Knowledge-Based Systems 261 (2023) 110206

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Growth Optimizer: A powerful metaheuristic algorithm for solving continuous and discrete global optimization problems

Qingke Zhang [a,*], Hao Gao [a], Zhi-Hui Zhan [b], Junqing Li [a], Huaxiang Zhang [a]

[a] School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China
[b] School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China

ARTICLE INFO

Article history:
Received 5 August 2022
Received in revised form 11 December 2022
Accepted 11 December 2022
Available online 17 December 2022

Keywords:
Metaheuristic algorithms
Growth optimizer
Global optimization
Multiple sequence alignment
Image segmentation

ABSTRACT

In this paper, a novel and powerful metaheuristic optimizer, named the growth optimizer (GO), is proposed. Its main design inspiration originates from the learning and reflection mechanisms of individuals in their growth processes in society. Learning is the process of individuals growing up by acquiring knowledge from the outside world. Reflection is the process of checking the individual's own deficiencies and adjusting the individual's learning strategies to help the individual's growth. This work simulates this growth behavior mathematically and benchmarks the proposed algorithm on a total of 30 international test functions of the 2017 IEEE Congress on Evolutionary Computation real-parameter boundary constraint benchmark (CEC 2017 test suite). A total of 50 state-of-the-art metaheuristic algorithms participated in the comparison process. The results of the convergence accuracy comparison and the two nonparametric statistics based on the Friedman test and the Wilcoxon signed-rank test showed that GO provides competitive results compared to the 50 state-of-the-art metaheuristic algorithms tested. In addition, to verify that GO has the ability to solve different real-world optimization problems, GO was applied to two different types of real-world optimization problems: the multiple sequence alignment (MSA) problem based on the hidden Markov model (HMM) and the multithresholding image segmentation problem based on Kapur's entropy method. GO provided more promising results than other metaheuristic techniques, especially in terms of solution quality and avoidance of local optima. The source code of the GO algorithm is publicly available at https://github.com/tsingke/Growth-Optimizer.

1. Introduction

Due to recent advances in computational science, the complexity of real-world problems to be solved is increasing, and these problems are often characterized by nonlinearity, a large scale, multimodality, and constraints [1]. Optimization algorithms are usually designed to solve real-world application problems, and they are classified into exact traditional algorithms and stochastic metaheuristics. Many nondeterministic polynomial-time hard (NP-hard) optimization problems are difficult to solve for exact algorithms, no matter how exponentially computational power grows. However, since the metaheuristic algorithms do not require gradient information, they are easy to implement and can bypass local optima [2]. This environment-adapted autonomous machine solves optimization problems by adjusting decision variables in an intelligent stochastic manner and minimizing or maximizing the objective function value of the problem [3,4]. As an alternative method to alleviate the dilemma of exact algorithms, it is commonly used in various fields due to its universal adaptability, flexibility, robustness, efficiency, and local optima avoidance [5].

Although metaheuristic algorithms use different operators and strategies, they must satisfy two basic preconditions when solving optimization problems: exploration and exploitation. Exploration is a global search capability that focuses on helping the algorithm discover promising search regions. Exploitation is a local search capability that enables the algorithm to search within a promising region to further improve the quality of the solution. If the two capabilities are effectively balanced, the quality of the best solution will be further improved [6,7].

In the literature, many classification standards of metaheuristic algorithms have appeared. Christian et al. gave their classification under different standards: single point search vs. population based, nature inspired vs. nonnature inspired, static search vs. dynamic search, single-neighborhood search vs. multi-neighborhood search, and memory usage vs. memory less methods [8].

Over time, it has become more common in the literature on metaheuristics to use division methods based on the source

* Corresponding author.
E-mail address: tsingke@sdnu.edu.cn (Q. Zhang).

https://doi.org/10.1016/j.knosys.2022.110206
0950-7051/

of inspiration. Akyol et al. classified them into nine categories based on their inspiration sources: biology-based, physics-based, social-based, mathematics-based, music-based, chemical-based, sports-based, swarm-based, and hybrid methods [9]. Alatas et al. added three new categories to this classification, which are plant-based, light-based, and water-based methods [10]. Ayyarao et al. classified metaheuristic algorithms into five types, which are evolutionary-based, swarm-based, human behavior-based, physics-based, and hybrid/advanced [11]. Dhiman and Jain et al. then simplified the metaheuristic algorithms into three types: evolutionary algorithms (EAs), swarm intelligence (SI), and physics-based algorithms (PbAs) [12,13].

Evolutionary algorithms: EAs were the first metaheuristic techniques to emerge. They were based on an in-depth observation of the potential relationship between biological evolution and optimization, linking natural and artificial evolution and innovatively metaphorically representing decision variables as genes in biology [14]. EAs mimic biological evolutionary mechanisms such as crossover, mutation, and selection, and they accomplish the evolutionary process by retaining highly adaptive individuals (solutions). In addition, some well-known EAs are shown in the first part of Table 1.

Swarm Intelligence: Unlike EAs, SI simulates the behavior of the division of labor and collaboration among organisms and drives the evolution of the whole population (the set of solutions) through the interaction between individuals. Common examples of SI are given in the second part of Table 1.

Physics-based Algorithms: PbAs are inspired by physical phenomena such as electromagnetic forces, gravity, and temperature variations. They focus more on theoretical and mathematical derivations. Since they may be based on theorems and formulas from certain fields, their convergence is easier to prove than those of other algorithms. Physics-based algorithms are listed in the third part of Table 1.

Although metaheuristic algorithms can be divided into different classes, they can be successfully applied to the same problem and do not change the algorithm structure [72]. In addition, they are simple, easy, versatile, and flexible and solve specific problems well without the need for problem gradients. Because their operators have a certain randomness, they can better explore the problem space and further find promising solutions. However, there are still some problems to be addressed, based on which we further summarize the motivation of the proposed work. They are divided into four separate points for them to be clearly displayed:

(1) Although existing metaheuristics have been shown to reliably find globally optimal solutions, this is limited to specific problems. As described by the theorem of no free lunch (NFL) [73], if an algorithm excels at a class of problems, its performance may be offset by other methods when the problem changes. Therefore, this theory keeps the openness of algorithm research and motivates novel algorithms such as growth optimizer (GO) proposed for the specific problem.
(2) Table 1 fully shows the development process of metaheuristic algorithms. It can be seen that most of the inspiration for these algorithms' design comes from nature. Compared to animals in nature, humans have superior logic and thinking skills, so optimization methods inspired by human creative behavior should be better than optimization methods inspired by the behavior of animals in nature [33]. Although human-based algorithms have started to gain attention as a new trend in the last decade, the number of publications in the literature is still relatively small. Thus, this further motivates the efforts of this study in this area.
(3) According to the first application in Table 1, along with the development of the timeline, the application scenarios where metaheuristic algorithms were first proposed have gradually switched from multi-scenario applications to function optimization and engineering design applications. In addition, based on the competitor number in Table 1, the number of competing subjects involved in the experiments of these algorithms is small, which does not provide a comprehensive view of the algorithm performance when the number of subjects increases. Therefore, it is more urgent to propose a new algorithm to consider more experimental subjects and to solve more challenging problems.
(4) Algorithms, including but not limited to the metaheuristic algorithms listed in Table 1, need to collect search direction information during evolution to determine their current search direction [74], but most of them are unable to provide individuals with extensive search direction information. They invariably approximate the search direction mainly by two or three vectors, but this approach has two main drawbacks. First, these vectors are not necessarily oriented in the correct direction. Second, these vectors do not provide enough extensive information about the correct search direction, which leads to entirely unreliable search directions. When the problem is complex and changing, the lack of directional information may upset the balance between exploration and exploitation, thus reducing the optimization performance of the algorithm. Therefore, this requires us to propose an algorithm to fuse more information to reduce the probability of stagnating the algorithm search due to the interference of wrong information.

As we can see, the above four reasons are the main drivers behind this study, and the GO algorithm in these studies expect to solve the more challenging problems and find better global solutions. To better demonstrate advantages and contributions of this research, we summarize them into the following three aspects: the algorithm aspect, experiment aspect, and application aspect.

From the algorithm aspect, this paper proposes a novel metaheuristic algorithm for global optimization, called the growth optimizer (GO). Through mathematical modeling of learning and reflective behavior, GO was divided into two phases: the learning phase and the reflection phase. The learning phase of GO is mainly based on the cooperative search operator. For each individual, the algorithm contributes four types of search direction information through five different specific individuals and self-adaptively balances these four types of direction information by combining the concepts of fitness value and Euclidean distance. The operator strongly reduces the chance of the algorithm falling into a local optimum due to the interference of incorrect direction information. The reflection phase of GO uses different computational methods for each dimension of the individual, thus improving the global convergence performance of GO.

From the experimental perspective, GO was evaluated using 30 benchmark test functions based on the IEEE Congress on Evolutionary Computation (IEEE CEC) real-parameter boundary constraint benchmark, CEC 2017 test suite [75]. Then, GO was compared with 50 metaheuristic algorithms from the literature. In the experimental results, GO provides more competitive performance than the other 50 state-of-the-art algorithms.

From the application aspect, GO was applied to a continuous optimization problem—multiple sequence alignment (MSA)—and a discrete optimization problem—multithresholding image segmentation—to demonstrate its validity and applicability. The dataset used in the MSA problem is from the National Center for Biotechnology Information (NCBI) database. The application results show that GO is able to train a high-quality hidden Markov model (HMM) to better align gene sequences even in the face of high-dimensional problems. In addition, the aligned

Table 1
A brief review of metaheuristic algorithms.

Classification	Algorithms	Inspiration	First application	Competitor number	Proposed time	Citations
Evolutionary algorithms	Evolutionary Programming (EP) [15]	The finite state machine	Function optimization	0	1966	5382
	Evolution Strategy (ES) [16]	The mechanisms of biological evolution	Engineering problem	0	1973	310
	Genetic Algorithm (GA) [17]	The Darwinian evolution	Function optimization	0	1975	17 613
	Memetic Algorithm (MA) [18]	The Darwinian principles of natural evolution and Dawkins' notion of a meme	Traveling salesman problem	1	1989	2587
	Genetic Programming (GP) [19]	The mechanisms of biological evolution	Robot control	0	1994	23 866
	Cultural Algorithm (CA) [20]	The evolution process of human culture	Cooperation problem	0	1994	1258
	Differential Evolution (DE) [21]	The mechanism of biological evolution	Function optimization	6	1997	29 477
	Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [22]	The covariance combined with the concept of biological evolution	Function optimization	0	2006	2050
	Differential Search (DS) [23]	The random motion of the organism	Function optimization and geographical coordinate conversion	17	2012	440
Swarm intelligence	Particle Swarm Optimization (PSO) [24]	The foraging behavior of birds	Function optimization and neural network training	0	1995	72 995
	Harmony Search (HS) [25]	The mechanism of improvisation performance	Function optimization, traveling salesman problem, and design of a pipeline network	6	2001	6517
	Ant Colony Optimization (ACO) [26]	The foraging behavior of ants	Asymmetric traveling salesman, quadratic assignment, and job-shop scheduling	2	2006	15 010
	Shuffled Frog-Leaping Algorithm (SFLA) [27]	The foraging behavior of frogs	Function optimization, groundwater model calibration, and water distribution system problem	1	2006	1253
	Artificial Bee Colony (ABC) [28]	The foraging behavior of honeybees	Function optimization	3	2007	7212
	Biogeography-Based Optimization (BBO) [29]	The migration mechanism of biological	Function optimization and aircraft engine health estimation	7	2008	3658
	Firefly Algorithm (FA) [30]	The behavior mechanism of fireflies	Function optimization	2	2009	4293
	Cuckoo Search (CS) [31]	The obligate brood parasitic behavior of the cuckoo	Function optimization	2	2009	6906
	Bacterial Foraging Optimization (BFO) [32]	The foraging behavior of bacteria	No	0	2010	259
	Brain Storm Optimization (BSO) [33]	The brainstorming process of humans	Function optimization	0	2011	591
	Teaching–Learning-Based Optimization (TLBO) [34]	The process of classroom teaching	Function optimization and mechanical design problems	6	2011	3423
	Flower Pollination Algorithm (FPA) [35]	The pollination process of flowers	Function optimization and engineering design problems	2	2012	2082
	Imperialist Competitive Algorithm (ICA) [36]	The mechanism of imperialist colonial competition	Function optimization	2	2014	2973
	Artificial Fish Swarm Algorithm (AFSA) [37]	The feeding, schooling, and tailing behavior of fish	Traveling salesman problem and PID controller optimization tuning	0	2014	366
	Grey Wolf Optimizer (GWO) [38]	The hierarchical mechanism and the hunting behavior of wolves	Function optimization and engineering design problems	18	2014	8819
	Moth-Flame Optimization (MFO) [39]	The navigation mechanism of moth	Function optimization and engineering design problems	26	2015	2514
	Whale Optimization Algorithm (WOA) [40]	The bubble-net hunting strategy of whales	Function optimization and engineering design problems	22	2016	5784
	Jaya [41]	No source of inspiration	No	11	2016	1490
	Salp Swarm Algorithm (SSA) [42]	The movement and foraging behavior of salp swarm	Function optimization and engineering design problems	28	2017	2413
	Coyote Optimization Algorithm (COA) [1]	The social behavior of coyotes	No	6	2018	295
	Harris Hawks Optimization (HHO) [43]	The cooperation and foraging behavior of Harris's hawks	Function optimization and engineering design problems	49	2019	1897
	Butterfly Optimization Algorithm (BOA) [44]	The foraging and mating behavior of butterflies	Function optimization and engineering design problems	25	2019	639
	Supply-Demand-Based Optimization (SDO) [45]	The mechanism of market supply and demand	Function optimization and engineering design problems	32	2019	66
	Manta Ray Foraging Optimization (MRFO) [46]	The foraging behavior of manta rays	Function optimization and engineering design problems	6	2020	304
	Gaining Sharing Knowledge (GSK) [47]	The process of gaining and sharing knowledge during the human life span	Function optimization and engineering design problems	10	2020	122
	Coronavirus Herd Immunity Optimizer (CHIO) [48]	The mechanism of herd immunity	Function optimization and engineering design problems	18	2020	64
	Heap-Based Optimizer (HBO) [49]	The company hierarchy	Function optimization and engineering design problems	22	2020	570
	Marine Predators Algorithm (MPA) [50]	The survival evolutionary mechanism of marine animals	Function optimization and engineering design problems	29	2020	479
	Aquila Optimizer (AO) [51]	The foraging behavior of Aquila	Function optimization and engineering design problems	48	2021	484
	White Shark Optimizer (WSO) [52]	The movement and foraging behavior of white sharks	Function optimization and engineering design problems	9	2022	8
	Honey Badger Algorithm (HBA) [53]	The foraging behavior of honey badgers	Function optimization and engineering design problems	10	2022	99
	Dwarf Mongoose Optimization Algorithm (DMOA) [54]	The foraging behavior of dwarf mongooses	Function optimization and engineering design problems	7	2022	63
	Snake Optimizer (SO) [55]	The foraging and reproductive behavior of snakes	Function optimization and engineering design problems	9	2022	16
	Prairie Dog Optimization Algorithm (PDO) [56]	The survival behavior of prairie dogs	Function optimization and engineering design problems	47	2022	2
	Reptile Search Algorithm (RSA) [57]	The foraging behavior of crocodiles	Function optimization and engineering design problems	50	2022	173
	Ebola Optimization Search Algorithm (EOSA) [58]	The mechanism of transmission of Ebola virus	Function optimization and medical image classification	10	2022	42
Physics-based algorithms	Simulated Annealing (SA) [59]	The principle of solid annealing	Function optimization and bates problem	0	1983	53 334
	Gravitational Search Algorithm (GSA) [60]	The law of gravity and Newton's second law	Function optimization	3	2009	6054
	Lightning Search Algorithm (LSA) [61]	The natural phenomenon of lightning	Function optimization and traveling salesman problem	5	2015	292
	Stochastic Fractal Search (SFS) [62]	The fractal concept	Function optimization and engineering design problems	21	2015	404
	Sine Cosine Algorithm (SCA) [63]	The sine and cosine functions	Function optimization and airfoil design	6	2016	2491
	Multi-Verse Optimizer (MVO) [64]	The cosmological concepts	Function optimization and engineering design problems	31	2016	1460
	Atom Search Optimization (ASO) [65]	The motion mechanism of atoms	Function optimization and hydrogeologic parameter estimation	5	2019	250
	Artificial Electric Field Algorithm (AEFA) [66]	The Coulomb's law of electrostatic force	Function optimization	10	2019	127
	Artificial Ecosystem-based Optimization (AEO) [67]	The production, consumption, and decomposition behaviors of organisms	Function optimization, engineering design and hydrogeologic parameter estimation	32	2019	142
	Equilibrium Optimizer (EO) [68]	The model of control volume mass balance	Function optimization and engineering design problems	31	2020	672
	Arithmetic Optimization Algorithm (AOA) [69]	The distribution behavior of the main arithmetic operators in mathematics	Function optimization and engineering design problems	34	2021	633
	Runge kutta optimizer (RUN) [70]	The Runge Kutta method	Function optimization and engineering design problems	13	2021	206
	weighted mean of vector (INFO) [71]	The weighted mean idea	Function optimization and engineering design problems	30	2022	48

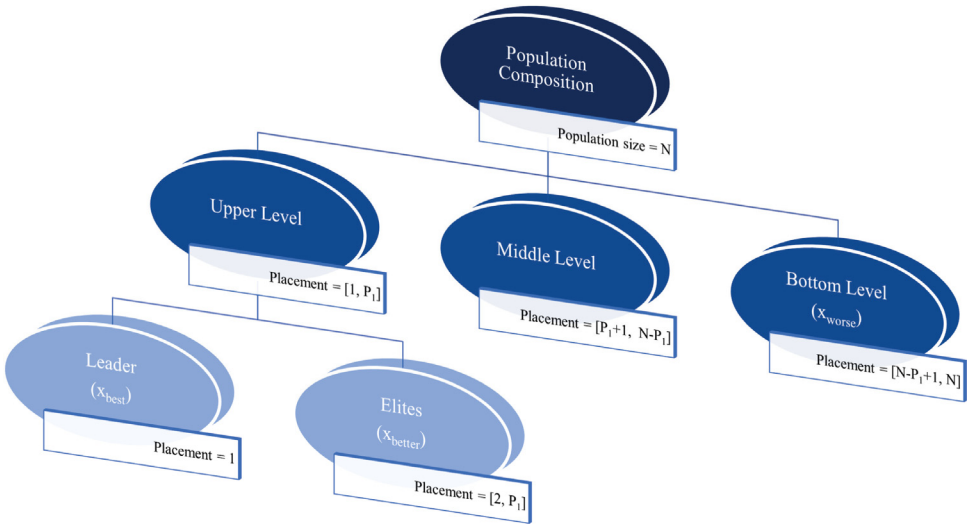


Fig. 1. The hierarchical distribution of individuals in the population.

sequences obtained by GO optimization confirmed that SARS-CoV-2, bat coronavirus, SARS, and other SARS-CoV-2 isolates have a certain homology. The dataset used in the multithreshold image segmentation problem is from the USC-SIPI image database. The application results show that the thresholds obtained from GO processing are sufficient for accurate image segmentation. This method also has potential application value in the fields of medical data analysis, machine vision, and pattern recognition. More importantly, GO can be flexibly applied to different problems and all the results obtained by GO are obtained without adjusting any parameters, which shows its robustness and stability.

The rest of the paper is organized as follows. Section 2 presents the design inspiration for GO, the mathematical model, and the difference with other similitude techniques. Section 3 describes the benchmark functions used and the setup of the test algorithm. Then, the comprehensive statistical experimental results and analysis of GO and 50 state-of-the-art metaheuristic algorithms are listed. Section 4 applies GO and other high-performing metaheuristic algorithms to MSA problem based on HMM. Section 5 applies GO and the same competing algorithms to the multithreshold image segmentation problem based on Kapur's entropy method. Finally, Section 6 contains conclusions and prospects for future research.

2. Growth optimizer

This section describes the inspiration for GO, the mathematical model, the structure of the algorithm, the flow of the algorithm, and how it differs from other humanoid metaheuristic technologies. Notably, the algorithm presented in this section is specific to the minimization problem.

2.1. Inspiration

Confucius, a great thinker and educator at the end of the Spring and Autumn period in China and the founder of the Confucian school, said, "Learning without thought is useless; thought without learning is perilous". From birth, a person must learn and benefit from the knowledge passed on to him or her to grow. When learning is not efficient, individuals usually stop thinking about the problem and correct it, and then they are able to promote their own growth. It is clear that learning and reflection are complementary and mutually reinforcing. This means that learning and reflection are key factors for an individual's growth.

In GO, the solution to a problem is called the individual. The components of the decision variables are the necessary elements that make up an individual, such as morality, beliefs, cultivation, emotions, perseverance, etc. A society or population containing a certain number of individuals is a set of decision variables, usually represented in the form of a matrix. For $i = 1, 2, 3, \ldots, N$, the ith individual is represented in the search space as $\vec{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D})$, where $x_{i,D}$ is the Dth element of the ith individual. In GO, the speed of individual growth is distinguished by growth resistance (GR). Generally, the objective function will receive the ith individual and return the corresponding output, which is the GR_i of the individual i. The less resistance an individual has to growing, the more knowledge they absorbs, and the more likely he or she is to become an elite member of society.

In some areas of society, there is a gap between individuals. In GO, the social structure of the individual is related to GR, and it is divided into three levels (Fig. 1) by P_1. The structure is composed of the upper level (including the leader and the elites), the middle level, and the bottom level. At the upper level, the leader ranked at the top is the global optimal solution, while the elite ranked in the range of 2 to P_1 represents the global suboptimal solution. Individuals in the middle stratum have a rank between P_1+1 and $N - P_1$. Members of the bottom stratum are between $N - P_1 + 1$ and N. Through extensive experiments, it has been shown that $P_1 = 5$ can guarantee the efficient and stable performance of the algorithm.

The GO algorithm divides society members' growth processes into two phases according to the stratified social structure. Fig. 1 shows that there are among three types of levels. Everyone has gaps in their accumulated knowledge, and it is because of these gaps that there is a learning process. First, an individual will see the gap between the elites and the leader. Second, an individual will also see the gap between the elites and leader and the lower classes. Finally, an individual sees the gap between two different individuals, and there is no hierarchy in this gap because different individuals have different strengths. When an individual fully understands these gaps and learns from them, it can help an individual grow faster and better. This is the learning phase of the process. Of course, after learning, there is a process of reflection, which is usually a process of giving up and gaining. When individuals realize that they may fall behind others or that there is a large gap between them, they urgently need to improve themselves by learning from the top-ranked individuals in society. When individuals believe that their cumulative knowledge

in a certain area may be too poor, they will readjust or change their current relatively poor state. Conversely, when individuals are more complete in all aspects of knowledge and perform better in groups, the process of reflection on a necessary element is temporarily ignored.

2.2. The GO process

Process is divided into two parts. The first part consists of the 'learning phase' and the second part consists of the 'reflection phase'. The 'learning phase' is when the individual learns from the gaps between other different individuals, and the 'reflection phase' is when the individual uses different strategies to check and make up for their own shortcomings.

