\sigma(\vec{x}) = \frac{504000}{t^2 b}, \quad \delta(\vec{x}) = \frac{65856000}{(30 \times 10^6) b t^3}
$$

$$
0.1 \le x_1, \ x_4 \le 2, \ 0.1 \le x_2, \ x_3 \le 10
$$

Table 19 tabulates the results obtained by solving this problem using GEO and other competing algorithms. This table shows that GEO outperforms GWO, GA, PSO, HS, and DA, and provides competitive results compared to CSA. This confirms the ability of the proposed GEO to solve problems with multiple nonlinear constraints efficiently.

Table 20 shows that the null hypothesis of the Kruskal-Wallis test is rejected for the welded beam design problem. Fig. 29 displays the results of the multiple comparisons test and shows that GEO performs statistically similar to GA and CSA.

6. Conclusion

This work proposed a new swarm-intelligence metaheuristic algorithm for solving optimization problems, called Golden Eagle Optimizer (GEO). The algorithm starts off with an initial population and mimics the hunting procedure of golden eagles to improve the fitness of the population and find the optimum. Particularly, GEO is based on the fact that golden eagles' behavior in any instance during the hunting flight is influenced by the propensity to attack and propensity to cruise. Golden eagles memorize the best preys they have visited and sometimes communicate prey's location with other eagles. The mathematical equations proposed for GEO simulate attack and cruise vectors to address exploitation and exploitation for solving optimization problems. Besides, the multi-objective version of the algorithm, called Multi-Objective Golden Eagle Optimizer (MOGEO), was proposed based on the main concepts of GEO with some modifications. The modification was implemented on prey selection, best solution preservation mechanism (external archive), and archive handling. MATLAB toolboxes and the source code are developed for GEO and MOGEO and publicly available.

To certify the performance and efficiency of the proposed algorithms, GEO was tested on 33 benchmark problems from different classes, including unimodal, multimodal, and composite benchmark functions. The CEC2017 test suite was utilized for composite benchmark functions. Results were compared to that of six other well-known metaheuristic algorithms via different statistical measures. It was revealed that GEO is capable of exploring the landscape through intense and abrupt movements in the initial stages of the search and converge toward the promising areas by exploiting the best solutions found over the course of iterations. GEO outperformed other algorithms in the majority of the benchmark problems while providing competitive results in the others. GEO was also used to solve real-world engineering problems, where it showed promising performance. The results indicate that GEO is able to find the global optimum of optimization problems with challenging and unknown search spaces.

MOGEO's performance was tested using the CEC2009 and DTLZ test suite, which are specially designed for testing multi-objective algorithms. The results of MOGEO was compared to that of four other well-known multi-objective algorithms. MOGEO was able to provide competitive results, and in many cases, outperform the other algorithms in approximating the true Pareto front in challenging multi-objective problems.

It worths noting that the proposed GEO and MOGEO algorithms treat single- and multi-objective problems as a black box; therefore, they can be applied to any type of optimization problems, including NP-hard ones, as long as the problem is properly formulated. In addition, since

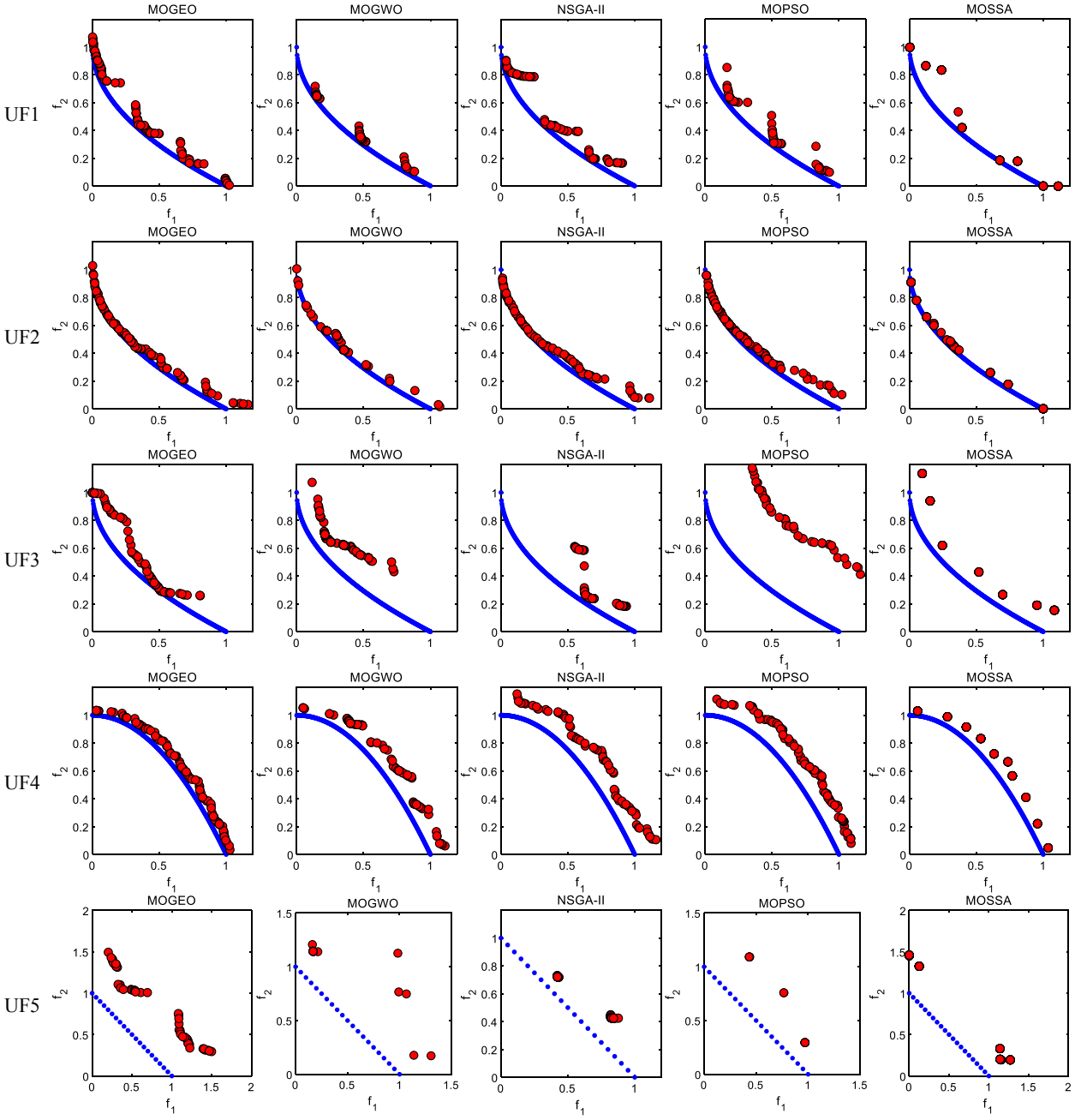

Fig. 19. Best Pareto fronts achieved by multi-objective solvers for the CEC 2009 test suite.

