Engineering Applications of Artificial Intelligence 117 (2023) 105622

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

A survey of recently developed metaheuristics and their comparative analysis

Abdulaziz Alorf

Department of Electrical Engineering, College of Engineering, Qassim University, Buraydah 52571, Saudi Arabia

ARTICLE INFO

Keywords:
Marine predator algorithm
Metaheuristics survey
Optimization algorithms
Political optimizer
Optimization survey
Optimization algorithm comparison
Engineering optimization

ABSTRACT

The aim of this study was to gather, discuss, and compare recently developed metaheuristics to understand the pace of development in the field of metaheuristics and make some recommendations for the research community and practitioners. By thoroughly and comprehensively searching the literature and narrowing the search results, we created with a list of 57 novel metaheuristic algorithms. Based on the availability of the source code, we reviewed and analysed the optimization capability of 26 of these algorithms through a series of experiments. We also evaluated the exploitation and exploration capabilities of these metaheuristics by using 50 unimodal functions and 50 multimodal functions, respectively. In addition, we assessed the capability of these algorithms to balance exploration and exploitation by using 29 shifted, rotated, composite, and hybrid CEC-BC-2017 benchmark functions. Moreover, we evaluated the applicability of these metaheuristics on four real-world constrained engineering optimization problems. To rank the algorithms, we performed a nonparametric statistical test, the Friedman mean rank test. Based on the statistical results for the unimodal and multimodal functions, we declared that the GBO, PO, and MRFO algorithms have better exploration and exploitation capabilities. Based on the results for the CEC-BC-2017 benchmark functions, we found the MPA, FBI, and HBO algorithms to be the most balanced. Finally, based on the results for the constrained engineering optimization problems, we declared that the HBO, GBO, and MA algorithms are the most suitable. Collectively, we confidently recommend the GBO, MPA, PO, and HBO algorithms for real-world optimization problems.

1. Introduction

The global minimum or maximum of a function/problem is found through global optimization. The importance of global optimization is evident from its critical role in a wide range of application areas, such as engineering (Deb, 1991; Sandgren, 1990; Golinski, 1973), machine learning (Berwick, 2003; Nadaraya, 1964; Attik et al., 2005), wireless sensor network optimization Jin et al. (2003), resource optimization and scheduling Hegazy and Kassab (2003), path planning Brand et al. (2010), bioinformatics Handl et al. (2007), health care Azcárate et al. (2008), vehicle routing Yu et al. (2009), image and signal processing Zibulevsky and Elad (2010), and process optimization Biegler et al. (2014). Intractably large and complex search spaces make optimization problems NP-hard (Hochba, 1997). **Exact algorithms** are a fundamental class of algorithms used to solve such problems. These algorithms include the exhaustive search, dynamic programming-based methods, and branch and bound algorithms as well as informed search algorithms such as the A* search algorithm (Pardalos and Romeijn, 2013). For NP-hard problems, exact algorithms require exponential time, and the optimal solution may not guaranteedly be found (Bianchi et al., 2008). In such cases, however, the **approximation algorithms** such as greedy algorithms come into play and offer a near-optimal solution by dealing with computational intractability (Gomes and Williams, 2005). Finding a near-optimal solution for inapproximable problems using traditional approximation algorithms may become as difficult as finding the exact optimal solution. **Metaheuristics** have been used to reasonably address the inapproximable nature of optimization problems (Gonzalez, 2007). In the last 3 to 4 decades, much research has been devoted to metaheuristics.

The term "metaheuristics", which was first coined by Glover (1986), comprises two words *meta* and *heuristic*. The term *heuristic* is derived from the Greek verb (heuriskein), whose meaning is 'to find'. One branch of heuristics covers local searches, which usually start from a random solution, and an attempt is made to improve the solution through iterations (Bianchi et al., 2008). A few well-known examples of local searches are the hill climbing, tabu search (Glover and Laguna, 1998), and local beam search algorithms (Ow and Morton, 1988). The term *meta* is a Greek suffix whose meaning is 'beyond, in an upper level'. Thus, metaheuristics are algorithms that combine heuristics (that are usually very problem-specific) in a more general framework to make them problem independent. Metaheuristics, in general, belong to the branch of approximation-based optimization algorithms; however, there are many dimensions to further categorize metaheuristics. For example, there are nature-inspired metaheuristics such as ant colony optimization (ACO) (Dorigo and Caro, 1999), the African vultures optimization algorithm (AVOA) (Abdollahzadeh et al., 2021), and artificial gorilla troops optimizer (AGTO) (Abdollahzadeh et al., 2021b) and

E-mail address: aaalorf@qu.edu.sa.

https://doi.org/10.1016/j.engappai.2022.105622
Received 8 March 2022; Received in revised form 15 October 2022; Accepted 7 November 2022
Available online 24 November 2022

nonnature inspired metaheuristics such as tabu search (Glover and Laguna, 1998). There are also single solution-based metaheuristics such as simulated annealing (SA) (Kirkpatrick et al., 1983) and population-based metaheuristics such as the genetic algorithm (GA) (Holland, 1992). Nature-inspired metaheuristics mimic the behaviour of living creatures such as humans, animals, birds, fishes, insects, germs, and bacteria or the laws of physics working behind natural processes and phenomena.

In the presence of imperfect information and resource constraints, the use of metaheuristics cannot guarantee that the global optimum will be found; however, a near-optimal solution may be found (Bianchi et al., 2008). Here, we discuss some common properties of metaheuristics that have been mentioned in the literature: (i) they are usually very simple and are easy to implement (Mirjalili and Lewis, 2016), (ii) they are problem independent, and they do not require problem-specific knowledge in advance to solve a problem; however, domain-specific knowledge can be utilized to guide the search (Blum and Roli, 2003; Boussaïd et al., 2013), (iii) a mechanism to globally search the space and avoid or escape local optima can be added (Blum and Roli, 2003; Boussaïd et al., 2013), (iv) they may utilize their search experience to guide the search (Blum and Roli, 2003; Boussaïd et al., 2013), (v) they are usually not dependent on the gradient information, unlike many other gradient-based algorithms, and (vi) they have tremendous capability to find good results for complex problems in limited time (Bianchi et al., 2008). Moreover, in metaheuristics, ideas to address an optimization problem are mapped based on a few common fundamental elements. These elements include the representation of the candidate solution (usually a vector), design variables involved in the problem (dimensions of the solution vector), evaluation function to measure the quality of the solution (objective/cost function), the ultimate goal (global optimum), and the position-updating mechanism (mathematical equation(s) and parameter(s)) to map the collaborative intelligent behaviour of the solutions in the whole population. Moreover, the mapping process is supposed to assure the following metaheuristic characteristics (Yang, 2010a; Bianchi et al., 2008; Blum and Roli, 2003; Boussaïd et al., 2013; Gandomi et al., 2013):

- **Exploration (diversification)**: The process of exhaustively searching the solution space to find the promising areas. An algorithm may not locate the area containing the global optimum if it does not have good exploration capability.
- **Exploitation (intensification)**: The process of searching in a promising region to find the best solution. An algorithm may not be able to find the optimum of the promising region without this capability.
- **Convergence**: The state when the characteristics of all solutions of a population-based metaheuristics become similar and the solutions stop improving. If lacking this ability, an algorithm may not be able to exploit the promising region well.
- **Premature convergence avoidance**: Premature convergence is the state that occurs when the whole population converges to some local optimum in early iterations. This usually happens when the algorithm cannot escape local optima due to having an unbalanced ratio of exploration and exploitation.
- **Convergence speed**: How quickly the characteristics of all solutions become similar is called the convergence speed of the algorithm. A reasonable algorithm is supposed to have a good convergence speed that prevents premature convergence. However, it allows the most promising areas to be exploited to reach the exact global optimum.

In this paper, a comprehensive review of recently developed metaheuristics is presented, and the optimization capability of 26 algorithms is comparatively analysed. To prepare the list of recent algorithms, several keywords, such as metaheuristics, metaheuristic optimization, numerical optimization, constrained optimization, global optimization, optimization algorithms, evolutionary algorithms, swarm intelligence, global search, computational intelligence, and heuristic search, were used. Our primary source was Google Scholar; however, we also searched other sources such as IEEE, Elsevier, Springer, ACM, and Hindawi. The result was a large list of more than 200 articles published in last two years. However, we shortened this list by removing variants, hybrid approaches, and application-based papers. Then, we created a list of 57 novel algorithms. In the second round, we searched different sources such as MathWorks, GitHub, ResearchGate and the websites of authors for the source codes of the algorithms on the list. We also emailed some authors for their source codes. Finally, based on the availability of the source codes, we included 26 algorithms for a comparative study using a diverse set of benchmark functions and engineering optimization problems. This paper contributes to the literature in the following respects:

1. Recently proposed metaheuristics are collected and presented together to allow heuristics developers to understand the pace and nature of developments in this field.
2. Based on the availability of the source code, the exploitation capability, exploration capability, convergence behaviour, and constrained engineering problem-solving ability of 26 metaheuristics are analysed. Such analysis will help heuristics practitioners choose appropriate algorithms to solve real-world applications.
3. By using the nonparametric Friedman mean rank test, these metaheuristics are ranked based on their ability to meet different optimization challenges. This ranking highlights the strengths and weaknesses of these metaheuristics, which will enable heuristics developers to improve the weak aspects of these algorithms.

The rest of the paper is structured as follows: A comprehensive literature review is presented in Section 2 by covering state-of-the-art and modern metaheuristics. The recently proposed algorithms that are being evaluated and comparatively analysed in this paper are discussed in Section 3. In Section 4, the comparative setup, benchmarks functions, and engineering problems that are used for the evaluation of the selected algorithms are discussed. Moreover, the generated results are presented in this section. The individual-level performance and comparative analysis of each algorithm along with its weaknesses and strengths are discussed in Section 5. Finally, this study is concluded in Section 6.

2. Literature review

Hundreds of nature-inspired metaheuristics have been proposed in the literature in the last 4 to 5 decades. Researchers have mapped the optimal behaviours of living creatures, physical phenomena and processes, social interaction-based behaviours, and hypothetical ideas to propose optimization algorithms. In this section, some state-of-the-art and well-known modern metaheuristics are reviewed.

2.1. State-of-the-art metaheuristics

One of the fundamental and state-of-the-art metaheuristics is the **genetic algorithm (GA)**. GA was first proposed by Fraser in 1957 (Fraser, 1957a,b) and then formalized and popularized by Holland in 1975 (Holland, 1975). In the canonical GA (Holland, 1975), a solution is encoded as a bitstring, the concept of the survival of the fittest is implemented through proportionate selection, and the phase of reproduction is mapped using one-point crossover. However, the mutation operator was later added to enhance the exploration capability. In 1962, Fogel proposed **evolutionary programming (EP)** (Fogel, 1962, 1964). The genetic model is evolved through GA, while the behavioural model is evolved through EP. Moreover, there is no recombination operator in EP; its framework includes only mutation, evaluation, and selection operators. Because there is no recombination operator, the mutation operator is responsible for generating new offspring and balancing exploration and exploitation. In 1965, Rechenberg proposed the **evolutionary strategy (ES)** (Rechenberg, 1965), which is also referred to as $(1 + 1)-$ES, to solve hydrodynamic problems. It involves only one solution along with a self-adapting strategic vector that is utilized during mutation to generate offspring from the only parent. Later, in 1978, Rechenberg introduced the concept of population in ES and named this variant $(\mu + 1)-$ES (Rechenberg, 1978). In this variant, two parents are selected randomly and recombined by using $n-$point crossover to produce an offspring that is mutated by using a strategic vector. Later, in 1993, Bäck and Schwefel generalized the scheme to $(\mu + \lambda)-$ES (Bäck and Schwefel, 1993). In this variant, λ offspring are generated in a row, and then selection is performed to obtain a new generation. In 1983, Kirkpatrick et al. proposed a trajectory-based algorithm, called **simulated annealing (SA)** (Kirkpatrick et al., 1983), inspired by the annealing process in metallurgy. In metallurgy, a material is heated and then cooled in a controlled way to remove defects. The concept is mapped through a temperature variable T and the fitness difference of the current and previous states of the solution. The value of temperature variable T is kept very high in early iterations, which promotes exploration, and it gradually decreases with the number of iterations to enhance exploitation. In 1989, Koza proposed **genetic programming (GP)** (Koza, 1989, 1990). GP is a specialization of GA, but unlike GA, individuals are encoded as tree structures in GP.