2.2.1. Learning phase

Facing the gap between individuals, exploring the causes of these gaps, and learning from them can greatly promote an individual's growth. In the learning phase of GO, four common gaps are mathematically modeled: the gap between the leader and the elite ($\vec{Gap}_1$), the gap between the leader and the bottom ($\vec{Gap}_2$), the gap between the elite and the bottom ($\vec{Gap}_3$), and the gap between two random individuals ($\vec{Gap}_4$). For each set of gaps, the mathematical model is described by Eq. (1).

$$\begin{cases} \vec{Gap}_1 = \vec{x}_{best} - \vec{x}_{better} \\ \vec{Gap}_2 = \vec{x}_{best} - \vec{x}_{worse} \\ \vec{Gap}_3 = \vec{x}_{better} - \vec{x}_{worse} \\ \vec{Gap}_4 = \vec{x}_{L1} - \vec{x}_{L2} \end{cases} \tag{1}$$

where $\vec{x}_{best}$ represents the leader of the society, while $\vec{x}_{better}$ represents one of the next $P_1 - 1$ best individuals, who are referred to as an elite. In GO, the leader and the elites together make up the upper echelon of society. $\vec{x}_{worse}$ is one of the P_1 lowest ranked individuals in the population and is at the bottom level of the social hierarchy. Both $\vec{x}_{L1}$ and $\vec{x}_{L2}$ are random individuals different from the ith individual. $\vec{Gap}_k (k = 1, 2, 3, 4)$ is the gap between two individuals. $\vec{Gap}_k$ allows learners to fully understand the differences between two individuals to benefit from them. It is worth noting that within the current iteration (It) of the GO algorithm, GR should be sorted in ascending order to facilitate the update as well as the selection of these individuals.

To reflect this variability, learning factor (LF) is introduced for each of the four disparity measures. For the ith individual, LF_k will influence its learning effect on the kth group gap, and LF_k is modeled as follows (Eq. (2)).

$$LF_k = \frac{\|\vec{Gap}_k\|}{\sum_{k=1}^{4} \|\vec{Gap}_k\|}, \quad (k = 1, 2, 3, 4) \tag{2}$$

where LF_k denotes the normalized ratio of the Euclidean distance of the kth group gap ($\vec{Gap}_k$), and it ranges from [0,1]. When the gap of the kth group is larger, LF_k will also be larger, and the ith individual will learn more from the kth gap.

People at different levels in the growth process perceive themselves differently. The ith individual uses SF_i to assess the range of acceptable knowledge for himself. A larger SF_i means that individual i needs to learn more knowledge to improve himself. SF_i is modeled by Eq. (3).

$$SF_i = \frac{GR_i}{GR_{max}} \tag{3}$$

where GR_i is the growth resistance of the ith individual, and GR_{max} is the largest growth resistance of all. In general, a smaller GR_i means that the higher the level of the ith individual, the more he will extract the essence of knowledge and absorb it. Therefore, the individual should obtain a smaller SF_i, which is biased toward performing local exploitation patterns. When GR_i is larger, it means that the ith individual is not good enough and the individual should fill the gaps in knowledge. Thus, the individual should get a larger SF_i, and the individual is biased toward performing global search patterns.

The learning and transformation of knowledge are lossy. For the kth group of gaps ($\vec{Gap}_k$), the individual i absorbs some knowledge from them, and this knowledge is the kth group of knowledge acquisition ($\vec{KA}_k$). For the ith individual, $\vec{KA}_k$ is obtained after the operations of LF_k and SF_i on the kth group gap, a process that is described by Eq. (4).

$$\vec{KA}_k = SF_i \cdot LF_k \cdot \vec{Gap}_k, (k = 1, 2, 3, 4) \tag{4}$$

where $\vec{KA}_k$ is the knowledge acquired by the ith individual from the kth group of the gap. SF_i is an assessment of its own situation, while LF_k is an assessment of the external situation. The ith individual, under the influence of both assessments, identifies his own needed knowledge $\vec{KA}_k$ from $\vec{Gap}_k$ and thus completes the learning process.

By absorbing the knowledge gap between different individuals, the ith individual completes a rich knowledge accumulation process, and the specific learning process of the ith individual is given by Eq. (5).

$$\vec{x}_i^{It+1} = \vec{x}_i^{It} + \vec{KA}_1 + \vec{KA}_2 + \vec{KA}_3 + \vec{KA}_4 \tag{5}$$

where It is the number of current iterations, and $\vec{x}_i$ is the ith individual who absorbs the knowledge acquired during the learning phase to grow.

The quality of the candidate solution represented by each individual may progress or lag after the adjustment of the learning phase. Thus, it is necessary to verify whether it has truly made progress. If progress is made, GR_i of the ith individual will decrease and its rank will arise. If the ith individual regresses, there is a high probability that the ith individual will give up some acquired knowledge, but since the learning process consumes the time and effort of the ith individual, the acquired knowledge may be retained with a small probability. Here, P_2 is responsible for controlling this retention probability. This process is described by Eq. (6).

$$\vec{x}_i^{It+1} = \begin{cases} \vec{x}_i^{It+1} & if\ f\left(\vec{x}_i^{It+1}\right) < f\left(\vec{x}_i^{It}\right) \\ \begin{cases} \vec{x}_i^{It+1} & if\ r_1 < P_2 \\ \vec{x}_i^{It} & else \end{cases} & else \end{cases} \tag{6}$$

where r_1 is a uniformly distributed random number in the range [0,1], $ind(i)$ is the ranking of the ith individual according to GR ascending order, and P_2 determines whether the newly acquired knowledge is retained when the ith individual fails to update. Here, P_2 is 0.001. Due to space limitations of Eq. (6), the complete conditional judgment statement for retaining the newly acquired knowledge (when the individual update fails) is given here: $r_1 < P_2$ && $ind(i) \sim= ind(1)$. This not only means that when an individual update fails, the individual has a 0.001 probability of entering the next generation population, but also ensures that the current global optimal solution cannot be replaced; otherwise, the algorithm may not converge.

The learning operator is a cooperative search strategy that uses five vectors to contribute four pieces of information related to the convergence direction. By balancing this four-directional information through the distance and the fitness value between the vectors, the risk of the algorithm falling into a local optimum trap is greatly reduced.

2.2.2. Reflection phase

Learning and reflection complement each other. Therefore, an individual should not only learn but also learn to reflect. This means an individual should check and make up for deficiencies in all aspects and his knowledge should be retained. For their bad aspects, they should learn from outstanding individuals, while their good aspects should be retained. When the lesson of a certain aspect cannot be remedied, the previous knowledge should be abandoned, and systematic learning should be carried out again. The reflective process of GO is mathematically modeled through Eqs. (7) and (8).

$$x_{i,j}^{It+1} = \begin{cases} \begin{cases} lb + r_4 \times (ub - lb) & if\ r_3 < AF \\ x_{i,j}^{It} + r_5 \times \left(R_j - x_{i,j}^{It}\right) & else \end{cases} & if\ r_2 < \mathrm{P}_3 \\ x_{i,j}^{It} & else \end{cases} \tag{7}$$

$$AF = 0.01 + 0.99 \times \left(1 - \frac{FEs}{MaxFEs}\right) \tag{8}$$

where ub and lb are the upper and lower bounds of the search domain, and r_2, r_3, r_4, and r_5 are uniformly distributed random numbers in the range [0,1]. The value of P_3 controls the probability of reflection, and is generally set to 0.3. The attenuation factor (AF) is composed of the current number of evaluations (FEs) and the maximum number of evaluations ($MaxFEs$). As the algorithm iterates, the value of AF will gradually converge to 0.01, which means that as an individual progresses, they will avoid wasting unnecessary time with frequent initialization. In the reflection phase, the jth aspect of the ith individual will be guided by some upper-level individual ($\vec{R}$). $\vec{R}$ denotes an individual at the high level, and it serves as a reflective learning guider for the current individual i. R_j is the knowledge of the jth aspect of $\vec{R}$. The range of $\vec{R}$ is defined as the top $\mathrm{P}_1 + 1$ individuals (the leader or the elite) in the population, which means that when the jth aspect of the ith individual truly requires learning from others, there will be an upper-level individual as $\vec{R}$ to guide it.

The same point with the learning stage is that each time the ith individual has finished his reflection, he should verify whether he has grown. The whole verification process and the subsequent operations, are still performed using Eq. (6).

2.3. Implementation of GO for optimization

In this subsection, the detailed implementation of GO is provided. Algorithm 1 gives the pseudocode of GO.

Step 1: Define the optimization problem as minimizing $f(\vec{x})$. Initialize parameters, including: population size (N), population dimension (D), number of evaluations (FEs), maximum number of evaluations ($MaxFEs$), search domain upper bound (ub), and search domain lower bound (lb). $f(\vec{x})$ is the objective function, $\vec{x}_i$ is the ith decision variable or individual, its jth dimension component is x_{ij}, and $lb \le x_{ij} \le ub$.

Step 2: Initialize the population. The population is initialized according to N, D, ub and lb. For GO, the population represents the set of individuals, and the components of the individual represent the necessary elements (such as morality, beliefs, cultivation, emotions, and perseverance). An individual is a row vector with one row and D columns, and the whole population is a matrix with N rows and D columns.

Step 3: Enter the loop iteration process. GO enters the iteration and first sorts the individuals by GR, to return an ascending ranking array of the entire population and stores it in ind. The best individuals in the current iteration are stored using $\vec{x}_{best}$. Note that the optimal solution ($\vec{x}_{best}$) during the current iteration is updated before each iteration, while the real-time global optimal solution ($\overrightarrow{gbestx}$) is updated at each evaluation.

Algorithm 1: The pseudocode of GO algorithm

Input: N, D, ub, lb, $\mathrm{P}_1 = 5$, $\mathrm{P}_2 = 0.001$, $\mathrm{P}_3 = 0.3$, FEs=0
Output: Global optimal solution:$\overrightarrow{gbestx}$
Initialize the population using $\vec{x} = lb + (ub - lb) \cdot rand(N, D)$ and evaluate ($i = 1, \ldots, N$)
while $FEs <= MaxFEs$ **do**
  [~,ind]=$sort(GR)$
  $\vec{x}_{best} = \vec{x}(ind(1), :)$
  %Learning phase:
  for $i = 1 : N$ **do**
    $\vec{x}_{better} = \vec{x}(ind(randi([2, 1 + \mathrm{P}_1], 1)), :)$
    $\vec{x}_{worse} = \vec{x}(ind(randi([N - \mathrm{P}_1, \mathrm{N}], 1)), :)$
    Find two random individuals that are different from $\vec{x}_i$: $\vec{x}_{L1}$ and $\vec{x}_{L2}$
    Compute $\vec{Gap}_k$, $k = 1, 2, 3, 4$ by Eq. (1)
    Compute LF_k, $k = 1, 2, 3, 4$ according to Eq. (2)
    Compute SF_i according to Eq. (3)
    Compute $\vec{KA}_k$, $k = 1, 2, 3, 4$ according to Eq. (4)
    Complete the learning process for the ith individual once according to Eq. (5)
    Complete the update of the ith individual according to Eq. (6)
    Real time update $\overrightarrow{gbestx}$
    $FEs = FEs + 1$
  end
  %Reflection phase:
  for $i = 1 : N$ **do**
    Complete the reflection process for the ith individual once according to Eq. (7) and Eq. (8)
    Complete the update of the ith individual according to Eq. (6)
    Real time update $\overrightarrow{gbestx}$
    $FEs = FEs + 1$
  end
end
Output $\overrightarrow{gbestx}$

Step 4: Learning phase. For the ith individual, one $\vec{x}_{better}$ and one $\vec{x}_{worse}$ will be selected to participate in the learning process, and of course, $\vec{x}_{best}$ will also participate in the learning process of the individual i, using Eqs. (1) to (6) to complete the update of the individual i and update the global optimal solution $\overrightarrow{gbestx}$ in real-time. Record the number of evaluations using FEs.

Step 5: Reflection Phase. For the jth dimension of the ith individual, the algorithm will process the jth dimension according to Eqs. (7) and (8). There are three specific processing methods. The first is to keep the original dimension; the second is to select an upper-class person to guide the jth dimension of the ith individual, and the third is to rebuild the jth dimension of an individual with a small probability on the basis of the second method. Eq. (6) is then used to update the ith individual, and update the global optimal solution $\overrightarrow{gbestx}$ in real-time. Record the number of evaluations using FEs.

Step 6: Termination criterion. If the current number of evaluations (FEs) reaches the maximum number of evaluations ($MaxFEs$), output $\overrightarrow{gbestx}$ and stop the program. Otherwise, return to step 3.

The design of the core operator in the learning phase of GO is inspired by the influence exerted by different individuals, which means that the operator itself avoids the strong influence of a few individuals and could decrease the chance of errors in the search direction of the algorithm. The reflection phase of GO relies on an outstanding person to guide or relearn with a small probability, while maintaining most of its advantages to strengthen the local exploitation ability of the algorithm.

2.4. Comparative study of GO with TLBO, GSK, BSO, and HBO

This subsection discusses the differences between GO and similar metaheuristic techniques, such as TLBO, GSK, BSO, and HBO, in order to determine its novelty. To better illustrate these differences, they are summarized in Table 2 and are explained in detail below.

Table 2
The main differences between GO and TLBO, GSK, BSO, and HBO.

Main differences	GO	TLBO	GSK	BSO	HBO
Inspiration	Mechanisms of learning and reflection in the process of growth of members of society	The process of teacher teaching and student interaction in the classroom	The process of gaining and sharing knowledge during the human life span	The process of human brainstorming	The process of employees achieving a common goal based on the company's hierarchy
Agent	The individual or the members of society	The teacher and the student	The individual	The individual	The company member
Population division	Use hyperparameters to classify populations into three levels, but use only two levels	No division	Proportionally divide the population into 3 levels and use them	Using k-means to cluster individuals into 5 classes	Dividing populations based on the ternary search tree
Core operator	The learning operator is under adaptive control of European distance and fitness values, and the reflection operator are controlled by random numbers, hyperparameters, and linear variables	The teacher operator is under random number and hyperparameter control, and the learner operator is under control of the random number	The junior gaining-sharing knowledge operator is under the control of the random number and hyperparameter, and the senior gaining-sharing knowledge operator is under the control of the hyperparameter	The core operator is under the control of the random number, hyperparameter, and non-linearly variables	The core operator is under the control of the random number and linearly variable
Convergence Strategy	Multi-elite collaborative guidance	Single-elite guidance	Single-elite guidance	Non-elite guidance	Single-elite guidance
Convergence direction	4 directions	1 direction	2 directions	1 direction	1 direction
Location information	5 kinds of information	2 kinds of information	3 kinds of information	2 kinds of information	a kind of information
Update strategy	Greedy selection & Simplified Monte Carlo sampling	Greedy selection	Greedy selection	Greedy selection	Greedy selection

GO, like TLBO, GSK, BSO, and HBO, is a population-based technique, which is based on a set of solutions to find the global optimal solution.

In terms of inspiration, GO is inspired by the learning and reflection behavior of the human growth process. TLBO is inspired by the teaching behavior of teachers and the communication behavior of students. GSK is inspired by the process of gaining and sharing knowledge in the human lifespan. BSO is inspired by the process of team members brainstorming together. HBO is inspired by the process of different levels of company members working together to achieve a task.

In terms of metaphors of agents or solutions, GO metaphorically refers to them as individuals or social members, TLBO metaphorically refers to them as students and teachers, GSK and BSO metaphorically refer to them as individuals, and HBO metaphorically refers to them as members of a company.

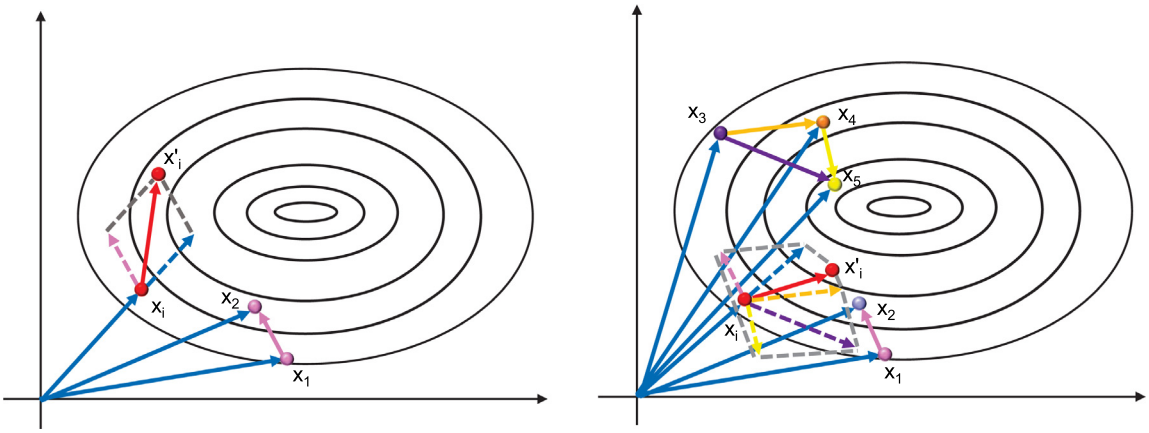
In terms of population partitioning, GO and GSK are most similar in that they both divide the population into 3 levels. However, the difference is that GO is based on the integer-based segmentation method and uses only two levels, while GSK is based on the proportional-based segmentation method and uses all levels. BSO is based on the k-means cluster method for population clustering operations. HBO is based on the concept of the ternary search tree or heap for population partitioning. The most special one is TLBO, which does not partition the population.

Core operator, convergence strategy, convergence direction, and position information are closely related and will be discussed comprehensively here. Five types of position information (containing multiple elite positions) are used in the learning operator of GO to provide four convergence directions for the algorithm, and these directions are controlled using Euclidean distance and fitness values to self-adaptively determine the current search direction of the algorithm. The reflection operator of GO incorporates three modes of operation and probabilistically guides each dimension of an individual based on different elite location information to achieve asymptotic convergence. The core operators of TLBO, GSK, BSO, and HBO use less position information to determine their search direction and use a large number of random numbers to control these convergence directions, which undoubtedly introduces blind, nonadaptive, and scarce error information to the algorithm, thus reducing the optimization performance of the algorithm. Fig. 2 shows more intuitively the effect of the directional information number on the convergence direction of the algorithm. As seen in Fig. 2(a), the one piece of search direction information contributed by two individuals tends to cause the algorithm to converge in the wrong direction. As seen in Fig. 2(b), GO uses five individuals to contribute four pieces of search directions, which are mutually constrained, giving GO the potential to avoid falling into local optima.

In terms of update strategy, TLBO, GSK, BSO, and HBO use the most common greedy selection strategy. However, GO inspired by Monte Carlo sampling, simplifies and integrates it into the greedy selection strategy to maintain population diversity. Of course, this method is not the first that uses this strategy to be proposed, and SA also uses Monte Carlo sampling. However, GO uses simplified Monte Carlo sampling.

3. Experimental results and analysis

In this section, problems and experiments related to numerical optimization testing with respect to metaheuristic algorithms are discussed in Section 3.1. Section 3.2 analyzes GO search pattern by means of an example. Section 3.3 performs parameter sensitivity testing of the GO and determines the parameters used for the experiment. Considering the complex real-world application scenarios with nondifferentiable targets, noise, and uncertainty, all experiments must be optimized by black-box optimization [76]. Since the test function of CEC 2017 is characterized by a large number of local optima, dynamic uncertainty interference, and the absence of linkage between variables, the real performance of the metaheuristic algorithms can be verified by experimentally



(a) The search process under the influence of single-direction information. (b) The search process under the influence of multi-direction information.

Fig. 2. The effect of directional information on the direction of convergence.

solving these problems [77]. Therefore, Section 3.4 conducts numerical optimization experiments of GO with 50 state-of-the-art metaheuristic algorithms and perform statistical analysis of the experimental results.

3.1. Some problems of the experiment

Testing metaheuristic algorithms on an international benchmarking set is necessary to verify their performance. However, there are some problems in conducting these experiments in much of the literature, and they need to be pointed out for the benefit of the development of the optimization field.

(1) Many papers related to metaheuristic algorithms use the relevant test suite of the IEEE Conference on Evolutionary Computation but do not use the experimental setup required by the test suite standard; that is, they do not follow the criterion $MaxFEs = D \times 10{,}000$, where $MaxFEs$ is the maximum number of evaluations, and D is the problem dimension. One thing they have in common is that they use fewer iterations, which means fewer evaluations, which is suspicious. One explanation is that they chose the optimal iteration interval of the algorithm, i.e., the algorithm may outperform other algorithms in the intentionally chosen iteration interval.
(2) Since the choice of test functions is not rigid, this gives researchers of these metaheuristic algorithms the freedom to choose the functions to be tested. The proposed algorithms may perform better than other algorithms on these intentionally chosen test functions.
(3) In most of the literature on metaheuristic algorithms, the number of compared optimization algorithms is far from adequate, and the problem of selecting the competing algorithms is somewhat subjective.

3.2. An example of optimization by GO

This subsection uses the multimodal Schwefel function to show the movement of an individual to the minimum point. This function has a large search domain and a large number of local extrema, and solving this function can fully show the performance of the algorithm, whose specific mathematical model is described by Eq. (9) as follows.

$$\min f\left(\vec{x}\right) = 418.9829D - \sum_{j=1}^{D} x_j \sin\left(\sqrt{|x_j|}\right) \tag{9}$$

where with global optimum $\vec{x} = [0, 0, \ldots, 0]$ and $f(\vec{x}) = 0$ for $-500 \leq x_j \leq 500$.

GO is used to solve the Schwefel function by reducing the number of decision variables to 10 and setting the number of iterations to 100, which helps to clearly show the search pattern of GO. Fig. 3 shows the four metrics used to evaluate GO performance and effectiveness: average fitness value error, 1D search history, 2D search history, and 3D search history. It is worth noting that the metrics in each row of Fig. 3 are derived from samples of the 1st generation, the 30th generation, the 50th generation, and the 100th generation.

The first metric in Fig. 3 is the average fitness error, which represents the change in the error between the fitness of the global optimal solution searched by the algorithm and the functional theoretical optimal solution from the 1st generation to the 100th generation of the population evolution. As the convergence curve declines, it fully shows the process of switching from global exploration to local exploitation of the algorithm and gradually approaches the global optimal solution. In addition, the sample points are marked on the convergence curve to facilitate the analysis of the relationship between the current number of iterations and other metrics.

The second metric in Fig. 3 is the 1D search history. The fitness errors for all individuals and their corresponding 1D positions are shown on the longitudinal profile of the 2D function. At the beginning of the iteration, the positions of all individuals are scattered, and there is no rule to follow for their positions. As the iterations proceed, the errors in the fitness values of all individuals decline, and they continue to converge to the best position. This behavior shows that GO has the clustering effect of SI on the one hand, and demonstrates the convergence property of GO on the other.