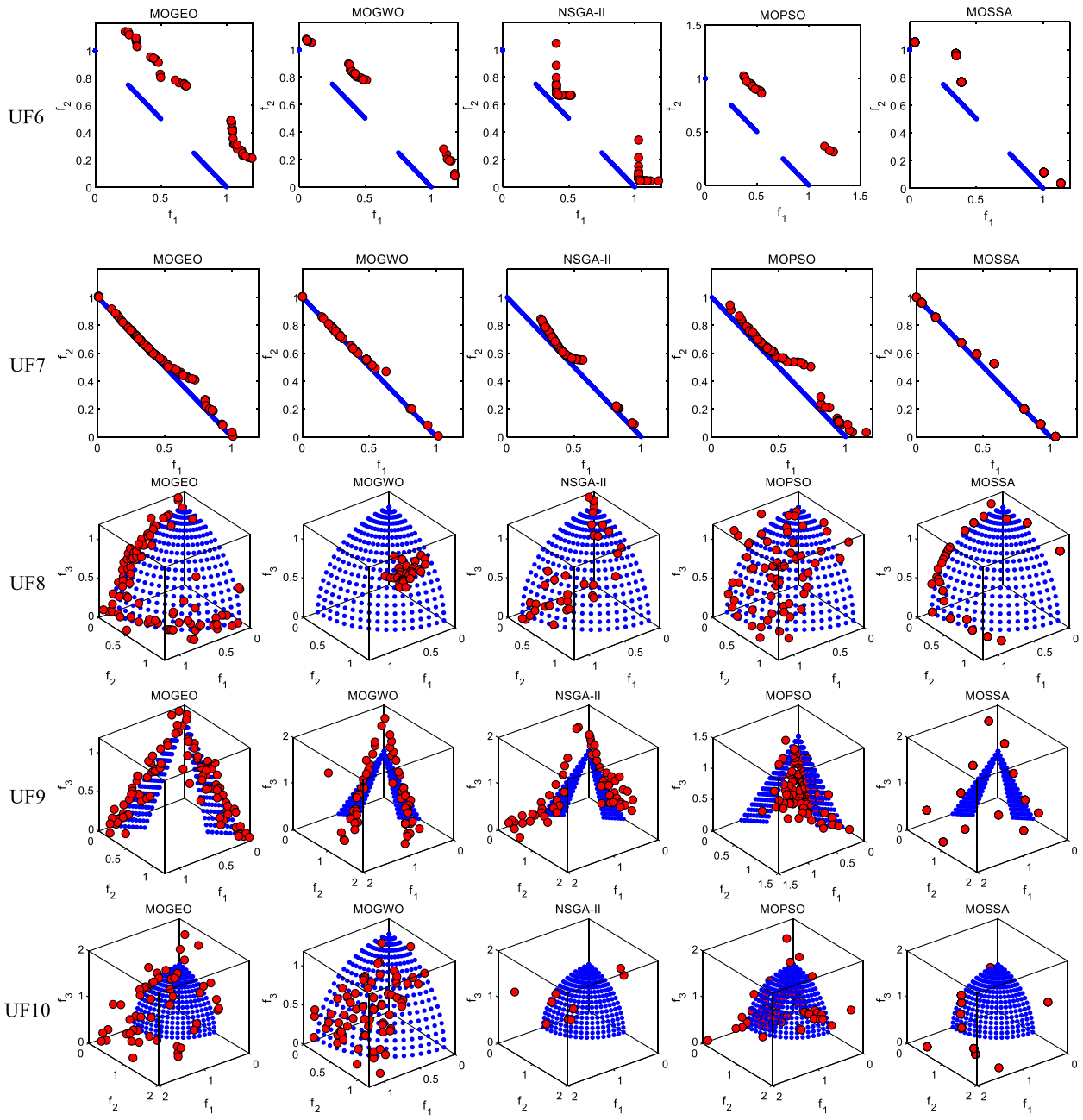

Fig. 19. (*continued*).

Table 12
Results of IGD scores for DTLZ multi-objective benchmark functions.

		2 objectives					3 objectives				
		MOGEO	MOGWO	NSGA-II	MOPSO	MOSSA	MOGEO	MOGWO	NSGA-II	MOPSO	MOSSA
DTLZ 1	Mean	21.485	20.203	**5.1113**	8.0282	19.033	10.32	19.551	11.144	**7.8028**	22.037
	Std	4.9369	4.09	**1.192**	3.9383	3.9375	**2.9302**	4.2953	5.644	3.0162	9.8469
DTLZ 2	Mean	6.9452	6.9041	**6.3703**	7.0545	6.714	8.8323	8.7484	7.9095	8.0977	**7.8462**
	Std	**0.0517**	0.0877	0.3375	0.069	0.0829	0.1367	0.6491	0.5309	**0.0633**	0.1516
DTLZ 3	Mean	18.697	19.729	**7.7265**	11.992	14.603	17.828	29.794	18.911	**12.314**	27.624
	Std	6.3404	5.8729	**1.8664**	4.5917	4.3449	**3.7691**	4.8473	10.023	4.7149	8.7055
DTLZ 4	Mean	7.3187	7.4069	**5.582**	7.0472	7.4759	10.862	12.235	10.419	11.81	**9.6378**
	Std	**0.1105**	0.6238	2.5732	0.8244	0.639	0.2894	0.3679	2.164	0.3363	**0.1586**
DTLZ 5	Mean	6.8398	7.5905	7.6684	7.0497	**6.4997**	6.9365	9.1043	8.1875	7.2588	**6.494**
	Std	**0.0449**	0.1959	1.9841	0.1487	0.0781	0.0877	0.436	2.2108	0.2541	**0.0643**
DTLZ 6	Mean	**7.238**	8.9986	18.839	13.906	8.2068	**7.5248**	17.023	21.946	27.102	9.8256
	Std	**0.1464**	0.7214	2.2866	1.0908	0.3778	0.3463	1.1571	1.3502	**0.0761**	0.8706
DTLZ 7	Mean	8.6588	10.611	15.664	8.6916	**8.5328**	**10.767**	14.47	24.438	12.288	12.412
	Std	**0.0824**	1.197	1.7947	1.1061	0.9813	0.2797	6.0019	2.4592	**0.2514**	0.8748

