Vol. 33, No. 1 (2023) | DOI: 10.37190/ord230108

OPEN ACCESS

Operations Research and Decisions

www.ord.pwr.edu.pl

OPERATIONS RESEARCH AND DECISIONS QUARTERLY

Gold rush optimizer. A new population-based metaheuristic algorithm

Kamran Zolfi[1]

[1] *Department of Industrial Engineering, Lenjan Branch, Islamic Azad University, Isfahan, Iran*

[*] *Corresponding author: zolfi@iauln.ac.ir*

Abstract

Today's world is characterised by competitive environments, optimal resource utilization, and cost reduction, which has resulted in an increasing role for metaheuristic algorithms in solving complex modern problems. As a result, this paper introduces the gold rush optimizer (GRO), a population-based metaheuristic algorithm that simulates how gold-seekers prospected for gold during the Gold Rush Era using three key concepts of gold prospecting: migration, collaboration, and panning. The GRO algorithm is compared to twelve well-known metaheuristic algorithms on 29 benchmark test cases to assess the proposed approach's performance. For scientific evaluation, the Friedman and Wilcoxon signed-rank tests are used. In addition to these test cases, the GRO algorithm is evaluated using three real-world engineering problems. The results indicated that the proposed algorithm was more capable than other algorithms in proposing qualitative and competitive solutions.

Keywords: *gold rush optimizer, metaheuristic, global optimization, population-based algorithm*

1. Introduction

The term optimization is a comprehensive concept encompassing various fields, from engineering design to entrepreneurial planning and internet routing to wedding planning. Nearly whatever we do, we make some effort to meet the current expectations, make specific accomplishments, or achieve predetermined objectives. Optimization of things is an essential part of all these activities. Tending to fall short of these resources in the real world, we optimize resources to gain profit, achieve higher quality, and save time. Therefore, we must use optimization to identify optimal ways to take advantage of these priceless resources, despite all the existing limitations. Combining optimization with mathematics magic, also known as mathematical optimization, deals with such challenges in planning and designing by using mathematical tools [76]. The optimization approach has a critical part in minimising or maximising a function concerning decision variables. In the real-life environment, we face many problems tackling

Received 17 May 2022, accepted 8 February 2023, published online 16 April 2023
ISSN 2391-6060 (Online)/

The costs of publishing this issue have been cofinansed by the program *Development of Academic Journals* of the Polish Ministry of Education and Science under agreement No. RCN/SP/0241/2021/1

which is done through an essentially intricate, with numerous solution spaces. They comprise nonlinear constraints, a non-convex optimization process that takes significant computational resources and is quite costly, complicating the problem-solving task, given the substantial number of constraints and variables. Moreover, despite its capability in providing approximately optimum solutions, the classical approaches cannot ensure delivering the best solutions to real-world optimization problems. Accordingly, researchers have developed and suggested many metaheuristic optimization algorithms that have significantly impacted solving complicated problems [30].

When combined with heuristics in mathematical algorithms, the term meta refers to beyond or higher level, displaying an overall better performance, compared to simple heuristics. The metaheuristic algorithms take advantage of the local and global search exchange. Randomization is often the mechanism allowing the generation of diversified solutions. Although metaheuristics is an already well-established concept with growing usage, it suffers from the absence of a generally accepted definition, even for the term heuristic, as many scholars do not differentiate between the word heuristics and metaheuristics. However, the new trend tends to bring all stochastic algorithms using randomization and global exploration under metaheuristics. Randomization properly paves the way for getting away from local search and working toward global-scale search. Hence, almost all metaheuristic algorithms are often suitable for nonlinear modelling and global optimization. The principal elements of any metaheuristic algorithms include intensification (exploitation) and diversification (exploration) [21]. Exploration refers to a search algorithm's ability to discover various solutions' distribution within different search space regions. However, exploitation highlights the idea of reinforcing the search process over promising regions of the solution space to identify optimum solutions or enhance the current solutions. Accordingly, the empirical experimentation has displayed a solid association between the specific search method's exploitation-exploration ability and its convergence rate. Exploitation mechanisms famously improve convergence speed toward a global optimum along with elevating the probability of entrapment into local optima. On the contrary, search strategies favouring explorations over exploitation are inclined to elevate the probability of identifying regions within the search space where the global optimum is more likely to be located, at the cost of worsening the algorithm's convergence speed [49].

A main part of the latest metaheuristics has been advanced before 2000. Despite those classical metaheuristic algorithms' achievements, innovative and fresh evolutionary methods also arose successfully in the last two decades. The studies in this field, on metaheuristic algorithms, have particularly relied on evolution and processes pertinent to behaviours as inspiration sources for many new metaheuristics. In numerous cases, this modern metaheuristic approach generates the best solutions for some benchmark problem sets that have remained unresolved [15]. As suggested by the No Free Lunch theory [73], no optimization algorithm can effectively solve all optimization problems. Broadly, a specific metaheuristic can show promising results on some problems. On the other hand, the same algorithm can perform unsatisfactorily on other problems. Accordingly, drawing new metaheuristic optimization algorithms is highly embraced as long as there is considerable added significance to the field. The NFL adds the super activeness element to this area, leading to the improvement in the existing approaches and introducing new metaheuristics each year.

The wisdom of crowds refers to the idea that the obtained solutions or judgments through group decision-making are generally superior to individual solutions. This idea can be used to solve problems

and find optimal solutions [80]. Although a lot of metaheuristic algorithms have been introduced so far, few have been built based on human wisdom, behaviour, or historical events; there is a lack of research in this area. In addition, a great challenge the metaheuristic algorithms face is being trapped in the local optimal point and the scarcity of the proper balance between exploration and exploitation capabilities. Another drawback of the metaheuristic algorithms is their large number of parameters whose adjustment needs sufficient knowledge and experience of the corresponding problem or trial and error. This article introduces a novel optimization algorithm called gold rush optimizer (GRO), inspired by the gold rush historical event and how gold seekers found gold. This algorithm has an appropriate equilibrium between exploitation and exploration, capable of perfectly giving a wide berth to the local optima. Accordingly, this algorithm is expected to solve complex optimization problems satisfactorily. The exploitation power of the GRO algorithm was examined using unimodal benchmark test cases. Also, its exploration power was examined by the multimodal benchmark and complex CEC2005 functions. The statistical results revealed that the algorithm could generate high-quality solutions competitive to other comparable metaheuristic algorithms. The statistical tests also indicated that this algorithm outperformed most algorithms, such as WCA, DE, SCA, WOA, GSA, FA, GA, and PSO. Furthermore, the GRO algorithm was applied to three engineering problems for optimal design, and the results were compared with those of other algorithms.

The paper is organized into sections as follows. In Section 2, a summary of metaheuristic algorithms is provided. In Section 3, the GRO algorithm's characteristics are described, and a comparative study is proposed in Section 4. Section 5 presents three engineering problems solved by this algorithm, and their results are examined. Finally, the concluded remarks are stated in Section 6 .

2. Literature review

The metaheuristic algorithms are classified from different aspects. According to the number of candidate solutions during algorithm execution, these are divided into single–solution and population-based types. The algorithm initiates a random solution for the first category, improving it to reach the final solution at stop time. In the second category, the algorithm starts with more than one solution and enhances them. Each of these two has some advantages and disadvantages. The first method has low computational costs due to starting with merely one solution. However, it may get stuck in a local optimum, called premature convergence. In the second method, since the solutions are distributed in various problem spaces, and there is data interaction between them, the model can avoid the local optimum, finding the global optimum. Nevertheless, the evaluation function must be calculated for more solutions, and therefore, it requires higher computational costs [43].

The metaheuristic algorithms are classified into two nature-inspired and non-nature-inspired types based on their origin. The nature-inspired algorithms are classified into evolutionary, swarm intelligence, and physics/chemistry-based ones [51].

The methods founded on evolution rely on natural evolution laws as a source of inspiration. The search process starts with a population randomly generated and evolved over succeeding generations. The main advantage of these methods is that the best individuals tend to create the next generation of individuals together, optimising the population over several generations [46]. The genetic algorithm

(GA) [26], primarily operating based on population, was influenced mainly by Darwin's evolutionary theory. It provides a simulation of the fitter's survival and their genes where a parameter represents a gene. Each solution corresponds to a chromosome, assessing each individual's fitness in the population by using an objective function. Different selection mechanisms, such as the roulette wheel, are used for poor solutions enhancement and the best solutions identification. Concerning the proportional quality of the probability to the objective value, this operator gives a higher probability to identify the best solutions. Also, the probability of worse solutions selection elevates the local optima avoidance, meaning that other solutions can function as saviours if good solutions are trapped in a local optimum. The differential evolution (DE) [67] is a stochastic multi-agent search technique effectively applied in global optimization problems. It uses simple arithmetic and classical operators – mutation, crossover, and selection – to evolve from a randomly created initial population to a final solution. It has three control parameters: amplification factor of the difference vector (F), crossover control parameter (CR), and population size (NP). Advanced versions of DE, such as success-history-based parameter adaptation differential evolution (SHADE) [70], SHADE with linear population size reduction (LSHADE) [71], LSHADE with ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighbourhood (LSHADE–EpSin) [6], have been developed. Other common evolutionary-based algorithms include evolutionary Rao algorithm (ERA) [69], genetic programming (GP), evolution strategy (ES), population-based incremental learning [8], fast evolutionary programming [79], biogeography-based optimizer [66], and learner performance-based behaviour algorithm [59].

Primarily depending on the decentralization principle, Swarm Intelligence (SI) imitates the individuals' self-organized behaviour in a group of individual birds or insects [1]. Particle Swarm Optimization (PSO) [35] algorithm simulates birds' and fish populations' behaviours. Besides, since being introduced, it has gained more popularity. This simulation technique allows a population of particles to convey information cooperatively and find the optimal regions in a particular space through two vital tasks: exploration and exploitation. The interaction of all particles follows exploration. Exploitation is carried out at the final iterations when the global search is finished, and particles are left passive to more search. Each particle (individual) flies at a certain velocity in the searching space. The particles' mobility in the searching space is based on their velocities. The velocity vector is updated concerning the best previous positions of the particles (Pbests) and the swarm's global best position (Gbests). Therefore, the particles do not necessarily discard their low-fitness Pbests and approach other regions of the search space. Hence, the particles can be trapped in local optima, intensifying premature convergence. ant colony optimization (ACO) [16] is another approximate optimization technique, taking inspiration from actual ant colonies, particularly from the ants' foraging behaviour. This behaviour is essentially based on the indirect interaction between the ants using chemical pheromone trails, enabling them to trace food scraps from their nest. Artificial ants in ACO are stochastic solution construction procedures generating candidate solutions by exploiting artificial pheromone information adapted based on the ants' foraging experience and probably available heuristic data. Karaboga [29] introduced the artificial bee colony algorithm (ABC) for real-parameter optimization problems stimulating a bee colony's foraging behaviour. The ABC classifies artificial foraging bees into three groups: employed bees, onlooker bees, and scout bees. The colony is made of about %50 employed bees, and the rest are onlooker bees. Employed bees rely on their memory to forage for food around the sources. In the meantime, they transfer their food-related information to

onlooker bees. Onlooker bees evaluate food sources from all employed bees, picking them out with a selection method; then, they forage the food around the chosen source. Scout bees are translated from a few employed bees leaving their food sources and searching for new ones. The seeker optimization algorithm [11] is another swarm intelligence algorithm inspired by the human's intelligent search via memory, experience, and uncertainty reasoning. In this algorithm, like particle swarm optimization (PSO), each seeker uses memory to store its best solution and the best global solution, and uncertainty is handled using the cloud theory. The marine predators algorithm [18] employs the Lévy strategy and Brownian motion to simulate the food-seeking behaviour of ocean predators. Suyanto et al. [68] presented the Komodo Mlipir algorithm (KMA) with inspiration from the Komodo Dragon life in Indonesia and the Javanese gait. The results of running KMA on 23 test function benchmarks indicated its higher level of guarantee to obtain global optimum than other algorithms. Li et al. [40] introduced the Slime Mould Algorithm (SMA) and used the Freedman test to show its first rank among other algorithms. Compared to evolutionary algorithms, the advantage of swarm intelligence algorithms is that they store and use information during the search in the problem space. However, evolutionary algorithms use evolutionary operators in each iteration. Other swarm intelligence metaheuristic algorithms include glowworm swarm optimization [39], firefly algorithm [75], bat algorithm [77], cuckoo search [78], grey wolf optimizer [47], whale optimization algorithm [46], harris hawks optimization [25], red deer algorithm [19], tunicate swarm algorithm [30], golden eagle optimizer [48], preaching optimization algorithm [72], wild horse optimizer [53], and sheep flock optimization algorithm [38].

Physics-inspired heuristics are another type of EAs activated by physics laws. Simulated annealing [37] is a single-solution metaheuristic algorithm that emulates the annealing process in metallurgy. It is capable of escaping from local optima to approximate the global minimum of combinatorial optimization problems. Gravity Search Optimization (GSA) [60] is introduced as an optimization algorithm that follows the law of gravity and mass interactions. In the proposed algorithm, the search agents are a collection of masses that interact based on Newtonian gravity and the laws of motion. Kaveh and Talatahari [34] utilized the governing Coulomb law from electrostatics and the Newtonian laws of mechanics and introduced Charged System Search (CSS). The CSS is a population-based algorithm in which each individual is a Charged Particle (CP). CPs can affect each other based on their fitness values and separation distances. The electrostatics laws and Newtonian mechanics laws determine the net force's quantity and movement quality, respectively. Electromagnetic Field Optimization (EFO) [2] is inspired by the behaviour of electromagnets with different polarities and uses the famous, nature-inspired golden ratio. In EFO, an electromagnetic particle creates a potential solution. The number of the optimization problem's variables determines the number of electromagnets. EFO is a population-based algorithm dividing the population into three negative, positive, and neutral fields, among which the attraction-repulsion forces direct the particles toward the global minimum. The golden ratio determines the ratio of attraction and repulsion forces that help accelerate and improve effective particle convergence. The Transient Search Optimization (TSO) algorithm [58] is inspired by the transient response of switched electrical RLC circuits. The Billiards-inspired Optimization Algorithm (BOA) [32] mimics the billiards game in which several physics laws are involved formulating the basis of BOA plus vector algebra. The physics behind these games mainly involves collisions between balls. In this algorithm, each candidate solution containing many decision variables is considered a multidimensional billiards ball. These balls are indeed the

agents of the optimization problem, and each dimension represents a variable. Briefly, the process starts with the initial generation of balls with random distribution. Then, some of the best ones are selected as pockets. When the balls hit other balls, vector algebra,5 and conservation laws determine the balls' final positions in the optimization search space. Other physics-based metaheuristic algorithms include ray optimization [33], blackhole [23], Henry gas solubility optimization [22], Lichtenberg algorithm [57], gradient-based optimizer [3], mementum search algorithm [13], thermal exchange metaheuristic optimization algorithm [31], and atomic orbital search [7].

Some papers propound Human-based metaheuristic (HM) algorithms as a new category [46, 50]. These algorithms imitate human behaviours and characteristics. Human social behaviour aiming at solving intricate problems inspires social group optimization [63]. Whenever an individual resolves a problem/task, the problem/task becomes too complex to solve, or the problem tends to stay unsolvable. Nevertheless, if a group of individuals solves an identical problem, the intricacy diminishes, and it becomes manageable and solvable. The members of a social group are influenced by the successful individual's features (i.e., traits). Accordingly, they tend to adapt by changing their traits and gaining the competence to solve/tackle complicated issues/challenges. On the contrary, the population in the imperialist competitive algorithm [5] is categorised into colonies and imperialist states, in which each individual is called a country. The imperialist competitive algorithm is, at its core, founded on the competition between imperialists to seize other colonies. In which the weak empires disintegrate over time, this competition optimistically leads the solutions to converge to the global minimum. At the end of the competition, merely one imperialist remain ruling over other colonies.

In the next section, the GRO and its characteristics are described. According to the literature review, the GRO algorithm can be categorised into two population-based and human-based metaheuristic algorithms.