In 1992, Dorigo proposed the **ant algorithm (AA)** (Dorigo, 1992) in his Ph.D. thesis to solve the travelling salesman problem. AA was the preliminary version of **ant colony optimization (ACO)** (Dorigo and Caro, 1999) proposed by Dorigo and Caro in 1999. Ants lay pheromones on a path when they find a food source, encouraging other ants to follow the same path. A shorter path begins to emerge when more ants follow it, and the pheromone concentration on the path increases (Dorigo et al., 1996). To solve the travelling salesman problem in ACO, the behaviour of ants who stochastically and incrementally build solutions by choosing next cities according to their selection probability, which is determined by the amount of pheromone on the link and the desirability of the link, is mimicked. In 1997, Storn and Price proposed another state-of-the-art evolutionary algorithm named **differential evolution (DE)** (Storn and Price, 1997; Price et al., 2006). DE shares the general model of evolutionary algorithms; however, DE extends the model by utilizing distance and direction-related information of the solutions. DE was developed to solve problems from the continuous domain. Unlike GA, mutation is performed first to generate a trial vector, which is used to perform crossover. In 1995, Kennedy and Eberhart proposed **particle swarm optimization (PSO)** (Kennedy and Eberhart, 1995) by mapping the flying behaviour of a flock of birds that adapt to reach a roost. The search agents (birds) fly in a hyperdimensional search space, where the search agents update their position based on their previous velocity (inertia component), the position of the best search agent of the flock, called the global best solution (social component), and their own historical best position, called the personal best position (the cognitive component). Exploration and exploitation are balanced by tuning the parameters. After PSO and ACO, hundreds of novel swarm-based metaheuristics, variants, and hybrids of existing approaches were proposed. Some prominent algorithms are the **artificial bee colony (ABC)** (Karaboga, 2005) algorithm inspired by the foraging behaviour of honeybees and proposed by Karaboga and the **firefly algorithm (FA)** (Yang, 2010b) inspired by the flashing behaviour of fireflies, the **cuckoo search (CS)** (Yang and Deb, 2009) inspired by the egg-laying behaviour of cuckoos in the nests of other birds, and the **bat algorithm (BA)** (Yang, 2010c) inspired by the echolocation behaviour of microbats, all of which were proposed by Yang.

2.2. *Well-known modern age metaheuristics*

In 2001, Geem et al. proposed one of the early human-behaviour-based metaheuristics called **harmony search (HS)** (Geem et al., 2001). It was inspired by the improvisation process of jazz musicians. Finding the global optimum resembles producing a perfectly pleasing harmony. Each harmonic is considered a candidate solution, and the total harmony memory is regarded as the population. Each jth design variable (pitch note) of the new solution (improvised harmonic) is updated by replacing it with the jth variable of a randomly selected harmonic (memory consideration phase) by locally perturbing the jth variable of the existing harmonic (pitch adjustment phase) or randomly initializing the jth variable in the feasible range (random selection phase). An improvised solution replaces the worst solution of the population if it is better than the worst solution. In HS, exploration is ensured through the random selection phase, and exploitation is incorporated in a controlled manner in memory consideration and pitch adjustment. Parameter tuning is required to balance exploration and exploitation. In 2002, Xie et al. proposed the **social cognitive optimization algorithm (SCOA)** (Xie et al., 2002) inspired by social cognitive theory, which assumes that humans learn by observing the behaviour of others and symbolizes/models the behaviour of others to guide behaviour. These capabilities are referred to as vicarious capability and symbolizing capability, respectively. In SCOA, a learning agent is considered a candidate solution whose position is a knowledge point in the knowledge space. In contrast, the knowledge space may be considered the library. Symbolizing capability is mapped through the library, and vicarious capability is implemented via tournament selection and position updates through neighbourhood searches.

In 2003, Ray and Liew proposed the **society and civilization (SC)** optimization algorithm (Ray and Liew, 2003) inspired by the intra- and intersociety interactions between the individuals in society and the civilization model. An individual is considered a candidate solution. A group of individuals forms a society, and societies collectively form the civilization, which is regarded as the population. The individuals with better fitness are called leaders. Ordinary individuals in a society update their position by interacting with their leaders, and the leaders of different societies interact with leaders in the civilization, who are chosen from the leaders of the societies. In 2006, Erol and Eksin proposed the **big bang-big crunch** (BBBC) algorithm (Erol and Eksin, 2006) inspired by the theory of creation, expansion, and shrinkage of the universe. BBBC involves two phases: big bang and big crunch. The big bang phase promotes exploration, and the big crunch phase focuses on exploitation, leading to convergence. In 2007, Atashpaz-Gargari and Lucas proposed the **imperialist competitive algorithm (ICA)** (Atashpaz-Gargari and Lucas, 2007). This algorithm was inspired by the concept of imperialism. Imperialism is the process of constructing powerful empires by holding control over other countries. The countries that have such control are called imperialists. Each country is considered a candidate solution, and the set of all countries is the population. The solutions that have better fitness are known as imperialists and the others as colonies. The dominance of imperialists is mapped by associating the colonies (weak solutions) with the imperialists (strong solutions). Colonies update their positions with respect to imperialists and replace them if the fitness of the colonies becomes better than the fitness of the imperialists. The total strength of each empire is computed, and the weakest colony of the weakest empire is shifted to some other empire. Empires with no colonies are finally eliminated.

In 2009, Rashedi et al. proposed the **gravitational search algorithm (GSA)** (Rashedi et al., 2009) inspired by Newton's law of gravitation and the second law of motion. In GSA, a particle is a candidate solution whose fitness is used to compute its mass. Exploitation is achieved by moving heavier objects (good solutions) slowly, and exploration is achieved through the comparatively fast movement of lighter objects (poor solutions). However, GSA suffers from the lack of balance between exploitation and exploration (Yazdani and Meybodi, 2015). In 2011, Rao et al. proposed **teaching learning-based optimization (TLBO)** (Rao et al., 2011) inspired by the teaching-learning environment in the classroom. The search agents represent a class of learners who interact with each other to find the global optimum. The algorithm involves two phases: the teacher phase and the learner phase. In the teacher phase, the best solution is considered as the teacher, and the mean of the whole class is shifted towards it. In the learner phase, the concept of learning is mimicked through interaction with some random learners. In the learner phase, a search agent updates its position based on the difference between two randomly selected solutions. In 2012, Yang proposed the **flower pollination algorithm (FPA)** (Yang, 2012) inspired by the pollination process in flowers, which involves two methods: cross-pollination and self-pollination. Cross-pollination promotes exploration, and self-pollination promotes exploitation. A flower is considered a candidate solution, where in the cross-pollination phase, its position is updated in reference to the global best by using the Levy distribution. In the phase of self-pollination, its position is updated with respect to the difference between two randomly selected solutions.

In 2014, Moosavian and Roodsari proposed the **soccer league competition algorithm (SLCA)** (Moosavian and Roodsari, 2014) inspired by competitions between the teams (clubs) in a soccer league. A total of n teams participate in a league, each playing a match against every other team. A team has 11 fixed players as well as a few substitute players. The teams struggle to earn points, and players on a team struggle to become star players on their teams and superstar players in the league. In a match, the winner is decided by comparing the average fitness of all fixed players of one team with the average fitness of all fixed players of another team. After each match, two operators are applied: imitation and provocation. In the imitation phase, the fixed players of the winning team update their positions, and in the provocation phase, substitute players attempt to become fixed players by updating their positions. In 2014, Mirjalili et al. proposed **grey wolf optimizer (GWO)** (Mirjalili et al., 2014) inspired by the social hierarchy and the hunting behaviour of grey wolves. To simulate the social hierarchy (leadership), the population of grey wolves is divided into four levels: alpha (α), beta (β), delta (δ), and omega (ω). The hunting behaviour is divided into four steps: searching for the prey, encircling the prey, hunting the prey, and attacking the prey. The encircling and hunting behaviour are modelled by updating the position of the search agent with respect to the best solution(s) and taking the average of the updated positions. However, the attacking and searching behaviours are modelled through different values of the parameters. Later, in 2016, Mirjalili and Lewis proposed the **whale optimization algorithm (WOA)** (Mirjalili and Lewis, 2016) inspired by the bubble-net foraging behaviour of humpback whales. To hunt their prey, whales generate bubbles, rotate around their prey in a spiral fashion, and move upwards. The algorithm is divided into three phases: searching for the prey, encircling the prey, and mimicking the bubble-net attacking method. The encircling and searching phases are implemented similarly to GWO, but in the searching phase, a random solution is referenced instead of the best solution. However, the bubble-net attacking method is modelled through a logarithmic spiral equation. Exploration and exploitation are balanced by the adaptive parameters. In the same year, Mirjalili proposed the **sine cosine algorithm (SCA)** (Mirjalili, 2016) inspired by two trigonometric functions, sine and cosine. SCA randomly initiates a population of N search agents, and the fittest solution is proclaimed as the destination solution. All search agents update their positions with respect to the destination solution $\vec{P}$ by using either the sine function or cosine function. The algorithm performs exploration in the first half of the iterations and performs exploitation in the second half.

2.3. Metaheuristics developed in 2017–2018

Some well-known metaheuristics developed in 2017 are the vibrating particles system algorithm (VPSA) (Kaveh and Ghazaan, 2017) inspired by the free vibration of single-degree-of-freedom systems with viscous damping, the tree growth algorithm (TGA) (Cheraghalipour and Hajiaghaei-Keshteli, 2017) inspired by the competition of trees when acquiring light and food, the cyclical parthenogenesis algorithm (CPA) (Kaveh and Zolghadr, 2017) inspired by the reproduction and social behaviour of some zoological species, such as aphids, which can reproduce with and without mating, the water evaporation algorithm (WEA) (Saha et al., 2017) inspired by the evaporation (vaporization) of small quantities of water particles from dense surfaces, the thermal exchange optimization algorithm (TEOA) (Kaveh and Dadras, 2017) inspired by Newton's law of cooling, spotted hyena optimizer (SHO) algorithm (Dhiman and Kumar, 2017) inspired by the behaviour of spotted hyenas, and the human mental search algorithm (HMSA) (Mousavirad and Ebrahimpour-Komleh, 2017) inspired by the exploration strategies of the bid space in online auctions. Moreover, well-known metaheuristics proposed in 2018 are the farmland fertility algorithm (FFA) (Shayanfar and Gharehchopogh, 2018) inspired by farmland fertility in nature, earthworm optimization algorithm (EOA) (Wang et al., 2018) inspired by earthworm contributions in nature, the rhino herd algorithm (RHA) (Gao et al., 2018) inspired by the herding behaviour of rhinos, queuing search algorithm (QSA) (Zhang et al., 2018) inspired by human activities when queuing, car tracking optimization algorithm (CTOA) (Chen et al., 2018) inspired by observing the programming methods of other metaheuristic algorithms, the self-defence mechanism of the plants (SDMP) algorithm (Caraveo et al., 2018) inspired by the self-defence mechanisms of plants, and the coyote optimization algorithm (COA) (Pierezan and Coelho, 2018) inspired by the Canis latrans species.