The third column of Fig. 3 shows the 2D search history, with the contour color fading from red to blue, indicating the change in fitness values from high to low. The individual positions are displayed on the 2-dimension transverse profile of the function, allowing a clear view of the individual search pattern. At the beginning of the iteration, the individuals are uniformly distributed in the search domain. As the iterations proceed, some individuals are clustered around different local optima, demonstrating the global search capability of the algorithm, while others are located at the boundaries of the search domain, which demonstrates the boundary exploration capability of the algorithm. In the later iterations, through the collaboration of all individuals, all individuals gradually converge to the global optimal point, and the combined analysis of the first two metrics indicates that the

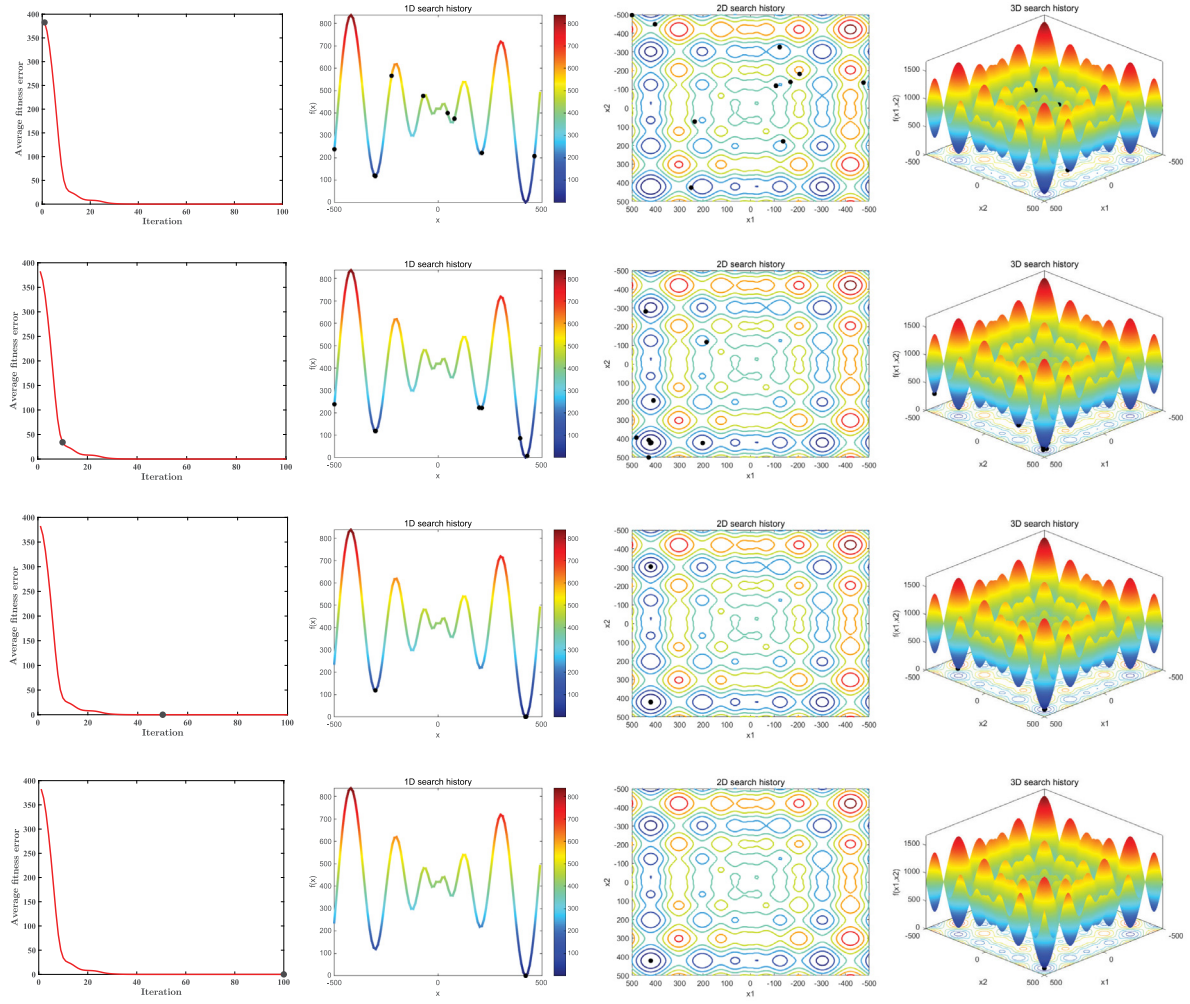


Fig. 3. Qualitative metrics: average fitness error, 1D search history, 2D search history, and 3D search history.

algorithm function has changed from global exploration to local exploitation.

The fourth metric in Fig. 3 is the 3D search history, which is located in the fourth column of Fig. 3. By analyzing this metric, it is easy to understand the complexity of the problem structure on the one hand and to observe the search pattern of the analysis algorithm from a three-dimensional perspective on the other. As with the previous metrics, the color of the contour line is closely related to the fitness value. Since the initialization determines the initial position of the individual, as the iteration proceeds, the individual may converge to a nearby local optimum, a phenomenon that can be confirmed in the second and third plots in the fourth column. The collaborative guidance model of GO allows individuals to be forced to move away from worse individuals and toward better ones while expecting to change the fixed search pattern based on the perturbation of random individuals.

3.3. Parameter sensitivity analysis

In this subsection, parameter sensitivity tests are performed for parameters P_1, P_2, and P_3. For each parameter, four possible values were chosen. The effect of different values of the parameters on the convergence of the algorithm will be experimentally investigated to determine the sensitivity of these parameters. The experiments will also determine the optimal parameters for subsequent experiments. For each parameter, comprehensive experiments and analyses are performed based on four different types of functions (the unimodal function F1, the multimodal function F10, the hybrid function F20, and the composition function F30) and four different dimensions ($D = 10$, $D = 30$, $D = 50$, and $D = 100$) of the CEC 2017 test suite. The experimental settings were $Run = 31$, $N = 40$, $D = 10/30/50/100$, and $MaxFEs = D * 10{,}000$. Table 3 shows the results of parameter sensitivity tests for P_1, P_2, and P_3.

First, a parametric sensitivity analysis was performed on P_1, which affects the division of the entire population. When $P_1 = 3$, the optimization performance of GO decreases significantly with dimension for the unimodal function F1, while it is stable for the other functions. When $P_1 = 4$ or $P_1 = 6$, GO performs better on the unimodal function F1 and the composition function F30 but underperforms on the multimodal function F10 and the hybrid function F20. When $P_1 = 5$, GO is stable on all functions and does not show extreme cases, which is a good compromise. The experimental results also show that GO is not sensitive to P_1, however as its value increases significantly, the convergence rate of GO may decrease.

Second, the parameter sensitivity of P_2 is analyzed, which affects the update state of the solution. When $P_2 = 0.1$, the optimization performance of GO is poor for all functions. When $P_2 = 0.0001$, GO performs well only on the unimodal function F1. When $P_2 = 0.01$, the performance of GO deteriorates on the unimodal function F1. When $P_2 = 0.001$, GO can bring better results than other conditions. The experiment shows that

Table 3
The effect of different values of parameters P_1, P_2, and P_3 on the average fitness value error.

Function	P_1					P_2					P_3				
	Parameter	$D = 10$	$D = 30$	$D = 50$	$D = 100$	Parameter	$D = 10$	$D = 30$	$D = 50$	$D = 100$	Parameter	$D = 10$	$D = 30$	$D = 50$	$D = 100$
F1	$P_1 = 3$	0.00E+000	1.96E−023	2.17E−003	6.93E+003	$P_2 = 0.1$	2.68E+003	2.14E+003	4.82E+003	1.80E+005	$P_3 = 0.2$	0.00E+000	0.00E+000	4.89E−008	4.50E+003
	$P_1 = 4$	0.00E+000	0.00E+000	4.72E−003	4.13E+003	$P_2 = 0.01$	5.36E+002	1.08E+001	3.44E+003	1.39E+003	$P_3 = 0.3$	0.00E+000	0.00E+000	1.26E−002	5.13E+003
	$P_1 = 5$	0.00E+000	7.24E−024	1.19E−002	4.09E+003	$P_2 = 0.001$	0.00E+000	3.06E−023	9.87E−002	4.18E+003	$P_3 = 0.4$	1.09E−023	9.62E−022	5.30E+001	2.12E+003
	$P_1 = 6$	0.00E+000	2.63E−025	1.72E−002	1.62E+003	$P_2 = 0.0001$	0.00E+000	6.57E−026	7.95E−004	2.69E+003	$P_3 = 0.5$	0.00E+000	6.15E−014	9.32E+002	6.76E+003
F10	$P_1 = 3$	4.43E+001	2.58E+003	3.37E+003	7.52E+003	$P_2 = 0.1$	1.86E+001	4.75E+003	1.11E+004	2.74E+004	$P_3 = 0.2$	4.43E+001	3.63E+003	2.74E+003	7.60E+003
	$P_1 = 4$	1.48E+002	1.47E+003	2.91E+003	8.37E+003	$P_2 = 0.01$	1.16E+002	1.46E+003	3.28E+003	6.54E+003	$P_3 = 0.3$	1.48E+002	2.18E+003	2.74E+003	7.41E+003
	$P_1 = 5$	1.23E+002	2.95E+003	2.63E+003	7.97E+003	$P_2 = 0.001$	8.79E+001	3.05E+003	4.78E+003	7.51E+003	$P_3 = 0.4$	1.23E+002	2.51E+003	2.94E+003	7.57E+003
	$P_1 = 6$	1.01E+002	3.66E+003	2.98E+003	7.27E+003	$P_2 = 0.0001$	1.46E+002	4.21E+003	9.00E+003	2.44E+004	$P_3 = 0.5$	1.01E+002	1.67E+003	2.86E+003	7.46E+003
F20	$P_1 = 3$	7.31E−001	1.18E+002	1.62E+002	1.65E+003	$P_2 = 0.1$	1.10E+000	4.57E+001	6.12E+002	3.53E+003	$P_3 = 0.2$	7.31E−001	4.69E+001	2.34E+002	2.28E+003
	$P_1 = 4$	7.74E−001	6.69E+001	2.30E+002	1.84E+003	$P_2 = 0.01$	6.64E−001	8.13E+001	7.44E+001	5.05E+002	$P_3 = 0.3$	7.74E−001	9.64E+001	1.40E+002	1.27E+003
	$P_1 = 5$	3.52E−001	8.22E+001	1.70E+002	1.34E+003	$P_2 = 0.001$	6.84E−001	4.56E+001	2.34E+002	2.03E+003	$P_3 = 0.4$	3.52E−001	4.20E+001	8.83E+001	1.47E+003
	$P_1 = 6$	5.83E−001	9.27E+001	2.20E+002	2.71E+003	$P_2 = 0.0001$	1.00E+000	9.45E+001	5.16E+002	2.40E+003	$P_3 = 0.5$	5.83E−001	1.36E+002	1.76E+002	9.51E+002
F30	$P_1 = 3$	3.97E+002	2.09E+003	5.81E+005	2.55E+003	$P_2 = 0.1$	2.74E+005	4.38E+003	1.05E+006	1.92E+004	$P_3 = 0.2$	3.97E+002	1.97E+003	6.19E+005	2.56E+003
	$P_1 = 4$	3.95E+002	1.96E+003	5.80E+005	3.13E+003	$P_2 = 0.01$	1.15E+003	2.09E+003	6.22E+005	4.76E+003	$P_3 = 0.3$	3.95E+002	2.11E+003	6.03E+005	3.29E+003
	$P_1 = 5$	4.18E+002	2.08E+003	6.04E+005	2.48E+003	$P_2 = 0.001$	3.95E+002	2.39E+003	6.20E+005	2.70E+003	$P_3 = 0.4$	4.18E+002	2.09E+003	5.81E+005	4.29E+003
	$P_1 = 6$	3.95E+002	2.06E+003	5.80E+005	3.01E+003	$P_2 = 0.0001$	4.85E+002	2.15E+003	6.71E+005	3.37E+003	$P_3 = 0.5$	3.95E+002	2.06E+003	5.80E+005	5.75E+003

Table 4
The CEC 2017 benchmark suite is used in experimental studies.

Test suite	Number of function	Function types	Search range	Fitness error
CEC 2017	30	Unimodal functions (F1–F3)	[−100,100]	0
		Simple Multimodal functions (F4–F10)	[−100,100]	0
		Hybrid functions (F11–F20)	[−100,100]	0
		Composition functions (F21–F30)	[−100,100]	0

GO is very sensitive to the value of P_2, and a better value can significantly improve its optimization performance.

Finally, the parameter sensitivity of P_3 is analyzed, which affects the probability of elite guidance. On the unimodal function F1, a smaller P_3 leads to a better convergence, but $P_3 = 0.4$ is better for $D = 100$. On the multimodal function F10, a larger P_3 tends to lead to better convergence. For the hybrid function F20, the value of P_3 has little effect on the convergence accuracy of GO, but it is better when $P_3 = 0.4$. For the composition function F30, a larger P_3 value will improve GO's optimization performance, but for $D = 100$, a smaller P_3 value will be better. In summary, excessively extreme values may lead to verly extreme results, so $P_3 = 0.3$ or $P_3 = 0.4$ is an option to improve the comprehensive optimization ability of GO. However, the result of $P_3 = 0.3$ is better than the result of $P_3 = 0.4$, so $P_3 = 0.3$ is the better solution.

In summary, GO's sensitivity to P_1 and P_3 is low, while P_2 needs to be carefully tuned to improve the quality of the global optimal solution obtained by GO. Therefore, all experiments in this study used $P_1 = 5$, $P_2 = 0.001$, and $P_3 = 0.3$ as GO's parameter selection scheme to improve GO's ability to deal with different optimization problems.

3.4. Comparison of GO and 50 metaheuristic algorithms

To test and verify the search performance of the proposed algorithm, comprehensive experiments have been conducted on GO. All experiments were implemented in MATLAB r2020b and run on an Intel (R) Core (TM) i7−10700 CPU @ 2.90 GHz, 16 GB RAM, and X64 processor.

3.4.1. Description of the benchmark functions

Table 4 gives information about the benchmark suites for CEC 2017, which are the test suite name and the range of test function numbers, the number of test functions included, the test function type, the search domain, and the fitness error. The search range for all test functions is [−100,100]. Unimodal problem is used to test the local search (exploitation) capability of the algorithm. Multimodal problems are used to test the global search (exploration) capability of the algorithm. Hybrid problems are used to test the algorithm's ability to balance exploitation and exploration. Composition problems are used to test the performance of algorithms in search spaces with high complexity. Different from the classical and common test functions, they are endowed with more complex structures through the operations of shift, rotation, hybridization, and composition. These test functions are chosen to avoid nonnormal convergence to the global optimum. For example, if the global optimum of a function is $\vec{x} = [0, 0, \ldots, 0]$; then, the algorithm can converge abnormally to the global optimum by multiplying the solution update formula by a number in the range [−1,1]. On the other hand, considering their complexity, they are closer to real-world optimization problems. In these test problems, the difference between the optimal fitness value ($f(\vec{x})$) found by the algorithm and the theoretical optimal fitness value ($f(\vec{x}^*)$) is the optimal error, i.e., optimal fitness error $= min(error) = min(f(\vec{x}) - f(\vec{x}^*))$, where $\vec{x}^*$ is the theoretical optimal solution. It is worth noting that in many papers, when the best fitness error of an algorithm reaches 1E-8 orders of magnitude, the best fitness error is forced to 0, which means that the algorithm has basically found the global optimal solution. However this raises a problem: if most of the algorithms reach 1E-8 for a particular test function, which means that they all find the global optimal solution, then how can we compare the performance of these algorithms? To test the true convergence performance of each algorithm, the experiments in this study do not use the threshold of 1E−8 to determine the global optimal point found by the algorithm.

3.4.2. Experimental setup and involved algorithms

The maximum number of evaluations (*MaxFEs*) was used as the termination criterion for the algorithms, and the conditions for obtaining relevant data for each algorithm included: 30 benchmark problems, 30/50/100 dimensions, and 31 runs. All algorithms participating in the experiment used the same initialization method, and they were initialized uniformly at random in the search space. The random seed was time-based, for MATLAB users, used rand ('state', sum(100 ∗ clock)). Because of the random initialization technique, the starting search point of all algorithms was uncertain. The uniform parameter settings for all algorithms are given in Table 5. The information and parameter settings of the 50 metaheuristic algorithms are detailed in Table 6, where the first column is the number of the algorithm, the second column is the full name of the metaheuristic algorithms and its abbreviation, the third column is the year of the algorithm was proposed, the fourth column is the parameter settings used for each algorithm in this study, the fifth column is the number of

Table 5
The unified parameter settings for all algorithms.

N	*D*	*MaxFEs*	*Run*	Fitness error	Initialization method	random seed
40	30	300,000	31	0	Uniform random	rand ('state', sum(100*clock))
40	50	500,000	31	0	Uniform random	rand ('state', sum(100*clock))
40	100	1,000,000	31	0	Uniform random	rand ('state', sum(100*clock))

Table 6
The algorithm information and parameter settings.

No.	Algorithms	Proposed time	Parameters settings	Citations	References
1	Genetic Algorithms (GA)	1975	pc = 0.8, pm = 0.2	17 613	[17]
2	Simulated Annealing (SA)	1983	T_0 = 0.1, α = 0.99, c = 5, μ = 0.5, σ=0.1 × (*ub* − *lb*)	53 334	[59]
3	Cultural Algorithm (CA)	1994	pAccept = 0.35, α = 0.3	1258	[20]
4	Particle Swarm Optimization (PSO)	1995	w = 1, w_p = 0.99, c_1 = 1.5, c_2 = 2.0	72 995	[24]
5	Differential Evolution (DE)	1997	F = 0.5, CR = 0.9	29 477	[21]
6	Harmony Search (HS)	2001	HMCR = 0.9, PAR = 0.1, *FW*=0.02 × (*ub* − *lb*), FW_{damp} = 0.995	6517	[25]
7	Ant Colony Optimization (ACO)	2006	q = 0.5, ζ=1	15 010	[26]
8	Covariance Matrix Adaptation Evolution Strategy (CMA-ES)	2006	σ = 0.3	2050	[22]
9	Shuffled Frog-Leaping Algorithm (SFLA)	2006	Memeplexes = 5, Offsprings = 3, Step = 2, Run = 5	1253	[27]
10	Artificial Bee Colony (ABC)	2007	a = 1, $L = round(0.6 \times N \times D)$	7212	[28]
11	Biogeography-Based Optimization (BBO)	2008	KeepRate = 0.2, α = 0.9, Mutation = 0.1, σ = 0.02× (*ub* − *lb*)	3658	[29]
12	Firefly Algorithm (FA)	2009	α = 0.2, β = 2, γ = 1, δ = 0.05 × (*ub* − *lb*), Damping Ratio = 0.98	4293	[30]
13	Cuckoo Search (CS)	2009	Pa = 0.25, α = 1, β = 1.5, σ_v = 1	6906	[31]
14	Gravitational Search Algorithm (GSA)	2009	G_0 = 100, α = 20, Rpower = 1	6054	[60]
15	Bacterial Foraging optimization (BFO)	2010	c = 25, s = 4, r = 4, p = 0.5, $Sr = N/2$	259	[32]
16	Brain Storm Optimization (BSO)	2011	m = 5, p_{5a} = 0.2, p_{6b} = 0.8, p_{6bi} = 0.4, p_{6c} = 0.5, k = 20, μ = 0, σ = 1	591	[33]
17	Teaching–Learning-Based Optimization (TLBO)	2011	T_F = 1 or 2	3423	[34]
18	Differential Search (DS)	2012	$p_1 = 0.3 \times rand$, $p_2 = 0.3 \times rand$	440	[23]
19	Flower Pollination Algorithm (FPA)	2012	p = 0.8	2082	[35]
20	Artificial Fish Swarm Algorithm (AFSA)	2014	T = 100, visual = 0.5, step = 2, δ = 0.618	366	[37]
21	Imperialist Competitive Algorithm (ICA)	2014	nEmp = 10, α = 1, β = 1.5, pR = 0.05, μ = 0.1, ζ = 0.2	2973	[36]
22	Grey Wolf Optimizer (GWO)	2014	$\alpha = 2 - 2 \times (FEs/MaxFEs)$	8819	[38]
23	Moth–Flame Optimization (MFO)	2015	$r = -1 - (FEs/MaxFEs)$, $t = r + (1 - r) \times rand$, b = 1	2514	[39]
24	Lightning Search Algorithm (LSA)	2015	maxchtime = 10, $direct = sign(unifrnd(-1, 1))$	292	[61]
25	Stochastic Fractal Search (SFS)	2015	MDN = 2, Walk = 1	404	[62]
26	Whale Optimization Algorithm (WOA)	2016	b = 1, $a_1 = 2 - (2 \times FEs/MaxFEs)$, $a_2 = -1 - (FEs/MaxFEs)$	5784	[40]
27	Jaya	2016	No parameters	1490	[41]
28	Sine Cosine Algorithm (SCA)	2016	a = 2	2491	[63]
29	Multi-Verse Optimizer (MVO)	2016	WEP_{Max} = 1, WEP_{Min} = 0.2	1460	[64]
30	Salp Swarm Algorithm (SSA)	2017	$c_1 = 2 \times exp((4 \times FEs/MaxFEs)^2)$	2413	[42]
31	Coyote Optimization Algorithm (COA)	2018	N_c = 5, N_p = 8	295	[1]
32	Atom Search Optimization (ASO)	2019	α = 50, β = 0.2	250	[65]
33	Artificial Electric Field Algorithm (AEFA)	2019	K_0 = 500, α = 30	127	[66]
34	Harris Hawks Optimization (HHO)	2019	$E_0 = 2 \times rand - 1$, $E_1 = 2 - 2 \times (FEs/MaxFEs)$	1897	[43]
35	Butterfly Optimization Algorithm (BOA)	2019	P = 0.8, α = 0.1, c = 0.01	639	[44]
36	Artificial Ecosystem-based Optimization (AEO)	2019	$a = rand \times (FEs/MaxFEs)$, $c = randn/2 \times \lvert randn \rvert$	142	[67]
37	Supply-Demand-Based Optimization (SDO)	2019	$a_1 = 2 \times ((MaxFEs - FEs + 1)/MaxFEs)$	66	[45]
38	Manta Ray Foraging Optimization (MRFO)	2020	S = 2, $r_1 = rand$, $Coef = FEs/MaxFEs$	304	[46]
39	Gaining Sharing Knowledge (GSK)	2020	p = 0.1, k_r = 0.9, k_f = 0.5	122	[47]
40	Equilibrium Optimizer (EO)	2020	V = 1, a_1 = 2, a_2 = 1, GP = 0.5	672	[68]
41	Coronavirus Herd Immunity Optimizer (CHIO)	2020	MaxAge = 100, CO = 1, BRr = 0.05	64	[48]
42	Heap-Based Optimizer (HBO)	2020	$cycles = \lfloor MaxIt/25 \rfloor$, degree = 3	570	[49]
43	Marine Predators Algorithm (MPA)	2020	FADs = 0.2, P = 0.5	479	[50]
44	Arithmetic Optimization Algorithm (AOA)	2021	MOP_{Max} = 1, MOP_{Min} = 0.2, α = 5, μ = 0.499	633	[69]
45	Runge kutta optimizer (RUN)	2021	a = 20, b = 12	206	[70]
46	weighted mean of vector (INFO)	2022	$I = randi([2, 5])$	48	[71]
47	White Shark Optimizer (WSO)	2022	$f = [0.07, 0.75]$, $\tau = 4.11$, $p = [0.5, 1.5]$, $a_0 = 6.25$, $a_1 = 100$, $a_2 = 0.0005$	8	[52]
48	Honey Badger Algorithm (HBA)	2022	β = 6, c = 2, *flag* = −1 or 1	99	[53]
49	Dwarf Mongoose Optimization Algorithm (DMOA)	2022	b = 3, Lp = 0.6, peep = 2	49	[54]
50	Snake Optimizer (SO)	2022	$T_1 = 0.25$, $T_2 = 0.6$, $C_1 = 0.5$, $C_2 = 0.05$, $C_3 = 2$	12	[55]
51	Growth Optimizer (GO)	2022	$P_1 = 5$, $P_2 = 0.001$, $P_3 = 0.3$		

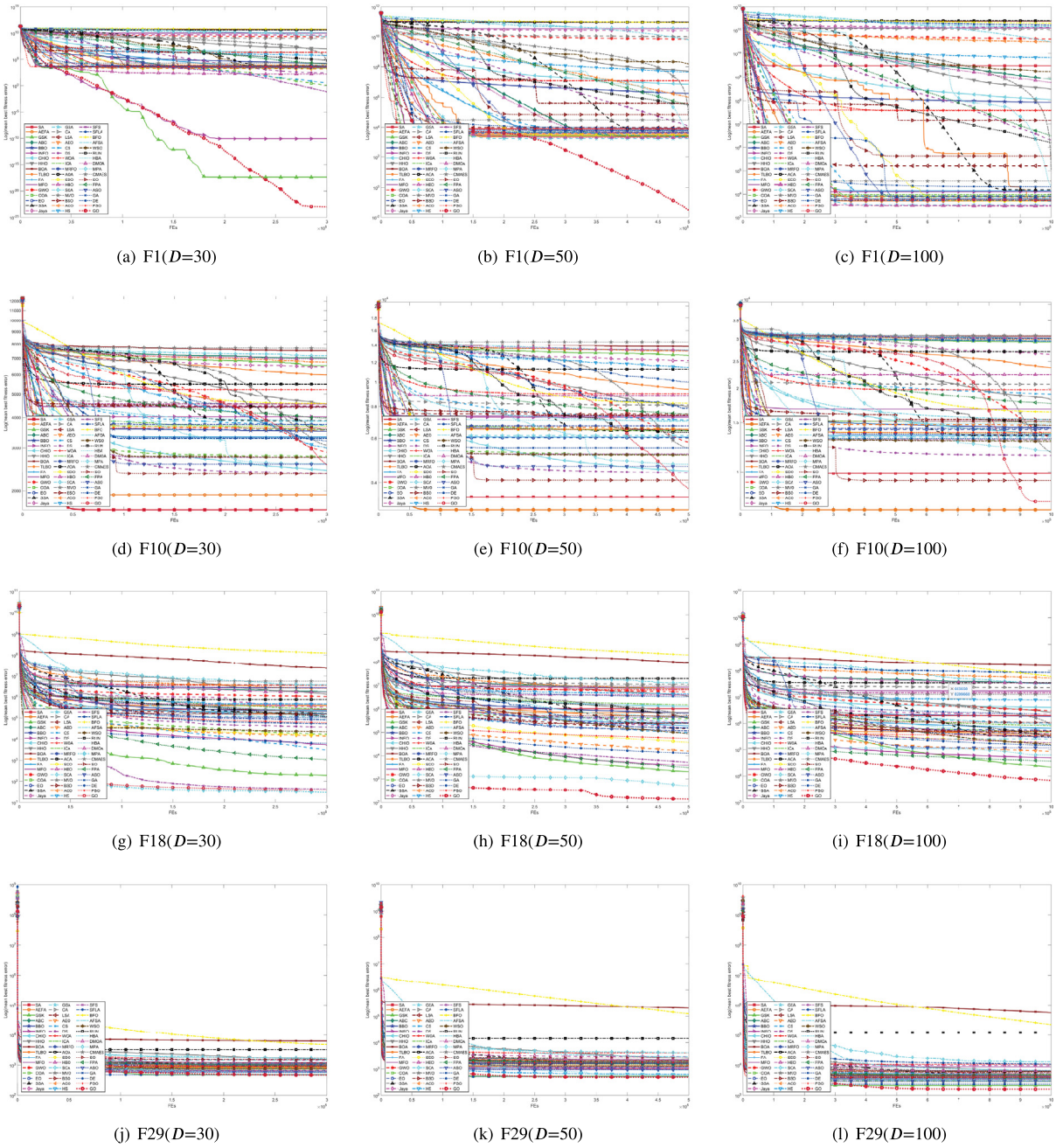
citations in the related literature, and the last column is the index of the related literature. If the decision variables violated the constraints in the experiment, GO restricted their values to the bounds, while for the other algorithms, the methods from the relevant literature are still used.