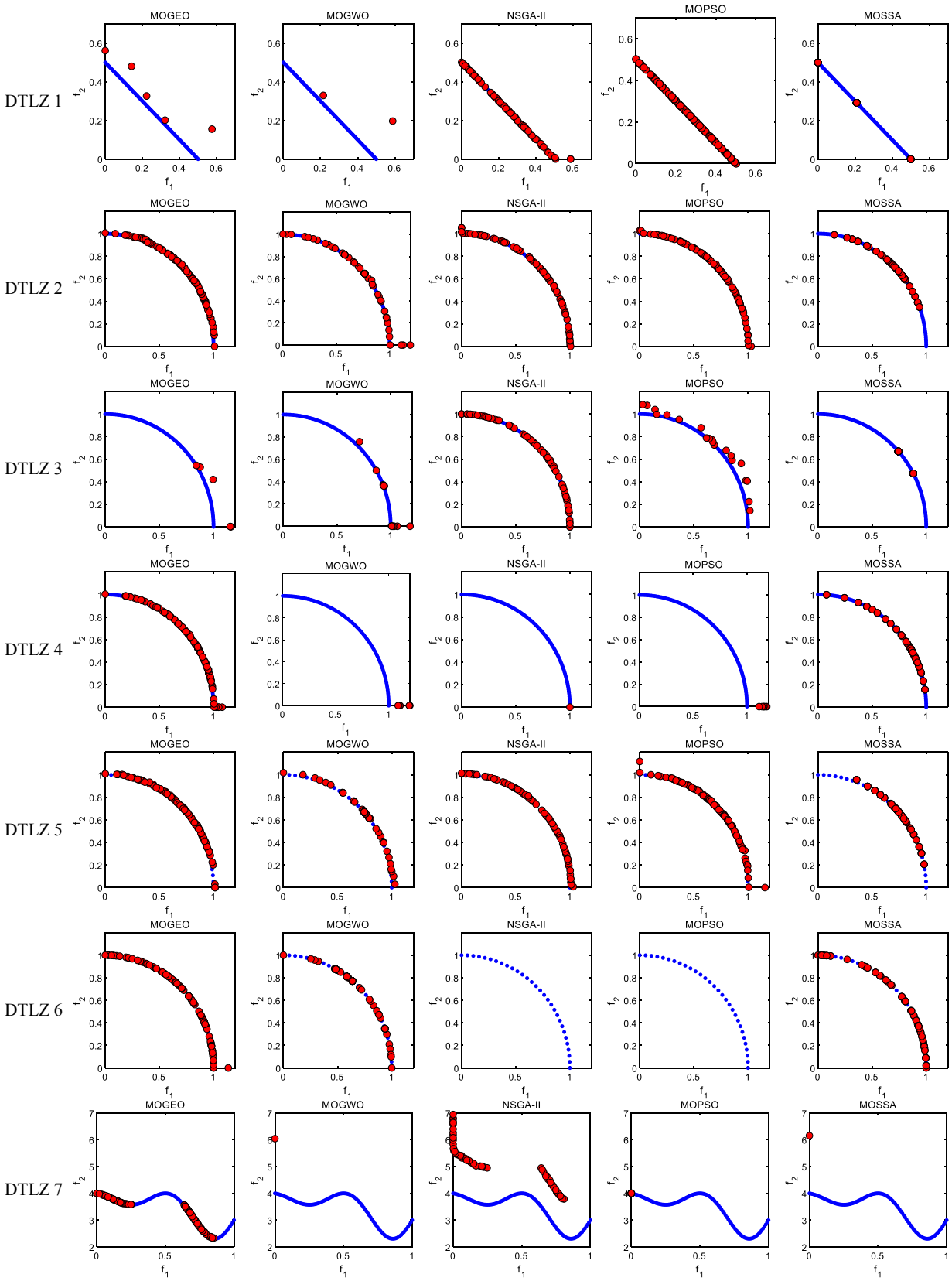

Fig. 20. Best Pareto fronts achieved by multi-objective solvers for the DTLZ test suite with two objectives.