2.4. Metaheuristics developed in 2019

Some well-known evolution/swarm-based metaheuristics developed in 2019 are the squirrel search algorithm (SqSA) (Jain et al., 2019) inspired by the foraging behaviour of flying squirrels, the seagull optimization algorithm (SOA) (Dhiman and Kumar, 2019) inspired by the migration and attacking behaviours of seagulls, the Hitchcock bird inspired algorithm (HBIA) (Morais et al., 2018) inspired by the aggressive bird behaviour portrayed by Hitchcock, the sea lion optimization (SLnO) algorithm (Masadeh et al., 2019) inspired by the hunting behaviour of sea lions, the emperor penguins colony (EPC) algorithm (Harifi et al., 2019) inspired by the spiral-like movement of penguins colonies, the sailfish optimizer (SFO) (Shadravan et al., 2019) inspired by the hunting strategies of sailfishes, the bald eagle search (BES) (Alsattar et al., 2019) inspired by the hunting strategies of bald eagles, the Naked mole-rat (NMR) algorithm (Salgotra and Singh, 2019) inspired by the mating patterns of naked mole-rats, the butterfly optimization algorithm (BOA) (Arora and Singh, 2018) inspired by the food searching and mating behaviours of butterflies, and Harris hawks optimization (HHO) (Heidari et al., 2019) inspired by the cooperative and chasing behaviours of Harris hawks. In addition to evolution/swarm-based algorithms, some of the human social interaction-based algorithms developed in 2019 are the ludo game-based swarm intelligence (LGSI) algorithm (Singh et al., 2019) inspired by ludo playing strategies, the poor and rich optimization (PRO) algorithm (Moosavi and Bardsiri, 2019) inspired by the behaviours of people as they obtain wealth and improve their economic situation, the expectation algorithm (ExA) (Shastri et al., 2019) inspired by societal individuals as problem variables, the social media inspired algorithm (SMIA) (Crawford et al., 2019) inspired by interactions on social media, the supply–demand-based optimization (SDO) algorithm (Zhao et al., 2019) inspired by the relations between suppliers and consumers, the nomadic people optimizer (NPO) algorithm (Salih and Alsewari, 2019) inspired by the behaviour of nomads, the social mimic optimization (SMO) algorithm (Balochian and Baloochian, 2019) inspired by assimilation to famous people, find–fix–finish–exploit–analyse (F3EA) (Kashan et al., 2019) inspired by war strategies, and the deer hunting optimization algorithm (DHOA) (Brammya et al., 2019) inspired by the deer hunting strategy of humans.

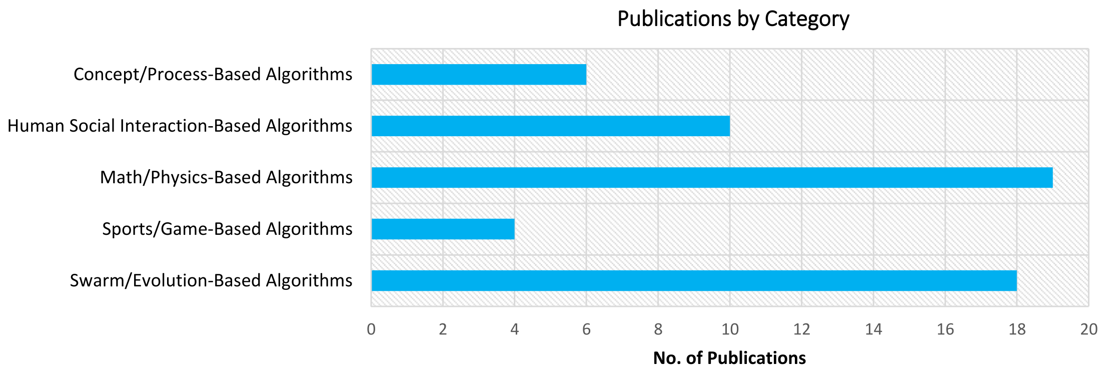

Fig. 1. Number of novel metaheuristics by category.

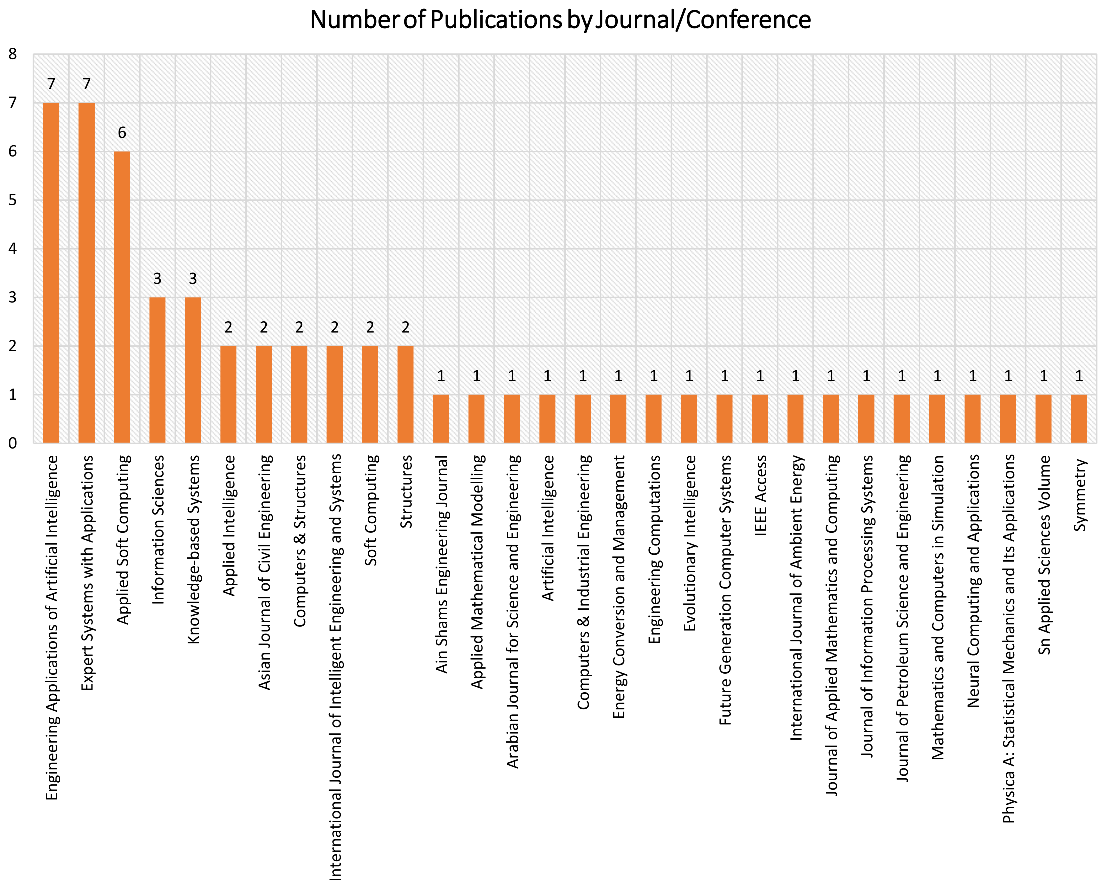

Fig. 2. Number of novel metaheuristics published in any journal/conference.

3. Recently proposed novel metaheuristics

After thoroughly searching the literature using approximately 30 keywords and narrowing the results by ignoring variant and hybrid approaches, we developed a list of 57 novel metaheuristics published in 2020–21. However, we do not claim that this list is all-inclusive. For a quick review, the metaheuristics along with their source of inspiration are presented in Table 1. Based on their source of inspiration, we classify the metaheuristics into five major groups: evolution/swarm-based algorithms, sports/game-based algorithms, math/physics-based algorithms, human social interaction-based algorithms, and concept/process-based algorithms. The development of algorithms in 2020–21 based on these categories is highlighted in Fig. 1. As shown in this figure, most of the recently developed metaheuristics belong to the math/physics-based algorithm and evolution/swarm-based algorithm categories. Moreover, in the course of our research, we have learned that most metaheuristic research is published in Expert Systems with Applications and Engineering Applications of Artificial Intelligence. For clarity, we have summarized the number of metaheuristics published in any journal/conference in Fig. 2.

3.1. *Swarm/evolution-based algorithms*

In this subsection, the mechanisms of recently proposed swarm-based or evolution-based algorithms are discussed.

3.1.1. *Barnacles mating optimizer (BMO)*

BMO (Sulaiman et al., 2020) is inspired by the mating behaviour of barnacles. Barnacles are microorganisms that are considered candidate solutions in this algorithm. The algorithm comprises of two phases: selection and reproduction. In the selection phase, two parent barnacles

Table 1
List of recently published metaheuristics.