3.4.3. Convergence results and analysis

This subsection gives the convergence curves of all competing algorithms on four groups of different types of problems (Fig. 4), namely the unimodal function F1, the multimodal function F10, the hybrid function F16, and the composition function F28. These four sets of questions are based on three dimensions ($D = 30$, $D = 50$, and $D = 100$). From Fig. 4, it can be seen that the performance of GO is most prominent when $D = 30$ or $D = 50$ for the unimodal function F1, and there is no tendency to fall into a local optimum. When D = 100, the optimization performance of GO is surpassed by most other algorithms, although its optimization accuracy is still high. However, its performance is no longer outstanding. On the multimodal function F10, the performance of all algorithms is clearly differentiated due to the complexity of the problem, with SA and AEFA being the best-performing algorithms. However, it is undeniable that the convergence curve of GO is slowly declining due to the collaborative search strategy, and because of this feature, the optimization ranking of GO increases rapidly as dimensions rise, and the convergence accuracy result of GO is in second place when $D = 100$. On the hybrid function F18 and the composition function F29, the convergence accuracy of GO rapidly diverges from the other algorithms as the dimension of the problem increases. This also shows that GO is powerful and comprehensive not only for different problems but also for different dimensions.

3.4.4. Statistical results and analysis

The box plot is generally used to characterize discrete distributions of data, and its greatest advantage is that it is not affected by outliers. In Fig. 5, the statistical box plots of the four exemplars mentioned earlier are given, which show the distribution of the results of all the algorithms for 31 runs on a given problem, to analyze the stability of the algorithms. These exemplars are based on the unimodal function F1, multimodal function F10, hybrid function F18, and composition function F29 in three different dimensions. The first row of Fig. 5 shows the distribution of the results for each algorithm on the unimodal function F1 in 30, 50, and 100 dimensions. It can be seen that the convergence accuracy results of all algorithms become increasingly scattered as the dimension increases. The most stable one is GO, especially at $D = 30$ and $D = 50$. When $D = 100$, the distribution of GO results is similar to those of most of the algorithms. The second row of Fig. 5 is based on the multimodal function F10, whose complex topology makes the distribution of results of all algorithms very unstable. The stability of SA and AEFA is not affected by dimension change, and their box plot positions are low to



(a) F1(D=30) (b) F1(D=50) (c) F1(D=100)

(d) F10(D=30) (e) F10(D=50) (f) F10(D=100)

(g) F18(D=30) (h) F18(D=50) (i) F18(D=100)

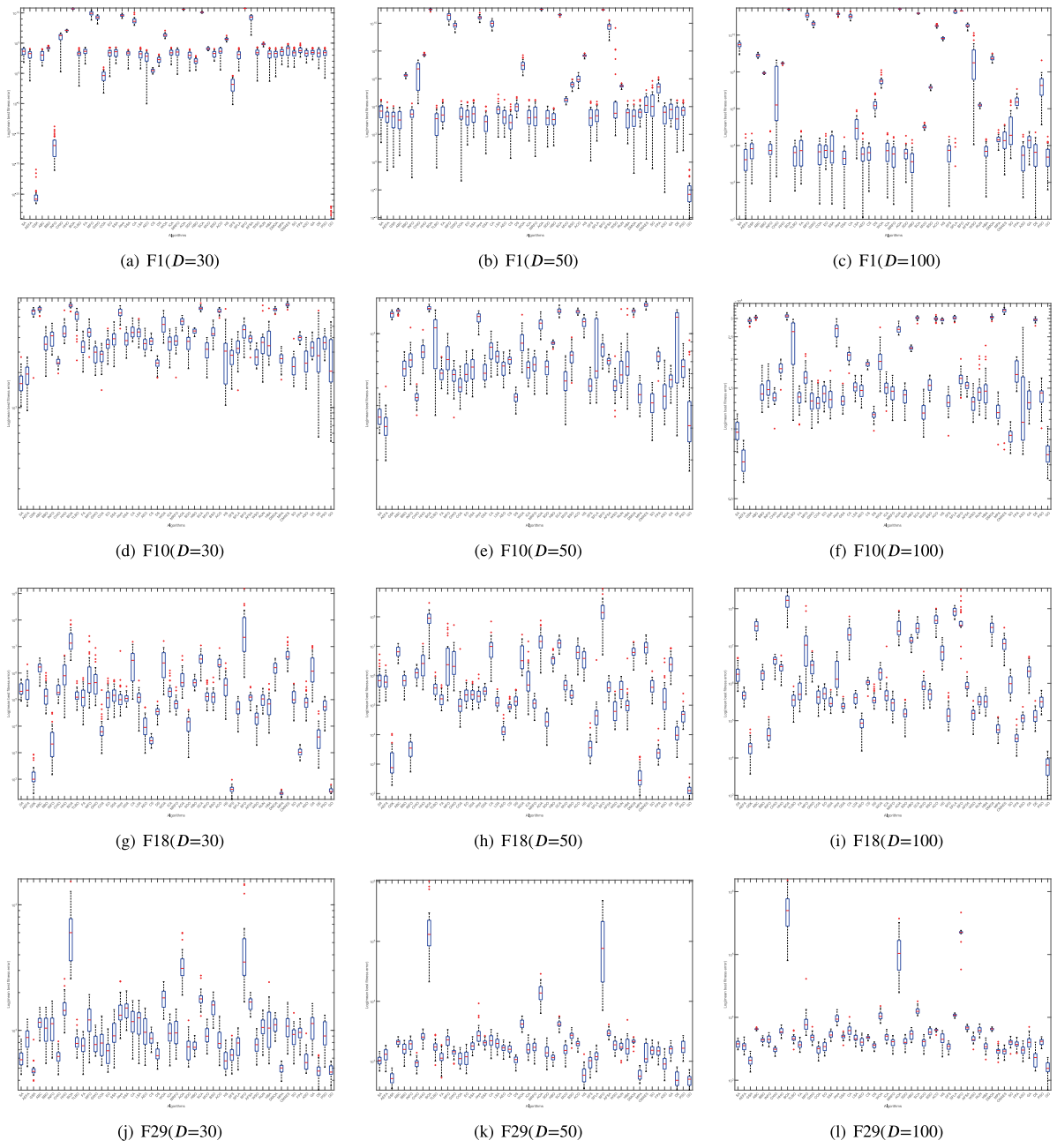
(j) F29(D=30) (k) F29(D=50) (l) F29(D=100)

Fig. 4. The convergence curves of GO and the other 50 metaheuristic algorithms on four groups of problems.

show better optimization performance. The data distribution of GO becomes more concentrated with increasing dimension, and the convergence accuracy results quickly catch up with those of most algorithms. On both functions discussed earlier, the lower edges of most box plots are far from the lower quartile, indicating that most algorithms have potential optimization performance. On the hybrid function F18, the fluctuation of the result distribution of most algorithms is independent of the dimension. The quality of the SFS and MPA results deteriorates with increasing dimensions, while GO is more stable and the box-plot positions stand out quickly with increasing dimensionality. The last row of Fig. 5 is based on the composition function F29, for which the stability of most of the algorithms is positively correlated with the dimension. However, BOA and BFO are exceptions, and they can be unstable in any dimension of F29. The box plot of GO has a flat shape, and its position is the lowest to demonstrate better robustness and stability.

The Friedman test is a nonparametric test that achieves the same purpose as analysis of variance (ANOVA) test. The invalid hypothesis in the test is the equivalence of intermediate values of different algorithms, and the alternative hypothesis is the difference of intermediate values of two or more algorithms. The Friedman rank test is calculated according to the average fitness value of 50 metaheuristic algorithms, and their statistical results are shown in Table 7. The visual results of the Friedman test mean ranks and optimal numbers of the best six algorithms and the worst six algorithms are given in Fig. 6.

In Table 7, the first column is the placement according to Friedman test mean ranks, the second column is the name of the algorithm, the third to fifth columns are Friedman test ranks of each algorithm in the different problem dimensions, and the



(a) F1(D=30) (b) F1(D=50) (c) F1(D=100)

(d) F10(D=30) (e) F10(D=50) (f) F10(D=100)

(g) F18(D=30) (h) F18(D=50) (i) F18(D=100)

(j) F29(D=30) (k) F29(D=50) (l) F29(D=100)

Fig. 5. Box plot of the four groups of problems.

last column is the Friedman test mean ranks. It can be seen from the table that GO ranks first, with an average ranking of 3.567, far ahead of GSK, which ranks second. To be sure, GO's rankings are stable and excellent for different issues. The ranking and performance of GSK, SFS, and DE deteriorate as the problem dimension rises, while the ranking and performance of the subsequent ASO and AFEA improve as the problem dimension rises, especially those of AEFA. In Fig. 6, the projection onto the xy-plane indicates the number of problems in which each algorithm ranks first among all problems, i.e., the optimal numbers. It can be seen that GO has obtained 34 firsts on the 90 (30 functions $\times$ 3 dimensions) problems, far ahead of AEFA's 20. The histogram on the xz-plane is a visualization of the Friedman test mean ranks. The smaller the rank result is, the higher the performance of the algorithm. From the graph, we can see that GO provides a competitive optimization ability.

The Wilcoxon signed-rank test is a pairwise comparison of GO with all competing algorithms to verify whether there is a significant difference between the algorithms at a significance level α of 0.05. The test is based on two hypotheses: the null hypothesis, which assumes that there is no significant difference between the two algorithms for a given problem, and the alternative hypothesis, which assumes that there is a significant difference between the two algorithms for a given problem. The result of the hypothesis test is returned as a logical value (h); when $p - value \leq \alpha$, then $h = 1$, which indicates that the null hypothesis is rejected at the significance level of α. Otherwise, if $p - value > \alpha$, then $h = 0$, which indicates that the null hypothesis is not rejected at the significance level of α. The indicator '$+/=/-$' in Table 8 is obtained based on the Wilcoxon signed-rank test, and it indicates the number of test problems in which the competing algorithms performed better, similar, or worse compared to GO in the condition of $\alpha = 0.05$.

Table 7
The Friedman test ranks for GO and the compared algorithms.

Placement	Algorithms	$D = 30$	$D = 50$	$D = 100$	Mean rank
1	GO	3.867	2.967	3.867	3.567
2	GSK	5.367	8.467	8.667	7.501
3	SFS	9.0333	12.300	13.133	11.489
4	DE	10.233	12.467	12.467	11.722
5	ASO	14.167	13.733	13.733	13.878
6	AEFA	18.033	14.033	10.433	14.166
7	MPA	13.066	14.267	18.600	15.311
8	EO	15.367	16.900	14.367	15.545
9	COA	16.433	17.900	13.467	15.933
10	DS	13.567	17.100	18.133	16.267
11	SDO	15.233	16.433	17.167	16.278
12	FA	18.733	16.767	15.133	16.878
13	SO	20.267	19.867	17.333	19.156
14	CS	17.833	20.033	22.767	20.211
15	MVO	24.367	20.633	16.733	20.578
16	AEO	22.133	22.433	18.500	21.022
17	INFO	20.733	22.667	19.800	21.067
18	HBO	21.067	20.233	21.967	21.089
19	SA	20.633	20.833	22.700	21.389
20	TLBO	19.767	21.400	23.100	21.422
21	CHIO	22.367	20.500	22.100	21.656
22	FPA	20.367	22.367	23.167	21.967
23	PSO	22.833	21.333	22.100	22.089
24	HBA	26.467	23.233	17.100	22.267
25	MRFO	24.100	23.600	19.200	22.300
26	BBO	24.433	22.533	20.433	22.466
27	CMA-ES	29.233	22.267	17.533	23.011
28	SSA	26.733	23.833	20.067	23.544
29	GA	27.633	24.667	20.600	24.300
30	HS	20.467	23.833	29.367	24.556
31	GSA	29.033	26.667	21.300	25.667
32	SFLA	16.300	15.567	45.733	25.867
33	WSO	24.167	27.200	26.677	26.015
34	ICA	28.833	27.733	25.133	27.233
35	RUN	28.600	27.600	26.300	27.500
36	DMOA	27.733	28.567	31.500	29.267
37	ABC	27.733	28.400	31.933	29.355
38	GWO	30.533	29.533	29.067	29.711
39	LSA	32.467	30.300	27.800	30.189
40	ACO	29.367	31.367	35.733	32.156
41	BSO	34.867	33.267	30.667	32.934
42	HHO	41.300	39.533	36.633	39.155
43	CA	39.333	39.333	41.867	40.178
44	Jaya	39.900	40.867	42.500	41.089
45	WOA	43.667	42.067	38.933	41.556
46	MFO	41.833	42.000	41.000	41.611
47	AFSA	42.633	42.100	40.668	41.801
48	SCA	44.667	45.833	45.133	45.211
49	AOA	45.800	46.933	47.467	46.733
50	BFO	49.533	47.800	45.633	47.655
51	BOA	48.267	49.633	49.267	49.056

Table 8
The Wilcoxon signed-rank test results of the algorithms.

vs. GO		$D = 30$	$D = 50$	$D = 100$
GSK	$+/=/-$	7/15/8	6/8/16	1/8/21
SFS	$+/=/-$	2/7/21	1/3/26	0/6/24
DE	$+/=/-$	0/12/18	0/7/23	0/6/24
ASO	$+/=/-$	1/6/23	2/2/26	2/5/23
AEFA	$+/=/-$	8/4/18	11/3/16	8/5/17

Due to the large number of algorithms involved in the test, only the statistics of the top 6 algorithms were counted, and these statistics are listed in Table 8. In the indicator '$+/=/-$', the number of times GSK and SFS outperformed GO decreased significantly as the dimension increased. The indicators '$+/=/-$' for DE, ASO, and AEFA remain stable. AEFA is the algorithm that defeats GO the most times, which is true for any dimension, but its ranking is at No. 6, which shows that the performance of this algorithm is extreme. In addition, GSK and GO have the lowest number of significant differences on different problems, while other algorithms have significant differences with GO on most problems.

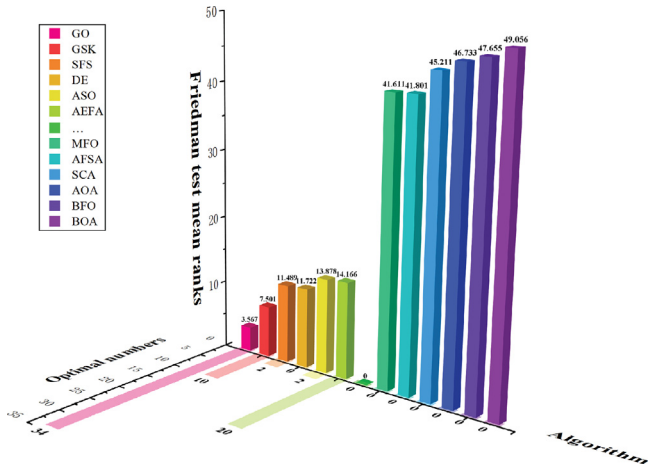


Fig. 6. The Friedman test mean ranks and the optimal numbers for better and worse algorithms.

3.4.5. Convergence behavior analysis of GO

The five most representative functions in the CEC 2017 test suite are selected for the experiments in this subsection, which are F1, F7, F10, F22, and F28. F1 is characterized by unimodality, shift, rotation, nonseparability, and a smooth but narrow ridge. This function is largely a test of the local exploitation ability of metaheuristic algorithms. Most metaheuristic algorithms experience slow convergence or even stagnation in the late evolutionary stage due to a large number of weak local extrema in this function. F7 is characterized by multimodality, nonseparability, asymmetry, continuity everywhere yet differentiable nowhere, and it is a function of offsets and rotations. F10 is one of the most difficult functions to optimize. In addition to its multimodality, shift, rotation, and nonseparability, it also has a large number of local extrema, and the global extrema are far from the local extrema. On this function, most metaheuristic algorithms will quickly fall into the traps of local optima. F22 and F28 use hybrid functions as basic functions, which can have different properties for different variable subcomponents. It is worth noting that the basic functions used in the composition function are the shift and rotation functions. Finding the global optimum in the F28 is a task that most metaheuristic algorithms cannot accomplish.

The four qualitative metrics of search history, optimization history, diversity history, and trajectory in Fig. 7 are used to analyze the performance of GO in optimizing different types of functions. It is worth noting that much of the literature obtains these metrics without indicating the dimensionality of the optimization problem, which may pose a problem: the use of 2-dimensional individuals does not capture the true optimization performance of the algorithm and the search pattern, but can mask the poor optimization performance of the algorithm itself. To better demonstrate the search pattern and optimization performance of GO, the parameter settings of the function optimization simulation experiment will be used to optimize the 30-dimensional problem with 40 individuals, using $MaxFEs = 300{,}000$ as the termination criterion of the algorithm.

The first column of Fig. 7 shows the 3-dimensional topology of the 2-dimensional function to demonstrate the structure and complexity of the problem. The second column in Fig. 7 shows the search history of individuals during the evolution of the population, keeping the position of each individual for each update. Individuals in the early, middle, and late stages of population

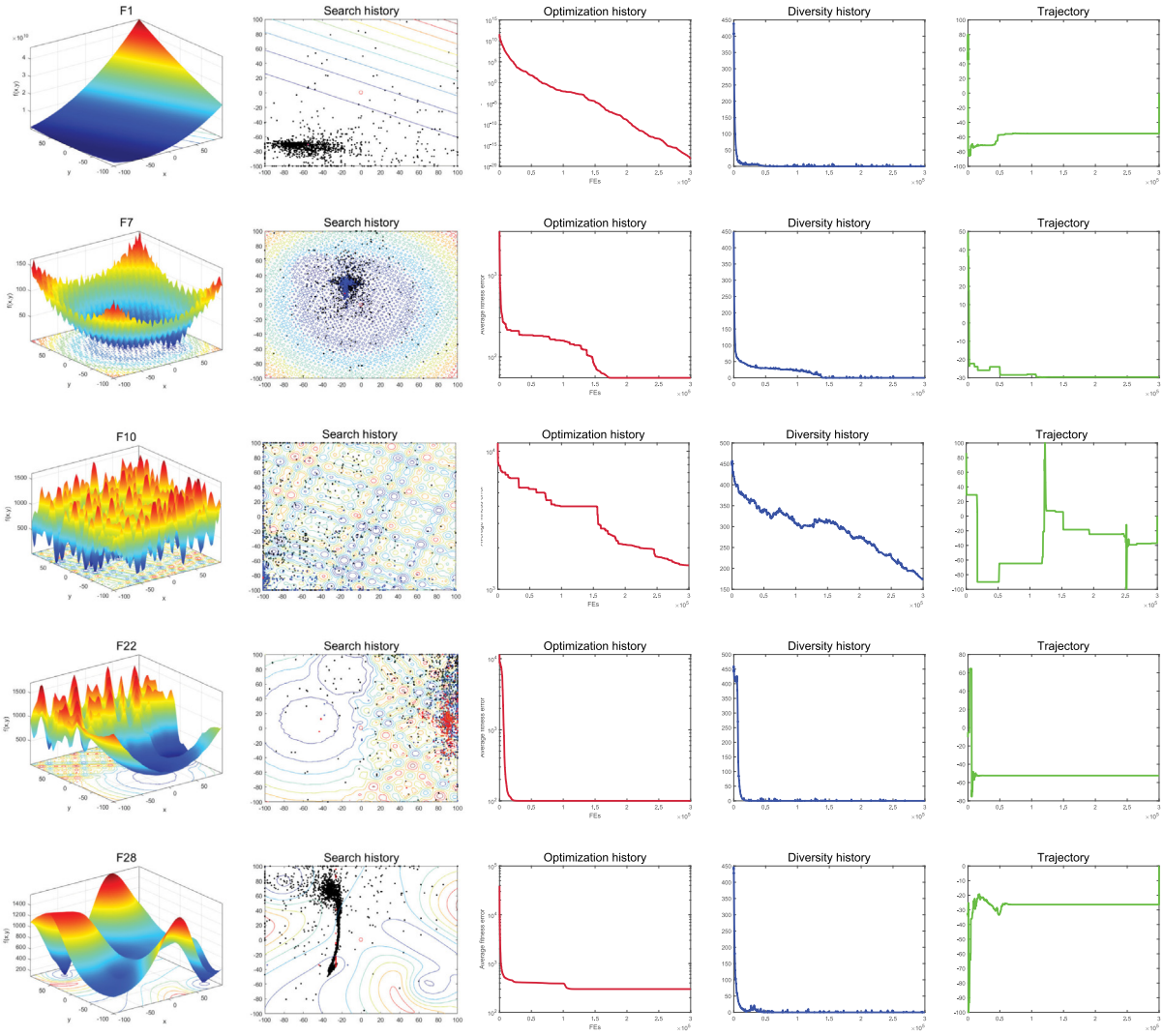


Fig. 7. Qualitative metrics: search history, optimization history, diversity history, and trajectory.

evolution are marked in black, blue, and red, respectively, to show the renewal status of individuals at different stages. Notably, the marker points in the search history are obtained by sampling the first 2 dimensions of a 30-dimensional individual. The search history reflects the different search patterns of GO for different modal functions. The search history shows that for the unimodal function F1, individuals tend to cluster in the optimal region and approach the optimal solution under the guidance of the optimal individual, which demonstrates the better local exploitation capability of the algorithm. For the multimodal functions F7 and F10, in the middle of population evolution, individuals are generated at different locations in the search domain, maintaining a rich population diversity. For the composition functions F22 and F28, the behavior of individuals shows their ability to optimize complex structural problems.