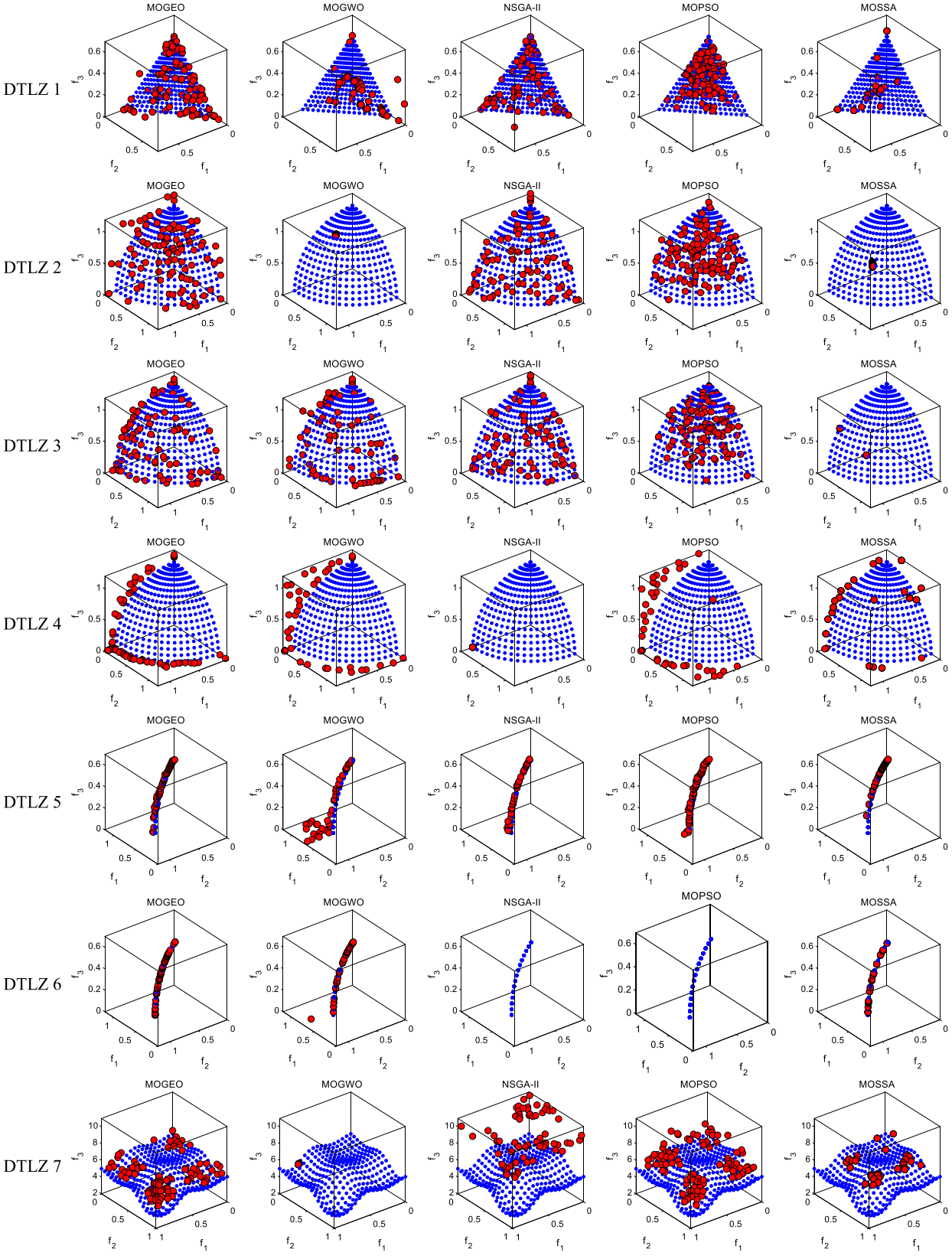

Fig. 21. Best Pareto fronts achieved by multi-objective solvers for the DTLZ test suite with three objectives.

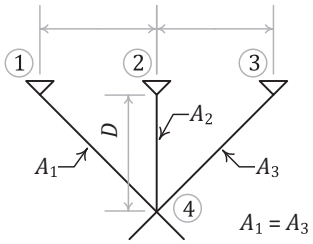

Fig. 22. The three-bar truss design problem.

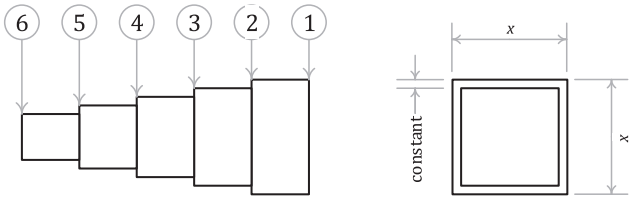

Fig. 24. Cantilever beam design problem.

Table 13
Best results obtained from algorithms for the three-bar truss design problem.

	GEO	GWO	GA	CSA	PSO	HS	DA
x_1	0.7886711	0.7887804	0.7886422	0.7886751	0.7882546	0.7895572	0.7883714
x_2	0.4082597	0.4079592	0.4083416	0.4082483	0.4094389	0.4060659	0.409108
g_1	−3.46E − 10	−6.54E − 06	−2.18E − 08	6.75E − 14	−1.34E − 08	−2.33E − 04	7.33E − 14
g_2	−1.46E + 00	−1.46E + 00	−1.46E + 00	−1.46E + 00	−1.46E + 00	−1.47E + 00	−1.46E + 00
g_3	−5.36E − 01	−5.36E − 01	−5.36E − 01	−5.36E − 01	−5.37E − 01	−5.34E − 01	−5.37E − 01
f(Weight)	263.89584	263.89671	263.89585	263.89584	263.89598	263.9271	263.89591

Table 14
Kruskal-Wallis table for the results of the three-bar truss design problem.

Source	SS	df	MS	χ^2	p-value
Groups	694319.9	6	115,720	188.0368	6.65E–38
Error	77406.13	203	381.311	–	–
Total	771,726	209	–	–	–

the proposed algorithms are able to solve optimization problems with continuous variables, some modifications may be needed for applying the GEO and MOGEO on problems with non-continuous decision space. There are opens avenues for future researches to proposed suitable operators to enhance the performance of the proposed algorithms on different types of problems. It is also perceived from the experiments that the introduction of the cruise vector provides good exploration in comparison to Exploitation capabilities in GEO and MOGEO. This

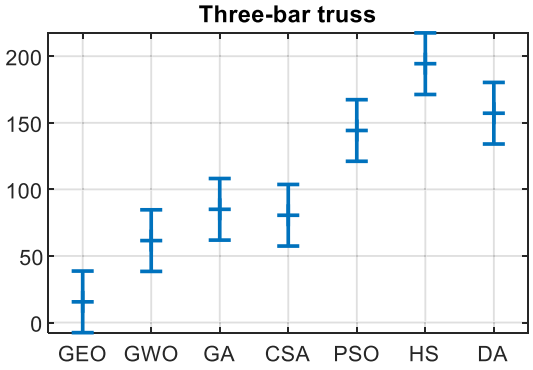

Fig. 23. Confidence intervals (95%) of the Tukey-Kramer multiple comparisons (Post hoc) test for the results of the three-bar truss design problem.

Table 15
Best results obtained from algorithms for the cantilever beam design problem.