3. Gold rush optimizer (GRO)

In this section, first, the main idea of the proposed algorithm is described, and then, the framework of the proposed method is expressed in detail.

3.1. Inspiration

Sitting among the 11th group in the periodic table with an atomic number of 79 and coming from Latin origins – aurum – gold (Au) is a shiny, yellow, noble metal that does not tarnish.

From ancient history to modern times, gold has enjoyed a spectacular position, unprecedented compared to any other metal, in our lives. Its chemical inactivity and lasting physical characteristics make it ideal for making precious coins and jewelry that maintain their shine even under harsh conditions for decades and even centuries [42]. Gold is a soft metal with excellent conductivity (thermal and electrical) and corrosion-resistive properties with the highest malleability and ductility among natural elements. It has multiple applications, such as minting, investment, making jewelry, various conductors, colored glass, gold leafing, and tooth restoration [55]. It is often employed as an alloy of other metals, such as silver, copper, platinum, or palladium, to increase toughness [52]. It is found in two main deposits types, including lode (or vein) deposits in which gold is found stored in rock cracks, and veins and placer (or

alluvial) deposits formed by moving water that has eroded gold out of lode deposits and deposited it in the sand, fissure, and stream beds; Native gold is undoubtedly the most frequent form of gold in ores, with a 90% gold content or more and is frequently accompanied by silver [54];

One of the main events in the history of gold is the gold rush. Gold rush refers to the unprecedented flood of people desiring to make a fortune. Hence, the name fortune seekers moved into the places where gold deposits were newly discovered. Major gold rushes occurred in the United States, Australia, Canada, and South Africa in the 19th century [4].

In the following, it is explained that the three main ideas of the gold rush inspired the GRO algorithm design.

3.2. Migration

When a gold mine was discovered, the news spread quickly and thousands of people were employed in these mines. Those who wanted to work in gold mines in that early period were supposed to have one paramount ability, including working for long hours in harsh conditions. They were also often young men traveling in the company of friends, brothers, cousins, and in-laws from their village. They often had a leader selected from their small community and some signed documents bounding them to travel together [9]. The dream of making a lifetime fortune overnight tempted people from every class and social or religious status, including military veterans, ordinary workers, village farmers, proficient miners, and traders worldwide. The easy wealth that was potentially available to all highlights their different motives and experiences and also their strife against both racial and religious prejudice [56].

The flood of men of all races pouring into the areas with gold mines was astonishing given the conditions at the time, including the short period of the rush, the relative global population, and the primary transportation system. The discovery of rich gold placers in other countries was an unanticipated consequence of the strike. While panning gold, prospectors came to realize the similarities between the rocks and geological formations there and the ones they had witnessed before, giving them the hunch that their countries might be a significant source of gold deposits. This hunch made them get back to their home, delve into the interior, and find gold [62].

3.3. Gold mining (gold panning)

The gold seekers' vision was to arrive in gold districts, gain considerable profits in a short time, and then leave the harsh working conditions. However, the mining process was time-consuming and required dwelling in the area, leading to the emergence of small towns, stores, eateries, and boardinghouses. The miners' working camps and individual miners declined and then faded away. Struggling for life and making arrangements to adapt to the working environment, the miners had to work tirelessly under grave conditions; they were on the move continually, whether digging or washing, transferring dirt out of the mine, or scouting out another mining site. This passion mirrored the unexpected and impractical character of the mining experience. On the verge of the second half of the 19th century, if someone wanted to pursue their dream in a gold mine, all they needed was a pick or shovel and a pan to sort out the gold from the debris [61].

3.4. Cooperation

Mining was becoming a cooperative work. Despite being in continuous motion, miners started organising themselves to create or use cradles or long toms, which were technically slanted boxes with bars on the bottom for catching gold. The cradle forced miners into an economic partnership through which they had to cooperate, while they may have preferred working separately. Sometimes miners would fight over the digging site and mining conditions. But, since they had to get together to use a cradle, they were forced to join partnerships. However uneasy they may have been [61].

3.5. Gold rush impacts

The adventures of gold seekers beginning with this valuable metal's discovery led to irreversible and drastic changes in societies in the economy, social and economic relationships, and personal values; short-term transformations were the prevalence of plague or war. Gold rapidly transformed the district into a gold-based cash economy. The discovery of gold was the first step to disintegrating many households and families in different communities during the gold rush [61]. There is also a negative side to the gold rush as it specifically led to irrevocable damages to streams, rivers, watersheds, and floodplains. And indispensably, it escalated the damages to environmental communities in nature [12]. Besides, some miners and emigrants were deprived of their land, hit, surrendered to unpaid labour, or murdered, and they were discriminated against and disrespected. Nowadays, in most countries, there are rules for mining that must be observed before taking any action.

In the next section, the mathematical model of the GRO metaheuristic algorithm will be introduced based on previous discussions.

3.6. Mathematical model and algorithm

In this section, first, mathematical models for gold prospectors, migration, mining, and collaboration between prospectors will be stated, and then the GRO metaheuristic algorithm will be explained.

3.6.1. Gold prospectors modeling

The GRO algorithm imitates the main events of the gold rush. In the GRO metaheuristic algorithm, prospectors play the same role as the population of GA and particles in the PSO algorithm. The location of gold prospectors is stored in a matrix named M_{GP} that is expressed as equation (1). In this equation, x_{ij} denotes the location of prospector i at jth dimension. d is the dimension size, and n is the number of gold prospectors.

$$M_{GP} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ x_{21} & x_{22} & \dots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix} \tag{1}$$

An objective function is needed to evaluate gold prospectors during optimization, and evaluation values of gold prospectors are stored in an evaluation matrix M_F according to Equation (2). In this equation, x_{ij} is the location of the prospector i at jth dimension, and f is the evaluation function.

$$M_F = \begin{bmatrix} f(x_{11} & x_{12} & \dots & x_{1d}) \\ f(x_{21} & x_{22} & \dots & x_{2d}) \\ \vdots & \vdots & \ddots & \vdots \\ f(x_{n1} & x_{n2} & \dots & x_{nd}) \end{bmatrix} \tag{2}$$

3.6.2. Migration of prospectors

After discovering a gold mine, gold prospectors migrate to it to obtain gold. The location of the richest gold mine is the optimal point of search space during the execution of the metaheuristic algorithm. Since the precise location of it is unknown, the location of the best gold prospector is used as an estimate for the location of the best gold mine (Figures 1, 2)

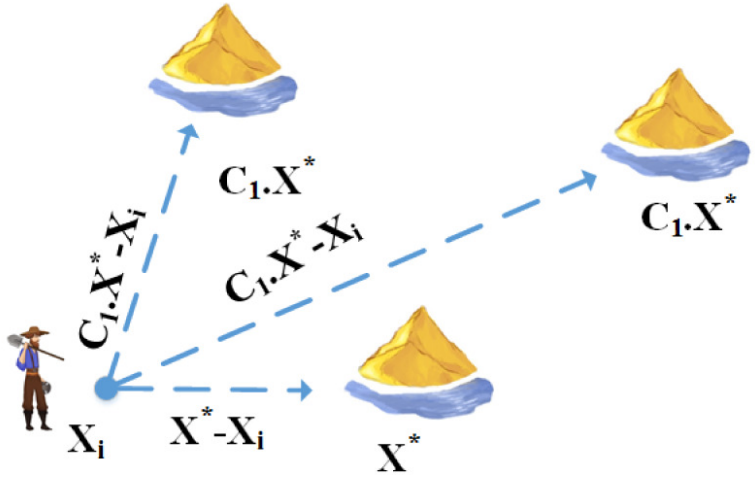

Figure 1. Schematic view of equation (3) in two dimensions

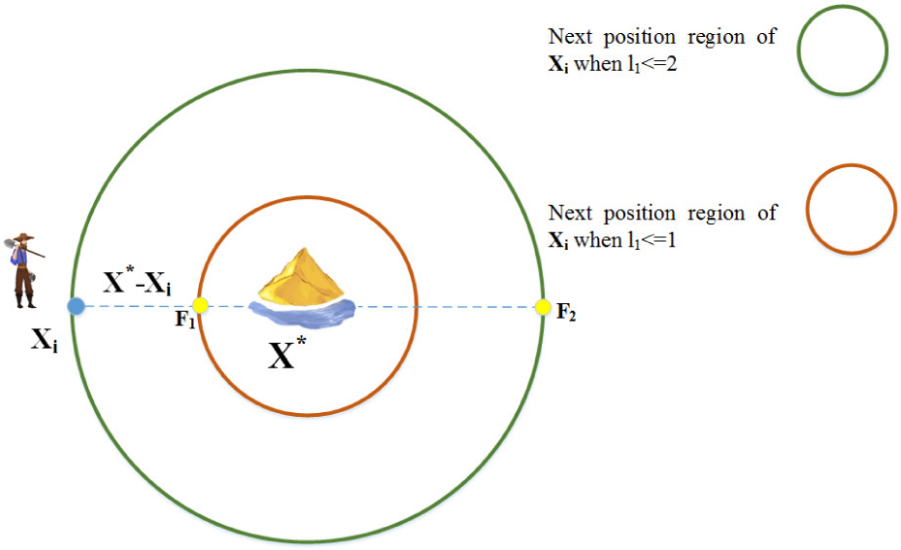

Figure 2. Schematic view of the operator of migration toward gold mine in two dimensions

For modeling the migration of a gold prospector to the gold mine, Equations (3) and (4) are applied.

$$\overrightarrow{D}_1 = \overrightarrow{C}_1 . \overrightarrow{X}^* (t) - \overrightarrow{X}_i (t) \tag{3}$$

$$\overrightarrow{X\text{new}}_i\,(t+1) = \overrightarrow{X}_i\,(t) + \overrightarrow{A}_1.\,\overrightarrow{D}_1 \tag{4}$$

where $\overrightarrow{X}^*$, $\overrightarrow{X}_i$ and t are the location of the best gold mine, location of the gold prospector i, and current iteration t, respectively. $\overrightarrow{X\text{new}}_i$ is the new location of the gold prospector i, and $\overrightarrow{A}_1$ and $\overrightarrow{C}_1$ are vector coefficients calculated by

$$\overrightarrow{A}_1 = 1 + l_1(\overrightarrow{r}_1 - \frac{1}{2}) \tag{5}$$

$$\overrightarrow{C}_1 = 2\overrightarrow{r}_2 \tag{6}$$

where $\overrightarrow{r}_1$ and $\overrightarrow{r}_2$ are random vectors with value in the range $[0, 1]$. l_1 is the convergence component defined by equation (7); If e is equal to one, it decreases linearly from 2 to $\frac{1}{\max_{\text{iter}}}$, and for values more than 1, it decreases non-linearly. Figure 3 shows the results for power values 1 and 2. Figure 4 shows the range of changes in A_1 over course of iterations.

$$l_e = \left(\frac{\max_{\text{iter}} - \text{iter}}{\max_{\text{iter}} - 1}\right)^e \left(2 - \frac{1}{\max_{\text{iter}}}\right) + \frac{1}{\max_{\text{iter}}} \tag{7}$$

Gold seekers settle in nearby places after they migrate to an assumed gold mine. Vector A_1 in Equation (5) is used for the mathematical modelling of migration. When its value is 1, the seeker migrates to the assumed gold mine, and lower and upper values mean migration to a place between the seeker location and the mine or a place after it, respectively. As seen in Figure 4, its value range varies around 1. Seekers may also consider similar regions other than the exact place of explored gold mines in order to migrate to find gold. Vector C_1 in Equation (3) is presented to mathematically model this matter. A value of 1 denotes migration to the exact place of a gold mine, and other values are considered for migration to places with similar coordinates. Figure 1 illustrates an instance of this equation in two dimensions that can be generalized to d dimensions. A schematic model of migration in two dimensions is illustrated in Figure 2. For example, when the value of (A_1, C_1) is $(0.5, 1)$, migration occurs at place F_1, and when the value is $(2, 1)$, migration occurs at place F_2. Although, this schematic can be extended to d-dimensional space; as can be observed, the migration region gradually approaches the mine location according to the value of l_1.

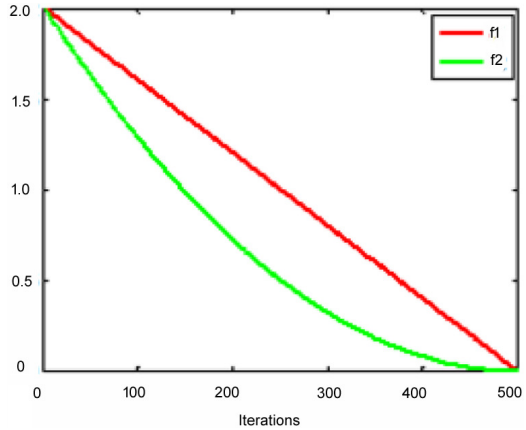

Figure 3. Graph for values of l_1 and l_2 during algorithm iteration

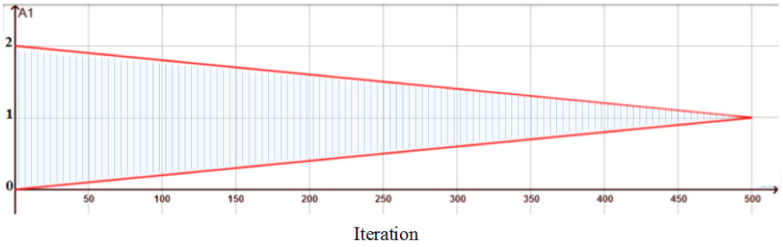

Figure 4. Range of changes in A_1 during algorithm iterations

3.6.3. Gold mining (gold panning)

Each gold prospector mines gold areas to find more gold. For mathematical modeling, the location of each gold prospector is regarded as an approximate location of a gold mine. The relevant mathematical relations of gold mining are considered as

$$\overrightarrow{D}_2 = \overrightarrow{X}_i(t) - \overrightarrow{X}_r(t) \tag{8}$$

$$\overrightarrow{X\text{new}}_i(t+1) = \overrightarrow{X}_r(t) + \overrightarrow{A}_2 . \overrightarrow{D}_2 \tag{9}$$

where $\overrightarrow{X}_r$, $\overrightarrow{X}_i$, t, and $\overrightarrow{X\text{new}}_i$ represent the location of a randomly selected gold prospector r, location of the gold prospector i, current iteration t, and the new location of the gold prospector i, respectively. $\overrightarrow{A}_2$ is the vector coefficient calculated by equation (10).

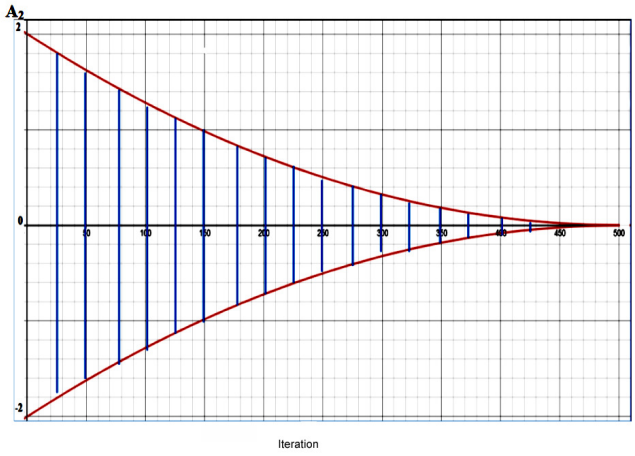

Figure 5. Permissible range of changes of A_2 during algorithm iteration

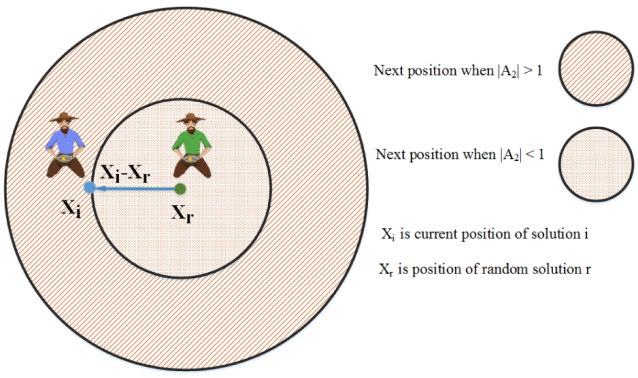

Figure 6. Schematic view of gold mining (panning) in 2D

In this equation, parameter l_2 is used instead of parameter l_1 to increase the exploitation capability of the mining method. The range of changes in the latter parameter is shown in Figure 5.

$$\overrightarrow{A}_2 = 2l_2\overrightarrow{r_1} - l_2 \tag{10}$$

To better understand the equations (8) and (9), a schematic 2D view of gold mining is illustrated in Figure 6. According to this figure, the gold prospector may approach or get away from the specified mine given the value of A_2. This concept can be extended to a d-dimensional search space.