The third column in Fig. 7 shows the optimization history, which records the variation in the average fitness error of the best individual. For different types of functions, the convergence curves are significantly different. For F1, the optimization history convergence curve of GO decreases steadily, the best convergence accuracy reaches 1E-30, and its approximate optimal solution is obtained. For F7 and F10, there are a large number of local extrema in their search domain, and the convergence curve drops in a ladder shape, indicating that individuals switch back and forth in the process of falling into a local optimum and jumping out of the local optimum, which indicates that the algorithm is constantly adjusting the balance between exploration and exploitation. For F22 and F28, the decrease in the convergence curve of GO slows down, indicating that GO does not find a higher quality solution for the time being.

The fourth column of Fig. 7 is the diversity history, which shows the change in the average distance between individuals in each iteration or each evaluation, which is a kind of diversity change. Combined with the corresponding fitness error convergence curve, it is convenient to analyze the relationship between the individual average distance and fitness error. Although there are explicit methods such as ancestral trees [78], diversity has become a common method to measure the global exploration and local exploitation ability of algorithms. Diversity is usually divided into two types: the diversity of fitness and the diversity of spatial location [79]. Rich diversity means that there are obvious differences between individuals, which should be reflected in distance. In a local optimal region, individuals with different fitness values may be distributed, but individuals with different distances cannot be located in the same local optimal region. It is obvious that the diversity history of F1, F7, F22, and F28 decreases rapidly with the evolution of individuals, which indicates that the algorithm changes from global search to local exploitation, and the curve fluctuates continuously in the process, indicating that all individuals still maintain a certain distance to avoid the

stagnation of the whole population's search. In F10, the individual diversity history is slightly different, and the curve still maintains an overall downward trend under strong shocks, which fully proves the balance between exploration and exploitation of GO.

The last column of Fig. 7 records the change in the first dimension component of the first individual in the population, and its change degree is closely related to the complexity of the search domain. In the initial stage, all one-dimensional trajectory curves exhibit obvious oscillation, which indicates the global exploration behavior of the algorithm. For F1, F7, F22, and F28, the one-dimensional trajectory of an individual basically does not fluctuate in the middle of the algorithm iteration, which implies that the algorithm has changed from global exploration to local exploitation. However, for these functions, the sudden oscillations in the late iterations suggest that the algorithm does not search for stagnation. For F10, due to the complexity of its search domain, the individual's one-dimensional trajectory shows intermittent oscillation during the whole period, which indicates that GO is constantly adjusting the search mode to expect to find higher-quality solutions.

3.4.6. Algorithm complexity results and analysis

Based on the problem definition and evaluation criteria of CEC 2017, the time complexity of GO and other good algorithms is investigated, and the complexity calculation of these algorithms is based on 30-dimensional, 50-dimensional, and 100-dimensional problems. We explore three complexity-related parameters $T0$, $T1$, and $T2$. $T0$ represents the runtime of the test program as defined in CEC 2017. $T1$ is the time for the D dimension function F18 to perform 200,000 evaluations. The complete computation time for 200,000 evaluations of the same D-dimensional function F18 is $T2$. We perform this step five times to obtain $\overline{T2} = mean\,(T2)$. The time complexity of the algorithm is reflected by $T0$, $T1$, $\overline{T2}$, and $(\overline{T2} - T1)/T0$. In particular, the method of calculating complexity should be fair. For example, if an algorithm evaluates N individuals simultaneously or uses parallelism or vectorization, then all algorithms also use the same evaluation method. However, in the related literature, the complexity results of the algorithm clearly appear to be unfair. The complexity in this experiment is calculated using the MATLAB programming language, evaluating one individual at a time, using the most general programming style for all algorithms, without considering vectorization or parallelism.

Table 9 shows the results of the complexity comparison of the top 6 algorithms. The complexity of all the algorithms in the table increases with increasing dimension. In particular, for GSK, ASO, and AEFA, their complexity grows faster with increasing dimension. Although GO has no significant advantage in terms of time complexity, its complexity growth is less influenced by dimension. The highest complexity is ASO, which has a high complexity on any problem.

Table 9
The algorithm complexity results.

Dimension	Algorithm	$T0$	$T1$	$\overline{T2}$	$(\overline{T2} - T1)/T0$
$D = 30$	GO	0.040	1.4574	24.1770	567.9900
	GSK	0.040	1.4574	25.4040	598.6600
	SFS	0.040	1.4574	20.2856	470.0705
	DE	0.040	1.4574	17.9420	412.1150
	ASO	0.040	1.4574	40.2200	969.0600
	AEFA	0.040	1.4574	21.7840	508.1600
$D = 50$	GO	0.040	2.0313	26.4120	609.5200
	GSK	0.040	2.0313	33.9840	798.8200
	SFS	0.040	2.0313	21.8750	496.0925
	DE	0.040	2.0313	18.1820	403.7675
	ASO	0.040	2.0313	49.2850	1181.3000
	AEFA	0.040	2.0313	29.6620	690.7700
$D = 100$	GO	0.040	4.3801	33.0590	716.9700
	GSK	0.040	4.3801	61.5180	1428.4000
	SFS	0.040	4.3801	27.5560	579.3975
	DE	0.040	4.3801	21.8620	437.0475
	ASO	0.040	4.3801	71.1610	1669.5000
	AEFA	0.040	4.3801	54.6020	1255.5000

4. Application I : multiple sequence alignment

Highly nonlinear real-world problems are a major challenge for all optimization algorithms, especially when the dimension of the problem is increasing, leading to a rapid deterioration in the optimization performance of the algorithm, a phenomenon known as "dimensional catastrophe" [80]. This phenomenon is more common in large-scale global optimization (LSGO) problems. A high problem dimension implies a highly complex search domain structure, and the algorithm not only converges slowly but also quickly falls into local optima [81].

Multiple sequence alignment (MSA) of nucleotides or amino acids is a kind of LSGO problem, that belongs to the frontier of multidisciplinary problems and is one of the most important and challenging tasks in bioinformatics. MSA is used to analyze and discover sequence function, gene structure, and biological evolutionary information in biological sequences. With the novel coronavirus, MSA plays an important role in the traceability, analysis, and detection of the virus. A set of aligned sequences represents a conserved region and is responsible for the evolutionary relationships between species and their ancestors [82]. The method of aligning characters in the same column of different sequences is called sequence alignment. The empty bits are placed in the sequence to be aligned to match the sequence that results in the maximum number of matches. When the same column of multiple matching sequences is replaced with the same character, it is considered a match, otherwise, it is regarded as a mismatch. If two sequences are involved in the alignment, then it is a pairwise sequence alignment (PSA). When the alignment contains more than two sequences, the alignment operation is called MSA [83].

The progressive multiple sequence alignment Baum–Welch algorithm is used to solve the MSA problem. However, with the development of MSA technology, the traditional progressive MSA algorithm has many shortcomings, such as the ease with which one can fall into local optima and the inability to jump out of the local optima. The hidden Markov model (HMM) is a powerful tool for solving MSA problems, but the process of sequence comparison becomes increasingly difficult by training the HMM alone, so GO was introduced to train the HMM and solve this challenging problem.

4.1. Topology of HMM for MSA

An HMM is a special discrete-time finite state chain. As a statistical model, it can output a symbolic sequence; in the MSA problem, the symbolic sequence is a sequence of nucleotides or amino acids. HMM has q states $(Q_1, Q_2, \ldots, Q_q)$. Unlike the Markov model (MM), its hidden states are separated from the observed states. An HMM is represented by the quintet: $\lambda = (Q, O, \pi, A, B)$, or simply: $\lambda = (\pi, A, B)$. $Q = \{Q_1, Q_2, \ldots, Q_q\}$ is the finite set of hidden states, and q is the total number of hidden states. $O = \{O_1, O_2, \ldots, O_p\}$ is the finite set of observed states, and p is the total number of observed states. π is the initial probability matrix, i.e., the distribution probability of each state in the HMM at $time = 0$. $A = [a_{ij}]_{q\times q}$ is the transfer probability matrix, where $a_{ij}(a_{ij} \geq 0$ and $1 \leq i, j \leq n)$ denotes the probability that a state Q_i transitions to a state Q_j at a given moment, and the sum of probabilities in each row of the state transfer matrix is 1.

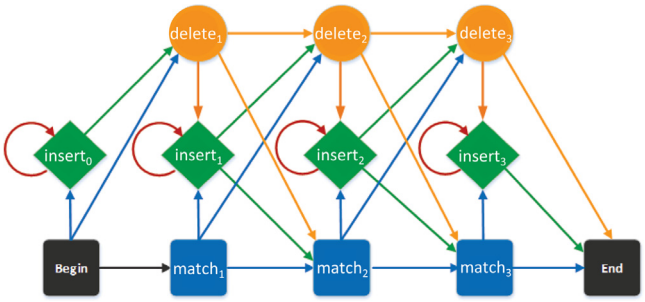


Fig. 8. Example of profile HMM of length 3 for the MSA problem.

$B = [b_j(k)]_{q\times p}$ is the observation probability matrix, where $b_j(k)$ ($b_j(k) \geq 0$ and $1 \leq j \leq q, 1 \leq k \leq m$) denotes the probability of observation O_k generated when state Q_i transitions to state Q_j at a certain moment, and the sum of the probabilities of each row of the observation matrix is 1.

When HMM is applied to MSA, the states of HMM include three basic states: match, insert, and delete, in addition to the start state (Begin) and the end state (End). Fig. 8 shows the simple topology of an HMM of length 3 used to solve the MSA problem.

From the start state to the end state, the HMM moves from one state to another with a certain period and outputs a symbol (observed value) at each move. Which state is transferred to and what symbols are output at the transfer are determined by the transfer probability (a_{ij}) and the output probability at the transfer ($b_j(k)$), respectively [84]. Notably, each transfer process yields a sequence of hidden states and a sequence of observations emitted from the sequence of hidden states.

4.2. GO trained HMM for MSA

For a given observation sequence (o), adjusting the λ of the HMM so that the probability $P(o|\lambda)$ of occurrence of the observation sequence (o) is maximized is the goal of training the HMM. The task of training an HMM is usually solved by statistical and re-estimation-based methods, such as the Baum–Welch algorithm. The Baum–Welch algorithm solves the parameter estimation problem of HMM to a certain extent, but since the Baum–Welch algorithm is a local optimization algorithm based on steepest gradient descent, it is very easy to become trapped in a local optimum during the training process. With the increasing improvement of metaheuristic techniques, metaheuristic algorithms are gradually being used to train HMMs with good results.

When GO is used to train the HMM, the individual is the parameter of the HMM, which consists of $\lambda = \{\pi, A, B\}$, and the objective function *Fitness* is used to evaluate the quality of the HMM (Eq. (10)). A sum-of-pairs (SOP) score was used to assess the quality of the paired sequence results (Eq. (11)). Since MSA is a maximization problem, a higher fitness value for an individual means a better parameter value for the HMM. Since the parameters of the HMM are all probabilities, all individuals in GO are normalized.

$$Fitness = -\frac{1}{num} \cdot \sum_{i=1}^{num} \frac{\log_2 P\left(o_i|\lambda\right)}{l_i} \tag{10}$$

where *num* is the number of entries in the sequence, and l_i is the length of the sequence o_i.

$$SOP = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} DIS\left(l_i, l_j\right) \tag{11}$$

where l_i is the compared sequence length, l_j is the sequence length to be compared, and *DIS* is the distance matrix between the two sequences. Note that the sequence used in this study has no reference sequence, so the *SOP* evaluation method without a reference sequence is used.

The following section describes the process of GO training the HMM and solving the MSA.

Step 1: Initialize HMM, read the gene sequence file, calculate the number of genes (*num*), the longest sequence length (L_{max}) and the compared sequence length ($L = 1.2 \times L_{max}$). After determining the length of the comparison sequence, the total number of parameters of the HMM is calculated ($D = 3 \times (3 \times L + 1) + 4 \times (2 \times L + 1)$), and the basic structure of the HMM is determined from this.

Step 2: Initiate the population of GO such that one individual corresponds to a set of HMM parameters, and each element of each individual corresponds to a parameter in the HMM. The population contains N groups. The dimension of each group is D, and the maximum number of evaluations of GO is set to *MaxFEs*.

Step 3: GO returns the processed parameters to the HMM and divides the D data in the individual into the corresponding parameters of the HMM according to the composition of the data in the HMM: π, A, B.

Step 4: The ith individual is evaluated according to Eq. (10). The computational principle of HMM is used to invoke the Viterbi algorithm to find the Viterbi sequence for each individual under the conditions of this HMM.

Step 5: The Viterbi algorithm calculates a series of implicit state sequences of insert, delete, and match states. According to the sequence matching criteria, the cryptic sequences were aligned with the observed sequences according to the three states of insertion, deletion, and matching. Matching and inserting operations insert characters at corresponding positions, while deleting operations insert symbols '.' at corresponding positions. Then, the aligned gene sequences are pruned. That means if a column is full of symbols '.', then this column is deleted.

Step 6: The quality of the alignment sequence obtained for the ith individual is evaluated using Eq. (11), and the evaluation result is returned as the objective function value. The following is a detailed description of the process. The aligned sequences are scored by the SOP function, and all sequences are compared one-to-one in turn. If the corresponding column characters of two sequences are the same, the score value is increased by 1. After traversing all the corresponding positions of the sequence, the alignment evaluation score of all the sequences is returned as the objective function value.

Step 7: Determine if the number of evaluations is *MaxFEs*, if not, return Step 3. If the condition is satisfied, return the optimal solution $\overrightarrow{gbestx}$.

Step 8: The $\overrightarrow{gbestx}$ is brought into the HMM, and the aligned gene sequence with the highest *SOP* value is obtained by backtracking with the Viterbi algorithm.

4.3. Gene sequence dataset and experimental setup

In December 2019, a new type of coronary pneumonia disease (COVID-19) caused by the SARS-CoV-2 virus broke out in Wuhan, Hubei Province, China [85]. According to studies, SARS-CoV-2 is closely related to SARS and bat coronavirus (RaTG13) [86]. For the virus traceability, the MSA technique becomes very meaningful. In this paper, 12 whole genome sequences of coronaviruses were selected on the basis of existing studies. Information about the genes is presented in Table 10. The first column of Table 10 is the reference sequence number; the second column is the length of the sequence, i.e., the problem dimension; the third column is the origin of the sequence; and the last column has other information

Table 10
The sequence details.

NCBI reference sequence	Sequence length	Source	Other information
NC045512.2	29 903 bp	China	Severe acute respiratory syndrome coronavirus 2 isolate
NC004718.3	29 751 bp	Canada	SARS coronavirus
MG772933.1	29 802 bp	China	Bat SARS-like coronavirus isolate
MZ202314.1	29 783 bp	South Africa	Severe acute respiratory syndrome coronavirus 2 isolate
MZ169911.1	29 903 bp	Brazil	Severe acute respiratory syndrome coronavirus 2 isolate
MZ318159.1	29 796 bp	India	Severe acute respiratory syndrome coronavirus 2 isolate
OU008837.1	29 836 bp	Switzerland	Severe acute respiratory syndrome coronavirus 2 isolate
MT956915.1	29 903 bp	Spain	Severe acute respiratory syndrome coronavirus 2 isolate
MW852494.1	29 666 bp	Italy	Severe acute respiratory syndrome coronavirus 2 isolate
ON672401.1	29 703 bp	New Zealand	Severe acute respiratory syndrome coronavirus 2 isolate
MZ310903.1	29 804 bp	America	Severe acute respiratory syndrome coronavirus 2 isolate
ON291775.1	29 702 bp	France	Severe acute respiratory syndrome coronavirus 2 isolate

Table 11
The sequence alignment results of each algorithm.

Algorithms	SOP score	Standard deviation	Normalized score	Average execution time (s)
GO	**761 532.0**	12 668.141	**9.867**	81 592.099
GSK	621 291.5	**399.515**	−38.168	98 915.946
SFS	625 387.0	2821.356	−3.953	114 776.203
DE	516 808.0	2251.428	−53.181	**74 501.061**
ASO	555 472.5	12 806.411	−6.330	87 696.604
AEFA	738 757.5	14 411.543	7.0928	118 044.991

about the sequence. It is worth noting that the above gene data were obtained from the NCBI database (https://www.ncbi.nlm.nih.gov).

Our experiment aligns 12 whole genome sequences using the top 6 performing algorithms in the function optimization experiment. Since this problem is a continuous optimization problem, the decision variables should be continuous variables. A decision variable corresponds to a set of HMM parameters, and its dimension (D) should be the total number of HMM parameters. Since all parameters are probability values, the value ranges of these parameters should be between [0,1]. If some component of the decision variable violates the boundary constraint, GO will be restricted to the boundary value. Notably, the decision variables should be normalized to match the HMM. The experimental parameters are set as follows: each algorithm runs 10 times, the population size (N) is set to 40, and the maximum number of evaluations ($MaxFEs$) is set to 40,000. The parameter settings of each algorithm will be consistent with the function test experiments, and these specific parameters can be found in Table 6.

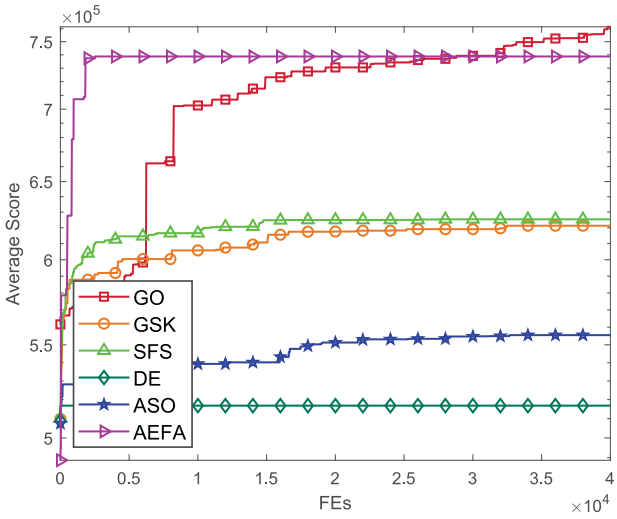


Fig. 9. The algorithm obtains an average score for the quality of the aligned sequences during HMM training.

4.4. Experiment results and analysis

Table 11 records the average SOP score, standard deviation (Std), normalized score, and average execution time for each algorithm. The normalized score will be given a specific mathematical description in Eq. (12). Fig. 9 records the convergence curves of each algorithm based on the average SOP scores.

$$Ns_i = \frac{SOP_i - \overline{SOP}}{\sigma_i} \tag{12}$$

where SOP_i is the average SOP score of the ith algorithm, $\overline{SOP}$ is the average SOP score of the algorithms and σ_i is the standard deviation of the SOP score of the ith algorithm.

Table 11 shows that GO has the highest average SOP score, which means that GO can train the HMM model better. Similarly, GO also obtained better results than other algorithms in terms of normalized score metrics. GSK has the smallest standard deviation, which means it has the most stable performance. DE has the smallest average execution time, which means that it can obtain results faster than other algorithms, but this does not guarantee the quality of the results. Fig. 9 clearly shows the historical scores of all algorithms in optimizing the HMM process. GSK and SFS, which performed well in the function optimization experiments, performed poorly in the training of HMM, while DE had the worst performance results. The poor performance of AEFA in the function optimization experiments but its outstanding performance in training the HMM reinforces the NFL theorem that each algorithm has its own range of application problems. GO has best performance, as its convergence curve is oscillating upward, which means it can escape from local optima. In contrast, AEFA's search behavior clearly shows the prematureness of the algorithm, so that it quickly falls into a local optimum and does not continue to converge.

The sequence alignment results of GO were visualized by means of sequence identity maps (Fig. 10) to better demonstrate the degree of sequence conservation and homology. Since the sequences are too long, three aligned sequence segments are intercepted for visualization and analysis. They are sequence segment 1–80, sequence segment 1000–1080, and sequence segment 3000–3080. In the sequence logo map, the accumulation of bases

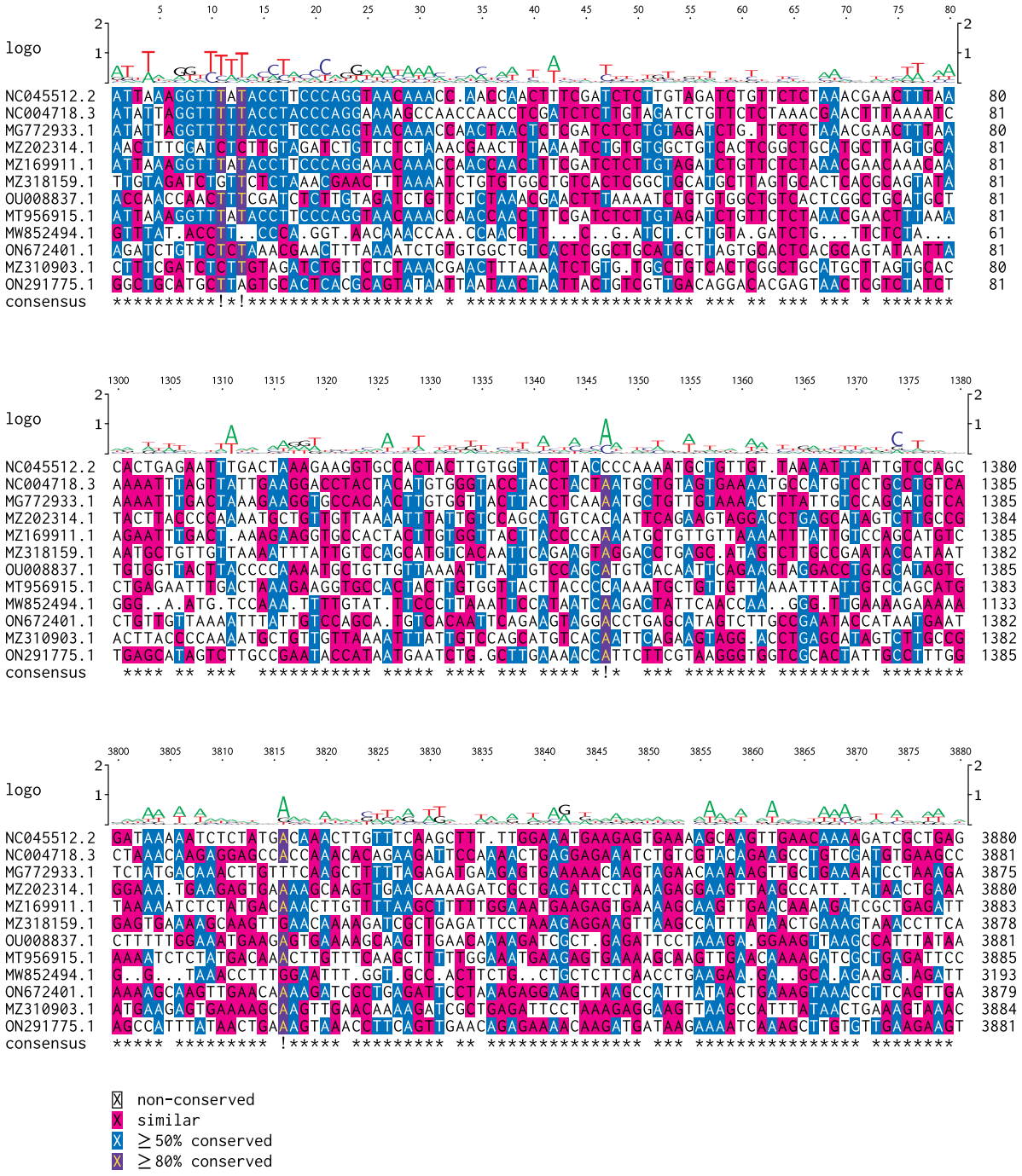


Fig. 10. The best resulting alignments generated by the HMM trained with GO for the sequences.

at each position reflects the consistency of the bases at that position. The size of the graphic character corresponding to a base is proportional to the frequency of occurrence of that base at that position. If only one graphic letter appears, it means that the position is very conserved. The maximum value of the y-axis of sequence logos is $log_2 4 = 2$ bits, where 4 is the number of base species of DNA. In the consensus position, the symbol '!' indicates that the frequency of the same base at a position is $\geq$ 80%, the symbol '*' indicates that the base species at a position is relatively similar, and the blank indicates that all sequences at that position have very low homology. As seen from Fig. 10, there are obvious sequence features at most positions of the sequence segments, and there is a high degree of consistency among some of the point sequences, which implies that there is some homology among the 12 sequences.