	GEO	GWO	GA	CSA	PSO	HS	DA
$x_1(h_1)$	6.0156663	6.0109041	6.0439109	6.016015	5.9776207	5.4129259	6.0643788
$x_2(h_2)$	5.30926	5.3127046	5.298085	5.3090164	5.3779792	5.4129259	5.111031
$x_3(h_3)$	4.4944048	4.491602	4.4836003	4.4939648	4.4484496	5.4129259	4.7138404
$x_4(h_4)$	3.5016424	3.4951881	3.4868247	3.5020552	3.5336466	3.6742979	3.4824003
$x_5(h_5)$	2.1526862	2.1635477	2.161796	2.1526086	2.1450825	2.2792842	2.1387489
g_1	−1.64E − 09	−1.93E − 05	−3.23E − 07	−6.94E − 09	−7.39E − 04	−3.67E − 02	2.00E − 15
f(Weight)	13.365206	13.365384	13.365553	13.365206	13.370881	13.812525	13.388073

Table 16
Kruskal-Wallis table for the results of the cantilever beam design problem.

Source	SS	df	MS	χ^2	p-value
Groups	713751.2	6	118958.5	193.3006	5.05E–39
Error	57969.3	203	285.5631	–	–
Total	771720.5	209	–	–	–

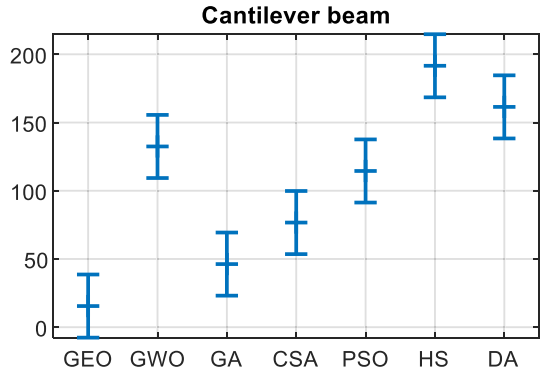

Fig. 25. Confidence intervals (95%) of the Tukey-Kramer multiple comparisons (Post hoc) test for the results of the cantilever beam design problem.

Fig. 26. Tension/compression spring design problem.

enables these algorithms to perform better on problems with unknown or more complex landscapes than unimodal functions. Future studies are encouraged to expand the concept of exploitation of GEO in unimodal functions. Therefore, the area for improvement of this algorithm is to modify the exploitation aspects of GEO.

Future works can also develop new mechanisms for the algorithm or enhance the existing ones for performance improvement. New prey selection mechanisms can be proposed to enhance the performance of the existing approach for both GEO and MOGEO based on, for example, statistical probability functions. For randomizing the attack, cruise, and the step vector, a uniform distribution is used in this work, which can be extended to other approaches for randomization, e.g., Lévy flights.

Table 18
Kruskal-Wallis table for the results of the tension/compression spring design problem.

Source	SS	df	MS	χ^2	p-value
Groups	630327.2	6	105054.5	170.708	3.18E–34
Error	141390.3	203	696.5039	–	–
Total	771717.5	209	–	–	–

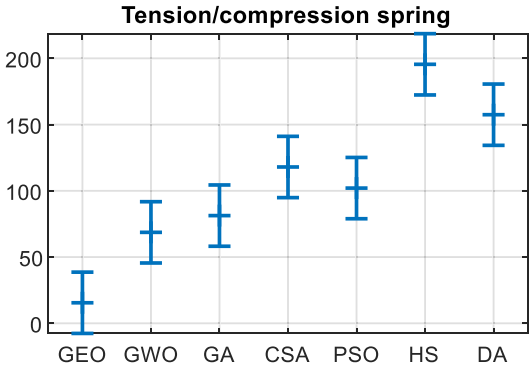

Fig. 27. Confidence intervals (95%) of the Tukey-Kramer multiple comparisons (Post hoc) test for the results of the cantilever beam design problem.

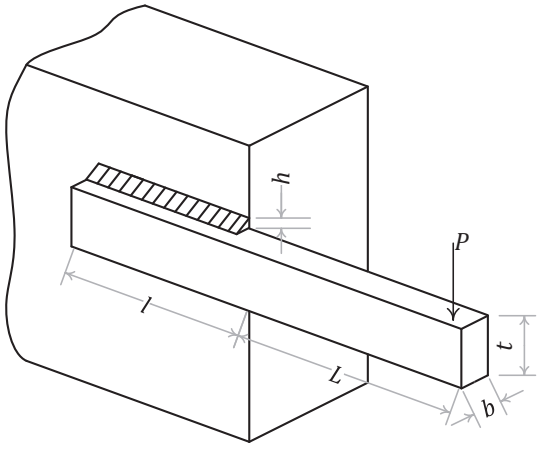

Fig. 28. The welded beam design problem.

Table 17
Best results obtained from algorithms for tension/compression spring design problem.

	GEO	GWO	GA	CSA	PSO	HS	DA
$x_1(d)$	0.0518499	0.0513858	0.0516977	0.0516892	0.050814	0.05	0.0516531
$x_2(D)$	0.3605987	0.3493298	0.3569189	0.3567214	0.3359981	0.3106913	0.3558539
$x_3(P)$	11.065069	11.743531	11.277477	11.288753	12.622433	15	11.347604
g_1	−2.99E − 06	−2.26E − 04	−6.76E − 07	−4.74E − 10	−4.29E − 04	−2.69E − 03	−6.89E − 04
g_2	−1.36E − 06	−3.14E − 04	−1.78E − 05	−8.42E − 11	−7.27E − 05	−1.67E − 02	−8.95E − 10
g_3	−4.06E + 00	−4.04E + 00	−4.05E + 00	−4.05E + 00	−4.01E + 00	−3.85E + 00	−4.05E + 00
g_4	−7.25E − 01	−7.33E − 01	−7.28E − 01	−7.28E − 01	−7.42E − 01	−7.60E − 01	−7.28E − 01
f(Weight)	0.0126658	0.0126771	0.0126657	0.0126652	0.012686	0.0132044	0.0126727

Table 19
Best results obtained from algorithms for the welded beam design problem.