3.6.4. Collaboration between prospectors

Since gold prospecting is sometimes performed through teamwork, the mathematical modeling of equations (11) and (12) is used to illuminate the collaboration between gold prospectors where g_1 and g_2 are two randomly selected gold prospectors. In this case, three-person collaboration is realized between prospectors i, $g1$, and $g2$, and $\overrightarrow{D}_3$ is the collaboration vector. The schematic view of the collaboration between prospectors in two dimensions is shown in Figure 7 that can be generalized into d dimensions.

$$\overrightarrow{D}_3 = \overrightarrow{X}_{g_2}(t) - \overrightarrow{X}_{g_1}(t) \tag{11}$$

$$\overrightarrow{X\text{new}}_i(t+1) = \overrightarrow{X}_i(t) + \overrightarrow{r}_1.\overrightarrow{D}_3 \tag{12}$$

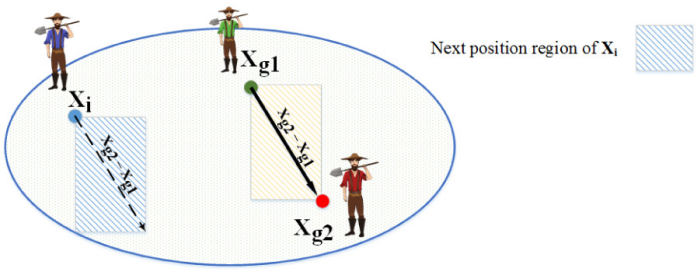

Figure 7. Schematic view of the collaboration between prospectors in two dimensions

As shown in Figure 7, the desired region for seeking gold is specified by seekers g_1 and g_2, but the exact seeking location in this region is determined randomly by seeker i. Accordingly, three-person cooperation is formed in this manner.

3.6.5. Prospectors relocation

Gold prospectors are constantly moving, and one critical parameter in their decision-making is to obtain more gold. Therefore, to decide whether the prospector remains in its previous location or moves to a new one, these two are compared via the evaluation function. In this process, the gold prospector updates its location if there was an improvement in the value of the objective function; otherwise, it remains in the previous location, which is modelled as equation (13) in minimization problems.

$$\overrightarrow{X}_i(t+1) = \overrightarrow{X\text{new}}_i(t+1) \text{ if } f(\overrightarrow{X\text{new}}_i(t+1)) < f(\overrightarrow{X}_i(t)) \tag{13}$$

3.6.6. Domain control

If the location of $X\mathrm{new}_{i,\,d}$ at dimension d is at the range between lower and higher bounds of dimension d, a new location is considered; otherwise, the previous location of $X_{i,\ d}$ remains unchanged.

Based on the previous concepts, the GRO algorithm begins with an initial population of prospectors randomly placed in the search space. The best solution obtained during searching is chosen as the location of the best gold mine (global optimum). Each prospector takes a new location in each iteration using one of the migration, gold mining, and collaboration methods. If the amount of gold (value of an objective function) in the new location is more than the current location (a reduction in objective function during minimization and an increase in objective function during maximization), the prospector moves to the new location; this process goes on until the iteration loop ends. The best solution found so far is considered the final solution of the algorithm.

3.6.7. Exploitation and exploration of GRO

In the gold mining method, a gold prospector searches gold mines to find more gold. When $|A_2| < 1$, the relative distance between the prospector and chosen mine reduces, and the algorithm's exploitation increases.

Since gold mining is done in teamwork, in addition to the essence of an individual's effort, gold prospectors prefer working in teams to find more gold. At first, since there is no agreement on the mining method and its place, teamwork is focused on investigating and searching, and then by achieving a collective agreement, more gold can be obtained. Considering that each prospector's next location in the collaboration method is calculated by the distance between two random prospectors' locations, the exploration ability of the algorithm is high at the start of the execution due to the spread distribution of prospectors' locations. Since prospectors gradually get close to the best gold mine, their distances to each other reduce, providing a higher exploitation ability. The random coefficient r_1 in this operator increases the ability of the algorithm for random searching in collaboration space.

The pseudocode of the algorithm is depicted in algorithm 1. The GRO algorithm can solve optimization problems theoretically. Some of its key points are as described:

- The solution of each search agent during the searching iteration is stored or enhanced. Thus, the best solution found during algorithm runtime is a member of the population.
- Parameters l_1 and l_2 cause the algorithm to change from the exploration phase to the exploitation phase. l_2 is used for a higher focus on the exploitation phase compared to l_1 and indicates an effortful process for gold mining.
- The parameters of the GRO algorithm do not require to be set for different problems.
- Applying the location of the best solution in the migration operator helps find inspirational space for finding a better solution.
- Parameter $|A_1|$ gradually limits the search region around the best solution, causing better exploitation.
- Utilising parameter A_2 makes the algorithm vary between exploitation $|A_2| < 1$ and exploration $|A_2| > 1$ phases.

- Applying the location of a random search agent in a mining operator increases the algorithm's potential for escaping the local optimum.
- Applying C_1 helps new solutions resulting from migration operators place in a hypersphere with different radii.
- In the collaboration method, using the locational difference between two solutions for the search region provides an adaptive dynamic radius for prospecting. In this method, in the early iterations, when the distance between search agents is long, the searching radius is large, and the exploration phase is performed by searching for a global optimum, and when the distance between them reduces, the searching process is done in lower radius and exploitation phase is done by local search.

Initialize the gold prospectors' population $X_i,\ \ i = 1, 2, \ldots, N$
Initialize the gold prospectors' new positions $X\text{new}_i = X_i\ ,\ \ i = 1, 2, \ldots, N$
Initialize $t, l_1,\ l_2$
X^* is the best search agent
while $t \leq$ maximum number of iterations **do**
  for each search agent i **do**
    Calculate the fitness of the current search agent at new position $X\text{new}_i$
    Update position of current search agent X_i according to equation (13)
    Update best search agent X^*
  end
  Update $l_1,\ l_2$ by equation (7)
  for each search agent i **do**
    calculate the next position of current search agent $X\text{new}_i$ with one of the migration, mining or collaboration methods
  end
  $t \leftarrow t + 1$;
end
return X^*;

Algorithm 1. GRO algorithm pseudocode

3.6.8. Comparison of GRO and GWO algorithms

In this section, the GRO algorithm is compared to the GWO algorithm:

- The GRO was inspired by gold prospectors' seeking approach during the Gold Rush Era. However, GWO was inspired by the grey wolves' food-seeking habits.
- The wolves' location in the GWO algorithm is obtained by changing the location of alpha, beta, and delta wolves. The result of these wolves' attacks determines the others' location. However, in the GRO algorithm, only the immigration method uses the location of the best gold prospector. Moreover, the basis for the prospector's new location is his current location and the movement toward the best prospector.
- The mining and collaboration methods in the GRO algorithm aim to perform the prospections in a random prospector's location and cooperation with two random prospectors, respectively. However, there is no similar concept in the GWO algorithm. Therefore, all prospectors participate actively in gold prospection in GRO, while alpha, beta, and delta wolves lead the entire group in GWO.

4. Result and discussion

Noting that metaheuristic algorithms have a random nature, one algorithm's efficiency may differ from one execution to another. Therefore, it is required that algorithm behaviour be examined in different problems several times [36]. In this section, 23 benchmark functions, which were used by many researchers to assess metaheuristic algorithms, were applied. Their characteristics are given in Table 1[1]. These functions are of three unimodal, multimodal, and fixed-dimension multimodal types. The first group lacks a local optimum and has a global optimum that is mainly applied to evaluate algorithm exploitation. The second and third types have multiple local optima with a global optimum and are employed for assessing algorithm exploration. Also, six composite functions were selected from CEC2005 [41], and their definitions are given in Table 2 (dimension 10, range [–5, 5], $f_{\min} = 0$). These functions are shifted, rotated, expanded, and a combined type of classic functions and are more complex, providing a proper platform for assessing the quality of metaheuristic algorithm solutions. For validating the effectiveness of the suggested GRO algorithm, it was compared with genetic algorithm (GA), particle swarm optimization (PSO), differential evolution (DE) [67], gravitational search algorithm (GSA) [60], improved grey wolf optimization (IGWO) [51], Salp swarm algorithm (SSA) [45], firefly algorithm (FA) [75], sine cosine algorithm (SCA) [44], water cycle algorithm (WCA) [17], whale optimization algorithm (WOA) [46], Komodo Mlipir algorithm (KMA) [68], and slime mold algorithm (SMA) [40].

Thirty search agents with 500 iterations for each algorithm were considered to solve these benchmark functions. For a fair assessment of algorithms, each algorithm was executed 30 times independently for each problem, and results were recorded. 15,000 iterations were carried out in the KMA runs. A summary of the statistical results is shown in Tables 3–6. In these tables, the average of the best solutions during 30 runtimes and their deviations are reported. The bold numbers indicate the best results of the algorithms compared. Underlined numbers represent the global optimum solutions with a standard deviation of 0 or less than the decimal numbers of a global solution. Moreover, the bold underlined numbers indicate the best solution with the global optimum guarantee.

4.1. Ability evaluation

One of the required features for an efficient metaheuristic algorithm is the exploitation ability that enables it to search for the best solution in the form of an inspirational region and propose better solutions. AS unimodal functions only have one optimal global solution, they can be used to examine the exploitation ability of metaheuristic algorithms. A 2D schematic view of these problems is shown in Figure 8. The results of executing the GRO algorithm on these functions are presented in Table 3. KMA and SMA algorithms obtained the best solutions, and the GRO algorithm could also generate competitive and high-quality solutions. One of the other main capabilities of the metaheuristic algorithm for avoiding the local optimum and finding the optimal global solution is the exploration in which the algorithm leaves the current proper region for a more appropriate region. Further, the GRO algorithm was executed on six multimodal benchmark problems. The comparison results of the executions on multimodal test cases are given in Table 4 from which it can be deduced that the GRO algorithm can find solutions competitive with other algorithms. A schematic view of these problems is shown in Figure 9 in 2D.

[1]All tables presented in the paper, the reader will find in the Appendix.