In Table 12, the similarity and identity (expressed as percentages) between the two alignment sequences are shown. The similarity is the number of residues that are identical or similar between two sequences as a percentage of the total length of the sequences. The identity is defined as the number of residues that are identical between two sequences as a percentage of the total sequence length. It is worth noting that the sequences are not compared with themselves, so the corresponding position in the table is indicated as '–'. It can be seen that the similarity of all sequences is above 30%, while their identity is mainly distributed between 20% and 30%. The unrooted evolutionary tree generated from aligned sequences can show the relative relationship

Table 12
The similarity and identity between aligned sequences.

	NC045512.2	NC004718.3	MG772933.1	MZ202314.1	MZ169911.1	MZ318159.1	OU008837.1	MT956915.1	MW852494.1	ON672401.1	MZ310903.1	ON291775.1
NC045512.2	—	35.8	55.5	35.4	50.6	49.3	37.9	44.4	42.4	49.3	44.4	40.7
NC004718.3	20.9	—	48.7	48.7	42.6	42.6	47.5	52.4	35.8	57.3	54.8	45.1
MG772933.1	37.0	35.3	—	48.7	47.5	39.0	46.2	39.0	40.2	41.4	43.9	48.7
MZ202314.1	13.9	35.0	33.7	—	42.5	50.0	47.4	43.7	45.4	47.5	50.0	57.5
MZ169911.1	33.3	23.1	26.8	30.0	—	43.9	36.2	48.7	44.7	48.7	53.6	50.0
MZ318159.1	30.8	23.1	21.9	28.7	28.0	—	42.5	42.6	35.8	41.4	41.4	45.1
OU008837.1	18.9	27.5	30.0	29.4	18.7	30.0	—	42.5	40.9	52.5	50.0	50.0
MT956915.1	29.6	39.0	23.1	25.0	32.9	26.8	28.7	—	34.3	54.8	41.4	46.3
MW852494.1	27.2	16.4	25.3	28.7	31.3	29.8	28.7	20.8	—	35.8	38.8	46.2
ON672401.1	32.0	39.0	26.8	28.7	26.8	28.0	32.5	35.3	22.3	—	47.5	48.7
MZ310903.1	29.6	34.1	28.0	30.0	32.9	25.6	28.7	23.1	13.4	26.8	—	50.0
ON291775.1	18.5	28.0	36.5	37.5	31.7	26.8	31.2	26.8	26.8	29.2	31.7	—

% identity

% similarity

between genes, and the distance between nodes indicates the proximity of genes to each other. By evolutionary tree analysis (Fig. 11), the 12 genome-wide strains used in this experiment may all belong to the same viral species.

Through this experiment, the powerful optimization performance of GO in solving the HMM-based MSA problem has been well demonstrated, and GO can be used as an alternative to other state-of-the-art metaheuristic algorithms. The aligned sequences obtained by GO optimization confirm that SARS-CoV-2, bat coronavirus, SARS, and other SARS-CoV-2 isolates are closely related.

5. Application II : multilevel thresholding image segmentation

Image segmentation is an important part of image processing. It divides the image into several regions with the same properties and extracts the interesting objects from them. Although there are many image segmentation techniques, the threshold-based segmentation method is relatively popular because of its simplicity, accuracy, and stability [87].

Generally, threshold segmentation techniques are classified into bilevel and multilevel types. Images containing more than two homogeneous regions cannot be segmented using the bilevel thresholding technique, so the multilevel thresholding technique is more generalizable. Parametric methods are usually used to obtain the optimal threshold value. Among parametric methods, the Otsu method [88] can be used to separate regions with close pixel intensities by maximizing the interclass variance, or homogeneous regions can be found by using Kapur's method [89] on the basis of maximizing the interclass variance [90]. However, the two methods mentioned above have a significant drawback: As the number of thresholds increases, their time complexity also increases exponentially. As a result, a technique using metaheuristics combined with traditional threshold segmentation methods became popular, and over time, the number of successful cases became countless.

To verify the ability of GO to handle real-world problems, GO was combined with Kapur's entropy method and used to perform multilevel threshold image segmentation. By comparing its optimization performance with that of other state-of-the-art metaheuristic algorithms and analyzing the optimization metrics, it is shown that GO can outperform other algorithms for multilevel thresholding image segmentation problems.

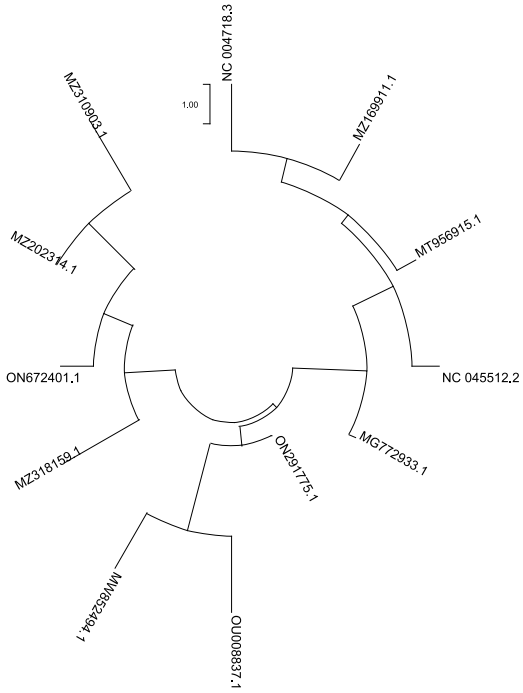


Fig. 11. The rootless evolutionary tree.

This section is organized as follows. Section 5.1 introduces the mathematical model of Kapur's entropy method. Section 5.2 presents the steps of GO to solve the multilevel thresholding image segmentation problem. Section 5.3 conducts experiments and discussion.

5.1. Kapur's entropy method

Kapur's entropy method segments the image by finding a series of optimal thresholds, and this process is achieved by maximizing the objective function (Eq. (13)). For an 8-bit grayscale image, the threshold (t) takes values between 0 and 255, and the level of thresholding (K) takes values between 1 and 255. When the threshold level is K, the image is segmented by $K-$

1 thresholds ($(t_1, t_2, \ldots, t_{K-1})$) segmented into $E_1, E_2, E_3, \ldots, E_K$ parts.

$$F(t_1, t_2, \ldots, t_{K-1}) = E_1 + E_2 + E_3 + \cdots + E_K \tag{13}$$

$$E_1 = -\sum_{i=0}^{t_1-1} \frac{X_i}{T_1} \cdot \ln \frac{X_i}{T_1}, \quad T_1 = \sum_{i=0}^{t_1-1} X_i, X_i = \frac{N_i}{T_1} \tag{14}$$

$$E_2 = -\sum_{i=t_1}^{t_2-1} \frac{X_i}{T_2} \cdot \ln \frac{X_i}{T_2}, \quad T_2 = \sum_{i=t_1}^{t_2-1} X_i, X_i = \frac{N_i}{T_2} \tag{15}$$

$$E_3 = -\sum_{i=t_2}^{t_3-1} \frac{X_i}{T_3} \cdot \ln \frac{X_i}{T_2}, \quad T_3 = \sum_{i=t_2}^{t_3-1} X_i, X_i = \frac{N_i}{T_3} \tag{16}$$

$$E_K = -\sum_{i=t_{K-1}}^{255} \frac{X_i}{T_K} \cdot \ln \frac{X_i}{T_K}, \quad T_K = \sum_{i=t_{K-1}}^{255} X_i, X_i = \frac{N_i}{T_K} \tag{17}$$

where the number of pixels with grayscale value i is N_i, T_K is the total number of pixels in the Kth part, and E_K is the objective function for the evaluation of the Kth part.

5.2. Optimization process of GO

The working principle of GO for solving multilevel thresholding image segmentation problems is described as follows.

Step 1: Read images in TIFF format.

Step 2: Convert the read-in image into a grayscale image and obtain its grayscale histogram.

Step 3: Transfer image data to GO.

Step 4: Initialize GO. Notably, the composition of individual i is a series of thresholds in ascending order, and the thresholds are in the form of integers with values in the range [0,255].

Step 5: GO processes the thresholds according to its own operation mechanism and passes the processed thresholds into the objective function.

Step 6: The objective function evaluates the incoming thresholds and returns the evaluation results to GO. The specific evaluation steps of the objective function are described based on Eqs. (13)–(17). When a set of thresholds $(t_1, t_2, \ldots, t_{K-1})$ in ascending order are passed into the objective function, all thresholded segmented Kpart images are evaluated separately. First, the image segmentation quality (entropy) of the $[0, t_1]$ part is evaluated, then the image quality of the $[t_1 + 1, t_2]$ part is evaluated, and so on, until the image segmentation quality of the $[t_{K-1}, 255]$ part is evaluated. Then, the entropy of the K part images is summed up to obtain the quality evaluation result of this set of threshold segmentation images, which is returned to GO as the objective function value. GO processes the thresholds based on the evaluation results.

Step 7: Determine if the termination condition is met. If the termination condition is not met, return to step 5.

Step 8: GO outputs the optimal thresholds and segments the image based on these thresholds.

5.3. Experiments and discussion

In this subsection, SFS, GSK, DE, ASO, and AEFA are chosen as competing algorithms for GO in the optimization of the multilevel thresholding image segmentation problem, and these algorithms perform best in the function optimization experiments above. The optimization objective is accomplished by maximizing the objective function. Since the problem is a discrete integer optimization problem and the objective function is evaluated from low to high thresholds, the decision variables should be an ascending integer array, and their values should be integers in the range [0,255]. If the value of a dimensional component of the decision variable violates the boundary, GO handles it by restricting it to the boundary value. If its value is not an integer, the rounding principle is applied. The uniform parameters of the experiment were set as $MaxFEs = 45\,000$, $N = 30$, $K = 2/4/6/8/10/15/20$, and $Run = 20$. All the specific parameter settings of the algorithms involved in the experiments are the relevant parameter settings from the function optimization experiments (in Table 6).

5.3.1. Image benchmark test suite

A set of test images was selected from the USC-SIPI image database (https://sipi.usc.edu/database/) for image segmentation quality evaluation. They are the cameraman, the pepper, the fighter, the woman, the sailboat, the monkey, the tank, and the house, in order. Fig. 12 shows their grayscale images and grayscale histograms, where the grayscale image reflects the lightness and darkness of the pixels, and the grayscale histogram shows the distribution of different grayscale values.

5.3.2. Quality evaluation metrics

Seven metrics were used to evaluate the optimization results of the algorithms involved in the experiments. F_{ave} is the average fitness value, and the larger it is, the higher the quality of the threshold obtained by the algorithm. The standard deviation (*Std*) reflects the stability of the algorithm when optimizing the problem, and the more *Std* tends to 0, the more stable the algorithm is. *Time* is the average running time of the algorithm. The rest of the metrics are described in detail below.

The peak signal-to-noise ratio (*PSNR*) [91] is used to express the similarity between the original image and the segmented image, which is expressed in decibels and in logarithmic form. Eq. (18) gives the mathematical description of *PSNR*.

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right) \tag{18}$$

$$MSE = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} |A(i,j) - S(i,j)|}{m \times n} \tag{19}$$

where $n \times m$ represents the image size; 255 represents the maximum pixel value; *MSE* represents the mean square error; $A(i, j)$ represents the original image, and $S(i, j)$ represents the segmented image. Note, that the larger *PSNR* is, the better, while the smaller *MSE* is, the better.

The maximum absolute error (*MAE*) [92] denotes the maximum error between the original image and the segmented image, and Eq. (20) is its mathematical description.

$$MAE = \max(|A - S|) \tag{20}$$

where A is the original image, S is the segmented image, and max is the maximum value function. Usually, the smaller *MAE* is, the better.

The structured similarity index metric (*SSIM*) [91] measures the similarity of two images in terms of structure. Its mathematical model is described by Eq. (21).

$$SSIM(A, S) = \frac{(2\mu_A\mu_S + a)(2\sigma_{AS} + b)}{(\mu_A^2 + \mu_S^2 + a)(\sigma_A^2 + \sigma_S^2 + b)} \tag{21}$$

where μ_A is the mean of A, μ_S is the mean of S, σ_A^2 is the standard deviation of A, σ_S^2 is the standard deviation of S, σ_{AS} is the covariance of A and S, and a and b are constants, which are 0.001 and 0.003 respectively. Note that a larger *SSIM* means better performance.

The feature similarity index measure (*FSIM*) is used to check the local quality of an image and consists of phase congruency

(a) Cameraman (b) Cameraman image histogram (c) Pepper (d) Pepper image histogram

(e) Fighter (f) Fighter image histogram (g) Woman (h) Woman image histogram

(i) Sailboat (j) Sailboat image histogram (k) Monkey (l) Monkey image histogram

(m) Tank (n) Tank image histogram (o) House (p) House image histogram

Fig. 12. The test image and the image histogram.

(*PC*) and gradient magnitude (*GM*). *FSIM* is calculated by Eqs. (22) to (26).

$$FSIM\ (x) = \frac{\sum_{x\epsilon\Omega} S_L\ (x)\ PC_m\ (x)}{\sum_{x\varepsilon\Omega} PC_m\ (x)} \tag{22}$$

$$PC_m\ (x) = \max\left(PC_1\ (x)\ ,\ PC_2\ (x)\right) \tag{23}$$

$$S_L\ (x) = \left(S_{PC}\ (x)\right)^{\alpha} \left(S_G\ (x)\right)^{\beta} \tag{24}$$

$$S_{PC}\ (x) = \frac{2PC_1\ (x)\ PC_2\ (x) + T_1}{\left(PC_1\ (x)\right)^2 \left(PC_2\ (x)\right)^2 + T_1} \tag{25}$$

$$S_G\ (x) = \frac{2G_1\ (x)\ G_2\ (x) + T_2}{\left(G_1\ (x)\right)^2 \left(G_2\ (x)\right)^2 + T_2} \tag{26}$$

where Ω denotes the range of pixel values, $S_L(x)$ denotes the similarity quotient, and $PC_1(x)$ and $PC_2(x)$ are the phase consistency features of two weighting functions. The gradient (G) local similarity index is denoted by S_G, the local similarity index of PC is denoted by S_{PC}, and T_1, T_2, α and β are constants. Note that a larger *FSIM* means a higher quality segmentation of the image.

5.3.3. Results discussion and analysis

Based on 8 test images, GO, GSK, SFS, DE, ASO, and AEFA are selected for image segmentation, and their parameter settings follow the settings in the function test. The optimization results of each algorithm for different threshold levels are given in Table 13. Fig. 13 shows the image segmentation results of GO for different threshold levels. It is obvious that as the number of threshold levels increases, the segmented image shows more details.

Fig. 13. The image segmentation results of GO at different threshold levels.

Table 13 listed the nine metrics used to evaluate the performance of the algorithms, and all the algorithms involved in the test have some effect on this application. From Table 13, it can be seen that the performance of all algorithms differs very little for small threshold problems, and the performance gap between algorithms gradually increases as the number of thresholds increases. GO shows excellent optimization results on all tested images, and still outperforms other algorithms in general on *PSNR*, *MSE*, *MAE*, *SSIM*, and *FSIM*. However, the average running time (*Time*) and standard deviation (*Std*) of GO have some disadvantages compared with those of other algorithms.

6. Conclusion

This study proposes a new metaheuristic algorithm called the growth optimizer (GO). It is a population-based algorithm inspired by the observation of society membership structures and the behavior of social members themselves. GO simulates the learning and reflection mechanisms involved in the growth process of the individual and mathematically models them into two phases: the learning phase and the reflection phase. Unlike most metaheuristic algorithms, GO's collaborative search model provides the individual with search information in multiple directions. It self-adaptively balances directional information using the

Table 13
Performance comparison on test images.

Test image	K	Algorithms	F_{ave}	*Std*	*Time*	*PSNR*	*MSE*	*MAE*	*SSIM*	*FSIM*
Cameraman	2	GO	**1.219E+01**	9.324E−03	1.990	13.626	2821.715	127	0.529	0.678
		GSK	**1.219E+01**	5.468E−15	1.817	**16.463**	**1468.314**	**118**	**0.841**	**0.806**
		SFS	**1.219E+01**	**5.467E−15**	1.154	13.626	2821.715	127	0.529	0.678
		DE	**1.219E+01**	5.468E−15	1.011	13.477	2919.717	128	0.525	0.676
		ASO	**1.219E+01**	5.468E−15	1.202	13.535	2881.428	128	0.526	0.676
		AEFA	1.212E+01	1.226E−02	**0.895**	12.808	3406.142	135	0.504	0.661
	4	GO	**1.844E+01**	1.229E−02	1.799	**20.163**	**626.350**	57	0.663	0.829
		GSK	1.840E+01	2.313E−02	1.256	20.153	627.728	57	0.682	0.841
		SFS	1.840E+01	1.424E−02	1.402	20.060	641.912	**56**	**0.692**	**0.852**
		DE	1.843E+01	**0.000E+00**	1.258	20.115	633.286	**56**	0.675	0.837
		ASO	1.843E+01	**0.000E+00**	1.378	20.076	639.034	**56**	0.677	0.838
		AEFA	1.803E+01	2.050E−01	**1.058**	17.164	1249.300	97	0.584	0.765
	6	GO	2.374E+01	**2.982E−02**	2.211	19.546	721.956	56	0.722	0.886
		GSK	2.365E+01	5.851E−02	2.387	20.953	522.171	51	0.732	0.891
		SFS	2.372E+01	4.554E−02	1.648	**23.191**	**311.858**	57	**0.733**	**0.896**
		DE	2.368E+01	5.742E−02	1.451	21.621	447.696	**44**	0.710	0.878
		ASO	**2.376E+01**	7.256E−02	1.522	22.230	389.092	48	0.712	0.880
		AEFA	2.317E+01	1.904E−01	**1.252**	21.677	442.002	**44**	0.699	0.875
	8	GO	2.863E+01	**6.199E−02**	2.465	**25.021**	**204.633**	**34**	0.733	0.917
		GSK	2.839E+01	1.273E−01	1.619	23.133	316.083	35	0.724	0.897
		SFS	2.853E+01	7.113E−02	2.007	24.852	212.732	37	0.738	0.911
		DE	2.837E+01	1.256E−01	1.675	22.529	363.226	37	0.715	0.895
		ASO	**2.871E+01**	9.550E−01	1.766	24.610	224.965	**34**	**0.742**	**0.920**
		AEFA	2.748E+01	3.857E−01	**1.458**	23.947	262.073	55	0.707	0.884
	10	GO	3.290E+01	**1.168E−01**	3.161	24.760	**163.688**	**31**	0.751	**0.929**
		GSK	3.256E+01	1.707E−01	1.943	**25.848**	169.168	34	0.744	0.924
		SFS	3.279E+01	1.481E−01	2.151	24.887	211.025	32	0.734	0.918
		DE	3.258E+01	2.308E−01	1.852	25.385	188.183	32	0.744	0.923
		ASO	**3.311E+01**	1.284E−01	1.834	25.845	169.278	37	**0.754**	0.928
		AEFA	3.097E+01	5.144E−01	**1.746**	21.148	499.158	46	0.729	0.887
	15	GO	4.182E+01	2.478E−01	3.668	**28.561**	**90.568**	**22**	0.790	**0.957**
		GSK	4.127E+01	2.927E−01	3.713	26.663	140.197	28	0.779	0.947
		SFS	4.181E+01	**2.008E−01**	2.802	27.620	112.485	23	**0.860**	0.935
		DE	4.094E+01	4.316E−01	2.401	28.150	99.551	31	0.785	0.951
		ASO	**4.238E+01**	2.556E−01	2.446	26.514	145.091	31	0.749	0.935
		AEFA	3.870E+01	1.078E+00	**2.325**	27.554	114.210	27	0.786	0.938
	20	GO	4.960E+01	**2.283E−01**	5.820	**31.116**	**50.295**	**18**	0.900	**0.965**
		GSK	4.793E+01	4.582E−01	3.477	30.757	54.619	23	**0.902**	0.962
		SFS	4.912E+01	4.024E−01	3.822	30.162	62.642	19	0.900	0.962
		DE	4.720E+01	6.318E−01	3.380	29.425	74.237	23	0.842	0.958
		ASO	**4.961E+01**	5.965E−01	3.214	28.959	82.635	19	0.786	0.960
		AEFA	4.503E+01	9.479E−01	**2.676**	28.526	91.293	30	0.869	0.941
Pepper	2	GO	**1.263E+01**	7.925E−04	1.971	**16.489**	**1459.431**	84	**0.636**	**0.738**
		GSK	**1.263E+01**	**5.468E−15**	1.865	16.445	1474.447	81	0.620	0.735
		SFS	**1.263E+01**	**5.468E−15**	1.166	16.445	1474.447	81	0.628	0.735
		DE	**1.263E+01**	**5.468E−15**	1.024	16.426	1480.649	**80**	0.624	0.734
		ASO	**1.263E+01**	**5.468E−15**	1.263	16.426	1480.649	**80**	0.624	0.734
		AEFA	**1.263E+01**	4.178E−03	**0.882**	16.450	1472.889	81	0.623	0.733
	4	GO	1.896E+01	2.358E−02	1.812	18.044	1020.262	58	0.714	0.808
		GSK	1.896E+01	2.331E−02	1.978	18.459	927.318	63	0.712	0.808
		SFS	1.895E+01	1.805E−02	1.440	18.456	927.881	62	0.714	0.810
		DE	**1.898E+01**	**2.284E−03**	1.235	18.436	932.111	63	0.714	0.810
		ASO	1.892E+01	5.985E−02	1.358	18.038	1021.549	79	0.675	0.789
		AEFA	1.848E+01	1.576E−01	**1.124**	**19.428**	**741.789**	**55**	**0.761**	**0.845**
	6	GO	**2.447E+01**	**6.409E−02**	2.149	21.206	492.556	47	0.807	0.872
		GSK	2.431E+01	9.772E−02	2.392	21.092	505.739	**43**	0.807	0.876
		SFS	2.438E+01	6.110E−01	1.670	**21.475**	462.992	54	**0.822**	**0.877**
		DE	2.438E+01	7.553E−02	1.435	20.771	544.528	57	0.790	0.866
		ASO	2.438E+01	1.234E−01	1.574	21.433	**467.445**	47	0.806	0.873
		AEFA	2.356E+01	3.683E−01	**1.252**	21.183	495.245	65	0.784	0.847
	8	GO	**2.923E+01**	**9.185E−02**	2.518	23.412	296.385	**37**	0.851	0.910
		GSK	2.897E+01	1.201E−01	2.521	23.116	317.287	**37**	0.840	0.906
		SFS	2.914E+01	1.092E−01	1.942	**24.276**	**242.924**	40	0.859	0.920
		DE	2.895E+01	1.920E−01	1.663	22.151	396.285	45	0.846	0.886
		ASO	2.922E+01	1.851E−01	1.733	23.747	274.374	41	**0.861**	**0.930**
		AEFA	2.760E+01	5.924E−01	**1.440**	21.716	437.983	48	0.820	0.884
	10	GO	**3.349E+01**	**9.173E−02**	2.412	24.481	231.707	39	0.879	0.928
		GSK	3.298E+01	1.911E−01	2.587	24.629	223.964	**31**	0.867	0.925

(*continued on next page*)

Table 13 (*continued*).