	GEO	GWO	GA	CSA	PSO	HS	DA
$x_1(h)$	0.2443688	0.2443158	0.2443271	0.2443689	0.243871	0.1585629	0.2411605
$x_2(l)$	3.0630204	3.0652528	3.0635288	3.0630243	3.0648441	7.7555863	2.9106552
$x_3(t)$	8.2914827	8.2924051	8.2931239	8.2914718	8.305948	8.2746943	8.6619439
$x_4(b)$	0.2443689	0.2443838	0.2445089	0.244369	0.2447052	0.2547101	0.2419407
g_1	−6.09E−04	−6.08E+00	−2.44E+00	−4.62E−05	−5.19E+00	−2.81E+03	−2.00E−03
g_2	−7.24E−02	−8.57E+00	−2.91E+01	−3.39E−04	−1.46E+02	−1.10E+03	−2.24E+03
g_3	−2.34E−01	−2.34E−01	−2.34E−01	−2.34E−01	−2.34E−01	−2.35E−01	−2.36E−01
g_4	−8.66E−08	−6.79E−05	−1.82E−04	−1.05E−08	−8.34E−04	−9.61E−02	−7.80E−04
g_5	−1.64E−03	−1.56E+00	−1.11E+01	−3.20E−05	−3.21E+01	−7.85E+02	−9.09E−13
g_6	−1.19E−01	−1.19E−01	−1.19E−01	−1.19E−01	−1.19E−01	−3.36E−02	−1.16E−01
g_7	−3.27E+00	−3.27E+00	−3.27E+00	−3.27E+00	−3.27E+00	−2.77E+00	−3.23E+00
f(Cost)	1.8653598	1.8659235	1.8666563	1.8653589	1.8700306	2.4214036	1.891987

Table 20
Kruskal-Wallis table for the results of the welded beam design problem.

Source	SS	df	MS	χ^2	p-value
Groups	661,891	6	110315.2	179.2554	4.88E−36
Error	109830.5	203	541.0369	–	–
Total	771721.5	209	–	–	–

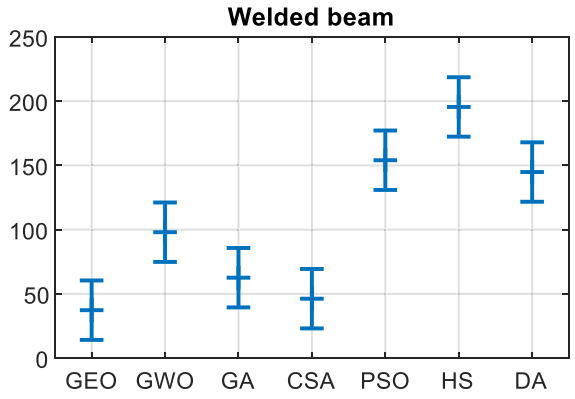

Fig. 29. Confidence intervals (95%) of the Tukey-Kramer multiple comparisons (Post hoc) test for the results of the welded beam design problem.

CRediT authorship contribution statement

Abdolkarim Mohammadi-Balani: Software, Writing - original draft, Visualization. **Mahmoud Dehghan Nayeri:** Conceptualization, Validation, Writing - review & editing, Supervision. **Adel Azar:** Conceptualization, Writing - review & editing. **Mohammadreza Taghizadeh-Yazdi:** Methodology, Validation.

Acknowledgements

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

References

Ahmadi, A., Tiruta-Barna, L., Capitanescu, F., Benetto, E., & Marvuglia, A. (2016). An archive-based multi-objective evolutionary algorithm with adaptive search space partitioning to deal with expensive optimization problems: Application to process eco-design. *Computers & Chemical Engineering., 87*, 95–110. https://doi.org/10.1016/j.compchemeng.2015.12.008

Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Computers & Structures., 169*, 1–12. https://doi.org/10.1016/j.compstruc.2016.03.001

Awad, N. H., Ali, M. Z., Suganthan, P. N., Liang, J. J., & Qu, B. Y. (2017) Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, https://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2017/CEC2017.htm (accessed April 13, 2020).

Bozorg-Haddad, O., Solgi, M., & Loaiciga, H. A. (2017). *Meta-heuristic and evolutionary algorithms for engineering optimization*. Hoboken, NJ: John Wiley & Sons.

Cai, L., Qu, S., & Cheng, G. (2018). Two-archive method for aggregation-based many-objective optimization. *Information Sciences., 422*, 305–317. https://doi.org/10.1016/j.ins.2017.08.078

Chen, L., Li, Q., Zhao, X., Fang, Z., Peng, F., & Wang, J. (2019). Multi-population coevolutionary dynamic multi-objective particle swarm optimization algorithm for power control based on improved crowding distance archive management in CRNs. *Computer Communications., 145*, 146–160. https://doi.org/10.1016/j.comcom.2019.06.009

Coello Coello, C. A., Lechuga, M. S. (2002) MOPSO: a proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), IEEE, Honolulu, HI, USA, 2002: pp. 1051–1056. https://doi.org/10.1109/CEC.2002.1004388.

Cui, Y., Geng, Z., Zhu, Q., & Han, Y. (2017). Review: Multi-objective optimization methods and application in energy saving. *Energy., 125*, 681–704. https://doi.org/10.1016/j.energy.2017.02.174

Das, S., & Suganthan, P. N. (2011). Differential Evolution: A Survey of the State-of-the-Art. *IEEE Transactions on Evolutionary Computation, 15*, 4–31. https://doi.org/10.1109/TEVC.2010.2059031

Davis, L. (1991). Bit-climbing, representational bias, and test suit design, Proc. Intl. Conf. Genetic Algorithm, 1991, 18–23.

Deb, K., Agrawal, S., Pratap, A., Meyarivan, T., & Fast, A. (2000). Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, & H.-. P. Schwefel (Eds.), *Parallel Problem Solving from Nature PPSN VI* (pp. 849–858). Berlin Heidelberg, Berlin, Heidelberg: Springer. https://doi.org/10.1007/3-540-45356-3_83.

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2005). Scalable Test Problems for Evolutionary Multiobjective Optimization. In A. Abraham, L. Jain, & R. Goldberg (Eds.), *Evolutionary Multiobjective Optimization* (pp. 105–145). London: Springer-Verlag. https://doi.org/10.1007/1-84628-137-7_6.

Dhiman, G., & Kumar, V. (2017). Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Advances in Engineering Software., 114*, 48–70. https://doi.org/10.1016/j.advengsoft.2017.05.014

Dhiman, G., & Kumar, V. (2018). Emperor penguin optimizer: A bio-inspired algorithm for engineering problems. *Knowledge-Based Systems., 159*, 20–50. https://doi.org/10.1016/j.knosys.2018.06.001

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

Eagle (heraldry), Wikipedia. (2020). https://en.wikipedia.org/w/index.php?title=Eagle_(heraldry)&oldid=943863753 (accessed April 12, 2020).