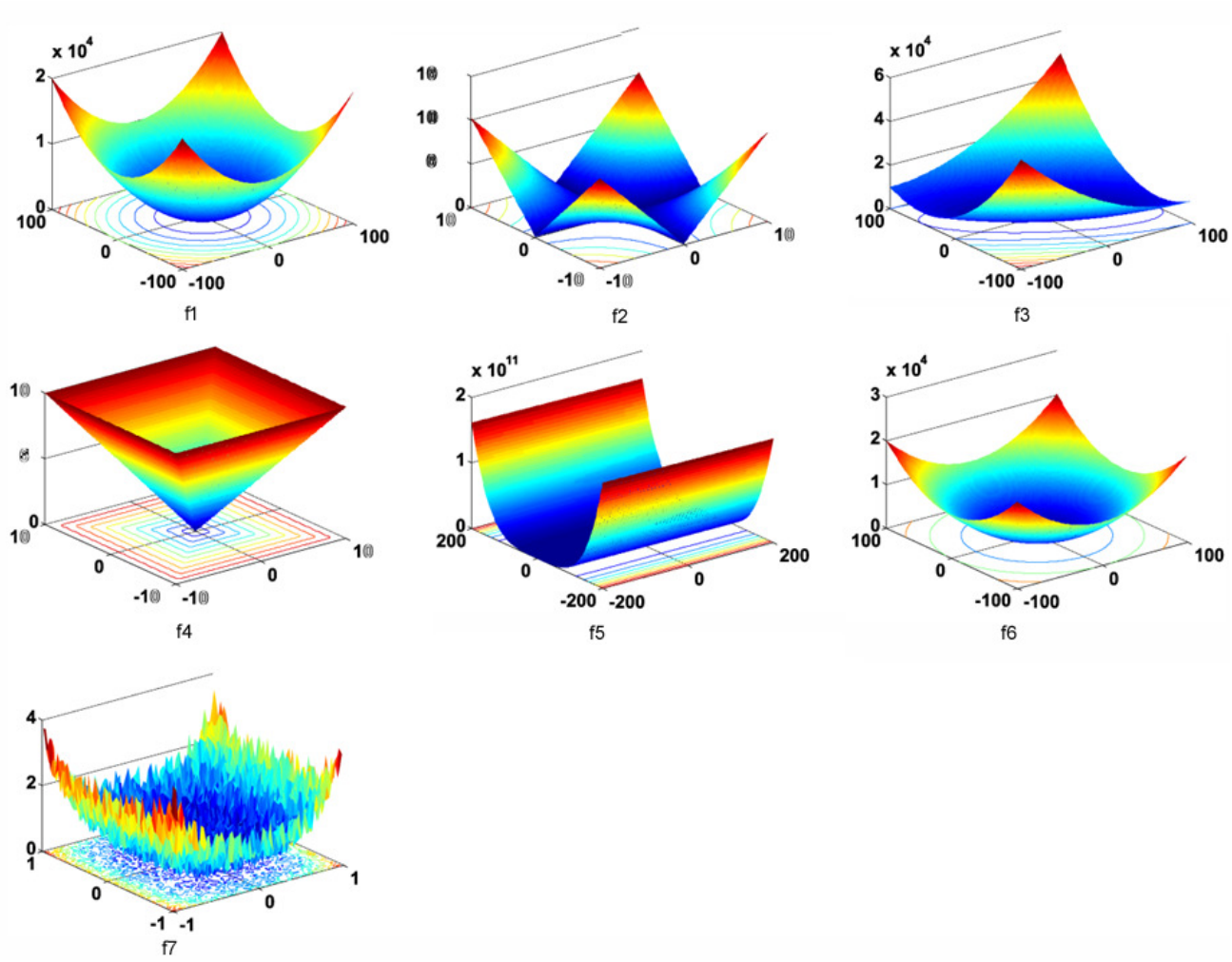

Figure 8. Unimodal benchmark functions in 2D

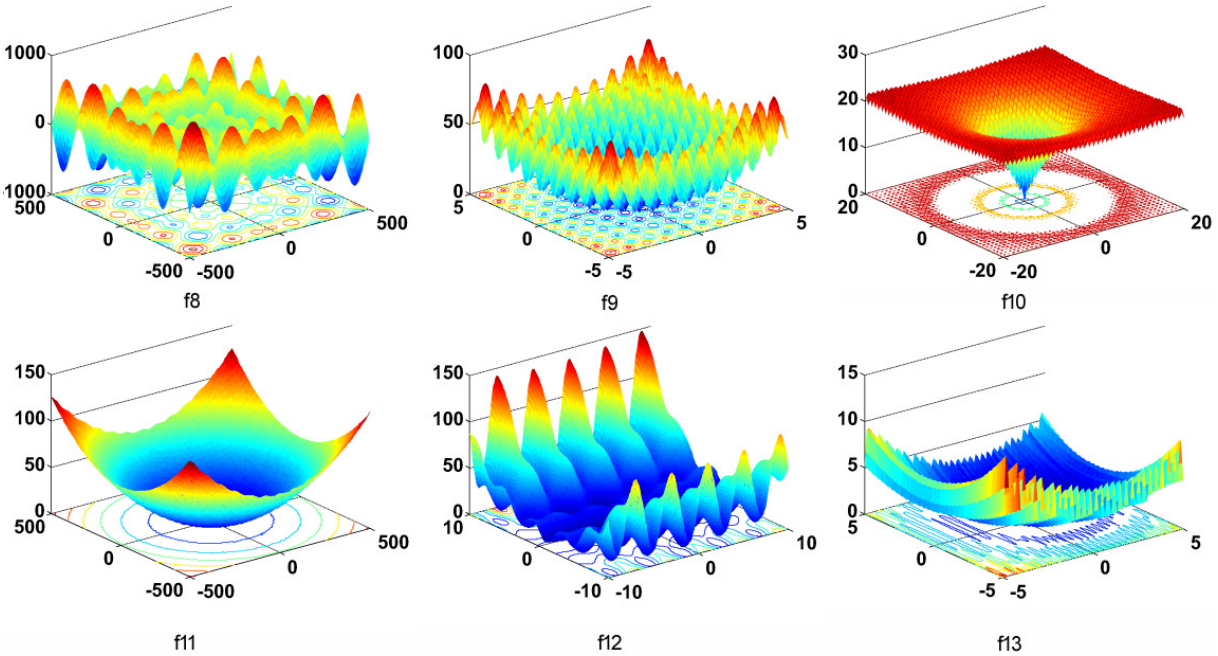

Figure 9. Multimodal benchmark functions in 2D

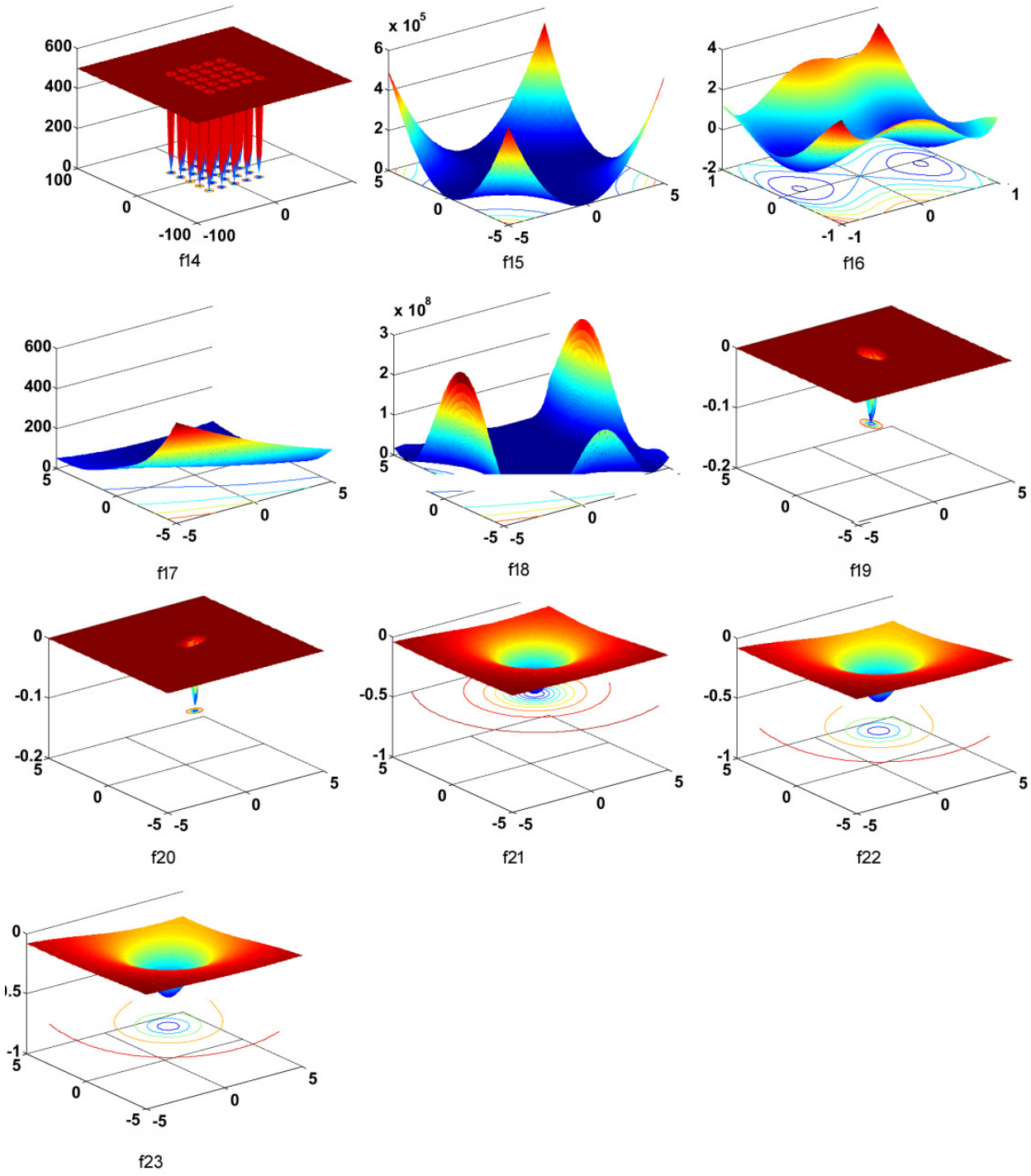

Figure 10. Fixed-dimension multimodal functions in 2D

In the third part, the results for executing the GRO algorithm on ten fixed-dimension multimodal problems in comparison to others are stated. Considering that multimodal functions have multiple local optimum solutions and only one global optimum solution, they are suitable for the exploration ability evaluation of metaheuristic algorithms. The results for applying the GRO algorithm on these functions are given in Table 5, and the schematic view is shown in Figure 10. In this table, the results for executing fixed-dimension multimodal test cases are mentioned. As is evident, in almost all of the test cases, it could find a more proper solution than other algorithms by converging to the global optimum, indicating the high exploration ability of the GRO algorithm in finding a global optimum solution and escaping the

local optimum. Moreover, as seen from the table, none of the other algorithms could guarantee the global optimum of all ten test functions.

4.2. Local minimum avoidance

For better examining the metaheuristic algorithm performance in more complex problems, test cases of the fourth group, proper for evaluating the metaheuristic algorithms' exploration ability, were studied in this paper. A 2D schematic view of these is shown in Figure 11, and a summary of the execution of the GRO algorithm, along with others, is mentioned in Table 6. As can be seen, the GRO algorithm could obtain a better average solution in three out of six test cases and had an acceptable performance in the other three.

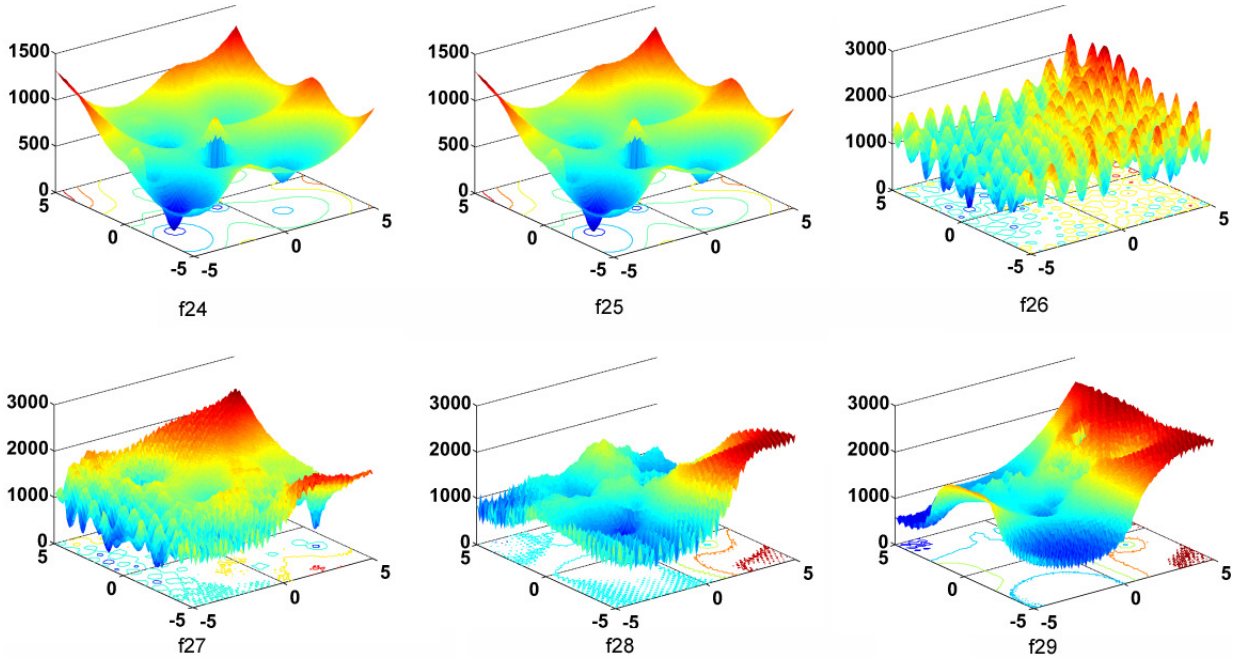

Figure 11. Composite benchmark functions in 2D

4.3. Convergence speed evaluation

The rate of convergence to the final solution is one of the crucial features of a metaheuristic algorithm. This property was investigated by taking into account the average of the best solution obtained in each iteration on 30 executions of each algorithm. The KMA algorithm has not been mentioned in these comparisons due to different iterations in every run.

As shown in Figure 12, a high convergence rate and providing competitive solutions are desirable properties of the GRO algorithm. Furthermore, this algorithm is likely to find the optimal global solution, and therefore, it has a faster convergence rate than most other algorithms.

In multimodal test cases, the final solution is more important since these have many local optima that complicate finding the global optimum. Hence, this algorithm must be capable of avoiding the local optimum and approaching the global optimum. As can be seen in Figures 13 and 14 regarding the comparison between the convergence rates of algorithms in solving these problems, GRO algorithm, in addition to high convergence speed, has succeeded in approaching the global optimal solution.