Test image	*K*	Algorithms	F_{ave}	*Std*	*Time*	*PSNR*	*MSE*	*MAE*	*SSIM*	*FSIM*
		SFS	3.331E+01	1.687E−01	2.157	**25.890**	**167.520**	38	**0.901**	**0.942**
		DE	3.300E+01	1.805E−01	1.950	24.382	237.055	34	0.879	0.934
		ASO	3.332E+01	3.640E−01	1.897	24.997	205.786	43	0.881	0.934
		AEFA	3.124E+01	5.017E−01	**1.742**	20.927	525.313	59	0.825	0.871
	15	GO	**4.217E+01**	2.350E−01	3.937	**29.251**	**77.265**	**25**	**0.941**	0.971
		GSK	4.138E+01	3.539E−01	3.961	26.776	136.594	39	0.900	0.946
		SFS	4.204E+01	**2.190E−01**	2.602	28.533	91.164	27	**0.941**	0.969
		DE	4.108E+01	3.982E−01	2.689	26.553	143.800	27	0.915	0.956
		ASO	4.221E+01	6.296E−01	2.454	28.235	97.626	29	0.927	**0.975**
		AEFA	3.915E+01	7.226E−01	**2.260**	25.889	167.562	32	0.906	0.934
	20	GO	**4.985E+01**	**3.613E−01**	4.737	**31.066**	**50.868**	**18**	**0.967**	**0.982**
		GSK	4.830E+01	4.115E−01	5.173	28.329	95.528	30	0.921	0.970
		SFS	4.918E+01	4.139E−01	3.783	30.162	62.6422	19	0.900	0.962
		DE	4.747E+01	6.389E−01	3.820	28.451	92.888	25	0.950	0.968
		ASO	4.939E+01	9.241E−01	2.948	30.206	62.007	21	0.961	**0.982**
		AEFA	4.532E+01	1.168E+00	**2.638**	27.747	109.245	28	0.924	0.948
Fighter	2	GO	**1.221E+01**	4.933E−03	1.987	**16.458**	**1550.349**	**95**	**0.814**	**0.795**
		GSK	**1.221E+01**	**3.645E−15**	2.130	15.859	1687.300	101	0.810	0.790
		SFS	**1.221E+01**	**3.645E−15**	1.172	15.859	1687.300	101	0.810	0.790
		DE	**1.221E+01**	**3.645E−15**	0.996	15.887	1676.296	102	0.809	0.790
		ASO	**1.221E+01**	**3.645E−15**	1.234	15.951	1651.820	101	0.810	0.790
		AEFA	1.220E+01	1.102E−02	**0.882**	16.450	1572.890	101	0.623	0.733
	4	GO	1.859E+01	9.431E−02	1.875	18.865	844.425	**60**	0.868	0.859
		GSK	1.860E+01	4.797E−02	2.058	17.762	1088.454	71	0.862	0.852
		SFS	1.863E+01	3.621E−01	1.398	18.597	898.275	68	0.870	0.860
		DE	**1.867E+01**	**2.136E−02**	1.310	**18.939**	**830.236**	69	0.865	0.859
		ASO	1.853E+01	1.211E−01	1.482	18.263	970.042	74	0.868	0.865
		AEFA	1.813E+01	1.431E−01	**1.104**	18.458	927.411	66	**0.884**	**0.886**
	6	GO	2.340E+01	**7.087E−02**	2.114	19.624	709.110	**52**	**0.918**	**0.917**
		GSK	2.383E+01	1.090E−01	2.321	20.609	565.137	66	0.888	0.876
		SFS	**2.393E+01**	8.243E−02	1.680	18.901	837.482	55	0.893	0.887
		DE	2.388E+01	1.312E−01	1.635	19.664	702.496	55	0.891	0.879
		ASO	2.389E+01	1.435E−01	1.676	20.214	618.915	55	0.904	0.903
		AEFA	2.306E+01	3.298E−01	**1.275**	**22.828**	**339.052**	61	0.895	0.915
	8	GO	**2.887E+01**	1.556E−01	3.437	**24.188**	**247.900**	47	**0.935**	**0.933**
		GSK	2.842E+01	1.604E−01	3.612	23.516	289.405	**37**	0.923	0.932
		SFS	2.874E+01	**1.317E−01**	1.902	22.791	341.979	43	0.914	0.916
		DE	2.844E+01	1.987E−01	1.794	21.664	443.255	44	0.917	0.911
		ASO	2.855E+01	2.185E−01	1.932	17.682	1108.789	62	0.878	0.875
		AEFA	2.705E+01	3.689E−01	**1.434**	23.088	319.349	73	0.881	0.909
	10	GO	3.315E+01	**1.826E−01**	2.860	24.276	242.955	33	0.907	0.930
		GSK	3.239E+01	1.836E−01	2.997	24.808	214.936	42	0.919	0.931
		SFS	3.295E+01	2.228E−01	2.381	24.707	219.981	43	**0.941**	0.933
		DE	**4.065E+01**	4.863E−01	2.640	**27.79**9	**107.935**	42	0.936	0.959
		ASO	3.287E+01	4.819E−01	2.637	26.376	149.775	**31**	0.938	**0.956**
		AEFA	3.041E+01	5.647E−01	**1.767**	21.858	423.879	43	0.908	0.908
	15	GO	**4.207E+01**	3.142E−01	3.677	**28.402**	**93.938**	29	**0.943**	**0.964**
		GSK	4.115E+01	4.647E−01	**2.106**	25.876	168.060	29	0.913	0.944
		SFS	4.177E+01	**2.845E−01**	2.585	25.391	187.938	**26**	0.919	0.941
		DE	4.061E+01	4.600E−01	3.531	26.499	145.590	39	0.927	0.943
		ASO	4.182E+01	5.325E−01	2.658	29.095	95.092	37	0.962	0.957
		AEFA	3.856E+01	7.442E−01	2.221	26.670	139.984	44	**0.943**	0.942
	20	GO	**4.967E+01**	4.410E−01	3.971	30.158	62.702	**20**	0.962	0.977
		GSK	4.767E+01	**3.076E−01**	4.298	28.089	100.964	26	0.950	0.959
		SFS	4.908E+01	3.848E−01	3.805	27.892	105.659	21	0.947	0.954
		DE	4.649E+01	6.226E−01	3.689	23.287	305.032	33	0.901	0.910
		ASO	4.882E+01	6.374E−01	2.978	**31.185**	**49.494**	25	**0.977**	**0.983**
		AEFA	4.430E+01	1.027E+00	**2.632**	29.290	76.570	23	0.955	0.969
Woman	2	GO	**1.230E+01**	1.884E−02	1.715	**14.412**	**2354.256**	**104**	**0.457**	**0.646**
		GSK	**1.230E+01**	**5.468E−15**	1.876	14.383	2369.915	105	0.450	0.641
		SFS	**1.230E+01**	**5.468E−15**	1.160	14.383	2369.915	105	0.450	0.641
		DE	**1.230E+01**	**5.468E−15**	1.099	14.257	2440.127	106	0.446	0.638
		ASO	**1.230E+01**	**5.468E−15**	1.302	14.307	2411.904	106	0.446	0.637
		AEFA	**1.230E+01**	3.622E−03	**0.923**	14.383	2369.915	105	0.450	0.641
	4	GO	1.807E+01	1.620E−02	2.247	16.663	1402.212	76	0.552	0.748
		GSK	1.806E+01	1.988E−01	2.387	16.592	1425.355	74	0.550	0.747
		SFS	1.807E+01	1.735E−01	1.423	16.548	1439.770	77	0.549	0.746
		DE	**1.809E+01**	**1.191E−02**	1.311	16.610	1419.240	**72**	0.548	0.746
		ASO	1.808E+01	3.017E−02	1.507	16.567	1433.371	74	0.549	0.750

(*continued on next page*)

Table 13 (*continued*).

Test image	*K*	Algorithms	F_{ave}	*Std*	*Time*	*PSNR*	*MSE*	*MAE*	*SSIM*	*FSIM*
		AEFA	1.787E+01	8.993E−02	**1.067**	**19.897**	**665.895**	76	**0.714**	**0.758**
	6	GO	**2.338E+01**	**3.928E−02**	2.104	21.778	431.757	**47**	0.760	0.814
		GSK	2.323E+01	6.591E−02	2.340	17.603	1129.255	69	0.569	0.788
		SFS	2.334E+01	5.841E−02	1.668	**21.958**	**414.223**	**47**	**0.766**	**0.818**
		DE	2.328E+01	1.024E−01	1.591	20.497	579.998	50	0.758	0.808
		ASO	**2.338E+01**	1.037E−01	1.717	21.680	441.638	48	0.757	0.817
		AEFA	2.280E+01	2.059E−01	**1.265**	20.840	535.937	49	0.726	0.787
	8	GO	**2.815E+01**	1.253E−01	3.094	**23.036**	**323.184**	39	0.786	0.830
		GSK	2.788E+01	**1.165E−01**	3.510	22.240	388.240	44	0.775	0.828
		SFS	**2.815E+01**	1.259E−01	1.892	22.634	354.521	41	0.766	0.827
		DE	2.786E+01	1.357E−01	1.870	22.211	390.786	40	0.782	0.833
		ASO	**2.815E+01**	2.435E−01	2.048	22.574	359.446	**36**	**0.802**	**0.843**
		AEFA	2.697E+01	3.092E−01	**1.467**	21.324	479.381	46	0.759	0.803
	10	GO	**3.244E+01**	1.791E−01	3.323	24.820	214.037	**32**	0.827	0.880
		GSK	3.197E+01	2.025E−01	3.585	20.153	627.784	56	0.720	0.832
		SFS	3.236E+01	**1.346E−01**	3.355	24.620	224.405	38	0.803	0.865
		DE	3.205E+01	2.302E−01	2.058	23.032	323.533	34	0.803	0.855
		ASO	3.234E+01	1.959E−01	2.344	**25.849**	**169.119**	37	**0.841**	**0.895**
		AEFA	3.051E+01	4.916E−01	**1.730**	20.231	616.634	60	0.669	0.797
	15	GO	4.138E+01	2.353E−01	3.633	**27.985**	**103.408**	**28**	0.860	**0.916**
		GSK	4.046E+01	3.131E−01	4.034	27.895	105.576	29	**0.870**	0.903
		SFS	3.239E+01	**1.682E−01**	**2.135**	25.209	195.946	40	0.805	0.871
		DE	4.026E+01	3.229E−01	2.716	26.276	153.265	29	0.851	0.895
		ASO	**4.145E+01**	4.635E−01	2.781	25.638	177.542	29	0.850	0.896
		AEFA	3.830E+01	9.884E−01	2.209	25.733	173.683	32	0.832	0.867
	20	GO	**4.909E+01**	4.025E−01	4.942	28.180	98.881	**20**	0.879	0.924
		GSK	4.730E+01	5.090E−01	5.554	27.884	105.854	28	0.852	0.910
		SFS	4.842E+01	**3.027E−01**	3.795	30.066	64.038	22	0.906	0.936
		DE	4.653E+01	5.949E−01	3.575	26.367	150.103	32	0.878	0.903
		ASO	4.850E+01	7.542E−01	3.125	**30.979**	**51.902**	21	**0.922**	**0.952**
		AEFA	4.450E+01	1.054E+00	**2.648**	21.311	480.826	51	0.756	0.834
Sailboat	2	GO	**1.252E+01**	**0.00E+00**	1.656	14.611	2248.895	92	0.585	**0.819**
		GSK	**1.253E+01**	**0.00E+00**	1.838	14.643	**2232.109**	**91**	**0.587**	**0.819**
		SFS	**1.253E+01**	**0.00E+00**	1.143	**14.644**	**2232.109**	**91**	**0.587**	**0.819**
		DE	**1.252E+01**	**0.00E+00**	1.066	14.579	2265.643	92	0.584	**0.819**
		ASO	**1.252E+01**	**0.00E+00**	1.282	14.556	2277.397	92	0.583	**0.819**
		AEFA	**1.252E+01**	4.918E−03	**0.879**	14.484	2315.730	95	0.580	0.814
	4	GO	**1.836E+01**	9.989E−03	2.383	17.268	1219.810	70	0.693	0.881
		GSK	**1.836E+01**	6.850E−03	2.523	**17.511**	**1153.305**	**67**	0.705	**0.886**
		SFS	**1.836E+01**	4.171E−03	1.410	17.133	1258.392	72	0.686	0.877
		DE	**1.836E+01**	**1.295E−04**	1.380	17.199	1239.341	71	0.690	0.880
		ASO	**1.836E+01**	2.682E−03	1.514	17.152	1252.737	71	0.689	0.881
		AEFA	1.823E+01	5.033E−02	**1.085**	17.348	1197.645	69	**0.714**	0.876
	6	GO	**2.363E+01**	**9.087E−03**	2.141	20.840	535.888	47	0.840	0.916
		GSK	2.351E+01	8.590E−02	2.541	**21.811**	**428.501**	**45**	**0.862**	**0.923**
		SFS	2.361E+01	8.408E−02	1.686	19.528	724.913	54	0.786	0.903
		DE	2.355E+01	1.122E−01	1.580	18.386	942.960	64	0.736	0.901
		ASO	2.356E+01	9.723E−01	1.749	18.630	891.462	60	0.749	0.898
		AEFA	2.302E+01	2.325E−01	**1.289**	20.323	603.605	51	0.811	0.903
	8	GO	**2.842E+01**	1.120E−01	2.553	**24.018**	**257.789**	**38**	0.919	0.953
		GSK	2.788E+01	1.165E−01	2.758	22.287	384.068	46	0.877	0.923
		SFS	2.834E+01	**9.492E−02**	1.909	22.333	379.967	**38**	0.897	0.937
		DE	2.814E+01	1.554E−01	1.830	21.060	509.412	57	0.859	0.906
		ASO	2.820E+01	1.827E−01	2.043	23.960	261.255	41	**0.926**	**0.957**
		AEFA	2.716E+01	3.113E−01	**1.478**	22.938	330.595	42	0.878	0.934
	10	GO	**3.268E+01**	**1.101E−01**	3.257	24.880	211.405	39	0.919	0.951
		GSK	3.227E+01	2.378E−01	3.641	25.357	189.388	**31**	**0.936**	**0.968**
		SFS	3.263E+01	1.794E−01	2.140	23.800	271.076	43	0.915	0.949
		DE	3.220E+01	2.291E−01	2.129	22.149	396.464	48	0.860	0.920
		ASO	3.256E+01	2.788E−01	2.710	**25.407**	**187.225**	36	**0.936**	0.963
		AEFA	3.095E+01	5.388E−01	**1.963**	23.932	262.956	42	0.928	0.952
	15	GO	**4.201E+01**	2.614E−01	3.671	**29.362**	**75.317**	**24**	**0.969**	0.981
		GSK	4.096E+01	3.011E−01	4.091	25.976	164.236	29	0.937	0.964
		SFS	4.170E+01	**2.231E−01**	2.612	28.516	91.505	27	0.967	**0.985**
		DE	4.052E+01	4.583E−01	2.665	25.268	193.312	35	0.944	0.961
		ASO	4.171E+01	5.542E−01	2.769	27.048	128.323	27	0.944	0.974
		AEFA	3.861E+01	9.546E−01	**2.251**	24.800	215.327	43	0.917	0.953
	20	GO	**4.957E+01**	3.539E−01	5.999	30.019	64.748	25	**0.976**	**0.989**
		GSK	4.780E+01	5.137E−01	6.508	28.869	84.377	30	0.968	0.982

(*continued on next page*)

Table 13 (*continued*).

Test image	K	Algorithms	F_{ave}	*Std*	*Time*	*PSNR*	*MSE*	*MAE*	*SSIM*	*FSIM*
		SFS	4.912E+01	**2.874E−01**	3.915	29.964	65.569	25	0.973	0.984
		DE	4.711E+01	6.337E−01	3.346	27.546	114.402	24	0.956	0.978
		ASO	4.893E+01	6.740E−01	3.259	**30.753**	**54.672**	**22**	**0.976**	**0.989**
		AEFA	4.512E+01	1.195E+00	**2.562**	26.558	143.652	31	0.944	0.966
Monkey	2	GO	**1.222E+01**	**0.000E+00**	1.708	15.990	1637.224	**86**	**0.764**	**0.864**
		GSK	**1.222E+01**	**0.000E+00**	1.838	15.987	1638.239	88	0.761	0.862
		SFS	**1.222E+01**	**0.000E+00**	1.281	15.987	1638.239	88	0.761	0.862
		DE	**1.222E+01**	**0.000E+00**	1.297	15.835	1696.473	88	0.758	**0.864**
		ASO	**1.222E+01**	8.338E−06	1.285	15.982	1640.250	87	0.760	0.863
		AEFA	**1.222E+01**	1.995E−03	**0.919**	**16.016**	**1627.470**	88	0.760	0.861
	4	GO	**1.845E+01**	2.922E−02	1.957	**18.736**	**869.933**	76	**0.845**	**0.903**
		GSK	1.841E+01	3.192E−02	2.260	18.004	1029.512	82	0.832	0.893
		SFS	1.842E+01	2.210E−02	1.421	18.421	935.439	**72**	0.833	0.897
		DE	**1.845E+01**	1.364E−02	1.330	18.464	926.254	77	0.841	0.902
		ASO	1.835E+01	**1.027E−02**	1.518	18.391	941.926	80	0.835	0.895
		AEFA	1.806E+01	1.316E−01	**1.064**	16.458	1470.033	91	0.775	0.875
	6	GO	**2.386E+01**	**4.229E−02**	2.262	21.180	495.519	**48**	0.906	0.945
		GSK	2.373E+01	7.566E−02	2.371	21.313	480.564	50	0.909	0.947
		SFS	2.379E+01	6.505E−02	1.704	21.782	431.416	54	0.910	0.949
		DE	2.381E+01	1.063E−01	1.573	**22.260**	**386.438**	58	**0.924**	**0.955**
		ASO	2.377E+01	1.403E−01	1.690	19.814	678.666	87	0.866	0.903
		AEFA	2.294E+01	2.822E−01	**1.261**	19.736	690.945	60	0.859	0.892
	8	GO	**2.864E+01**	**8.709E−02**	2.664	**25.090**	**201.401**	46	**0.958**	**0.976**
		GSK	2.834E+01	1.216E−01	3.377	22.555	361.078	42	0.926	0.955
		SFS	2.853E+01	6.561E−01	1.885	23.667	279.505	**37**	0.942	0.967
		DE	2.829E+01	1.417E−01	1.822	21.195	493.793	50	0.896	0.926
		ASO	2.839E+01	1.865E−01	2.077	23.148	314.954	45	0.928	0.948
		AEFA	2.718E+01	4.757E−01	**1.501**	21.964	413.697	43	0.908	0.936
	10	GO	**3.287E+01**	1.180E−01	3.707	**27.084**	**127.243**	35	**0.971**	**0.981**
		GSK	3.241E+01	1.749E−01	4.302	23.405	296.851	40	0.927	0.954
		SFS	3.275E+01	**1.163E−01**	2.164	23.146	315.114	36	0.924	0.953
		DE	3.238E+01	1.728E−01	2.044	25.139	199.150	44	0.945	0.961
		ASO	3.273E+01	4.002E−01	2.446	24.992	205.985	**31**	0.960	**0.981**
		AEFA	3.069E+01	4.490E−01	**1.754**	21.710	438.649	44	0.906	0.944
	15	GO	4.177E+01	2.873E−01	3.351	28.055	101.756	23	0.977	0.990
		GSK	4.095E+01	**2.610E−01**	3.562	25.979	164.110	37	0.960	0.974
		SFS	4.159E+01	3.327E−01	2.598	27.551	114.277	26	0.972	0.986
		DE	4.068E+01	4.018E−01	2.712	29.024	81.415	28	0.982	0.989
		ASO	**4.184E+01**	5.518E−01	2.784	**29.144**	**79.187**	**19**	**0.985**	**0.993**
		AEFA	3.850E+01	8.368E−01	**2.232**	26.924	132.047	25	0.969	0.982
	20	GO	**4.946E+01**	3.894E−01	5.562	30.721	55.074	29	0.985	**0.992**
		GSK	4.756E+01	3.552E−01	6.221	29.321	76.028	26	0.987	0.983
		SFS	4.889E+01	**3.175E−01**	4.064	**31.276**	**48.475**	**22**	**0.988**	**0.992**
		DE	4.685E+01	4.850E−01	3.273	26.484	146.123	28	0.966	0.983
		ASO	4.876E+01	9.925E−01	3.313	28.286	96.493	23	0.975	0.986
		AEFA	4.460E+01	1.159E+00	**2.570**	27.735	109.553	36	0.975	0.983
Tank	2	GO	**1.087E+01**	6.393E−03	1.755	14.225	2458.180	125	0.528	0.637
		GSK	1.084E+01	1.242E−01	1.886	14.225	2458.180	125	0.528	0.637
		SFS	**1.087E+01**	**5.468E−15**	1.219	14.225	2458.180	125	0.528	0.637
		DE	**1.087E+01**	**5.468E−15**	1.189	14.292	2420.542	124	0.530	0.640
		ASO	1.086E+01	1.783E−02	1.243	**14.372**	**2376.167**	**122**	**0.533**	**0.648**
		AEFA	1.040E+01	1.667E−01	**0.884**	11.358	4756.515	153	0.449	0.534
	4	GO	**1.631E+01**	3.864E−02	2.155	17.528	1148.970	113	0.681	0.754
		GSK	1.626E+01	3.497E−02	2.311	15.945	1654.247	122	0.627	0.699
		SFS	1.623E+01	2.961E−02	1.446	15.960	1648.493	122	0.627	0.697
		DE	1.630E+01	**1.389E−02**	1.326	16.477	1463.333	119	0.644	0.716
		ASO	1.622E+01	4.748E−02	1.499	**20.310**	**605.408**	100	**0.769**	**0.830**
		AEFA	1.538E+01	3.121E−01	**1.066**	14.999	2056.645	**86**	0.594	0.661
	6	GO	2.075E+01	1.471E−01	2.221	23.044	322.635	92	0.853	0.893
		GSK	2.073E+01	1.878E−01	2.339	22.835	338.496	**87**	0.857	0.894
		SFS	**2.079E+01**	**1.233E−01**	1.684	22.857	336.828	91	0.858	0.892
		DE	2.073E+01	2.206E−01	1.592	**23.402**	**297.079**	89	**0.875**	**0.908**
		ASO	2.062E+01	2.713E−01	1.705	16.408	1486.860	120	0.640	0.713
		AEFA	1.947E+01	5.287E−01	**1.254**	19.065	806.436	106	0.699	0.759
	8	GO	2.470E+01	2.510E−01	2.562	21.840	425.672	**51**	0.798	0.854
		GSK	2.448E+01	2.202E−01	2.772	22.312	381.846	92	0.843	0.885
		SFS	**2.475E+01**	2.257E−01	1.934	19.040	811.105	108	0.707	0.780
		DE	2.421E+01	**2.199E−01**	1.886	17.670	1111.983	89	0.664	0.734
		ASO	2.460E+01	2.408E−01	1.909	**24.954**	**207.822**	87	**0.900**	**0.927**

(*continued on next page*)

Table 13 (*continued*).