Algorithm	Source of inspiration
Swarm/evolution-based algorithms	
Barnacles mating optimizer (BMO) (Sulaiman et al., 2020)	Mating behaviour of barnacles
Bear smell search algorithm (BSSA) (Ghasemi-Marzbali, 2020)	Movement and sensing behaviours of bears and their ability to smell
Black widow optimization algorithm (BWOA) (Hayyolalam and Kazem, 2020)	Mating behaviour of black widow spiders
Caledonian crow learning algorithm (CCLA) (Al-Sorori and Mohsen, 2020)	Learning behaviour of New Caledonian crows as they develop tools from Pandanus trees to obtain food
Chimp optimization algorithm (ChOA) (Khishe and Mosavi, 2020)	Individual intelligence and sexual motivation of chimps in group hunting
Coronavirus optimization algorithm (CvOA) (Martínez-Álvarez et al., 2020)	How coronaviruses spread and infect their hosts
Dynamic group-based cooperative optimization (DGBCO) (Fouad et al., 2020)	Cooperative behaviour is adopted by individuals in a swarm to achieve global goals
Manta ray foraging optimization (MRFO) (Zhao et al., 2020)	Intelligent foraging behaviour of manta rays
Marine predators algorithm (MPA) (Faramarzi et al., 2020a)	Lévy and Brownian movements in ocean predators
Mayfly algorithm (MA) (Zervoudakis and Tsafarakis, 2020)	Flight and mating behaviours of mayflies
Parallel fully dynamic iterative bio-inspired (PFDIBI) algorithm (Arslan, 2020)	Physarum Polycephalum foraging
Parasitism-predation algorithm (PPA) (Mohamed et al., 2020)	Crow–cuckoo–cat system model
Red deer algorithm (RDA) (Fathollahi-Fard et al., 2020)	Mating behaviour of Scottish red deer in breeding season
Shuffled shepherd optimization method (SSOM) (Kaveh and Zaerreza, 2020)	Using animal instinct to find the best way
Sine tree-seed algorithm (STSA) (Jiang et al., 2020)	Relation between trees and their seeds
Slime Mould Algorithm (SMA) (Li et al., 2020)	Oscillation mode of slime mould in nature
Tunicate swarm algorithm (TSA) (Kaur et al., 2020)	Jet propulsion and swarm behaviours of the tunicates
Water strider algorithm (WSA) (Kaveh and Eslamlou, 2020)	Tunicate life cycle of water strider bugs
Ebola optimization search algorithm (EOSA) (Oyelade et al., 2022)	Propagation mechanism of the Ebola virus
Aquila optimizer (AO) (Abualigah et al., 2021b)	Prey catching behaviour of Aquila
Reptile search algorithm (RSA) (Abualigah et al., 2022)	Hunting behaviour of crocodiles
Arithmetic optimization algorithm (RrOA) (Abualigah et al., 2021a)	Arithmetic operators
Dwarf mongoose optimization (DMO) (Agushaka et al., 2022)	Foraging behaviour of the dwarf mongoose
Sports/game-based algorithms	
Billiards-inspired optimization algorithm (BIOA) (Kaveh et al., 2020a)	Billiard ball collisions
Darts game optimizer (DGO) (Dehghani et al., 2020a)	Simulating the rules of the darts game
Kho-kho optimization algorithm (KKOA) (Srivastava and Das, 2020)	Strategies used by players in a tag-team game
Hide objects game optimization (HOGO) (Dehghani et al., 2020b)	Search agents who try to find a hidden object in a given space
Math/physics-based algorithms	
Archimedes optimization algorithm (AOA) (Hashim et al., 2020)	Archimedes' principle
Artificial electric field algorithm (AEFA) (Yadav et al., 2020)	Coulomb's law of electrostatic force and Newton's law of motion
Balancing composite motion optimization (BCMO) (Le-Duc et al., 2020)	Balancing composite motions
Black hole mechanics optimization (BHMO) (Kaveh et al., 2020b)	Black hole mechanics
Dynamic differential annealed optimization (DDAO) (Ghafil and Jármai, 2020)	Random search and classical SA
Equilibrium optimizer (EO) (Faramarzi et al., 2020b)	Control volume mass balance models
Filter nonmonotone adaptive trust region method (FNATRM) (Wang et al., 2020)	Nonmonotone trust-region ratio
Generalized normal distribution optimization (GNDO) (Zhang et al., 2020)	The generalized normal distribution model
Gradient-based optimizer (GBO) (Ahmadianfar et al., 2020)	Newton's gradient-based method
Grey prediction evolution (GPE) (Hu et al., 2020)	Even grey model
Limited memory Q-BFGS algorithm (LMQA) (Lai et al., 2020)	BFGS-type update using q-derivatives
Momentum search algorithm (MSA) (Dehghani and Samet, 2020)	Newton's law of conservation of momentum
Newton metaheuristic algorithm (NMA) (Gholizadeh et al., 2020)	Newton's gradient-based method
Photon search algorithm (PSA) (Liu and Li, 2020)	Photon properties in the field of physics
Quantum-inspired algorithm (QIA) (Mu et al., 2020)	Quantum computing
Transient search optimization (TSO) (Qais et al., 2020)	Transient behaviour of switched electrical circuits
Turbulent flow of water-based optimization (TFWO) (Ghasemi et al., 2020)	Whirlpools created in turbulent water
Vapour–liquid equilibrium (VLE) algorithm (Taramsco et al., 2020)	Vapour–liquid equilibrium process
Volcano eruption algorithm (VEA) (Hosseini et al., 2020)	Nature of volcano eruption
Human social interaction-based algorithms	
Adolescent identity search algorithm (AISA) (Bogar and Beyhan, 2020)	Identity development/search of adolescents
Forensic-based investigation (FBI) algorithm (Chou and Nguyen, 2020)	Suspect investigation–location–pursuit process of police officers
Giza pyramid construction (GPC) (Harifi et al., 2020)	Pushing stones on ramps by workers to construct pyramids
Group teaching optimization algorithm (GTOA) (Zhang and Jin, 2020)	Group teaching mechanism
Heap-based optimizer (HBO) (Askari et al., 2020a)	Corporate rank hierarchy
Human dynasties optimization algorithm (HDOA) (Wagan et al., 2020)	Social behaviour in human dynasties
Human urbanization algorithm (HUA) (Ghasemian et al., 2020)	Human behaviour for urbanization and improving life situations
Interactive autodidactic school (IAS) (Jahangiri et al., 2020)	Interactions between students in an autodidactic school
Political optimizer (PO) (Askari et al., 2020b)	Multiparty political system
Search and rescue optimization algorithm (SROA) (Shabani et al., 2020)	Exploration behaviour of humans during search and rescue operations

(continued on next page)

Table 1 (*continued*).

Algorithm	Source of inspiration
Concept/process-based algorithms	
Group optimization (GO) (Dehghani et al., 2020)	Whole group involvement in position updating
Lévy flight distribution (LFD) (Houssein et al., 2020)	Lévy flight random walk for exploration
Color harmony algorithm (CHA) (Zaeimi and Ghoddosian, 2020)	Combining harmonic colours based on their relative positions around the hue circle
Opposition-based high dimensional optimization (OHDO) algorithm (GhaemiDizaji et al., 2020)	Angular movement according to a few selected dimensions
Optimization algorithm based on OCM And PCM (OAOP) (Gong and razmjooy, 2020)	Cost minimization and payment cost minimization
Rain optimization algorithm (ROA) (Moazzeni and Khamehchi, 2020)	Movement of rain drops to the minimum point

are selected based on the length of their penises (pl). In reproduction phase, the Hardy–Weinberg principle is used for offspring generation. Suppose pl of the father's barnacle is in the selection range of the parent barnacles. Then, p% characteristics are selected from the father, and $(1-p)$% characteristics are selected from the mother; otherwise, a new offspring is generated by mutating only the mother's characteristics. The former approach promotes exploitation, and the latter promotes exploration.

The generation of offspring from the mating process of the parents is formulated in the following equation:

$$x_i^{N_new} = p x^N_{barnacle_d} + q x^N_{barnacle_m} \tag{1}$$

where p is a random number in the range of [0, 1], and q is equal to $(1 - p)$. The solution for the father is represented by $barnacle_d$ and the solution for the mother is represented by $barnacle_m$. The cap is exceeded if barnacle1 chooses barnacle8. Thus, the usual mating process is stopped. The process of casting sperm is currently used to generate the offspring. The sperm cast, as it is termed in BMO, describes the exploration process as follows:

$$x_i^{N_new} = rand() \times x^n_{barnacle_m} \tag{2}$$

where $rand()$ generates a random number in the range [0, 1].

3.1.2. Black widow optimization algorithm (BWOA)

BWOA (Hayyolalam and Kazem, 2020) is inspired by the mating behaviour of black widow spiders. Female black widow spiders typically construct their nets at night and disperse pheromones in particular areas in order to attract male black spiders to mate with. As male black widow spiders are drawn to this pheromone, they are drawn into the web. The female black widow spider eats the male black widow spider after or during mating. The female black widow lays her egg sacks in the net after mating. The eggs hatch into young, cannibalistic spiders after 11 days. Strangely, the mother may occasionally eat some of the young spiders while they are held in the mother's net. This suggests that the surviving spiders are the most healthy and fittest spiders, which is the motivation behind this new algorithm.

A black widow spider is considered as a candidate solution, and initially, a population of spiders is randomly generated. The algorithm involves three main phases: procreation, cannibalism, and mutation. In the procreation step, randomly selected pairs of spiders mate to generate offspring. The cannibalism phase involves three behaviours: (i) mother eats father, (ii) sibling eats other siblings, and (iii) offspring eat mother. Finally, in the mutation step, two randomly selected elements of an individual are exchanged. Rate exploration is incorporated in the procreation step, exploitation is introduced and the convergence speed is enhanced in the cannibalism step, and exploration and exploitation are balanced in the mutation step.

The reproduction process of BWOA is formulated in the following equation:

$$Y_{i,d} = \beta \times X_{i,d} + (1-\beta) \times X_{j,d}$$

$$Y_{j,d} = \beta \times X_{j,d} + (1-\beta) \times X_{i,d} \tag{3}$$

In above equation, i and j are random numbers that are generated in the range [0, 1], β is also a random number but its range is [1, N], and $Y_{i,d}$ and $Y_{j,d}$ are the offspring. In addition to reproduction, the mutation operation is also formulated, as shown below:

$$Z_{k,d} = Y_{k,d} + \alpha \tag{4}$$

where α is a random mutation, Z is the mutated spider, and Y is the randomly selected spider that is being mutated.

3.1.3. Chimp optimization algorithm (ChOA)

ChOA (Khishe and Mosavi, 2020) is a swarm-based algorithm inspired by the individual intelligence of chimps and their sexual motivation in group hunting. Each chimp is considered a candidate solution. The process of evolution is led by the top four solutions in the population, which are the attacker, driver, barrier, and chaser solutions. The algorithm involves two main phases: (i) driving, blocking, and chasing the prey, which promote exploration, and (ii) attacking the prey, which promotes exploitation. Each solution updates its position with respect to either the four best solutions or the mean of all four solutions. The transition between exploration and exploitation is controlled with the help of carefully designed parameters. Moreover, to escape local optima and enhance the convergence speed, six types of chaotic maps are used in the position-updating mechanism of ChOA.

The positions of the attacker, driver, barrier, and chaser solutions are updated using the following mathematical equations:

$$x_{attacker} = x_{attacker} - A_1(C_1 x_{attacker} - M_1 x)$$

$$x_{driver} = x_{driver} - A_2(C_2 x_{driver} - M_2 x)$$

$$x_{barrier} = x_{barrier} - A_3(C_3 x_{barrier} - M_3 x)$$

$$x_{chaser} = x_{chaser} - A_4(C_4 x_{chaser} - M_4 x)$$

In the above equations, $X_{attacker}$, X_{driver}, $X_{barrier}$, and X_{chaser} are the positions of the attacker, driver, barrier, and chaser, respectively. Furthermore, M_i is the chaotic map, and other parameters, such as A and C, are computed as follows:

$$A = 2 \cdot f \cdot r_1 - f$$

$$C = 2 \cdot r_2$$

where r_1 and r_2 are random variables with values in the range [0, 1], and f nonlinearly decreases from 2.5 to 0. The final position updating equation is presented below, in which the average of all top four chimps is calculated.

$$x(t+1) = \frac{x_{attacker} + x_{driver} + x_{barrier} + x_{chaser}}{4} \tag{5}$$

3.1.4. Manta ray foraging optimization (MRFO)

MRFO (Zhao et al., 2020) is inspired by the intelligent foraging behaviours of manta rays, which are large marine creatures. In this algorithm, a manta ray is considered a candidate solution, and the best-known solution is plankton, which is a food source for manta rays. The algorithm comprises three phases to simulate three different foraging behaviours of manta rays: chain foraging, cyclone foraging, and somersault foraging. In chain foraging, the manta rays follow each other by swimming in a row; the leader (best solution) is the nearest to the plankton. This behaviour is mapped by deriving an equation to update the position of each solution with regard to the global best and next index-wise solution, which is presented below:

$$x_i(t+1) = \begin{cases} x_i + r \cdot \left(x_{\text{best}} - x_i(t)\right) + \alpha \cdot \left(x_{best} - x_i(t)\right) & i = 1 \\ x_i + r \cdot \left(x_{i-1}(t) - x_i(t)\right) + \alpha \cdot \left(x_{best} - x_i(t)\right) & i = 2, \dots, N \end{cases} \tag{6}$$

where α is the weight coefficient and r is a random number in the range of [0, 1]. In cyclone foraging, the manta rays follow each other and swim around the plankton in a spiral form, making a cyclone in the water. This phase is mimicked in two ways. To promote exploitation, the position of each solution is updated with respect to the global best and next manta ray by constructing a spiral equation. This phase is mathematically expressed below:

$$x_i(t+1) = \begin{cases} x_{best} + r \cdot \left(x_{\text{best}} - x_i(t)\right) + \beta \cdot \left(x_{best} - x_i(t)\right) & i = 1 \\ x_{best} + r \cdot \left(x_{i-1}(t) - x_i(t)\right) + \beta \cdot \left(x_{best} - x_i(t)\right) & i = 2, \dots, N \end{cases} \tag{7}$$

where β is the weight coefficient and x_{best} is the best solution found thus far. To promote exploration, the position of each solution is updated with respect to a randomly selected manta ray and the next manta ray. The mathematical formulation of this phase is presented below:

$$x_i(t+1) = \begin{cases} x_{rand} + r \cdot \left(x_{\text{rand}} - x_i(t)\right) + \beta \cdot \left(x_{rand} - x_i(t)\right) & i = 1 \\ x_{rand} + r \cdot \left(x_{i-1}(t) - x_i(t)\right) + \beta \cdot \left(x_{rand} - x_i(t)\right) & i = 2, \dots, N \end{cases} \tag{8}$$

where x_{rand} is a randomly selected solution. It is important to mention here that the balance between exploration and exploitation is accomplished by giving more time to the exploration equation in early iterations. In contrast, the rate of selection of the other equation increases with the number of iterations. Finally, in somersault foraging, the manta rays move to and around the global best solution by considering it as the food for updating their positions. The equation is expressed below:

$$x_i^d(t+1) = x_i^d(t) + S \cdot \left(r_2 \cdot x_{\text{best}}^d - r_3 \cdot x_i^d(t)\right), i = 1, \dots, N \tag{9}$$

where r_2 and r_3 are two random values in the range [0, 1], S is the somersault factor that determines the somersault range of manta rays, and $S = 2$.