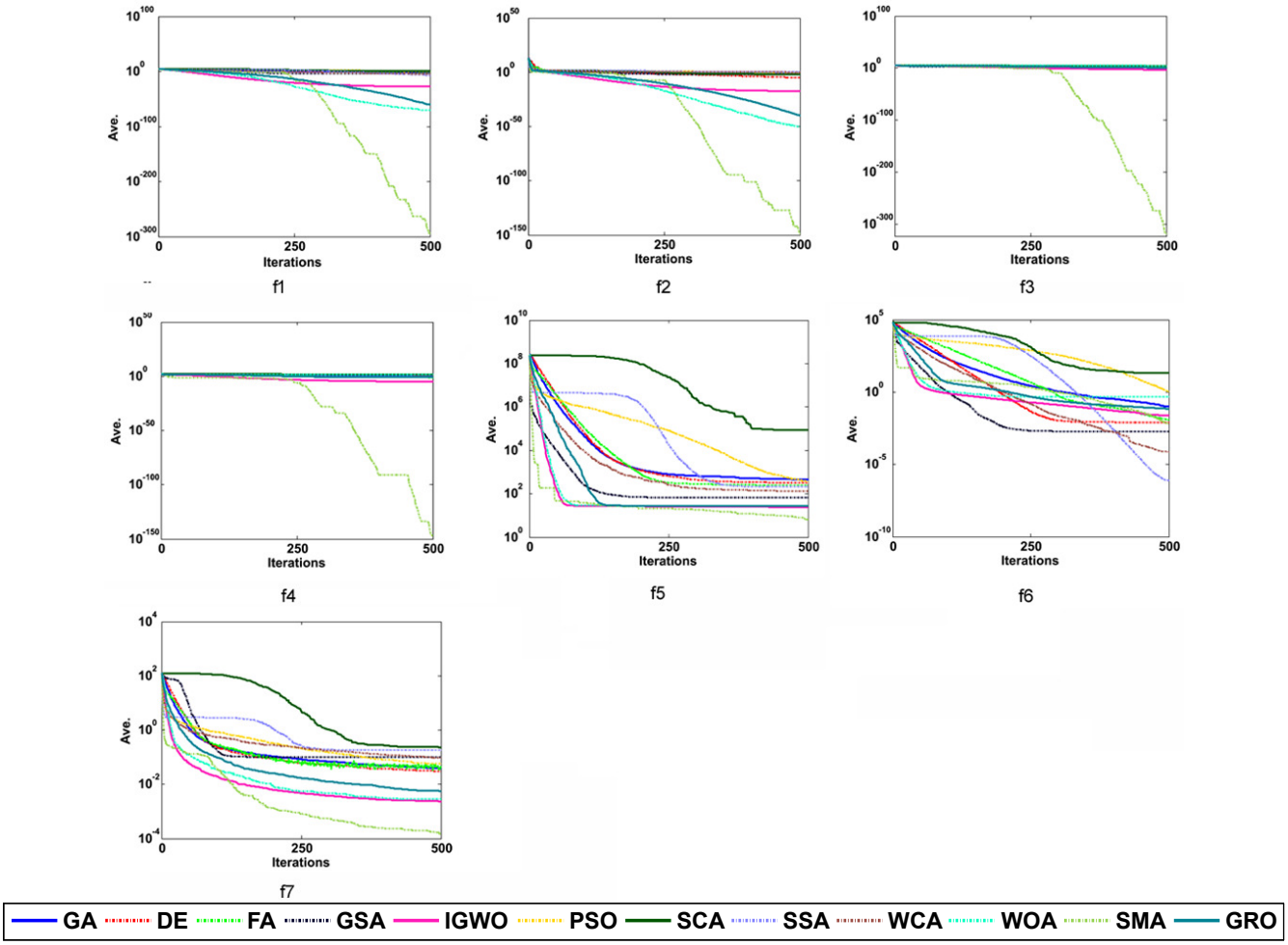

Figure 12. Convergence curve of unimodal benchmark functions

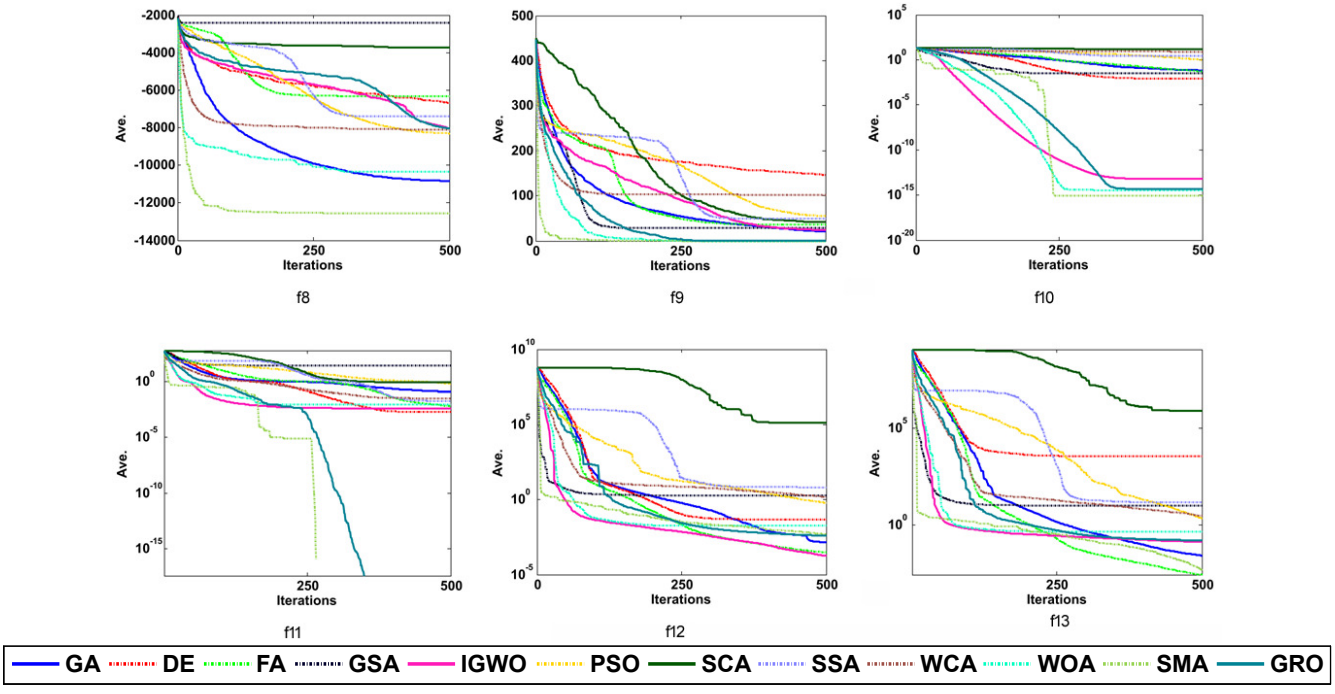

Figure 13. Convergence curve of multimodal benchmark functions

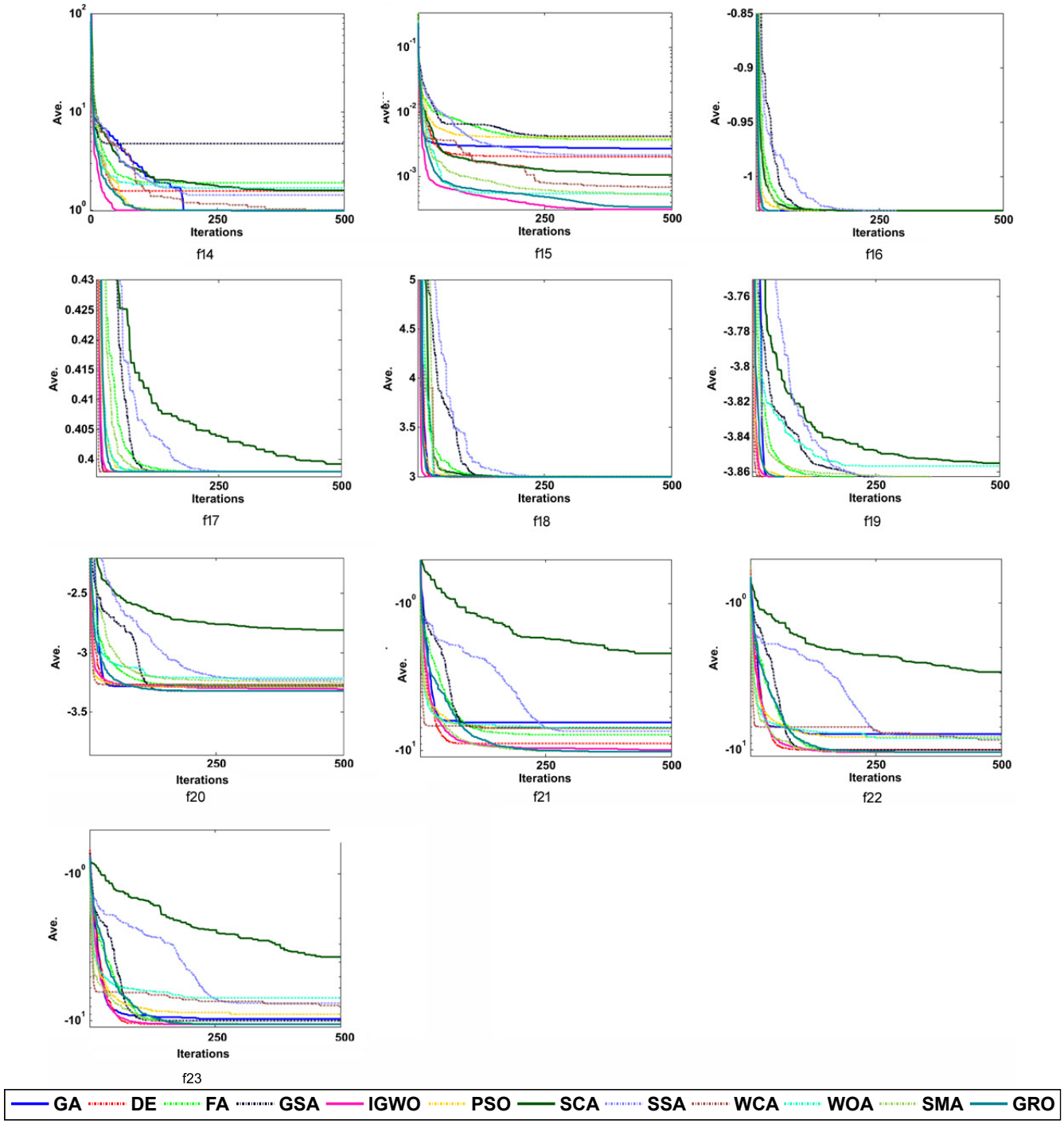

Figure 14. Convergence curve of fixed-dimension multimodal functions

In Figure 15, a comparison of the results for executing CEC2005 test cases is represented. As can be seen, the GRO algorithm had a better performance than other algorithms during the first half of the execution and outperformed others in the second half by using exploration ability and achieving a better average solution.

4.4. Global optimum guarantee

Table 7 presents the comparison between algorithms in terms of the global optimum guarantee, defined as certainly obtaining of global optimum with a standard deviation of 0 or less than its decimal numbers.

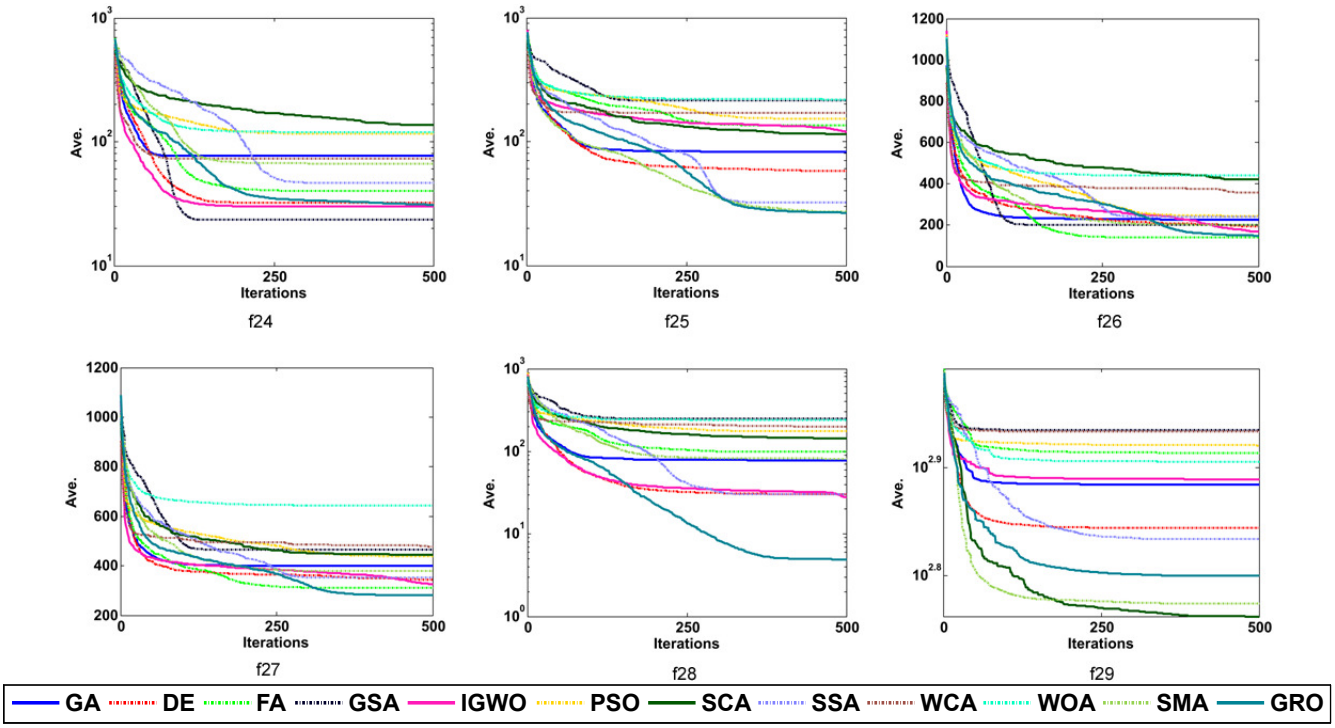

Figure 15. Convergence curve of composite benchmark functions

The KMA and GRO algorithms have the maximum global optimum guarantee with 51.7% (15 of 29 functions) and 37.9% (11 of 29 functions), respectively. However, SCA and WOA algorithms show the least global optimum guarantee with 0% and 3.7%, respectively. The global optimum guarantee is a comparative criterion in algorithms; however, its results are not comprehensive since the quality of solutions is not specified when the algorithm fails to reach the global optimum.

In conclusion, the effectiveness of the GRO algorithm was proven experimentally in 29 problems. It was found that in most cases, the algorithm was able to converge global optimum fast and effectively.

4.5. Statistical analysis

Although comparison of the results based on the average and standard deviation is an intuitive method for evaluating the performance of various algorithms, it is not comprehensive since the algorithm may not perform one execution as well as others, leading to a greater average value. Also, because the performance of algorithms is compared separately for each test case, it is essential to investigate whether it can be stated with certainty that one algorithm is superior to others or not. In this regard, statistical methods may be used. For a reliable scientific comparison between results obtained from each algorithm, a statistical test was carried out to illustrate the importance of the results. Here, a non-parametric Friedman rank test, which is of multiple comparison types, was used to rank algorithms. More explanations about non-parametric tests can be found in [14]. Based on this test, a lower tank indicates higher superiority and more effectiveness of the algorithm. Table 8 shows the results of this test. According to this table and the average value and standard deviations of ranks, the GRO algorithm provides the best solution method and more robustness compared to others. The obtained P-values indicate a significant difference between compared algorithms. In Table 9, a summary of the Wilcoxon signed-rank test is presented, in which positive and negative ranks are, respectively, the sum of ranks for problems with superiority of

the second algorithm to the first one and vice versa. As is shown in this table, at a significance level of 5%, the GRO algorithm was better than other algorithms except for IGWO, KMA, and SMA, since most comparisons showed that p-values were less than 5%, and negative ranks had greater values than positive ranks.

5. Engineering optimization problems

In the previous section, the GRO algorithm performance was studied in unconstrained problems. There is another group of problems that are crucial in engineering optimizations since they have constraints. These are known as constraint global optimization problems, and their general form with inequality, equality, and lower and upper bound constraints is based on equation (14).

$$\begin{aligned} &\min f(x) \\ &\text{s.t.} \\ &\quad g_j(x) = 0, \quad j = 1, 2, \ldots, k \\ &\quad g_j(x) \leq 0, \quad j = k+1, k+2, \ldots, m \\ &\quad l_i \leq x_i \leq u_i, \quad i = 1, 2, \ldots, n \end{aligned} \tag{14}$$

where $f(x)$ is the objective function, $g_j(x) = 0$ are equality constraints, $g_j(x) \leq 0$ are inequality constraints, and l_i and u_i are lower and upper bounds of parameter x_i, respectively.

In this study, three classic engineering problems mainly used by other researchers were solved by the GRO algorithm. The number of gold prospectors and iterations was considered to be 30 and 1000, respectively, and the total number of objective function evaluations was 30,000. A summary of the results for 30 independent executions of the GRO algorithm is given in Tables 10–15. For precisely comparing the proposed algorithm with other algorithms in solving engineering problems, 9 papers solving all of these engineering problems were selected. The result of the proposed algorithm was compared with the Lévy flight distribution (LFD) [27], bat algorithm (BA) [20], grey wolf optimizer (GWO) [47], co-evolutionary particle swarm optimization (CPSO) [24], co-evolutionary differential evolution (CDE) [28], transient search optimization (TSO) [58], multi-verse optimization algorithm (MVO) [64], bidirectional butterfly optimization algorithm (BBOA) [65], and hybrid genetic algorithm (HGA) [74].

5.1. Constraint handling

Constraint handling is one of the main challenges in solving real problems with metaheuristic algorithms. Coello [10] stated five constraint handling methods, including 1) penalty functions, 2) special representation and operators, 3) repair algorithms, 4) separation of objectives and constraint, and 5) hybrid methods. The simplest are penalty functions. There are various penalty functions: static penalty, annealing penalty, co-evolutionary penalty, adaptive penalty, dynamic penalty, and the death penalty. In this paper, a static penalty method was utilised for constraint handling. This method was implemented in [76] as well.

5.2. Pressure vessel design

In pressure vessel design, we aim to minimize overall costs, including material, forming, and welding costs of the cylindrical tank. There are four parameters in this problem that need to be optimised. These are the thickness of the head T_h (x_2), the thickness of the shell T_s (x_1), the inner radius R (x_3), and the length of the cylinder section of the vessel without semi-spherical head L (x_4). The general schematic of the problem is illustrated in Figure 16.

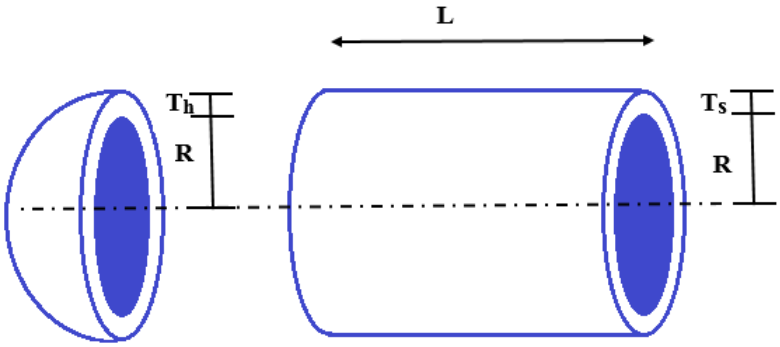

Figure 16. Pressure vessel design problem

The mathematical model of the problem with its constraints is according to equation:

$$\begin{aligned}
&\min f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
&\text{s.t.} \\
&\quad g_1(x) = -x_1 + 0.0193x_3 \leq 0 \\
&\quad g_2(x) = -x_2 + 0.00954x_3 \leq 0 \\
&\quad g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0 \\
&\quad g_4(x) = x_4 - 240 \leq 0 \\
&\quad 0 \leq x_1 \leq 99 \\
&\quad 0 \leq x_2 \leq 99 \\
&\quad 10 \leq x_3 \leq 200 \\
&\quad 10 \leq x_4 \leq 200
\end{aligned} \tag{15}$$

Statistical results of executing GRO are mentioned in Table 10. In this table, this algorithm was compared with advanced algorithms. As can be observed, the suggested algorithm performed better than others in finding the best solution. In Table 11, values of the best solution and objective function parameters are mentioned.

5.3. Tension/compression spring design problem

In the tension/compression problem, we aim to minimize the weight of a tension/compression spring. Three parameters need to be optimised, which are as follows: Wire diameter d (x_1), mean coil diameter D

(x_2), and a number of active coils N (x_3). A schematic representation of the problem is demonstrated in Figure 17.

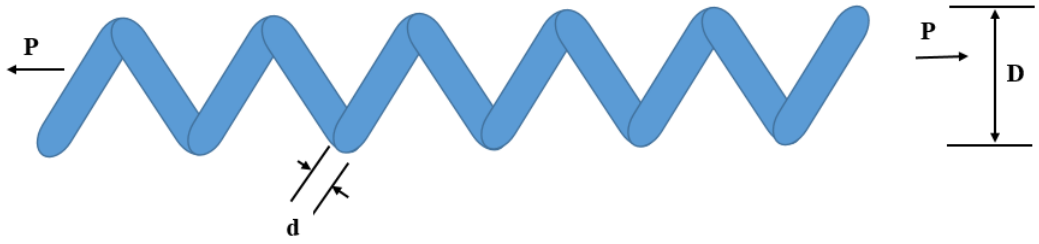

Figure 17. Tension/compression spring problem.

The associated mathematical model is based on the equation

$$\begin{aligned}
&\min f(x) = (x_3+2)x_2x_1^2 \\
&\text{s.t.} \\
&\quad g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0 \\
&\quad g_2(x) = \frac{4x_2^2 - x_1x_2}{12566\,(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \\
&\quad g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \\
&\quad g_4(x) = \frac{x_1+x_2}{1.5} - 1 \leq 0 \\
&\quad 0.05 \leq x_1 \leq 2 \\
&\quad 0.25 \leq x_2 \leq 1.30 \\
&\quad 2 \leq x_3 \leq 15
\end{aligned} \tag{16}$$

The efficiency of the GRO algorithm was compared with methods mentioned in previous literature. Table 12 shows a summary of the statistical results obtained from the GRO algorithm as compared with other advanced algorithms. Based on this table, the GRO algorithm performed well to find competitive solutions compared to other algorithms and the best solution obtained by GRO, BA, and TSO was superior to others. Table 13 displays the parameters of the best solutions obtained by compared algorithms.