Test image	*K*	Algorithms	F_{ave}	*Std*	*Time*	*PSNR*	*MSE*	*MAE*	*SSIM*	*FSIM*
		AEFA	2.248E+01	6.061E−01	**1.464**	19.087	802.458	60	0.709	0.789
	10	GO	**2.827E+01**	**2.441E−01**	3.864	23.361	299.876	45	0.864	0.903
		GSK	2.773E+01	3.588E−01	4.285	24.648	222.986	45	0.870	0.898
		SFS	2.824E+01	2.871E−01	2.140	23.146	315.114	**36**	0.924	**0.953**
		DE	2.747E+01	4.173E−01	2.115	25.158	198.267	41	0.883	0.897
		ASO	2.774E+01	5.180E−01	2.496	25.911	166.711	55	0.916	0.951
		AEFA	2.553E+01	7.501E−01	**1.610**	**26.088**	**160.065**	44	**0.925**	0.946
	15	GO	**3.565E+01**	4.353E−01	3.382	26.401	148.946	**27**	**0.937**	**0.961**
		GSK	3.448E+01	3.717E−01	3.671	21.751	434.484	45	0.817	0.874
		SFS	3.544E+01	**2.543E−01**	2.608	25.607	178.794	52	0.896	0.923
		DE	3.382E+01	5.787E−01	2.684	24.736	218.531	33	0.903	0.930
		ASO	3.483E+01	6.804E−01	2.679	**27.423**	**117.695**	41	0.930	0.956
		AEFA	3.118E+01	9.667E−01	**2.269**	21.402	470.826	55	0.812	0.865
	20	GO	**4.121E+01**	6.085E−01	5.707	**31.083**	**50.670**	41	**0.971**	**0.978**
		GSK	3.943E+01	5.583E−01	6.529	26.368	150.072	**25**	0.922	0.956
		SFS	4.077E+01	**3.903E−01**	3.954	25.495	183.495	29	0.888	0.933
		DE	3.796E+01	5.965E−01	3.338	25.481	184.078	53	0.873	0.904
		ASO	4.030E+01	1.161E+00	2.728	29.169	78.734	35	0.960	0.976
		AEFA	3.599E+01	8.518E−01	**2.576**	25.938	165.695	65	0.896	0.934
House	2	GO	**1.146E+01**	**0.000E+00**	1.743	**12.150**	**3963.114**	**101**	**0.558**	0.681
		GSK	**1.146E+01**	**0.000E+00**	1.844	**12.150**	**3963.114**	**101**	**0.558**	0.681
		SFS	**1.146E+01**	**0.000E+00**	1.163	**12.150**	**3963.114**	**101**	**0.558**	0.681
		DE	**1.146E+01**	**0.000E+00**	1.075	12.147	3966.009	102	0.561	0.683
		ASO	**1.146E+01**	5.020E−03	1.329	12.069	4038.215	102	**0.558**	0.683
		AEFA	1.135E+01	9.749E−02	**0.887**	12.082	4025.737	**101**	**0.558**	**0.685**
	4	GO	**1.723E+01**	7.627E−02	1.894	14.716	2175.162	78	0.702	0.735
		GSK	1.712E+01	9.327E−02	2.334	14.740	2183.137	84	0.696	0.740
		SFS	1.715E+01	7.636E−02	1.427	13.222	3096.826	95	0.666	0.718
		DE	**1.723E+01**	9.155E−02	1.398	14.725	2190.531	81	**0.711**	0.747
		ASO	1.699E+01	**7.518E−02**	1.521	12.798	3413.901	100	0.653	0.713
		AEFA	1.656E+01	1.176E−01	**1.075**	**16.175**	**1568.991**	**76**	0.707	**0.774**
	6	GO	**2.242E+01**	**1.038E−01**	2.709	22.655	352.878	**41**	**0.836**	**0.868**
		GSK	2.214E+01	1.183E−01	3.330	23.514	389.525	42	0.835	0.858
		SFS	2.229E+01	1.355E−01	1.936	20.140	629.695	69	0.764	0.819
		DE	2.215E+01	1.120E−01	1.578	21.062	509.184	59	0.774	0.823
		ASO	2.228E+01	1.208E−01	1.752	**23.805**	**270.749**	60	0.819	0.856
		AEFA	2.154E+01	3.335E−01	**1.277**	19.896	665.954	49	0.821	0.855
	8	GO	**2.734E+01**	**1.425E−01**	2.646	24.921	209.383	**42**	0.848	0.876
		GSK	2.690E+01	1.968E−01	3.290	**25.265**	**193.475**	46	0.848	0.879
		SFS	2.718E+01	1.550E−01	1.901	25.135	199.315	44	**0.860**	**0.900**
		DE	2.680E+01	2.570E−01	1.828	24.583	226.302	44	0.848	0.881
		ASO	2.696E+01	3.323E−01	1.890	21.303	481.763	46	0.845	0.879
		AEFA	2.560E+01	4.325E−01	**1.518**	24.559	227.581	**42**	0.825	0.870
	10	GO	**3.174E+01**	**2.037E−01**	3.363	25.404	**181.370**	44	0.877	**0.914**
		GSK	3.099E+01	2.467E−01	3.604	24.060	255.312	42	0.804	0.866
		SFS	3.167E+01	2.161E−01	2.179	24.851	212.797	**36**	0.849	0.894
		DE	3.087E+01	3.057E−01	2.049	**25.500**	183.257	62	0.844	0.879
		ASO	3.141E+01	4.605E−01	2.505	23.662	279.853	37	0.851	0.901
		AEFA	2.916E+01	6.601E−01	**1.730**	24.406	235.752	48	**0.900**	0.906
	15	GO	**4.074E+01**	**2.991E−01**	4.099	**28.412**	**93.736**	31	**0.916**	0.933
		GSK	3.954E+01	3.306E−01	4.577	27.340	119.971	32	0.906	**0.941**
		SFS	**4.074E+01**	4.476E−01	2.598	23.663	279.757	29	0.866	0.923
		DE	3.928E+01	5.578E−01	2.699	26.180	156.688	30	0.777	0.892
		ASO	4.058E+01	6.669E−01	2.807	27.993	103.227	**24**	0.894	0.938
		AEFA	3.694E+01	7.602E−01	**2.132**	28.362	94.814	37	0.873	0.923
	20	GO	**4.884E+01**	**3.929E−01**	4.845	**31.514**	**45.885**	**20**	**0.940**	**0.966**
		GSK	4.649E+01	5.362E−01	5.557	28.657	88.598	36	0.915	0.934
		SFS	4.823E+01	4.029E−01	3.846	30.483	58.175	21	0.924	0.960
		DE	4.574E+01	5.907E−01	3.253	26.898	132.832	32	0.880	0.925
		ASO	4.713E+01	8.320E−01	2.932	29.171	78.693	25	0.869	0.941
		AEFA	4.315E+01	1.111E+00	**2.666**	31.055	50.998	28	0.939	0.955

concepts of fitness values and Euclidean distances, which greatly reduces the chance of the algorithm falling into a local optima. The core operator of the learning phase is fully self-adaptive, which can adequately reduce the adverse effects of human-set parameters on the algorithm. The reflection phase then imposes 3 possible operations on each of the individual dimensions. Retaining individual partial evolutionary information through a high probability can ensure the stability of algorithm convergence. Probabilistic guidance based on different elite individuals makes the algorithm capable of local convergence. We rely on the *AF* control restart operation to improve the population diversity and improve the global search ability of the algorithm by forcing unsuccessful individuals to retain knowledge through a very small probability, improving the ability of the algorithm to jump out of local optima. GO balances its exploration and exploitation capabilities through the synergy of the two phases, which ultimately improves GO's overall optimization performance in solving different problems.

We refer to the CEC 2017 problem definitions and evaluation criteria, using the most rigorous and unique experimental setup. Fifty state-of-the-art metaheuristic algorithms were used as competing algorithms on 30 test functions of the CEC 2017 test suite, and the powerful optimization performance of GO was verified. In addition, the results of two nonparametric statistical tests based on the Friedman test and the Wilcoxon signed-rank test demonstrate that GO is statistically competitive. The results of the data distribution based on box plots also confirm the robustness and stability of GO under different functions and different dimensions.

To verify the ability of GO to solve real-world problems, we used it to solve the MSA problem and the image segmentation problem. The best-performing GSK, SFS, DE, ASO, and AEFA in function optimization experiments are also used as competing algorithms for GO. First, GO is applied to train the HMM and solve the MSA problem. The experiments are based on the gene sequence of the novel coronavirus, which reaches 30,000 dimensions. The experimental results show that GO outperforms other algorithms and can efficiently solve complex MSA problems. The results obtained by GO confirm that SARS-CoV-2, bat coronavirus, SARS, and other SARS-CoV-2 isolates have certain homology. Therefore, the aligned sequences obtained by GO alignment can be of greater value in the fields of virus control, disease diagnosis, and gene tracing. Second, in Kapur's entropy-based multithreshold image segmentation problem, GO gives better image segmentation results under different threshold conditions. The images segmented by GO can be used in the fields of object positioning, machine vision, pattern recognition, and so on. Based on these two different types of problems, GO achieves more potential optimization results and demonstrates stability and robustness in solving different types of real-world problems. Without adjusting any parameters of GO in advance, it can be considered an alternative optimizer to other metaheuristic algorithms, especially in the face of high-dimensional problems.

Of course, GO has certain limitations, which are listed in detail below. (1) Although GO simulates learning and reflection behavior, its form of simulation is specific and limited, and does not truly accomplish the mapping from physical simulants to mathematical models. (2) Although GO has been shown to be insensitive to parameters, constant parameters may not yield better results for a given problem. (3) Although GO has conducted different experiments in multiple dimensions, the algorithm's running time also increases significantly when the dimension explodes due to the restart operation and distance calculation involved. Similarly, although using different strategies for each dimension of an individual in the reflection phase will improve the optimization performance of GO, it will also incur a certain time overhead. (4) In addition to the optimization performance of GO proven on CEC 2017, it can still achieve good results on CEC 2005 and CEC 2014. However, due to its mechanism of preventing premature convergence of the algorithm, it cannot converge to the global optimum on simpler functions such as the sphere function within the standard evaluation times.

Future research should be devoted to GO. First, fixed parameters prevent GO from better responding to different problems, and parameter self-adaptive techniques combined with GO are expected to change this situation. Second, a binary version, a multiobjective version, and a parallel version of GO will be developed in the future to adapt to more challenging practical optimization problems. Finally, the combination of GO with large-scale optimization strategies such as the recursive differential grouping (RDG) strategy to improve optimization when solving high-dimensional, complex LSGO problems is an additional development direction for GO.

CRediT authorship contribution statement

Qingke Zhang: Conceptualization, Methodology, Investigation, Writing – original draft, Writing – review & editing, Software, Supervision, Project administration, Funding acquisition. **Hao Gao:** Methodology, Investigation, Software, Writing – original draft, Validation, Visualization, Data curation, Formal analysis. **Zhi-Hui Zhan:** Writing – review & editing, Resources. **Junqing Li:** Writing – review & editing, Resources. **Huaxiang Zhang:** Writing – review & editing, Resources, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 62006144), the Major Fundamental Research Project of Shandong, China (No. ZR2019ZD03), and the Taishan Scholar Project of Shandong, China (No. ts20190924).

References

[1] J. Pierezan, L.D.S. Coelho, Coyote optimization algorithm: a new metaheuristic for global optimization problems, in: 2018 IEEE Congress on Evolutionary Computation, IEEE, 2018, pp. 1–8.
[2] G. Dhiman, V. Kumar, Emperor penguin optimizer: A bio-inspired algorithm for engineering problems, Knowl.-Based Syst. 159 (2018) 20–50.
[3] A.H. Halim, I. Ismail, S. Das, Performance assessment of the metaheuristic optimization algorithms: an exhaustive review, Artif. Intell. Rev. 54 (3) (2021) 2323–2409.
[4] G. Dhiman, A. Kaur, STOA: a bio-inspired based optimization algorithm for industrial engineering problems, Eng. Appl. Artif. Intell. 82 (2019) 148–174.
[5] K. Hussain, M.N. Mohd Salleh, S. Cheng, Y. Shi, Metaheuristic research: a comprehensive survey, Artif. Intell. Rev. 52 (4) (2019) 2191–2233.
[6] B. Morales-Castañeda, D. Zaldivar, E. Cuevas, F. Fausto, A. Rodríguez, A better balance in metaheuristic algorithms: does it exist? Swarm Evol. Comput. 54 (2020) 100671.
[7] X. Tao, X. Li, W. Chen, T. Liang, Y. Li, J. Guo, L. Qi, Self-adaptive two roles hybrid learning strategies-based particle swarm optimization, Inform. Sci. 578 (2021) 457–481.
[8] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.
[9] S. Akyol, B. Alatas, Plant intelligence based metaheuristic optimization algorithms, Artif. Intell. Rev. 47 (4) (2017) 417–462.
[10] B. Alatas, H. Bingol, Comparative assessment of light-based intelligent search and optimization algorithms, Light Eng. 28 (6) (2020).
[11] T.S.L.V. Ayyarao, N.S.S. Ramakrishna, R.M. Elavarasan, N. Polumahanthi, M. Rambabu, G. Saini, B. Khan, B. Alatas, War strategy optimization algorithm: a new effective metaheuristic algorithm for global optimization, IEEE Access 10 (2022) 25073–25105.
[12] G. Dhiman, V. Kumar, Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications, Adv. Eng. Softw. 114 (2017) 48–70.
[13] M. Jain, V. Singh, A. Rani, A novel nature-inspired algorithm for optimization: squirrel search algorithm, Swarm Evol. Comput. 44 (2019) 148–175.
[14] A.E. Eiben, J. Smith, From evolutionary computation to the evolution of things, Nature 521 (7553) (2015) 476–482.
[15] D.B. Fogel, Artificial Intelligence Through Simulated Evolution, Wiley-IEEE Press, 1998.
[16] I. Rechenberg, Evolution strategy: optimization of technical systems by means of biological evolution, Fromman-Holzboog Stuttgart 104 (1973) 15–16.

[17] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1998.
[18] P. Moscato, et al., On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, Caltech Concurrent Computation Program, C3P Report 826, Citeseer, 1989, p. 1989.
[19] J.R. Koza, Genetic programming as a means for programming computers by natural selection, Stat. Comput. 4 (2) (1994) 87–112.
[20] R.G. Reynolds, An introduction to cultural algorithms, in: Proceedings of the Third Annual Conference on Evolutionary Programming, Vol. 24, World Scientific, 1994, pp. 131–139.
[21] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[22] N. Hansen, The CMA evolution strategy: a comparing review, Towards New Evol. Comput. (2006) 75–102.
[23] P. Civicioglu, Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm, Comput. Geosci. 46 (2012) 229–247.
[24] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, IEEE, 1995, pp. 1942–1948.
[25] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[26] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE Comput. Intell. Mag. 1 (4) (2006) 28–39.
[27] M. Eusuff, K. Lansey, F. Pasha, Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization, Eng. Optim. 38 (2) (2006) 129–154.
[28] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (3) (2007) 459–471.
[29] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (6) (2008) 702–713.
[30] X. Yang, Firefly algorithms for multimodal optimization, in: International Symposium on Stochastic Algorithms, Springer, 2009, pp. 169–178.
[31] X. Yang, S. Deb, Cuckoo search via Lévy flights, in: 2009 World Congress on Nature & Biologically Inspired Computing, IEEE, 2009, pp. 210–214.
[32] K.M. Passino, Bacterial foraging optimization, Int. J. Swarm Intell. Res. 1 (1) (2010) 1–16.
[33] Y. Shi, Brain storm optimization algorithm, in: International Conference in Swarm Intelligence, Springer, 2011, pp. 303–309.
[34] R.V. Rao, V.J. Savsani, D. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput. Aided Des. 43 (3) (2011) 303–315.
[35] X. Yang, Flower pollination algorithm for global optimization, in: International Conference on Unconventional Computing and Natural Computation, Springer, 2012, pp. 240–249.
[36] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition, in: 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 4661–4667.
[37] M. Neshat, G. Sepidnam, M. Sargolzaei, A.N. Toosi, Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications, Artif. Intell. Rev. 42 (4) (2014) 965–997.
[38] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[39] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, Knowl.-Based Syst. 89 (2015) 228–249.
[40] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[41] R. Rao, Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems, Int. J. Ind. Eng. Comput. 7 (1) (2016) 19–34.
[42] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp swarm algorithm: a bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191.
[43] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: algorithm and applications, Future Gener. Comput. Syst. 97 (2019) 849–872.
[44] S. Arora, S. Singh, Butterfly optimization algorithm: a novel approach for global optimization, Soft Comput. 23 (3) (2019) 715–734.
[45] W. Zhao, L. Wang, Z. Zhang, Supply-demand-based optimization: a novel economics-inspired algorithm for global optimization, IEEE Access 7 (2019) 73182–73206.
[46] W. Zhao, Z. Zhang, L. Wang, Manta ray foraging optimization: an effective bio-inspired optimizer for engineering applications, Eng. Appl. Artif. Intell. 87 (2020) 103300.
[47] A.W. Mohamed, A.A. Hadi, A.K. Mohamed, Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm, Int. J. Mach. Learn. Cybern. 11 (7) (2020) 1501–1529.
[48] M.A. Al-Betar, Z.A.A. Alyasseri, M.A. Awadallah, I. Abu Doush, Coronavirus herd immunity optimizer (CHIO), Neural Comput. Appl. 33 (10) (2021) 5011–5042.
[49] Q. Askari, M. Saeed, I. Younas, Heap-based optimizer inspired by corporate rank hierarchy for global optimization, Expert Syst. Appl. 161 (2020) 113702.
[50] A. Faramarzi, M. Heidarinejad, S. Mirjalili, A.H. Gandomi, Marine predators algorithm: a nature-inspired metaheuristic, Expert Syst. Appl. 152 (2020) 113377.
[51] L. Abualigah, D. Yousri, M. Abd Elaziz, A.A. Ewees, M.A. Al-Qaness, A.H. Gandomi, Aquila optimizer: a novel meta-heuristic optimization algorithm, Comput. Ind. Eng. 157 (2021) 107250.
[52] M. Braik, A. Hammouri, J. Atwan, M.A. Al-Betar, M.A. Awadallah, White shark optimizer: a novel bio-inspired meta-heuristic algorithm for global optimization problems, Knowl.-Based Syst. 243 (2022) 108457.
[53] F.A. Hashim, E.H. Houssein, K. Hussain, M.S. Mabrouk, W. Al-Atabany, Honey badger algorithm: new metaheuristic algorithm for solving optimization problems, Math. Comput. Simulation 192 (2022) 84–110.
[54] J.O. Agushaka, A.E. Ezugwu, L. Abualigah, Dwarf mongoose optimization algorithm, Comput. Methods Appl. Mech. Engrg. 391 (2022) 114570.
[55] F.A. Hashim, A.G. Hussien, Snake optimizer: A novel meta-heuristic optimization algorithm, Knowl.-Based Syst. 242 (2022) 108320.
[56] A.E. Ezugwu, J.O. Agushaka, L. Abualigah, S. Mirjalili, A.H. Gandomi, Prairie dog optimization algorithm, Neural Comput. Appl. (2022) 1–49.
[57] L. Abualigah, M. Abd Elaziz, P. Sumari, Z.W. Geem, A.H. Gandomi, Reptile search algorithm (RSA): a nature-inspired meta-heuristic optimizer, Expert Syst. Appl. 191 (2022) 116158.
[58] O.N. Oyelade, A.E.-S. Ezugwu, T.I. Mohamed, L. Abualigah, Ebola optimization search algorithm: a new nature-inspired metaheuristic optimization algorithm, IEEE Access 10 (2022) 16150–16177.
[59] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[60] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Inform. Sci. 179 (13) (2009) 2232–2248.
[61] H. Shareef, A.A. Ibrahim, A.H. Mutlag, Lightning search algorithm, Appl. Soft Comput. 36 (2015) 315–333.
[62] H. Salimi, Stochastic fractal search: a powerful metaheuristic algorithm, Knowl.-Based Syst. 75 (2015) 1–18.
[63] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowl.-Based Syst. 96 (2016) 120–133.
[64] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, Neural Comput. Appl. 27 (2) (2016) 495–513.
[65] W. Zhao, L. Wang, Z. Zhang, Atom search optimization and its application to solve a hydrogeologic parameter estimation problem, Knowl.-Based Syst. 163 (2019) 283–304.
[66] Yadav, A. Yadav, AEFA: artificial electric field algorithm for global optimization, Swarm Evol. Comput. 48 (2019) 93–108.
[67] W. Zhao, L. Wang, Z. Zhang, Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm, Neural Comput. Appl. 32 (13) (2020) 9383–9425.
[68] A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: A novel optimization algorithm, Knowl.-Based Syst. 191 (2020) 105190.
[69] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, A.H. Gandomi, The arithmetic optimization algorithm, Comput. Methods Appl. Mech. Engrg. 376 (2021) 113609.
[70] I. Ahmadianfar, A.A. Heidari, A.H. Gandomi, X. Chu, H. Chen, RUN beyond the metaphor: an efficient optimization algorithm based on runge kutta method, Expert Syst. Appl. 181 (2021) 115079.
[71] I. Ahmadianfar, A.A. Heidari, S. Noshadian, H. Chen, A.H. Gandomi, INFO: an efficient optimization algorithm based on weighted mean of vectors, Expert Syst. Appl. (2022) 116516.
[72] A.C. Johnvictor, V. Durgamahanthi, R.M. Pariti Venkata, N. Jethi, Critical review of bio-inspired optimization techniques, Wiley Interdiscip. Rev. Comput. Stat. 14 (1) (2022) 1528.
[73] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.

[74] A. Tzanetos, G. Dounias, Nature inspired optimization algorithms or simply variations of metaheuristics? Artif. Intell. Rev. 54 (3) (2021) 1841–1862.
[75] G. Wu, R. Mallipeddi, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2017 Competition and Special Session on Constrained Single Objective Real-Parameter Optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2017.
[76] Z. Meng, Y. Zhong, C. Yang, Cs-de: cooperative strategy based differential evolution with population diversity enhancement, Inform. Sci. 577 (2021) 663–696.
[77] L. Sabug Jr., F. Ruiz, L. Fagiano, SMGO-Δ: balancing caution and reward in global optimization with black-box constraints, Inform. Sci. 605 (2022) 15–42.
[78] M. Ćrepinšek, S. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms, ACM Comput. Surv. 45 (2013) 1–33.
[79] J. Arabas, K. Opara, Population diversity of nonelitist evolutionary algorithms in the exploration phase, IEEE Trans. Evol. Comput. 24 (6) (2019) 1050–1062.
[80] H. Dong, Z. Dong, Surrogate-assisted grey wolf optimization for high-dimensional, computationally expensive black-box problems, Swarm Evol. Comput. 57 (2020) 100713.
[81] F. Wang, X. Wang, S. Sun, A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization, Inform. Sci. 602 (2022) 298–312.
[82] S. Lalwani, R. Kumar, N. Gupta, A study on inertia weight schemes with modified particle swarm optimization algorithm for multiple sequence alignment, in: 2013 Sixth International Conference on Contemporary Computing (IC3), IEEE, 2013, pp. 283–288.
[83] M. Gollery, Bioinformatics: sequence and genome analysis, Clin. Chem. 51 (11) (2005) 2219–2220.
[84] A. Krogh, M. Brown, I.S. Mian, K. Sjölander, D. Haussler, Hidden markov models in computational biology: applications to protein modeling, J. Mol. Biol. 235 (5) (1994) 1501–1531.
[85] N. Zhu, D. Zhang, W. Wang, X. Li, B. Yang, J. Song, X. Zhao, B. Huang, W. Shi, R. Lu, et al., A novel coronavirus from patients with pneumonia in China, 2019, N. Engl. J. Med. (2020).
[86] P. Zhou, X.-L. Yang, X.-G. Wang, B. Hu, L. Zhang, W. Zhang, H.-R. Si, Y. Zhu, B. Li, C.-L. Huang, et al., A pneumonia outbreak associated with a new coronavirus of probable bat origin, Nature 579 (7798) (2020) 270–273.
[87] J. Kuruvilla, D. Sukumaran, A. Sankar, S.P. Joy, A review on image processing and image segmentation, in: 2016 International Conference on Data Mining and Advanced Computing, IEEE, 2016, pp. 198–203.
[88] N. Otsu, A threshold selection method from gray-level histograms, IEEE Trans. Syst. Man Cybern. 9 (1) (1979) 62–66.
[89] J.N. Kapur, P.K. Sahoo, A.K. Wong, A new method for gray-level picture thresholding using the entropy of the histogram, Comput. Vis. Graph. Image Process. 29 (3) (1985) 273–285.
[90] M. Abdel-Basset, V. Chang, R. Mohamed, A novel equilibrium optimization algorithm for multi-thresholding image segmentation problems, Neural Comput. Appl. 33 (17) (2021) 10685–10718.
[91] A. Hore, D. Ziou, Image quality metrics: PSNR vs. SSIM, in: 2010 20th International Conference on Pattern Recognition, IEEE, 2010, pp. 2366–2369.
[92] K. Chaurasia, N. Sharma, Performance evaluation and comparison of different noise, apply on pngimage format used in deconvolution wiener filter (FFT) algorithm, Evol. Trends Eng. Technol. 4 (2015) 8–14.