3.1.5. Marine predators algorithm (MPA)

MPA (Faramarzi et al., 2020a) is a swarm-based algorithm inspired by Levy and Brownian movements of predators and prey in the ocean. The algorithm comprises of three major phases: (i) position updating of the prey when the predator moves faster than the prey, (ii) position updating of the prey when the predator and prey both move at the same speed, and (iii) position updating of the prey when the prey moves faster than the predator. In phase 1, exploration is promoted in the first $\frac{1}{3}$ of iterations. The mathematical formulation of this phase is formulated below:

$$prey_i = prey_i + P * R \times R_B \times (Elite_i - R_B \times Prey_i) \tag{10}$$

where $prey_i$ is the ith solution, P is a constant and fixed at 5, R is a random number in the range of [0, 1], $\times$ denotes elementwise multiplication, and $Elite_i$ is the ith vector from the list of elite solutions. In phase 2, exploration as well as exploitation incorporated by mapping the transition between exploration and exploitation in the second $\frac{1}{3}$ of iterations. The mathematical form of this phase is expressed through the following equation:

$$prey_i = prey_i + P * R \times R_L \times (Elite_i - R_L \times Prey_i) \tag{11}$$

$$prey_i = Ellite_i + P * CF \times R_B \times (Elite_i - R_B \times Prey_i) \tag{12}$$

where the former equation is used to update the first half of the population and the latter is used to update the second half of the population. In the above equations, R_L is a vector that is generated randomly using the Levy flight distribution, while CF adaptively regulate the motion of the predator. Finally, phase 3 emphasizes exploitation in the last $\frac{1}{3}$ of the iterations and is formulated as follows:

$$prey_i = Ellite_i + P * CF \times R_L \times (R_L \times Elite_i - R_B) \tag{13}$$

The predator performs a different type of movement in each phase. Moreover, the idea of FAD's effect is incorporated to enable solutions to take longer jumps to escape the local optima. Finally, the concept of marine memory is used to allow a solution to update its position only if it has improved compared to its previous position.

3.1.6. Mayfly algorithm (MA)

MA (Zervoudakis and Tsafarakis, 2020) is a swarm-based algorithm inspired by the flight and mating behaviours of mayflies. The position of each mayfly in the search space is considered a candidate solution. The population is subdivided into two groups: male mayflies and female mayflies. The algorithm comprises three phases: movement of male mayflies, movement of female mayflies, and mating of mayflies. The movement of male mayflies is determined by the difference of current fitness of the male mayfly and his best historical fitness. If his current fitness is better, then the following equation is used to update the velocity of the male mayfly.

$$v_i(t+1) = g \cdot v_i(t) + a_1 \exp^{\beta r_p^2}[x_{h_i} - x_i(t)] + a_2 \exp^{\beta r_g^2}[x_g - x_i(t)] \tag{14}$$

where g is a variable whose value drops linearly from highest to lowest. The values are balanced using three constants: α_1, α_2, and β. The two variables r_p and r_g are used to indicate the Cartesian distance between solution's current position and previous best position. In contrast, if the current fitness is not better, then the following equation is used to update the velocity.

$$v_i(t+1) = g \cdot v_i(t) + d \cdot r_1 \tag{15}$$

where r_1 is randomly generated in the range [−1, 1]. The movement of female mayflies is mapped by attracting female mayflies towards male mayflies, which is again formulated in two ways depending on the relative fitness of the female mayfly.

$$v_i(t+1) = g \cdot v_i(t) + a_3 \exp^{\beta r_{mf}^2}[x_i(t)] - [y_i(t)] \tag{16}$$

where a_3 is a constant and r_m represents the Cartesian distance. If the fitness of the female mayfly is not better, then the following equation is used to update the velocity.

$$v_i(t+1) = g \cdot v_i(t) + fl \cdot r_2 \tag{17}$$

where r_2 is a randomly generated value in the range of [−1, 1]. Finally, the matting phase is accomplished by performing a crossover among a male mayfly, and a female mayfly selected randomly or based on their fitness. The authors incorporated a few modifications, such as limiting velocity, adding a gravity coefficient, reducing nuptial dances and random walks, and mutating offspring to handle premature convergence and stability issues.

3.1.7. Parasitism – Predation algorithm (PPA)

PPA (Mohamed et al., 2020) is inspired by the interaction between the predator (cats), the parasite (cuckoos), and the host (crows) in the crow–cuckoo–cat system model. PPA is designed to preserve the strengths and overcome the weaknesses of cat swarm optimization (CSO) (Chu et al., 2006), cuckoo search (CS) (Yang and Deb, 2009), and the crow search algorithm (CSA) (Askarzadeh, 2016). This algorithm comprises three main phases: the nesting phase, parasitism phase, and predation phase. The nesting phase, also called the crow's phase, promotes exploration. Crows (candidate solutions) fly from one state to the other and update their positions based on randomly selected crows by using Levy flight. The mathematical formulation of this phase is given below:

$$X_i^{t+1} = X_i^t + F(X_{r1} - X_i^t) \tag{18}$$

where X_{r1} is a randomly selected crow and F is a step size computed through the Levy flight distribution. In the parasitism phase, also called the crow–cuckoo phase, some nests (solutions) selected through a roulette wheel are replaced with new nests, whereas new nests are constructed by utilizing two random solutions. The equation for this phase is given below:

$$X_{i,new}^{cuckoo} = X_{i,old}^{cuckoo} + S_G \cdot K \tag{19}$$

where $X_{i,old}^{cuckoo}$ is chosen through a roulette wheel, S_G is step size generated through a uniform Gaussian distribution, and K is a random binary matrix. In the predation phase, also called the crow–cat phase, cats search nonparasitized nests. The movement of cats is mimicked by updating the velocities of the cats by utilizing the best solution and updating the positions based on the updated velocities. The equations to update the velocities and positions are given below:

$$v_{k,d} = v_{k,d} + r \cdot c \cdot (x_{best,d} - x_{k,d}) \tag{20}$$

where $x_{best,d}$ denotes the best cat position and $v_{k,d}$ denotes the velocity of the cat. Once the velocity is updated, the final position is updated as follows:

$$x_{k,d} = x_{k,d} + v_{k,d} \tag{21}$$

All three phases run sequentially to balance exploration and exploitation.

3.1.8. Red deer algorithm (RDA)

RDA (Fathollahi-Fard et al., 2020) is inspired by the mating behaviour of red deer in the breeding season. In this algorithm, a deer is considered a candidate solution, and the whole population is divided into two groups: males and hinds. Males are comparatively better solutions than hinds. The algorithm involves 7 steps: roar, select commander, fight between commanders and stags, form harems, mate commanders, mate stags, and select the next generation. In the roar phase, males randomly update their positions. In the commander selection phase, γ% of males are selected as commanders, and the remaining males are stags. The fight between commanders and stags is mimicked through the position-updating equations presented below:

$$new_1 = (C + S)/2 + b_1((UB - LB) * b_2 + LB) \tag{22}$$

$$new_2 = (C + S)/2 - b_1((UB - LB) * b_2 + LB) \tag{23}$$

where new_1 and new_2 are two new solutions that are produced as a result of the fight. In above equations, C is the commander solution and S denotes the stags. Moreover, b_1 and b_2 are random numbers that are generated in the range of [0, 1] through uniform distribution. In the harem formation phase, hinds are grouped together based on the strength of their commander. The mating phase is mapped by an equation involving a commander and a selected hind from the same group and a different group. The mating process is formulated in the following equation:

$$offs = \frac{C + Hind}{2} + (UB - LB) \times c \tag{24}$$

In the stag mating phase, each stag mates with the hind closest to it. Finally, the next generation is selected. Exploration is incorporated in the harem formation and commander mating phases. Exploitation is achieved in the roar, commander selection, fight between commanders and stags, and stag mating phases. The local optima are escaped through the next generation selection.

3.1.9. Slime mould algorithm (SMA)

SMA (Li et al., 2020) is a swarm-based algorithm inspired by the oscillation mode of slime mould (a kind of fungus) in nature. Each candidate solution corresponds to a unique location of the slime mould. The algorithm maps three behaviours of slime mould: (i) approaching food, (ii) wrapping the food, and (iii) oscillation. The slime mould approaches food by following its odour in the air, for which a smell index is calculated. With the help of two randomly selected solutions and the best solution, the positions of current search agents are updated, which is expressed through the following equation:

$$\vec{S}(t+1) = \begin{cases} \vec{S_b}(t) + \vec{vb} * \left(\vec{W} * \vec{S_A}(t) - \vec{S_B}(t)\right), & r < p \\ \vec{vC} * \vec{S}(t), & r \geq p \end{cases} \tag{25}$$

where the current position of the slime mould is denoted by $S(t)$, $S_A(t)$ and $S_b(t)$ are two randomly selected solutions, and $S_b(t)$ contains the most concentrated odour. Moreover, W is the slime mould weight, v_b and v_c are self-adaptive parameters, r is generated randomly in the range [0, 1], and p is computed as $p = \tanh(|f(i) - DF|)$, where f is the fitness of the current solution and DF is the fitness of the best solution. Exploration occurs in the approaching the food phase, and exploitation comes from the wrapping the food phase. The transition among both phases is controlled through the parameters.

3.1.10. Tunicate swarm algorithm (TSA)

TSA (Kaur et al., 2020) is inspired by jet propulsion and swarm behaviours of tunicates during navigation and foraging. The best solution is considered the food source, and in the jet propulsion phase, the tunicates (search agents) update their position with respect to the best solution because the objective is to keep the solutions close to the best solution. The mathematical equation to formulate this behaviour is given below:

$$P_p(x) = \begin{cases} X_{\text{best}} + A \cdot \vec{P}D, & \text{if } r_{\text{rand}} \geq 0.5 \\ X_{\text{best}} - A \cdot \vec{P}D, & \text{if } r_{\text{rand}} < 0.5 \end{cases} \tag{26}$$

where X_{best} is the best solution, and PD is computed as follows:

$$PD = |X_{best} - r_{rand} \cdot P_p(x)| \tag{27}$$

where $P - p$ is the position of the tunicate being updated and r_{rand} is a random number in the range [0, 1]. However, in the swarm behaviour phase, each search agent other than the first search agent updates its position by taking the mean of its position updated in the jet propulsion phase and the position of the previous search agent in the swarm. This behaviour is formulated in the following equation:

$$P_p(\vec{x}+1) = \frac{P_p(x) + P_p(\vec{x}+1)}{2 + c_1} \tag{28}$$

Exploration and exploitation are controlled through the parameters.