5.4. Welded beam design problem

Welded beam design is one of the problems investigated comprehensively by many researchers. This problem aims to minimize the costs of welded beams. Figure 18 displays a schematic of the welded beam. Four decision-making parameters need to be optimised, which are as follows: The weld thickness h (x_1), the length of the attached part of the bar l (x_2), the height t (x_3), and thickness of the bar b (x_4). The mathematical model of the problem is described by

$$
\begin{aligned}
&\min f(x) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 \left(x_2 + 14\right) \\
&\text{s.t.} \\
&\quad g_1(x) = \tau(x) - \tau_{\max} \leq 0 \\
&\quad g_2(x) = \sigma(x) - \sigma_{\max} \leq 0 \\
&\quad g_3(x) = \delta(x) - \delta_{\max} \leq 0 \\
&\quad g_4(x) = x_1 - x_4 \leq 0 \\
&\quad g_5(x) = P - P_c(x) \leq 0 \\
&\quad g_6(x) = 0.125 - x_1 \leq 0 \\
&\quad g_7(x) = 1.1047 x_1^2 + 0.04811 x_3 x_4 \left(14 + x_2\right) - 5 \leq 0 \\
&\quad 0.1 \leq x_1 \leq 2 \\
&\quad 0.1 \leq x_2 \leq 10 \\
&\quad 0.1 \leq x_3 \leq 10 \\
&\quad 0.1 \leq x_4 \leq 2 \\
&\quad \tau(x) = \sqrt{\left(\tau'\right)^2 + \frac{x_2}{2R} 2\tau''\tau' + \left(\tau''\right)^2} \\
&\quad \tau' = \frac{P}{\sqrt{2} x_1 x_2}, \quad \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right) \\
&\quad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \\
&\quad J = 2\left\{\sqrt{2} x_1 x_2 \left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \\
&\quad \sigma(x) = \frac{6PL}{x_4 x_3^2} \\
&\quad \delta(x) = \frac{4PL^3}{E x_4 x_3^3} \\
&\quad P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2 x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \\
&\quad P = 600 \text{ lb} \\
&\quad L = 14 \text{ in} \\
&\quad \delta_{\max} = 0.25 \text{ in} \\
&\quad \tau_{\max} = 13,600\, psi \\
&\quad \sigma_{\max} = 30,000\, psi \\
&\quad E = 30 \cdot 10^6\, psi \\
&\quad G = 12 \cdot 10^6\, psi
\end{aligned}
\tag{17}
$$

Welded beam problem was optimised using the GRO algorithm, and the results were compared with others shown in Table 14. According to this table, the GRO algorithm had the best performance in finding the best solution with a better average, worst solution, and standard deviation. Table 15 displays the parameters of the best solutions obtained by compared algorithms.

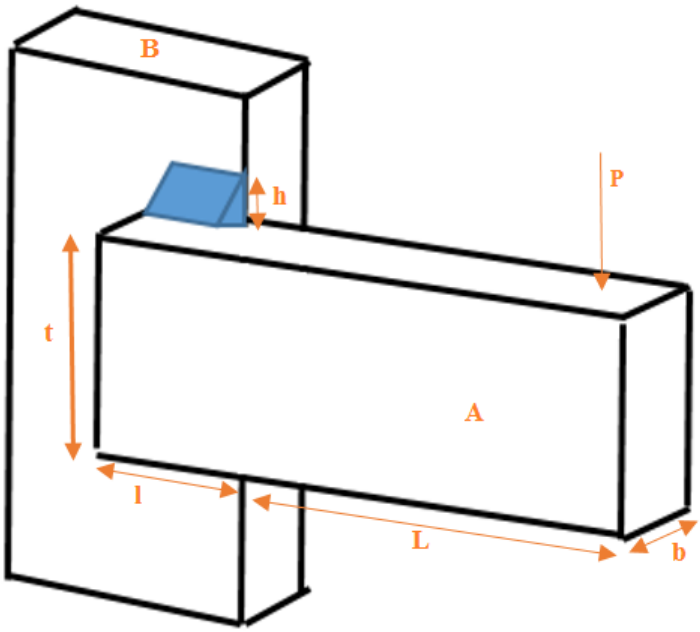

Figure 18. Welded beam problem

The performance of the GRO algorithm in three well-known engineering problems was examined briefly in this section. The best solutions were obtained for the first and third problems with acceptable performance in the second problem as compared with most of the algorithms. Considering the presence of constraints in these problems, the proposed method was successful in solving these types of problems.

6. Conclusion

A population-based metaheuristic algorithm, called GRO, was suggested that was inspired by the gold rush and imitated the gold prospecting by prospectors. Twenty-nine benchmark problems were used for assessing the introduced algorithm. The algorithm exploitation was compared with 12 known meta-heuristic algorithms, including GA, FA, DE, GSA, IGWO, PSO, SCA, SSA, WCA, WOA, KMA, and SMA, and the results have illuminated that the proposed algorithm was capable of generating qualitative solutions competitive with other algorithms that can escape local optimum and reach global optimum, with a higher convergence rate than most of the examined algorithms. Two statistical non-parametric tests were employed for a more detailed investigation of the results. Based on the Friedman test, the GRO algorithm had the best average rank with the least standard deviation, and according to Wilcoxon signed-rank test, at a significance of 5%, it was better than 9 out of 12 of the compared algorithms. In the following, the results of this algorithm in three engineering constraint problems were studied, and they were compared with nine other studies. In two problems, a better solution was achieved, and in one, this algorithm was able to generate competitive solutions with the best of the compared algorithms.

This study provides a proper platform for yet-to-come investigations in many ways. First, by expanding the GRO algorithm into binary and multi-objective types, it can solve the binary and multi-objective problems. Second, evolutionary operators, such as mutation, are suggested to improve algorithm performance. Third, the suggested algorithm can be used in combination with others so that the strengths of other algorithms can improve this algorithm's performance. Fourth, the proposed algorithm can be utilised to solve complex real-world problems and find the optimal solution for other problems.

References

[1] Abdel-Basset, M., Abdel-Fatah, L., and Sangaiah, A. K. Metaheuristic algorithms: A comprehensive review. In *Computational intelligence for multimedia big data on the cloud with engineering applications*, A. K. Sangaiah, M. Sheng and Z. Zhang, Eds., Academic Press, 2018, pp. 185–231.

[2] Abedinpourshotorban, H., Shamsuddin, S. M., Beheshti, Z., and Jawawi, D. N. A. Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm. *Swarm and Evolutionary Computation 26*, (2016), 8–22.

[3] Ahmadianfar, I., Bozorg-Haddad, O., and Chu, X. Gradient-based optimizer: A new metaheuristic optimization algorithm. *Information Sciences 540* (2020), 131–159.

[4] Andrist, R. K. *The Gold Rush*. New Word City, 2015.

[5] Atashpaz-Gargari, E., and Lucas, C. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In *2007 IEEE Congress on Evolutionary Computation*, (Singapure, 2007), IEEE, pp. 4661–4667, DOI: 10.1109/CEC.2007.4425083.

[6] Awad, N. H., Ali, M. Z., and Suganthan, P. N. Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving cec2017 benchmark problems. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, (Donostia, Spain), IEEE, pp. 372–379, DOI: 10.1109/CEC.2017.7969336.

[7] Azizi, M. Atomic orbital search: A novel metaheuristic algorithm. *Applied Mathematical Modelling 93* (2021), 657–683.

[8] Baluja, S. Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning. Technical report CMU-CS-94-163, Carnegie–Mellon University, Pittsburgh, PA, USA, 1994.

[9] Clay, K., and Jones, R. Migrating to riches? Evidence from the California gold rush. *The Journal of Economic History 68*, 4 (2008), 997–1027, DOI: 10.1017/S002205070800079X.

[10] Coello, C. A. C. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering 191*, 11-12 (2002), 1245–1287.

[11] Dai, C., Zhu, Y., and Chen, W. Seeker optimization algorithm. In *Computational Intelligence and Security*, Y. Wang, Y.-m. Cheung, H. Liu, Eds., Vol. 4456 of Lecture Notes in Computer Science, 2007, Springer, pp. 167–176.

[12] Dasmann, R. F. Environmental changes before and after the gold rush. *California History 77*, 4 (1998), 105–122.

[13] Dehghani, M., and Samet, H. Momentum search algorithm: A new meta-heuristic optimization algorithm inspired by momentum conservation law. *SN Applied Sciences 2*, 10 (2020), 1720.

[14] Derrac, J., García, S., Molina, D., and Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation 1*, 1 (2011), 3–18.

[15] Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., and Cosar, A. A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering 137* (2019), 106040.

[16] Dorigo, M., and Di Caro, G. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, (Washington, DC, USA, 1999), Vol. 2, IEEE, pp. 1470–1477.

[17] Eskandar, H., Sadollah, A., Bahreininejad, A., and Hamdi, M. Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures 110-111* (2012), 151–166.

[18] Faramarzi, A., Heidarinejad, M., Mirjalili, S., and Gandomi, A. H. Marine predators algorithm: A nature–inspired metaheuristic. *Expert Systems with Applications 152* (2020), 113377.

[19] Fathollahi-Fard, A. M., Hajiaghaei-Keshteli, M., and Tavakkoli-Moghaddam, R. Red deer algorithm (RDA): a new nature–inspired meta-heuristic. *Soft Computing 24*, 19 (2020), 14637–14665.

[20] Gandomi, A. H., Yang, X.-S., Alavi, A. H., and Talatahari, S. Bat algorithm for constrained optimization tasks. *Neural Computing and Applications 22*, 6 (2013), 1239–1255.

[21] Gandomi, A. H., Yang, X.-S., Talatahari, S., and Alavi, A. H. Metaheuristic algorithms in modeling and optimization. In *Metaheuristic applications in structures and infrastructures*, A. H. Gandomi, X.-S. Yang, S. Talatahari and A. H. Alavi, Eds., Elsevier, London, 2013, pp. 1-24.

[22] Hashim, F. A., Houssein, E. H., Mabrouk, M. S., Al-Atabany, W., and Mirjalili, S. Henry gas solubility optimization: A novel physics-based algorithm. *Future Generation Computer Systems 101* (2019), 646–667.

[23] Hatamlou, A. Black hole: A new heuristic optimization approach for data clustering. *Information Sciences 222* (2013), 175–184.

[24] He, Q., and Wang, L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence 20*, 1 (2007), 89–99.

[25] Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., and Chen, H. Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems 97* (2019), 849–872.

[26] Holland, J. H. Genetic algorithms. *Scientific American 267*, 1 (1992), 66–73.

[27] Houssein, E. H., Saad, M. R., Hashim, F. A., Shaban, H., and Hassaballah, M. Lévy flight distribution: A new metaheuristic algorithm for solving engineering optimization problems. *Engineering Applications of Artificial Intelligence 94* (2020), 103731.

[28] Huang, F.-z., Wang, L., and He, Q. An effective co-evolutionary differential evolution for constrained optimization. *Applied Mathematics and Computation 186*, 1 (2007), 340–356.

[29] Karaboga, D. An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes University, Kayseri, Turkey, 2005.

[30] Kaur, S., Awasthi, L. K., Sangal, A. L., and Dhiman, G. Tunicate swarm algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. *Engineering Applications of Artificial Intelligence 90* (2020), 103541.

[31] Kaveh, A. Thermal exchange metaheuristic optimization algorithm. In *Advances in Metaheuristic Algorithms for Optimal Design of Structures*, Springer, Cham, 2021, pp. 733–782.

[32] Kaveh, A., Khanzadi, M., and Moghaddam, M. R. Billiards-inspired optimization algorithm; a new meta-heuristic method. *Structures 27* (2020), 1722–1739.

[33] Kaveh, A., and Khayatazad, M. A new meta-heuristic method: ray optimization. *Computers & Structures 112-113* (2012), 283–294.

[34] Kaveh, A., and Talatahari, S. A novel heuristic optimization method: charged system search. *Acta Mechanica 213*, 3 (2010), 267–289.

[35] Kennedy, J., and Eberhart, R. Particle swarm optimization. In *Proceedings of ICNN'95 International Conference on Neural Networks*, (Perth, WA, Australia, 1995), Vol. 4, IEEE, pp. 1942–1948.

[36] Khalilpourazari, S., and Khalilpourazary, S. An efficient hybrid algorithm based on water cycle and moth-flame optimization algorithms for solving numerical and constrained engineering optimization problems. *Soft Computing 23*, 5 (2019), 1699–1722.

[37] Kirkpatrick, S. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics 34*, 5-6 (1984), 975–986.

[38] Kivi, M. E., and Majidnezhad, V. A novel swarm intelligence algorithm inspired by the grazing of sheep. *Journal of Ambient Intelligence and Humanized Computing 13*, 2 (2022), 1201–1213.

[39] Krishnanand, K., and Ghose, D. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm Intelligence 3*, 2 (2009), 87–124.

[40] Li, S., Chen, H., Wang, M., Heidari, A. A., and Mirjalili, S. Slime mould algorithm: A new method for stochastic optimization. *Future Generation Computer Systems 111* (2020), 300–323.

[41] Liang, J.-J., Suganthan, P. N., and Deb, K. Novel composition test functions for numerical global optimization. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, SIS 2005*, (Pasadena, CA, USA, 2005), IEEE, pp. 68–75.

[42] Mingos, D. M. P. *Gold clusters, colloids and nanoparticles I*, Vol. 161. Springer, 2014.

[43] Mirjalili, S. Moth-flame optimization algorithm: A novel nature–inspired heuristic paradigm. *Knowledge–Based Systems 89* (2015), 228–249.

[44] Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge–Based Systems 96* (2016), 120–133.

[45] Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., and Mirjalili, S. M. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software 114* (2017), 163–191.

[46] Mirjalili, S., and Lewis, A. The whale optimization algorithm. *Advances in Engineering Software 95* (2016), 51–67.

[47] Mirjalili, S., Mirjalili, S. M., and Lewis, A. Grey wolf optimizer. *Advances in Engineering Software 69* (2014), 46–61.

[48] Mohammadi-Balani, A., Nayeri, M. D., Azar, A., and Taghizadeh-Yazdi, M. Golden eagle optimizer: A nature–inspired metaheuristic algorithm. *Computers & Industrial Engineering 152* (2021), 107050.

[49] Morales-Castañeda, B., Zaldívar, D., Cuevas, E., Fausto, F., and Rodríguez, A. A better balance in metaheuristic algorithms: Does it exist? *Swarm and Evolutionary Computation 54* (2020), 100671.

[50] Mousavirad, S. J., and Ebrahimpour-Komleh, H. Human mental search: a new population-based metaheuristic optimization algorithm. *Applied Intelligence 47*, 3 (2017), 850–887.

[51] Nadimi-Shahraki, M. H., Taghian, S., and Mirjalili, S. An improved grey wolf optimizer for solving engineering problems. *Expert Systems with Applications 166* (2021), 113917.

[52] Nardes, R. C., Silva, M. S., Rezier, A. N. S., Sanches, F. A. C. R. A., Gama Filho, H. S., Santos, R. S., Oliveira, D. F., Lopes, R. T., Carvalho, M. L., Cesareo, R., Zanatta E. M., Assis J. T., and Anjos M. J. Study on Brazilian 18th century imperial carriage using x-ray nondestructive techniques. *Radiation Physics and Chemistry 154* (2019), 74–78.

[53] Naruei, I., and Keynia, F. Wild horse optimizer: a new meta-heuristic algorithm for solving engineering optimization problems. *Engineering with Computers 38*, Suppl. 4 (2022), 3025–3056.

[54] Norgate, T., and Haque, N. Using life cycle assessment to evaluate some environmental impacts of gold production. *Journal of Cleaner Production 29-30* (2012), 53–63.