Eskandar, H., Sadollah, A., Bahreininejad, A., & Hamdi, M. (2012). Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures., 110–111*, 151–166. https://doi.org/10.1016/j.compstruc.2012.07.010

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

Fausto, F., Cuevas, E., Valdivia, A., & González, A. (2017). A global optimization algorithm inspired in the behavior of selfish herds. *Biosystems., 160*, 39–55. https://doi.org/10.1016/j.biosystems.2017.07.010

Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation, 76*, 60–68. https://doi.org/10.1177/003754970107600201

Glover, F. (1989). Tabu Search—Part I. *ORSA Journal on Computing., 1*, 190–206. https://doi.org/10.1287/ijoc.1.3.190

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning, 3*, 95–99. https://doi.org/10.1023/A:1022602019183

Golden eagle, Wikipedia. (2020). https://en.wikipedia.org/w/index.php?title=Golden_eagle&oldid=943393767 (accessed March 2, 2020).
Golden eagles in human culture, Wikipedia. (2020). https://en.wikipedia.org/w/index.php?title=Golden_eagles_in_human_culture&oldid=942701659 (accessed April 12, 2020).
Hashim, F. A., Houssein, E. H., Mabrouk, M. S., Al-Atabany, W., & Mirjalili, S. (2019). Henry gas solubility optimization: A novel physics-based algorithm. *Future Generation Computer Systems., 101*, 646–667. https://doi.org/10.1016/j.future.2019.07.015
Hayyolalam, V., & Pourhaji Kazem, A. A. (2020). Black Widow Optimization Algorithm: A novel meta-heuristic approach for solving engineering optimization problems. *Engineering Applications of Artificial Intelligence., 87*, Article 103249. https://doi.org/10.1016/j.engappai.2019.103249
Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems., 97*, 849–872. https://doi.org/10.1016/j.future.2019.02.028
Hunting with eagles, Wikipedia. (2020). https://en.wikipedia.org/w/index.php?title=Hunting_with_eagles&oldid=940982958 (accessed April 12, 2020).
Husseinzadeh Kashan, A. (2015). A new metaheuristic for optimization: Optics inspired optimization (OIO). *Computers & Operations Research., 55*, 99–125. https://doi.org/10.1016/j.cor.2014.10.011
Husseinzadeh Kashan, A., Tavakkoli-Moghaddam, R., & Gen, M. (2019). Find-Fix-Finish-Exploit-Analyze (F3EA) meta-heuristic algorithm: An effective algorithm with new evolutionary operators for global optimization. *Computers & Industrial Engineering, 128*, 192–218. https://doi.org/10.1016/j.cie.2018.12.033
Jahani, E., & Chizari, M. (2018). Tackling global optimization problems with a novel algorithm – Mouth Brooding Fish algorithm. *Applied Soft Computing., 62*, 987–1002. https://doi.org/10.1016/j.asoc.2017.09.035
Jain, M., Singh, V., & Rani, A. (2019). A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm and Evolutionary Computation., 44*, 148–175. https://doi.org/10.1016/j.swevo.2018.02.013
Kaveh, A., & Dadras, A. (2017). A novel meta-heuristic optimization algorithm: Thermal exchange optimization. *Advances in Engineering Software., 110*, 69–84. https://doi.org/10.1016/j.advengsoft.2017.03.014
Kaveh, A., & Mahdavi, V. R. (2014). Colliding bodies optimization: A novel meta-heuristic method. *Computers & Structures., 139*, 18–27. https://doi.org/10.1016/j.compstruc.2014.04.005
Kennedy, J., Eberhart, R. (1948) Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, IEEE, Perth, WA, Australia, 1995: pp. 1942–1948. https://doi.org/10.1109/ICNN.1995.488968.
Khoroshiltseva, M., Slanzi, D., & Poli, I. (2016). A Pareto-based multi-objective optimization algorithm to design energy-efficient shading devices. *Applied Energy., 184*, 1400–1410. https://doi.org/10.1016/j.apenergy.2016.05.015
Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science, 220*, 671–680. https://doi.org/10.1126/science.220.4598.671
Martín, A., & Schütze, O. (2018). Pareto Tracer: A predictor–corrector method for multi-objective optimization problems. *Engineering Optimization., 50*, 516–536. https://doi.org/10.1080/0305215X.2017.1327579
Massan, S.-R., Wagan, A. I., & Shaikh, M. M. (2020). A new metaheuristic optimization algorithm inspired by human dynasties with an application to the wind turbine micrositing problem. *Applied Soft Computing., 90*, Article 106176. https://doi.org/10.1016/j.asoc.2020.106176
Mavrovouniotis, M., Li, C., & Yang, S. (2017). A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation., 33*, 1–17. https://doi.org/10.1016/j.swevo.2016.12.005
Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software, 83*, 80–98. https://doi.org/10.1016/j.advengsoft.2015.01.010
Mirjalili, S. (2016). Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing & Applications, 27*, 1053–1073. https://doi.org/10.1007/s00521-015-1920-1
Mirjalili, S. (2016). SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge-Based Systems., 96*, 120–133. https://doi.org/10.1016/j.knosys.2015.12.022
Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software., 114*, 163–191. https://doi.org/10.1016/j.advengsoft.2017.07.002
Mirjalili, S., Jangir, P., & Saremi, S. (2017). Multi-objective ant lion optimizer: A multi-objective optimization algorithm for solving engineering problems. *Applied Intelligence, 46*, 79–95. https://doi.org/10.1007/s10489-016-0825-8
Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-Verse Optimizer: A nature-inspired algorithm for global optimization. *Neural Computing & Applications, 27*, 495–513. https://doi.org/10.1007/s00521-015-1870-7
Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software., 69*, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
Mirjalili, S. Z., Mirjalili, S., Saremi, S., Faris, H., & Aljarah, I. (2018). Grasshopper optimization algorithm for multi-objective optimization problems. *Applied Intelligence, 48*, 805–820. https://doi.org/10.1007/s10489-017-1019-8