3.2. Math/physics-based algorithms

3.2.1. Archimedes optimization algorithm (AOA)

AOA (Hashim et al., 2020) is a physics-based algorithm inspired by Archimedes' principle, in which an immersed object is considered a candidate solution. A population of objects along with densities, volumes, and accelerations are randomly initialized. The algorithm comprises several steps. In step 1, the densities and volumes of the objects are updated by using the following equations.

$$den_i^{t+1} = den_i^t + rand \times (den_{best} - den_i^t) \tag{29}$$

$$vol_i^{t+1} = vol_i^t + rand \times (vol_{best} - vol_i^t) \tag{30}$$

where den_{best} and vol_{best} are the densities and volumes of the best solution found thus far. In step 2, two operators, the transfer operator and density decreasing factor, are calculated. The density decreasing factor helps to incorporate exploration, and the transfer operator controls the transition from exploration to exploitation. In step 3, the acceleration of each object is updated either by considering the collision with a random material or no collision with any object. In the exploration phase, it is assumed that the objects collide. In the exploitation phase, it is assumed that the objects do not collide. In step 4, the accelerations are normalized, and the positions of the objects are updated either with reference to a randomly selected object (exploration phase) or the global best object (exploitation phase) by using the following equations:

$$x_i^{t+1} = x_i^t + C_1 \times rand \times acc_{i-norm}^{t+1} \times d \times (x_{rand} - x_i^t) \tag{31}$$

$$x_i^{t+1} = x_best^t + F \times C_2 \times rand \times acc_{i-norm}^{t+1} \times d \times (T \times x_{best} - x_i^t) \tag{32}$$

where C_1 is a constant with a value of 2 and C_2 is a constant with a value of 6. Moreover, acc_i is the acceleration of the current solution, d is the density decreasing factor, and $rand$ is a random number in the range [0, 1].

3.2.2. Equilibrium optimizer (EO)

EO (Faramarzi et al., 2020b) is a physics-based meta-heuristic inspired by the control volume mass balance models used to estimate both dynamic and equilibrium states. The particles are considered solutions, and their positions map the concentration of the particles. In this algorithm, an equilibrium pool of five reference solutions (four best so-far particles and the arithmetic mean of them) called equilibrium candidates is constructed. The equilibrium pool is formulated below:

$$\overrightarrow{X}_{e,pool} = \{\overrightarrow{X}_{e(1)}, \overrightarrow{X}_{e(2)}, \overrightarrow{X}_{e(3)}, \overrightarrow{X}_{e(4)}, \overrightarrow{X}_{e(ave)}\} \tag{33}$$

Each particle updates its position in reference to a randomly selected candidate from the pool. The algorithm contains two carefully designed parameters called the exponential term (E) and generation rate (G) that are computed below:

$$\overrightarrow{E} = e^{-\overrightarrow{\lambda}(t-t_0)} \tag{34}$$

where t and t_0 are computed as:

$$t = (1 - \frac{Iter}{Max\_iter})^{(a_2 \frac{Iter}{Max\_iter})} \tag{35}$$

$$\overrightarrow{t}_0 = \frac{1}{\overrightarrow{\lambda}} \ln(-a_1 sign(\overrightarrow{r} - 0.5)[1 - e^{-\overrightarrow{\lambda} t}]) + t \tag{36}$$

By combining the above equations, the final equation for E is given below:

$$\overrightarrow{E} = -a_1 sign(\overrightarrow{r} - 0.5)[e^{-\overrightarrow{\lambda} t} - 1] \tag{37}$$

Moreover, the concept of memory saving, which allows a solution to update its concentration only if it is improved as compared to the previous concentration, is implemented. The final position updating equation of EO is presented below:

$$\overrightarrow{X} = \overrightarrow{X}_{eq} + (\overrightarrow{X} - \overrightarrow{X}_{eq}) \cdot \overrightarrow{E} + \frac{\overrightarrow{R}}{\overrightarrow{\lambda} V}(1 - \overrightarrow{E}) \tag{38}$$

Exploration, exploitation, and their balance are controlled through the following parameters: equilibrium pool and generation probability.

3.2.3. Gradient-based optimization (GBO)

GBO (Ahmadianfar et al., 2020) is inspired by Newton's gradient method. The algorithm involves two main phases: the gradient search rule and the local escaping operator. Newton's method is a root-finding algorithm that estimates the nearby solution of the current solution by using the Taylor series. The gradient search rule is derived by modifying Newton's gradient-based method to solve indifferentiable problems. The position updating equation is presented below:

$$x_n^{m+1} = r_a \times (r_b \times X1_n^m + (1 - r_b) \times X2_n^m) + (1 - r_a) \times X3_n^m \tag{39}$$

where r_a and r_b are random numbers. To enhance the performance of GBO and deal with complex problems, a local escaping operator is introduced, which utilizes the best solution and five other solutions. The mathematical formulation of this operator is given below:

$$X_{LEO}^m = X_n^{m+1} + f_1(u_1 x_{best} - u_2 x_k^m) + f_2 p_1(u_3(x2_n^m - x1_n^m)) + u_2(x_{r1}^m - x_{r2}^m)/2 \tag{40}$$

where u_1, u_2, and u_3 are randomly generated values. Furthermore, f_1 and f_2 are also random numbers and can have values of either 1 or −1. The gradient search rule incorporates both exploration and exploitation through two different equations; however, the balance between exploration and exploitation is attained by modifying this rule through a random parameter p_1.

3.2.4. Transient search optimization (TSO)

TSO (Qais et al., 2020) is a physics-based algorithm inspired by the transient behaviour of switched electrical circuits. These electrical circuits can be either first-order circuits (containing a single storing component) or second-order circuits (containing two storage elements). Differential equations to compute the transient responses of both types of circuits are used to derive the position updating equations for the candidate solutions. Each candidate solution (a state of the circuit) updates its position with respect to the best solution (steady-state or final value of the circuit) according to one of the two equations derived for both circuits. The first-order equation incorporates exploitation, and the second-order equation promotes exploration. The second order equation is formulated below:

$$X(t+1) = X^*(t) + e^{-t}[\cos(2\pi T) + \sin(2\pi T)]|X(t) - C_1 X^*(t)| \tag{41}$$

where $X(t)$ denotes current position, $X(t+1)$ denotes the updated position, $X^*(t)$ is the global best position, T is a random coefficient and computed as $T = 2 \times z \times r_1 - z$, and C_1 is a random coefficient and computed as $C_1 = k \times z \times r_2 + 1$. On the other hand, the first-order equation is formulated as follows:

$$X(t+1) = X^*(t) + [X(t) - C_1 \cdot X^*(t)] \cdot e^{-T} \tag{42}$$

The balance between exploration and exploitation is attained through parameter T.

3.2.5. Turbulent flow of water-based optimization (TFWO)

TFWO (Ghasemi et al., 2020) is a physics-based algorithm inspired by the behaviour of whirlpools created in water due to turbulent flow. In this algorithm, the population is divided into several groups. The best solution of each group is treated as a whirlpool. The other solutions in the group are considered objects that are rotating around the whirlpool hole in a spiral fashion. The algorithm calculates two forces on an object to update its position. The first force is the centripetal force, which causes the object to be pulled down towards the centre of the whirlpool. The second force is the centrifugal force, which works against the centripetal force to pull objects away from the whirlpool centre. Moreover, the interaction between whirlpools is also mimicked so that whirlpools update their positions concerning other whirlpools. This interaction is mathematically formulated below:

$$\begin{aligned} \Delta X_i = {} & (\cos(\delta_i^{new}) * rand(1, D) * (Wh_f - X_i) \\ & - \sin(\delta_i^{new}) * rand(1, D) * (Wh_w - X_i)) \\ & * (1 + |\cos(\delta_i^{new}) - \sin(\delta_i^{new})|) \end{aligned} \tag{43}$$

$$X_i^{new} = Wh_j - \Delta X_i \tag{44}$$

where δ_i is the angle of the ith object and Wh_w and Wh_f are two whirlpools. In the original paper, a discussion on how the algorithm meets optimization challenges is not presented; however, according to the best of our understanding, the centripetal force incorporates exploitation, centrifugal force promotes exploration, and convergence comes from the interaction among the whirlpools.

3.2.6. Artificial electric field algorithm (AEFA)

AEFA (Yadav et al., 2020) is inspired by Coulomb's law of electrostatic force between charged particles and Newton's law of motion. According to Coulomb's law, there is an electrostatic force between two charged particles that is directly proportional to the product of the charge on those particles and inversely proportional to their squared distance. This force can be attractive or repulsive. However, AEFA considers the attractive force only. In AEFA, each charged particle is considered a candidate solution whose charge is computed from its fitness. The force between the charged particles is computed and then used to calculate the acceleration of a particle. The total electrostatic force acting on particle i is calculated by using the following equation:

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N} rand() F_{ij}^d(t) \tag{45}$$

where F_i is the resultant force, $rand()$ gives a random number in the range [0, 1], N is the total number of particles, and $F_{ij}^d(t)$ is the force acting on charge i from charge j. By using the total force, the acceleration of the ith particle is computed by using the following equation:

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \tag{46}$$

where $M_i(t)$ is the mass of the ith particle at time t. Acceleration is used to find the velocity based on the following equation:

$$V_i^d(t+1) = rand() * V_i^d(t) + a_i^d(t) \tag{47}$$

Once the velocity is updated, the current position can be updated by adding the updated velocity into the current position by using the following equation:

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \tag{48}$$

3.2.7. Balancing composite motion optimization (BCMO)

BCMO (Le-Duc et al., 2020) is a population-based metaheuristic algorithm that works by balancing the composite motion properties of individuals in the solution space. Equalizing the global and local searches via a probabilistic selection model creates a movement mechanism for each individual. The steps include initializing of the search space and determining of the instant global point, best individual, and the composite motions of individuals in solution space. The position-updating equation of the candidate solution is formulated below:

$$x_i^{t+1} = x_i^t + v_{i/j} + v_j \tag{49}$$

where $v_{i/j}$ and v_j are computed as follows:

$$v_{i/j} = \alpha_{ij}(x_j - x_i) \tag{50}$$

$$v_j = \alpha_j(x_{oin} - x_j) \tag{51}$$

In the above equations, α_{ij} and α_j are computed using the following equations.

$$\alpha_j = L_{GS} \times dv_j \tag{52}$$

$$\alpha_{ij} = L_{LS} \times dv_{ij} \tag{53}$$

where L_{GS} is a global step size scaling the movement of the jth individual, L_{LS} can be fixed at 1 and dv_{ij} is the direction vector between ith and jth individuals. The mathematical model based on random tests is built to control the movement trends of candidate solutions, which probabilistically equalizes and balances the exploration and exploitation capabilities of each individual.