[55] Oujja, M., Camacho, J. J., Sanz, M., Castillejo, M., and de Nalda, R. Optical diagnostics of gold plasmas produced by infrared laser ablation. *Journal of Quantitative Spectroscopy and Radiative Transfer 256* (2020), 107308.

[56] Owens, K. N. *Riches for all: the California gold rush and the world*. University of Nebraska Press, 2002.

[57] Pereira, J. L. J., Francisco, M. B., Diniz, C. A., Oliver, G. A., Cunha Jr, S. S., and Gomes, G. F. Lichtenberg algorithm: A novel hybrid physics-based meta-heuristic for global optimization. *Expert Systems with Applications 170* (2021), 114522.

[58] Qais, M. H., Hasanien, H. M., and Alghuwainem, S. Transient search optimization: a new meta-heuristic optimization algorithm. *Applied Intelligence 50*, 11 (2020), 3926–3941.

[59] Rahman, C. M., and Rashid, T. A. A new evolutionary algorithm: Learner performance based behavior algorithm. *Egyptian Informatics Journal 22*, 2 (2021), 213–223.

[60] Rashedi, E., Nezamabadi-pour, H., and Saryazdi, S. GSA: a gravitational search algorithm. *Information Sciences 179*, 13 (2009), 2232–2248.

[61] Rohrbough, M. J. *Days of gold: The California gold rush and the American nation*. University of California Press, 1997.

[62] Roske, R. J. The world impact of the California gold rush 1849-1857. *Arizona and the West 5*, 3 (1963), 187–232.

[63] Satapathy, S., and Naik, A. Social group optimization (SGO): a new population evolutionary optimization technique. *Complex & Intelligent Systems 2*, 3 (2016), 173–203.

[64] Sayed, G. I., Darwish, A., and Hassanien, A. E. A new chaotic multi-verse optimization algorithm for solving engineering optimization problems. *Journal of Experimental & Theoretical Artificial Intelligence 30*, 2 (2018), 293–317.

[65] Sharma, T. K., Sahoo, A. K., and Goyal, P. Bidirectional butterfly optimization algorithm and engineering applications. *Materials Today: Proceedings 34*, 3 (2021), 736-741.

[66] Simon, D. Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation 12*, 6 (2008), 702–713.

[67] Storn, R., and Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization 11*, 4 (1997), 341–359.

[68] Suyanto, S., Ariyanto, A. A., and Ariyanto, A. F. Komodo mlipir algorithm. *Applied Soft Computing 114* (2022), 108043.

[69] Suyanto, S., Wibowo, A. T., Al Faraby, S., Saadah, S., and Rismala, R. Evolutionary Rao algorithm. *Journal of Computational Science 53* (2021), 101368.

[70] Tanabe, R., and Fukunaga, A. Success-history based parameter adaptation for differential evolution. In *2013 IEEE Congress on Evolutionary Computation*, (Cancun, Mexico, 2013), IEEE, pp. 71–78.

[71] Tanabe, R., and Fukunaga, A. S. Improving the search performance of SHADE using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation*, (Beijing, China, 2014), IEEE, pp. 1658–1665.

[72] Wei, D., Wang, Z., Si, L., and Tan, C. Preaching-inspired swarm intelligence algorithm and its applications. *Knowledge–Based Systems 211* (2021), 106552.

[73] Wolpert, D. H., and Macready, W. G. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[74] Yan, X., Liu, H., Zhu, Z., and Wu, Q. Hybrid genetic algorithm for engineering design problems. *Cluster Computing 20*, 1 (2017), 263–275.

[75] Yang, X.-S. Firefly algorithms for multimodal optimization. In *Stochastic Algorithms: Foundations and Applications*, O. Watanabe and T. Zeugmann, Eds., Vol. 5792 of Lecture Notes in Computer Science, Springer, Berlin, pp. 169–178.

[76] Yang, X.-S. *Nature–inspired metaheuristic algorithms*. Luniver Press, 2010.

[77] Yang, X.-S. A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, J. R. González, D. A. Pelta, C. Cruz, G. Terrazas and N. Krasnogor, Eds., Vol. 284 of *Studies in Computational Intelligence*, Springer, Berlin, 2010, pp. 65–74.

[78] Yang, X.-S., and Deb, S. Cuckoo search via Lévy flights. In *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, (Coimbatore, India, 2009), IEEE, pp. 210–214.

[79] Yao, X., and Liu, Y. Fast evolutionary programming. In *Proceedings of the Fifth Annual Conference on Evolutionary Programming, EP 1996*, (San Diego, CA, USA), L. J. Fogel, P. I. Angeline, and T. Back, Eds., Cambridge, MA, 1996, MIT Press, pp. 451–460.

[80] Yi, S. K. M., Steyvers, M., Lee, M. D., and Dry, M. J. The wisdom of the crowd in combinatorial problems. *Cognitive Science 36*, 3 (2012), 452–470.

7. Appendix

Table 1. Specification of the unimodal, multimodal, and fixed-dimension multimodal benchmark functions

Function	Dimension, Range, $F_{\min}$
$f_1 = \sum_{i=1}^{n} x_i^2$	30, [–100, 100], 0
$f_2 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30, [–10, 10], 0
$f_3 = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30, [–100, 100], 0
$f_4 = \max\{\lvert x_i \rvert\}$	30, [–100, 100], 0
$f_5 = \sum_{i=1}^{n-1} [100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2]$	30, [–30, 30], 0
$f_6 = \sum_{i=1}^{n} (x_i + 0.5)^2$	30, [–100, 100], 0
$f_7 = \sum_{i=1}^{n} i x_i^4 + \text{random}\,[0,\ 1)$	30, [–1.28, 1.28], 0
$f_8 = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30, [–500, 500], –418.98D
$f_9 = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30, [–5.12, 5.12], 0
$f_{10} = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n} x_j^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30, [–32, 32], 0
$f_{11} = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30, [–600, 600], 0
$f_{12} = \frac{\pi}{4}\{10\sin(\pi y_1)^2 + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})]$ $+(y_n - 1)^2\} + \sum_{i=1}^{n} u\,(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}, \quad u\,(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30, [–50, 50], 0
$f_{13} = 0.1\sin^2(3\pi x_1)] + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})]\}$ $+(x_{n-1})^2[1 + \sin^2(2\pi x_n) + \sum_{i=1}^{n} u\,(x_i, 5, 100, 4)$	30, [–50, 50], 0
$f_{14} = \left(\frac{1}{500} + \sum_{i=1}^{25}\frac{1}{i + \sum_{j=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2, [–65.54, 65.54], 1
$f_{15} = \sum_{i=1}^{11}\left[a_i - \frac{x_1\left(b_i^2 + b_i x_2\right)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4, [–5, 5], 0.0003
$f_{16} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	2, [–5, 5], –1.0316
$f_{17} = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x1) + 10$	,2, [–5, 5], 0.398

Continuation of Table 1

Function	Dimension, Range, $F_{\min}$
$f_{18} = \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right]$ $\times \left[30 + (2x_1 - 3x_2)^2 \times \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$	2, [–2, 2], 3
$f_{19} = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3, [1, 3], –3.86
$f_{20} = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6, [0, 1], –3.32
$f_{21} = -\sum_{i=1}^{5} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4, [0, 10], -10.1532
$f_{22} = -\sum_{i=1}^{7} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4, [0, 10], –10.4028
$f_{23} = -\sum_{i=1}^{10} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4, [0, 10], –10.5363

Table 2. Specification of composite benchmark functions

f_{24} (CF1)
$f_1, f_2, \ldots, f_{10}$ – sphere function $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [5/100, 5/100, \ldots, 5/100]$
f_{25} (CF2)
$f_1, f_2, \ldots, f_{10}$ – Griewank's function $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [5/100, 5/100, \ldots, 5/100]$
f_{26} (CF3)
$f_1, f_2, \ldots, f_{10}$ – Griewank's function $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [1, 1, \ldots, 1]$
f_{27} (CF4)
f_1, f_2 – Ackley's function f_3, f_4 – Rastrigin's function f_5, f_6 – Weierstrass' function f_7, f_8 – Griewank's function f_9, f_{10} – sphere function $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$
f_{28} (CF5)
f_1, f_2 – Rastrigin's function f_3, f_4 – Weierstrass function f_5, f_6 – Griewank's function f_7, f_8 – Ackley's function f_9, f_{10} – sphere function $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32, 5/100, 5/100]$
F_{29} (CF6)
f_1, f_2 – Rastrigin's function f_3, f_4 – Weierstrass function f_5, f_6 – Griewank's function f_7, f_8 – Ackley's function f_9, f_{10} – sphere function $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$ $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [0.1 \times 1/5, 0.2 \times 1/5, 0.3 \times 5/0.5, 0.4 \times 5/0.5, 0.5 \times 5/100$ $0.6 \times 5/100, 0.7 \times 5/32, 0.8 \times 5/32, 0.9 \times 5/100, 1 \times 5/100]$

Table 3. Results of executing unimodal benchmark functions

Function	Quantity	GA	DE	FA	GSA	IGWO	PSO	SCA
f1	Ave.	0.133225	0.383876	0.011638	0.00033	3.751E–28	0.509689	8.04
	Std.	0.123437	2.1	0.003929	0.001689	1.013E–27	0.318177	20.86
f2	Ave.	0.020962	0.000011	0.411286	0.015714	5.43E–18	0.102025	0.020669
	Std.	0.017782	0.000004	0.123289	0.086069	3.92E–18	0.053114	0.028093
f3	Ave.	3965.75	19652.15	2379.88	925.94	0.001282	1931.42	7320.27
	Std.	1554.22	5343.37	662.79	306.91	0.003836	686.34	4664.88
f4	Ave.	8.06	13.21	0.089565	6.89	0.000014	8.25	35.06
	Std.	2.88	6.73	0.017417	1.87	0.00001	1.21	11.54
f5	Ave.	437.36	325.92	247.62	67.11	24.19	364.44	85836.58
	Std.	668.38	1010.39	475.59	69.16	0.869251	520.58	192066.87
f6	Ave.	0.102311	0.008304	0.011635	0.001909	0.024615	1.01	21.97
	Std.	0.099086	0.027745	0.00428	0.007349	0.074921	0.948836	26.4
f7	Ave.	0.037937	0.029735	0.043929	0.102857	0.002371	0.057214	0.233735
	Std.	0.016588	0.009711	0.036366	0.058656	0.001102	0.014477	0.526834

Function	Quantity	SSA	WCA	WOA	KMA	SMA	GRO
f1	Ave.	1.959E–07	0.000031	3.342E–71	**0**	2.727E–294	2.024E–61
	Std.	3.43E–07	0.00005	1.83E–70	0	0	1.067E–60
f2	Ave.	1.94	1.38	1.15E–50	**0**	4.256E–150	1.195E–40
	Std.	1.35	4.33	5.569E–50	0	2.33E–149	3.766E–40
f3	Ave.	1318.69	363.07	47740.78	**0**	4.007E–320	8.18
	Std.	688.52	1136.32	13494.05	0	0	19.34
f4	Ave.	10.8	20.92	44.1	**0**	2.423E–147	0.107588
	Std.	3.17	5.48	25.06	0	1.327E–146	0.589282
f5	Ave.	211.83	128.46	27.96	24.57	**6.64**	26.67
	Std.	304.35	121.58	0.439249	9.28	10.84	0.316622
f6	Ave.	**7.27E–07**	0.000071	0.46056	0.210988	0.006177	0.070432
	Std.	0.000002	0.000154	0.276193	0.205541	0.004642	0.051007
f7	Ave.	0.185488	0.618082	0.002835	0.00036	**0.000153**	0.005677
	Std.	0.097735	0.291299	0.003279	0.000222	0.00015	0.005617

Table 4. Results of executing multimodal benchmark functions

Function	Quantity	GA	DE	FA	GSA	IGWO	PSO	SCA
f8	Ave.	-10845.92	-6682.67	-6311.86	-2411.3	-8038.13	-8306.83	-3715.63
	Std.	310.73	641.19	624.22	276.64	1657.34	549.8	306.49
f9	Ave.	20.92	146.52	35.73	28.62	24.55	55.16	42.49
	Std.	5.24	13.22	11.7	8.17	21.85	13.31	35.19
f10	Ave.	0.06346	0.008094	0.054169	0.031043	5.856E–14	1.01	15.22
	Std.	0.031378	0.043296	0.019379	0.170032	1.025E–14	0.656047	8.31
f11	Ave.	0.119276	0.001946	0.006281	29.48	0.004031	0.596379	0.84701
	Std.	0.108791	0.003688	0.001884	7.5	0.006297	0.264622	0.312651
f12	Ave.	0.001403	0.048003	0.000313	1.92	**0.000178**	0.600018	134988.5
	Std.	0.001236	0.103451	0.000244	1.17	0.00093	0.759963	724443.09
f13	Ave.	0.025278	3668.38	**0.002559**	9.9	0.132176	2.15	822455.14
	Std.	0.021894	17552.32	0.001153	6.04	0.10664	2.23	3332216.7

Function	Quantity	SSA	WCA	WOA	KMA	SMA	GRO
f8	Ave.	-7411.79	-8115.29	-10352.91	-9724.69	**-12568.94**	-8051.51
	Std.	884.73	675.07	1919.9	1144.28	0.538186	656.07
f9	Ave.	50.21	102.19	1.895E–15	**0**	**0**	0.416243
	Std.	18.91	32.72	1.038E–14	0	0	2.28
f10	Ave.	2.64	7.92	3.612E–15	8.36E–11	8.882E–16	4.559E–15
	Std.	0.727022	4.36	2.586E–15	3.166E–10	0	6.486E–16
f11	Ave.	0.019165	0.032991	0.008927	**0**	**0**	**0**
	Std.	0.01346	0.029126	0.048896	0	0	0
f12	Ave.	6.77	1.57	0.017894	0.00304	0.004463	0.003884
	Std.	3.12	2.06 0.007641	0.004427	0.004384	0.003351	
f13	Ave.	14.83	3.02	0.435132	0.086491	0.004592	0.152654
	Std.	11.89	6.39	0.215847	0.110225	0.005607	0.093099

Table 5. Results of executing fixed-dimension multimodal benchmark functions

Function	Quantity	GA	DE	FA	GSA	IGWO	PSO	SCA
f14	Ave.	**0.998**	1.587	1.921	4.762	**0.998**	**0.998**	1.596
	Std.	1.806E–08	1.97	0.717521	3.49	1.129E–16	1.091E–16	0.923258
f15	Ave.	0.0027	0.002	0.0037	0.0042	**0.0003**	0.0039	0.0011
	Std.	0.004759	0.004991	0.00673	0.002311	7.613E–09	0.007497	0.00041
f16	Ave.	**–1.03163**	**–1.03163**	**–1.03163**	**–1.03163**	**–1.03163**	**–1.03163**	**–1.03163**
	Std.	1.674E–08	6.712E–16	3.992E–09	4.61E–16	6.116E–16	5.758E–16	0.00006
f17	Ave.	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	0.39921
	Std.	8.521E–08	0	1.464E–09	0	0	0	0.001286
f18	Ave.	3.00004	**3**	**3**	**3**	**3**	**3**	3.00007
	Std.	0.000126	1.267E–15	4.002E–08	4.128E–15	1.345E–15	2.139E–15	0.000091
f19	Ave.	**–3.8628**	**–3.8628**	**–3.8628**	**–3.8628**	**–3.8628**	**–3.8628**	–3.8551
	Std.	4.461E–08	2.71E–15	1.192E–09	2.356E–15	2.479E–15	2.494E–15	0.003111
f20	Ave.	–3.282	–3.273	–3.276	**–3.322**	–3.306	–3.29	–2.811
	Std.	0.057005	0.058202	0.061312	1.641E–15	0.041071	0.053475	0.497179
f21	Ave.	-6.4128	–8.9887	–7.8053	–6.9646	–9.8872	–7.0668	–2.1891
	Std.	3.8	2.68	3.43	3.71	0.910161	3.45	1.7
f22	Ave.	–7.793	–9.9577	**–10.4029**	–9.9922	**–10.4029**	–8.1695	–2.9389
	Std.	3.5	1.69	0.000002	1.57	1.705E–07	3.28	1.92
f23	Ave.	–9.7551	**–10.5364**	**–10.5364**	–10.002	**–10.5364**	–9.037	–3.6682
	Std.	2.38	1.776E–15	0.000001	2.03	4E–09	2.83	1.62