Mirjalili, S., Saremi, S., Mirjalili, S. M., & Coelho, L. dos S. (2016). Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. *Expert Systems with Applications., 47*, 106–119. https://doi.org/10.1016/j.eswa.2015.10.039
Nematollahi, A. F., Rahiminejad, A., & Vahidi, B. (2017). A novel physical based meta-heuristic optimization method known as Lightning Attachment Procedure Optimization. *Applied Soft Computing., 59*, 596–621. https://doi.org/10.1016/j.asoc.2017.06.033
Nematollahi, A. F., Rahiminejad, A., & Vahidi, B. (2020). A novel meta-heuristic optimization method based on golden ratio in nature. *Soft Computing, 24*, 1117–1151. https://doi.org/10.1007/s00500-019-03949-w
Piotrowski, A. P., Napiorkowski, M. J., Napiorkowski, J. J., & Rowinski, P. M. (2017). Swarm intelligence and evolutionary algorithms: Performance versus speed. *Information Sciences., 384*, 34–85. https://doi.org/10.1016/j.ins.2016.12.028
Qi, X., Zhu, Y., & Zhang, H. (2017). A new meta-heuristic butterfly-inspired algorithm. *Journal of Computational Science., 23*, 226–239. https://doi.org/10.1016/j.jocs.2017.06.003
Rahmanzadeh, S., & Pishvaee, M. S. (2019). Electron radar search algorithm: A novel developed meta-heuristic algorithm. *Soft Computing.* https://doi.org/10.1007/s00500-019-04410-8
Rakotonirainy, R. G., & van Vuuren, J. H. (2020). Improved metaheuristics for the two-dimensional strip packing problem. *Applied Soft Computing.* , Article 106268. https://doi.org/10.1016/j.asoc.2020.106268
Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2011). Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design., 43*, 303–315. https://doi.org/10.1016/j.cad.2010.12.015
Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A Gravitational Search Algorithm. *Information Sciences., 179*, 2232–2248. https://doi.org/10.1016/j.ins.2009.03.004
Salimi, H. (2015). Stochastic Fractal Search: A powerful metaheuristic algorithm. *Knowledge-Based Systems., 75*, 1–18. https://doi.org/10.1016/j.knosys.2014.07.025
Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper Optimisation Algorithm: Theory and application. *Advances in Engineering Software., 105*, 30–47. https://doi.org/10.1016/j.advengsoft.2017.01.004
Shadravan, S., Naji, H. R., & Bardsiri, V. K. (2019). The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. *Engineering Applications of Artificial Intelligence., 80*, 20–34. https://doi.org/10.1016/j.engappai.2019.01.001
Sierra, M. R., & Coello Coello, C. A. (2005). Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ∈-Dominance. In C. A. Coello Coello, A. Hernández Aguirre, & E. Zitzler (Eds.), *Evolutionary Multi-Criterion Optimization* (pp. 505–519). Berlin Heidelberg, Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-31880-4_35.
Tabari, A., & Ahmad, A. (2017). A new optimization method: Electro-Search algorithm. *Computers & Chemical Engineering., 103*, 1–11. https://doi.org/10.1016/j.compchemeng.2017.01.046
Tack, J. D., Noon, B. R., Bowen, Z. H., Fedy, B. C. (2020) Ecosystem processes, land cover, climate, and human settlement shape dynamic distributions for golden eagle across the western US. Animal Conservation 23 (2020) 72–82. https://doi.org/10.1111/acv.12511.
Tikkanen, H., Rytkönen, S., Karlin, O.-P., Ollila, T., Pakanen, V.-M., Tuohimaa, H., & Orell, M. (2018). Modelling golden eagle habitat selection and flight activity in their home ranges for safer wind farm planning. *Environmental Impact Assessment Review., 71*, 120–131. https://doi.org/10.1016/j.eiar.2018.04.006
Van Veldhuizen, D. A., & Lamont, G. B. (1998). Multiobjective evolutionary algorithm research: A history and analysis. *Citeseer*.
Veldman, R., 2018. Golden eagle, (2018). https://pixabay.com/photos/golden-eagle-bird-raptor-eagle-4780267/ (accessed March 4, 2020).
Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*, 67–82. https://doi.org/10.1109/4235.585893
Yang, X.-S. (2014). *Nature-inspired optimization algorithms* (First edition). Amsterdam; Boston: Elsevier.
Yang, X.-S., Karamanoglu, M., 2013. Swarm Intelligence and Bio-Inspired Computation. In: Swarm Intelligence and Bio-Inspired Computation, Elsevier, 2013: pp. 3–23. https://doi.org/10.1016/B978-0-12-405163-8.00001-6.
Yapici, H., & Cetinkaya, N. (2019). A new meta-heuristic optimizer: Pathfinder algorithm. *Applied Soft Computing., 78*, 545–568. https://doi.org/10.1016/j.asoc.2019.03.012
Zahedi, Z. M., Akbari, R., Shokouhifar, M., Safaei, F., & Jalali, A. (2016). Swarm intelligence based fuzzy routing protocol for clustered wireless sensor networks. *Expert Systems with Applications., 55*, 313–328. https://doi.org/10.1016/j.eswa.2016.02.016
Zhang, Q., Zhou, A., Zhao, S., Suganthan, P. N., Liu, W., & Tiwari, S. (2009) Multiobjective optimization Test Instances for the CEC 2009 Special Session and Competition, https://www3.ntu.edu.sg/home/epnsugan/index_files/CEC09-MOEA/CEC09-MOEA.htm (accessed April 13, 2020).
Zhang, Y., Gong, D., Sun, J., & Qu, B. (2018). A decomposition-based archiving approach for multi-objective evolutionary optimization. *Information Sciences., 430–431*, 397–413. https://doi.org/10.1016/j.ins.2017.11.052
Zhang, Y., & Jin, Z. (2020). Group teaching optimization algorithm: A novel metaheuristic method for solving global optimization problems. *Expert Systems with Applications., 148*, Article 113246. https://doi.org/10.1016/j.eswa.2020.113246
Zhang, J., Xiao, M., Gao, L., & Pan, Q. (2018). Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems. *Applied Mathematical Modelling., 63*, 464–490. https://doi.org/10.1016/j.apm.2018.06.036
Zhao, W., Wang, L., & Zhang, Z. (2019). A novel atom search optimization for dispersion coefficient estimation in groundwater. *Future Generation Computer Systems., 91*, 601–610. https://doi.org/10.1016/j.future.2018.05.037