3.3. Human social interaction-based algorithms

3.3.1. Forensic-based investigation (FBI)

FBI (Chou and Nguyen, 2020) is inspired by the suspect investigation–location–pursuit process. The forensic investigation process involves five steps: opening the case, interpreting the findings, directing the inquiry, taking action, and prosecuting the case. The suspected location is considered a candidate solution in the investigation phase, and the location of the police agent is treated as the candidate solution in the pursuit phase. The investigation phase involves two steps: (i) interpretation of the findings, in which a solution updates its position with reference to the average of two randomly selected solutions (suspected locations), and (ii) direction of inquiry, in which the solution updates its position with respect to the global best and three randomly selected solutions (suspected locations). The first step is mathematically formulated as follows:

$$X_{ij} = X_{ij} + ((rand_1 - 0.5) \times 2) \times (X_{ij} - (X_{kj} + X_{hj})/2) \tag{54}$$

where k, h, and i are three suspected locations. Moreover, $((rand_1 - 0.5) \times 2)$ generates a random number in the range $[-1, 1]$. Similarly, the second step is formulated by using the following equation:

$$X_{ij} = X_{best} + X_{dj} + rand_5 \times (X_{ej} - X_{fj}) \tag{55}$$

where d, e, f, and i are four suspected positions. In addition, the pursuit phase is a combination of two steps. In the first step, called actions, the solutions (police agent locations) update their positions with respect to the global best, and in the second step, the solution (police agent location) updates its position with respect to the global best and two randomly selected police agents. The first step is mathematically formulated below:

$$X_{ij} = rand_6 \times X_{ij} + rand_7 \times (X_{best} - X_{ij}) \tag{56}$$

and the second step is formulated by using the following equation:

$$X_{ij} = X_{rj} + rand_8 \times (X_{rj} - X_{ij}) + rand_9 \times (X_{best} - X_{rj}) \tag{57}$$

where all $rand_i$'s are random numbers, and r and i are two police locations. To meet the optimization challenges, the investigation phase incorporates exploration, the pursuit phase incorporates exploitation, and the balance among both challenges is attained through the execution of both phases in a row.

3.3.2. Giza pyramids construction (GPC)

GPC (Harifi et al., 2020) is a human social interaction-based algorithm inspired by the movement of stones on a ramp by workers to construct polynomials. The authors claimed that GPC, which is inspired by the ancient past, is the first of its kind. The population comprises workers and the locations of the stones. Equations are derived to compute the displacement of stones and the amount of movement required by the workers to update the position of the solution. The number of stone blocks displaced is calculated by using the following equation:

$$d = \frac{v_0^2}{2g(\sin\theta + \mu_k \cos\theta)} \tag{58}$$

where v_0 is the initial velocity, g is the gravity constant, and μ_k is the coefficient of kinetic friction between the stone block and the ramp. The number of movements performed by the worker is computed as follows:

$$x = \frac{v_0^2}{2\ g \sin\theta} \tag{59}$$

Using the above equations, the position of a solution is updated using the following equation:

$$p = (p_i + d) \times x\epsilon_i \tag{60}$$

Moreover, GPC maps the concept of worker substitution by using a uniform crossover technique in which 50% of old solutions are substituted by randomly selected new solutions. In this paper, the role of different phases or parameters in relation to meeting optimization challenges is not explicitly discussed; however, it is stated that worker substitution balances exploration and exploitation.

3.3.3. Heap-based optimizer (HBO)

HBO (Askari et al., 2020a) is a human social behaviour-inspired algorithm that maps the concept of corporate rank hierarchy and interactions among individuals in the hierarchy. In this algorithm, the solutions are arranged in a 3-ary heap based on their fitness. The algorithm comprises three phases: interaction with the individual boss, interaction among colleagues, and self-contribution of an employee. Position updating equations are derived for all phases and are then combined into a single equation by using carefully designed probabilistic parameters, as formulated below:

$$x_i^k(t+1) = \begin{cases} x_i^k(t), & p \le p_1 \\ B^k + \gamma\ \lambda^k |B^k - x_i^k(t)|, & p > p_1\ \ and\ \ p \le p_2 \\ S_r^k + \gamma\ \lambda^k |S_r^k - x_i^k(t)|, & p > p_2\ and\ p \le p_3\ \ and\ \ f(\vec{S}_r) < f(\vec{x}_i(t)) \\ x_i^k + \gamma\ \lambda^k |S_r^k - x_i^k(t)|, & p > p_2\ and\ p \le p_3\ \ and\ \ f(\vec{S}_r) \ge f(\vec{x}_i(t)) \end{cases} \tag{61}$$

where x is the current solution, B is the immediate boss of x, S is the randomly chosen colleague of x, and γ and λ are computed using the following equations:

$$\gamma = \left|2 - \frac{(t\ mod\ \frac{T}{C})}{\frac{T}{4C}}\right| \tag{62}$$

$$\lambda^k = 2\ r - 1 \tag{63}$$

In the above equations, T denotes total number of iterations, r is a random number in the range [0, 1], and $C = \lfloor T/25 \rfloor$. The self-contribution phase incorporates exploration, interaction with colleagues promotes exploitation, and interaction with the immediate boss leads the population to convergence. Exploration and exploitation are balanced through the probabilistic parameters that give more priority to the self-contribution phase in early iterations and increase the rate of selection of the other two equations through iterations.

3.3.4. Political optimizer (PO)

PO (Askari et al., 2020b) is inspired by a multiparty political system. A political member is considered a candidate solution, and the prestige of the political member is treated as the fitness of the solution. The population is subdivided into political parties, and a political party comprises candidate solutions. The algorithm has several phases, such as party formation and constituency allocation, election campaigns, party switching, interparty elections, government formation, and parliamentary affairs. The main position updating mechanism of the algorithm is implemented in the election campaign phase by using the following equations:

$$p_{i,k}^j(t+1) = \begin{cases} m^* + r(m^* - p_{i,k}^j(t)), \\ \text{if } p_{i,k}^j(t-1) \le p_{i,k}^j(t) \le m^* \text{ or } p_{i,k}^j(t-1) \ge p_{i,k}^j(t) \ge m^* \\ \\ m^* + (2r-1)|m^* - p_{i,k}^j(t)|, \\ \text{if } p_{i,k}^j(t-1) \le m^* \le p_{i,k}^j(t) \text{ or } p_{i,k}^j(t-1) \ge m^* \ge p_{i,k}^j(t) \\ \\ m^* + (2r-1)|m^* - p_{i,k}^j(t-1)|, \\ \text{if } m^* \le p_{i,k}^j(t-1) \le p_{i,k}^j(t) \text{ or } m^* \ge p_{i,k}^j(t-1) \ge p_{i,k}^j(t) \end{cases} \tag{64}$$

$$p_{i,k}^j(t+1) = \begin{cases} m^* + (2r-1)|m^* - p_{i,k}^j(t)|, \\ \text{if } p_{i,k}^j(t-1) \le p_{i,k}^j(t) \le m^* \text{ or } p_{i,k}^j(t-1) \ge p_{i,k}^j(t) \ge m^* \\ \\ p_{i,k}^j(t-1) + r(p_{i,k}^j(t) - p_{i,k}^j(t-1)), \\ \text{if } p_{i,k}^j(t-1) \le m^* \le p_{i,k}^j(t) \text{ or } p_{i,k}^j(t-1) \ge m^* \ge p_{i,k}^j(t) \\ \\ m^* + (2r-1)|m^* - p_{i,k}^j(t-1)|, \\ \text{if } m^* \le p_{i,k}^j(t-1) \le p_{i,k}^j(t) \text{ or } m^* \ge p_{i,k}^j(t-1) \ge p_{i,k}^j(t) \end{cases} \tag{65}$$

where t denotes the current iteration, $t-1$ denotes the previous iteration, m can either be the party leader or constituency winner, and p_k^j denotes the kth dimension of the jth member of the ith political party. The former equation is used when the current fitness of the member is better than its previous fitness, while the latter equation is used to update the position. Two unique features of PO are a logical division of the population to assign a dual role to each candidate solution and utilization of recent positions of the candidates to update their positions. Position updating with respect to the area winner and party leader incorporates exploration and exploitation, the balance between exploration and exploitation is attained through party switching, and convergence is assured through parliamentary affairs.

3.3.5. Search and rescue optimization algorithm (SROA)

SROA (Shabani et al., 2020) is a human social interaction-based algorithm inspired by the exploratory behaviour of humans during a search and rescue operation. In this algorithm, an individual human is considered a candidate solution. The population of N humans and N clues is randomly initialized. A clue is also a candidate solution that points out the promising area to search for. The algorithm involves two main phases: the social phase and the individual phase. In the social phase, the position of a candidate solution (human) is updated with reference to a randomly selected clue. If the fitness of the clue is better, then the position is updated as follows:

$$X_{i,j}^{new} = C_{k,j} + r_i \times (X_{i,j} - C_{k,j}) \tag{66}$$

Otherwise, the position is updated by using the following equation:

$$X_{i,j}^{new} = X_{i,j} + r_i \times (X_{i,j} - C_{k,j}) \tag{67}$$

where r_i is a random number, $C_{k,j}$ is the randomly selected clue, and $X_{i,j}$ is the candidate solution. In the individual phase, the solution updates its position with reference to two randomly selected clues by using the following equation:

$$X_i^{new} = X_i + r3 \times (C_k - C_m) \tag{68}$$

where k and m are two random numbers in the range $[1, 2N]$. The individual phase mimics the idea of connecting the clues to search around the solution's own vicinity. There are a few other auxiliary phases, such as boundary controlling to push the solutions back to the feasible region, updating the solution and clue matrices based on the new position of the solution, abandoning clues to stop searching less significant areas, and restarting the strategy to regenerate matrices if the whole population is trapped in the local and infeasible region. The exploration, exploitation, and transition among them are controlled through two control parameters called SE (social effect) and MU (maximum unsuccessful search number).

3.4. Concept/process-based algorithms

3.4.1. Lévy flight distribution (LFD)

LFD (Houssein et al., 2020) is inspired by the Lévy flight random walk to explore wireless sensor networks (WSNs). In WSNs, the distance between two adjacent nodes is calculated to decide whether to move the nodes or keep them in their original positions. The goal is to cover all areas with minimum overlap. The position of each node is considered as a candidate solution, and all nodes to be deployed

Table 2
The characteristics of the unimodal benchmark functions used for exploitation analysis. $F1$–$F17$ are variable-dimensional functions, and $F18$–$F25$ are fixed-dimensional functions. *Range* defines the lower and upper bounds of the design variables. *Dim* denotes the dimensionality of the search space. F_{min} is the global optimum value.