Function	Quantity	SSA	WCA	WOA	KMA	SMA	GRO
f14	Ave.	1.428	**0.998**	1.692	**0.998**	**0.998**	**0.998**
	Std.	0.767328	2.781E–16	1.17	5.755E–11	9.732E–13	0
f15	Ave.	0.0022	0.0007	0.0005	**0.0003**	0.0005	**0.0003**
	Std.	0.004957	0.000438	0.000234	0.000003	0.000273	0.000037
f16	Ave.	**–1.03163**	**–1.03163**	**–1.03163**	**–1.03163**	**–1.03163**	**–1.03163**
	Std.	3.105E–14	4.965E–16	2.432E–09	5.975E–16	1.299E–09	6.519E–16
f17	Ave.	**0.39789**	**0.39789**	0.39789	**0.39789**	**0.39789**	**0.39789**
	Std.	1.164E–14	0	0.000013	0	1.267E–08	0
f18	Ave.	**3**	**3**	3.00017	**3**	**3**	**3**
	Std.	2.217E–13	3.769E–15	0.000799	1.694E–15	3.688E–10	1.056E–15
f19	Ave.	**–3.8628**	**–3.8628**	–3.8565	**–3.8628**	**–3.8628**	**–3.8628**
	Std.	4.63E–11	2.386E–15	0.006483	2.473E–15	1.96E–07	2.696E–15
f20	Ave.	–3.228	–3.267	–3.213	–3.314	–3.246	**–3.322**
	Std.	0.058197	0.060328	0.139149	0.030164	0.058578	0.000006
f21	Ave.	–7.3961	–7.0515	–7.0068	**–10.1532**	–10.1528	**–10.1532**
	Std.	3.31	3.27	2.64	0.000005	0.000433	8.102E–07
f22	Ave.	–7.946	–8.5855	–8.3503	**–10.4029**	–10.4025	**–10.4029**
	Std.	3.35	3.11	2.74	3.854E–08	0.000403	1.892E–07
f23	Ave.	–7.5737	–7.8505	–7.0084	**–10.5364**	–10.5361	**–10.5364**
	Std.	3.74	3.64	3.5	5.274E–08	0.000359	4.414E–12

Table 6. Results of executing composite benchmark functions

Function	Quantity	GA	DE	FA	GSA	IGWO	PSO	SCA
f24	Ave.	76.67	32.33	40	**23.33**	30	116.67	136.25
	Std.	97.14	53.03	49.83	43.02	46.61	117.69	36.32
f25	Ave.	82.59	57.63	134.68	215.56	119.84	153.2	115.38
	Std.	73.03	84.28	91.05	70.68	86.43	100.76	25.42
f26	Ave.	226.86	193.66	**139.45**	200.92	167.38	244.43	421.42
	Std.	73.17	50.86	38.64	113.1	58.63	123.95	89.97
f27	Ave.	401.02	342.8	311.39	464.82	326.29	439.1	444.8
	Std.	107.19	49.94	85.88	109.81	103.18	129.66	21.09
f28	Ave.	78.69	30.32	99.09	248.81	27.7	176.88	145.46
	Std.	79.91	53.58	124.17	76.95	44.42	239.66	67.73
f29	Ave.	765.83	698.02	819.4	859.15	774.27	832.44	**578.96**
	Std.	191.76	202.02	162.68	59.62	183.35	151.52	137.08

Function	Quantity	SSA	WCA	WOA	KMA	SMA	GRO
f24	Ave.	46.67	73.33	120.13	63.51	66.67	31.02
	Std.	68.14	90.72	101.12	49.02	80.23	46.25
f25	Ave.	32.19	170.65	219.39	72.22	27.31	**26.71**
	Std.	49.36	133.23	91.37	50.61	34.82	43.85
f26	Ave.	239.73	357.12	439.6	340.29	202.35	149.01
	Std.	66.38	151.51	150.14	96.51	142.42	37.5
f27	Ave.	351.47	478.63	644.34	365.67	378.59	**281.3**
	Std.	42.97	124.22	124.06	139.23	133.41	65.88
f28	Ave.	30.18	200.65	242.66	17.66	81.99	**4.93**
	Std.	59.88	211.74	169.09	41.04	118.15	6.16
f29	Ave.	682.11	856.13	804.34	766.23	594.64	631.35
	Std.	201.35	122.06	167.03	185.6	175.65	188.97

Table 7. Comparison between algorithms in terms of global optimum guarantee

Quantity	GA	DE	FA	GSA	IGWO	PSO	SCA	SSA	WCA	WOA	KMA	SMA	GRO
Gua (n)	4	5	6	5	8	5	0	4	5	1	15	9	11
Gua (%)	13.8	17.2	20.7	17.2	27.6	17.2	0	13.8	17.2	3.4	51.7	31.0	37.9

Table 8. Results of executing Friedman test

f	GA	DE	FA	GSA	IGWO	PSO	SCA	SSA	WCA	WOA	KMA	SMA	GRO	p-Value
f1	11.17	7.33	9.97	6.30	5.00	12.23	12.40	7.77	8.83	3.00	**1.43**	1.57	4.00	1.1E–67
f2	9.30	6.97	11.90	6.20	5.00	10.73	9.17	12.80	8.93	3.00	**1.00**	2.00	4.00	3.6E–67
f3	10.03	12.00	8.87	6.40	3.60	8.07	10.37	7.10	5.20	12.97	**1.47**	1.53	3.40	1.1E–66
f4	7.70	9.40	4.97	7.00	3.97	7.90	12.03	9.00	11.17	11.80	**1.00**	2.00	3.07	2.9E–63
f5	10.27	7.53	8.43	6.83	2.17	10.50	12.77	8.70	8.53	5.20	5.13	**1.73**	3.20	1.4E–50
f6	8.93	2.57	6.63	**1.37**	5.23	11.63	13.00	2.87	4.00	10.90	9.57	5.80	8.50	1.3E–63
f7	7.60	7.00	7.63	10.43	3.80	9.13	9.93	11.47	12.70	3.70	1.87	**1.30**	4.43	1.8E–62
f8	2.60	9.60	10.27	13.00	6.97	6.53	12.00	8.03	6.63	3.80	3.80	**1.00**	6.77	9.5E–59
f9	6.17	12.90	8.13	7.47	6.43	10.30	8.30	9.47	11.80	2.53	**2.47**	**2.47**	2.57	3.3E–62
f10	8.50	7.03	8.43	6.17	4.87	10.20	12.27	11.20	12.20	2.83	2.170	**1.63**	3.50	7.7E–67
f11	9.93	6.00	6.83	13.00	4.43	11.23	11.63	8.07	8.33	3.08	**2.82**	**2.82**	**2.82**	1.0E–64
f12	4.20	3.60	3.17	10.60	**1.93**	9.07	12.50	12.17	9.30	7.93	5.10	5.43	6.00	3.5E–55
f13	4.57	4.07	**2.23**	11.17	5.70	9.63	12.70	11.00	6.67	8.50	5.27	2.83	6.67	5.8E–50
f14	6.87	5.68	10.68	12.32	**4.33**	**4.33**	10.83	6.48	**4.33**	10.13	4.95	5.72	**4.33**	3.3E–48
f15	10.23	7.57	10.03	12.23	2.07	7.33	9.33	8.77	5.50	6.20	**2.03**	5.77	3.93	1.7E–43
f16	5.48	**5.05**	11.80	**5.05**	**5.05**	**5.05**	13.00	**5.05**	**5.05**	9.55	**5.05**	10.77	**5.05**	3.7E–65
f17	5.40	**4.97**	10.07	**4.97**	**4.97**	**4.97**	13.00	**4.97**	**4.97**	11.93	**4.97**	10.87	**4.97**	6.4E–68
f18	6.47	**5.02**	10.87	**5.02**	**5.02**	**5.02**	12.47	**5.02**	**5.02**	12.30	**5.02**	8.77	**5.02**	1.1E–61
f19	5.03	**4.80**	9.97	**4.80**	**4.80**	**4.80**	12.73	6.43	**4.80**	12.27	**4.80**	10.97	**4.80**	3.0E–66
f20	5.12	6.07	8.43	**2.72**	5.62	4.52	12.93	9.27	5.87	10.07	6.30	9.67	4.43	3.2E–37
f21	7.52	4.03	8.37	6.45	6.20	6.42	12.37	7.38	6.67	9.60	**3.77**	8.03	4.20	8.7E–24
f22	7.27	3.80	8.53	**3.77**	7.10	6.18	12.27	8.35	5.58	10.70	4.08	9.53	3.83	1.0E–36
f23	4.82	**3.47**	8.80	4.07	6.83	5.52	12.00	9.12	6.75	11.37	4.85	9.87	3.55	1.5E–42
f24	7.12	4.15	6.73	**3.28**	5.50	7.87	10.83	5.68	5.57	10.50	8.97	8.30	6.50	2.4E–20
f25	6.00	3.70	8.20	10.62	8.20	8.78	8.20	4.30	8.90	10.97	6.00	4.17	**2.97**	2.0E–29
f26	6.93	5.50	**3.00**	5.53	4.27	6.67	11.77	7.40	9.97	11.30	9.70	5.33	3.63	1.4E–35
f27	7.63	5.97	3.50	9.30	4.17	8.10	9.97	6.50	9.63	12.17	5.60	6.40	**2.07**	9.0E–35
f28	7.47	3.60	7.03	11.50	3.47	7.70	9.47	5.37	9.53	10.67	5.57	6.53	**3.10**	3.9E–32
f29	8.47	5.13	8.70	8.87	6.37	8.40	5.27	5.90	9.30	10.27	6.05	**3.85**	4.43	1.6E–16
Mean	7.20	6.02	8.01	7.46	4.93	7.89	11.22	7.78	7.65	8.59	4.51	5.40	**4.34**	5.4E–18
Std. dev.	2.09	2.49	2.58	3.33	1.50	2.31	1.87	2.46	2.53	3.56	2.35	3.36	**1.44**	2.9E–17

Table 9. Results for the execution of Wilcoxon signed-rank test

Comparison	Negative ranks	Positive ranks	p-Value
GRO vs. GA	288	37	0.000733
GRO vs. DE	292	8	0.000050
GRO vs. FA	235	41	0.003175
GRO vs. GSA	282	18	0.000162
GRO vs. IGWO	156	75	0.159224
GRO vs. PSO	278	22	0.000255
GRO vs. SCA	416	19	0.000018
GRO vs. SSA	321	4	0.000020
GRO vs. WCA	279	21	0.000228
GRO vs. WOA	327	51	0.000915
GRO vs. KMA	111	79	0.519657
GRO vs. SMA	137	139	0.975735

Table 10. Comparison of the statistical results for pressure vessel design

Algorithm	Statistical result			Std. dev.
	Best	Average	Worst	
LFD	6.08E+03	1.60E+04	3.62E+04	8.02E+03
BA	6059.71	6179.13	6318.95	137.223
GWO	6051.5639	N/A	N/A	N/A
CPSO	6061.0777	6147.1332	6363.8041	86.4545
CDE	6059.7340	6085.2303	6371.0455	43.0130
TSO	5888.408	5892.72	N/A	8.2
MVO	6059.9528	6326.3745	7201.8922	6.12E+02
BBOA	6059.714	N/A	N/A	N/A
HGA	6059.716776	6073.254823	6080.422852	15.0651
GRO	**5886.4068**	5912.5944	6000.9867	26.67

Table 11. Comparison of the parameters for the best solutions of pressure vessel design

Algorithm	Optimum variables				Optimum cost
	$x_1(T_s)$	$x_2(T_h)$	$x_3\,(R)$	$x_4\,(L)$	
LFD	0.8777	0.4339	45.4755	139.0654	6.08E+03
BA	0.8125	0.4375	42.0984456	176.6365958	6059.71
GWO	0.812500	0.434500	42.089181	176.758731	6051.5639
CPSO	0.812500	0.437500	42.091266	176.746500	6061.0777
CDE	0.812500	0.437500	42.098411	176.637690	6059.7340
TSO	0.77859	0.385053	40.34033	199.7883	5888.408
MVO	0.8125	0.4375	42.098	176.6502	6059.9528
BBOA	0.8125	0.4375	42.098456	176.62446	6059.714
HGA	0.8125	0.4375	42.098447	176.636692	6059.716776
GRO	0.7787153	0.384967	40.347943	199.6061	**5886.4068**

Table 12. Comparison of the statistical results for tension/compression spring problem

Algorithm	Statistical result			Std. dev.
	Best	Average	Worst	
LFD	0.0127	0.0138	0.0156	9.82E–04
BA	**0.012665**	0.01350052	0.0168954	0.001420272
GWO	0.012666	N/A	N/A	N/A
CPSO	0.012674	0.012730	0.012924	5.198500E–5
CDE	0.012670	0.012703	0.012790	2.7000E–5
TSO	**0.012665**	0.012668	N/A	2.7E–8
MVO	0.012698	0.0167897	0.0227750	3.82E–03
BBOA	0.012667	N/A	N/A	N/A
HGA	0.012666	0.012673	0.012705	1.18603E–05
GRO	**0.012665**	0.0126775	0.0127526	1.84E–05

Table 13. Comparison of the parameters for the obtained best solutions of tension/compression spring problem

Algorithm	Optimum variables			Optimum cost
	$x_1(d)$	$x_2\,(D)$	$x_3\,(N)$	
LFD	0.0517	0.3575	11.2442	0.0127
BA	0.05169	0.35673	11.2885	**0.012665**
GWO	0.05169	0.356737	11.28885	0.012666
CPSO	0.051728	0.357644	11.244543	0.012674
CDE	0.051609	0.354714	11.410831	0.012670
TSO	0.051735	0.357822	11.22456	**0.012665**
MVO	0.051689	0.3567409	11.288291	0.012698
BBOA	0.051344	0.334881	12.6223	0.012667
HGA	0.051678	0.356436	11.305945	0.012666
GRO	0.0517082206	0.35717883	11.2619852	**0.012665**

Table 14. Comparison of the statistical results for welded beam problem

Algorithm	Statistical result			Std. dev.
	Best	Average	Worst	
LFD	1.77E+00	2.30E+00	3.04E+00	3.16E–01
BA	1.7312065	1.8786560	2.3455793	0.2677989
GWO	1.72624	N/A	N/A	N/A
CPSO	1.728024	1.748831	1.782143	0.012926
CDE	1.733461	1.768158	1.824105	0.022194
TSO	1.72509	1.725821	N/A	5.1E–03
MVO	1.724855	1.800613	1.963254	8.15E–02
BBOA	1.72491	N/A	N/A	N/A
HGA	1.725236	1.728847	1.73578	0.0112
GRO	**1.7248523086**	1.72485383	1.72488705	5.72E–05

Table 15. Comparison of the parameters for best solutions of welded beam problem

Algorithm	Optimum variables				Optimum cost
	$x_1(h)$	$x_2(l)$	$x_3\,(t)$	$x_4\,(b)$	
LFD	0.1857	3.9070	9.1552	0.2051	1.77
BA	0.2015	3.562	9.0414	0.2057	1.7312
GWO	0.205676	3.478377	9.03681	0.205778	1.72624
CPSO	0.202369	3.544214	9.048210	0.205723	1.728024
CDE	0.203137	3.542998	9.033498	0.206179	1.733462
TSO	0.205695122	3.471597147	9.037394427	0.205731215	1.72509
MVO	0.20573	3.4705	9.036662	0.20574	1.724855
BBOA	0.20574	3.4798	9.03801	0.2078	1.72491
HGA	0.205712	3.470391	9.039693	0.205716	1.725236
GRO	0.20572964	3.47048867	9.03662391	0.20572964	**1.7248523086**