Function	*Range*	*Dim*	F_{min}
$F1$ — Sphere	$[-100, 100]$	50	0
$F2$ — Quartic Noise	$[-1.28, 1.28]$	50	0
$F3$ — Powell Sum	$[-1, 1]$	50	0
$F4$ — Schwefel's 2.20	$[-100, 100]$	50	0
$F5$ — Schwefel's 2.21	$[-100, 100]$	50	0
$F6$ — Step	$[-100, 100]$	50	0
$F7$ — Stepint	$[-5.12, 5.12]$	50	$25 - 6n$
$F8$ — Schwefel's 1.20	$[-100, 100]$	50	0
$F9$ — Schwefel's 2.22	$[-100, 100]$	50	0
$F10$ — Schwefel's 2.23	$[-10, 10]$	50	0
$F11$ — Rosenbrock	$[-30, 30]$	50	0
$F12$ — Brown	$[-1, 4]$	50	0
$F13$ — Dixon and Price	$[-10, 10]$	50	0
$F14$ — Powell Singular	$[-4, 5]$	50	0
$F15$ — Zakharov	$[-5, 10]$	50	0
$F16$ — Xin-She Yang	$[-20, 20]$	50	-1
$F17$ — Perm 0,D,Beta	$[-d_i, d_i]$	5	0
$F18$ — Three-Hump Camel	$[-5, 5]$	2	0
$F19$ — Beale	$[-4.5, 4.5]$	2	0
$F20$ — Booth	$[-10, 10]$	2	0
$F21$ — Brent	$[-10, 10]$	2	0
$F22$ — Matyas	$[-10, 10]$	2	0
$F23$ — Schaffer N. 4	$[-100, 100]$	2	0.29257
$F24$ — Wayburn Seader 3	$[-500, 500]$	2	21.35
$F25$ — Leon	$[-1.2, 1.2]$	2	0

represent the population. In this algorithm, two adjacent nodes are chosen, and their Euclidean distance is computed. However, their positions are not updated if the Euclidean distance is less than a given threshold. In this case, one solution is either updated with respect to the leader (node with the fewest neighbours) using the Lévy flight function or regenerated randomly in the search space. Position updating with respect to the leader is formulated below:

$$X_j(t+1) = L_f(X_j(t), X_L, L_b, U_b) \tag{69}$$

where L_f represents the sl value and the LF direction. In contrast, the solution is regenerated randomly using the following equation:

$$X_j(t+1) = L_b + (U_b - L_b)rd() \tag{70}$$

The former promotes exploitation and later incorporates exploration. In contrast, the other node (solution) updates its position with reference to the leader and global best solution according to the Lévy flight function and parameters such as the total target fitness of the neighbour and the fitness degree of each neighbour.

3.4.2. Rain optimization algorithm (ROA)

ROA (Moazzeni and Khamehchi, 2020) is inspired by the natural behaviour of raindrops and the lowest land point they reach. Each raindrop is considered a candidate solution, and the algorithm randomly initializes a population of raindrops. Each raindrop is accompanied by a parameter called the radius, which determines the life of the raindrop. The radius is computed in two ways: (i) if raindrops are too close, then they merge and form a large droplet, which is formulated as follows:

$$R = (r_i^n + r_2^n)^{1/n} \tag{71}$$

(ii) if the droplet does not move, the radius is computed using the following equation:

$$R = (\alpha r_1^n)^{1/n} \tag{72}$$

The radius is used to update the position of a solution and becomes smaller with the number of iterations. The solution keeps updating its position by moving in the same direction as long as the fitness of the solution improves in each iteration. Moreover, in ROA, three other behaviours are mimicked: (i) merging raindrops in the same vicinity, (ii) removing raindrops with a radius less than a certain threshold and (iii) generating new raindrops. Diversity is maintained through the introduction of new raindrops, and moving in the same direction enhances exploitation.

4. Comparative setup and results on benchmarks and engineering problems

In this section, we investigated the capability of recent metaheuristics to meet optimization challenges such as exploration, exploitation, and convergence. As discussed in the previous section, to the best of our knowledge, approximately 57 novel metaheuristics were proposed and published through well-reputed journals and conferences. We thoroughly searched MathWorks, GitHub, ResearchGate, and the websites of authors to obtain the source code of these algorithms. Moreover, we contacted the corresponding authors for codes that are not publicly available. As a result, we collected the implementations of 26 metaheuristics, which we will analyse in this section.

4.1. Experimental setup

To evaluate and analyse the optimization capabilities of the algorithms, several experiments are performed. The nature of the experiments is presented below:

- For exploitation analysis, 25 fixed/variable-dimensional unimodal benchmark functions are used. The details of the benchmarks are presented in Table 2.
- For exploration analysis, 25 fixed/variable-dimensional multimodal benchmark functions are used. The details of the benchmarks are presented in Table 3.
- To investigate the capability of the metaheuristics to balance exploration and exploitation, 29 CEC-BC-2017 benchmarks with shifted, rotated, hybrid, and composite landscapes are used. The characteristics of the benchmarks are presented in Table 4.
- To assess the convergence assurance, convergence speed, and premature convergence avoidance capabilities, the convergence curves of the top five algorithms for selected unimodal and multimodal benchmarks are compared and analysed in Figs. 4 and 6, respectively.

Table 3
The characteristics of the multimodal benchmark functions used for exploration analysis. $F26$–$F40$ are variable-dimensional functions, and $F41$–$F50$ are fixed-dimensional functions. *Range* defines the lower and upper bounds of the design variables. *Dim* denotes the dimensionality of the search space. F_{min} is the global optimum value.

Function	*Range*	*Dim*	F_{min}
$F26$ — Schwefel's 2.26	$[-500, 500]$	50	0
$F27$ — Rastrigin	$[-5.12, 5.12]$	50	0
$F28$ — Periodic	$[-10, 10]$	50	0.9
$F29$ — Qing	$[-500, 500]$	50	0
$F30$ — Alpine N. 1	$[-10, 10]$	50	0
$F31$ — Xin-She Yang	$[-5, 5]$	50	0
$F32$ — Ackley	$[-32, 32]$	50	0
$F33$ — Trignometric 2	$[-500, 500]$	50	1
$F34$ — Salomon	$[-100, 100]$	50	0
$F35$ — Styblinski-Tang	$[-5, 5]$	50	$-39.16599 \times n$
$F36$ — Griewank	$[-100, 100]$	50	0
$F37$ — Xin-She Yang N.4	$[-10, 10]$	50	−1
$F38$ — Xin-She Yang N.2	$[-2\pi, 2\pi]$	50	0
$F39$ — Gen. Penalized	$[-50, 50]$	50	0
$F40$ — Penalized	$[-50, 50]$	50	0
$F41$ — Egg crate	[−5, 5]	2	0
$F42$ — Ackley N.3	$[-32, 32]$	2	−195.629
$F43$ — Adjiman	$[-1, 2]$	2	−2.02181
$F44$ — Bird	$[-2\pi, 2\pi]$	2	−106.7645
$F45$ — Camel Six Hump	$[-5, 5]$	2	−1.0316
$F46$ — Branin RCOS	$[-5, 10]$	2	0.3978873
$F47$ — Hartman 3	$[0, 1]$	3	−3.862782
$F48$ — Hartman 6	$[0, 1]$	6	−3.32237
$F49$ — Cross-in-tray	[−10, 10]	2	−2.06261218
$F50$ — Bartels Conn	$[-500, 500]$	2	1

Table 4
Characteristics of the benchmark functions from CEC-BC-2017 (Awad et al., 2017). *Range* defines the lower and upper bounds of the design variables. *Dim* denotes the dimensionality of the search space. F_{min} is the global optimum value.

Function	*Dim*	*Range*	F_{min}
Unimodal functions			
$F51$ — Shifted and Rotated Bent Cigar Function	10	[−100, 100]	100
Multimodal functions			
$F52$ — Shifted and Rotated Rosenbrock's Function	10	[−100, 100]	300
$F53$ — Shifted and Rotated Rastrigin's Function	10	[−100, 100]	400
$F54$ — Shifted and Rotated Expanded Scaffer's $F6$ Function	10	[−100, 100]	500
$F55$ — Shifted and Rotated Lunacek Bi-Rastrigin Function	10	[−100, 100]	600
$F56$ — Shifted and Rotated Non-Continuous Rastrigin's Function	10	[−100, 100]	700
$F57$ — Shifted and Rotated Levy Function	10	[−100, 100]	800
$F58$ — Shifted and Rotated Schwefel's Function	10	[−100, 100]	900
Hybrid functions (N is basic number of functions)			
$F59$ — Hybrid Function 1 (N = 3)	10	[−100, 100]	1000
$F60$ — Hybrid Function 2 (N = 3)	10	[−100, 100]	1100
$F61$ — Hybrid Function 3 (N = 3)	10	[−100, 100]	1200
$F62$ — Hybrid Function 4 (N = 4)	10	[−100, 100]	1300
$F63$ — Hybrid Function 5 (N = 4)	10	[−100, 100]	1400
$F64$ — Hybrid Function 6 (N = 4)	10	[−100, 100]	1500
$F65$ — Hybrid Function 6 (N = 5)	10	[−100, 100]	1600
$F66$ — Hybrid Function 6 (N = 5)	10	[−100, 100]	1700
$F67$ — Hybrid Function 6 (N = 5)	10	[−100, 100]	1800
$F68$ — Hybrid Function 6 (N = 6)	10	[−100, 100]	1900
Composite functions (N is basic number of functions)			
$F69$ — Composite Function 1 (N = 3)	10	[−100, 100]	2000
$F70$ — Composite Function 2 (N = 3)	10	[−100, 100]	2100
$F71$ — Composite Function 3 (N = 4)	10	[−100, 100]	2200
$F72$ — Composite Function 4 (N = 4)	10	[−100, 100]	2300
$F73$ — Composite Function 5 (N = 5)	10	[−100, 100]	2400
$F74$ — Composite Function 6 (N = 5)	10	[−100, 100]	2500
$F75$ — Composite Function 7 (N = 6)	10	[−100, 100]	2600
$F76$ — Composite Function 8 (N = 6)	10	[−100, 100]	2700
$F77$ — Composite Function 9 (N = 6)	10	[−100, 100]	2800
$F78$ — Composite Function 10 (N = 3)	10	[−100, 100]	2900
$F79$ — Composite Function 11 (N = 3)	10	[−100, 100]	3000

- To evaluate the applicability of the metaheuristics to complex real-world problems, four constrained engineering optimization problems are used.
- To rank the algorithms, a nonparametric statistical test called Friedman mean rank test is performed for each set of benchmarks/problems.

Table 5
Parameter settings used for each algorithm and the source code links. NFEs denotes the number of objective function evaluations.

Algorithms	Parameter settings	NFEs
HBO	Nill	40,000
Source link:	https://github.com/qamar-askari/HBO	
PO	$\lambda = 1$	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/74577-political-optimizer-po	
GPC	g = 9.8, Minimum friction = 1, Maximum friction = 10, Substitution probability = 0.5	40,000
Source link:	http://www.harifi.com	
GBO	$\beta_{min} = 0.2, \beta_{max} = 1.2, pr = 0.5$	40,000
Source link:	http://imanahmadianfar.com/codes/	
RDA	$\alpha = 0.9, \beta = 0.4, \gamma = 0.7, Ncom = 15$	40,000
Source link:	https://www.researchgate.net/profile/Amir-Fathollahi-Fard	
MPA	$FADs = 0.2, P = 0.5$	40,000
Source link:	https://github.com/afshinfaramarzi/Marine-Predators-Algorithm	
LFD	Threshold $= 2, CSV = 0.5, \alpha 1 = 10, \beta = 1.5, \alpha 2 = 0.00005, \alpha 3 = 0.005, \gamma 1 = 0.9, \gamma 2 = 0.1$	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/76063-levy-flight-distribution-lfd	
TSA	PmiN = 1, Pmax = 4	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/75182-tunicate-swarm-algorithm-tsa	
BWOA	$pc = 0.8, pm = 0.4, pCannibalism = 0.5$	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/94080-black-widow-optimization-algorithm	
ROA	rain speed = 10, rain radius = 0.05(Xmax − Xmin), soil adsorptioN = 50%	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/65617-rain-water-algorithm	
ChOA	Nill	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/76763-chimp-optimization-algorithm	
TSO	$k = 2$, z in [0, 2]	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/82640-transient-search-algorithm-tso	
EO	$a1 = 2, a2 = 1, gp = 0.5$	40,000
Source link:	https://github.com/afshinfaramarzi/Equilibrium-Optimizer	
SMA	$z = 0.03$	40,000
Source link:	http://www.alimirjalili.com/SMA.html	
TFWO	nWh = 3	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/75868-turbulent-flow-of-water-based	
SROA	$\theta = 0.5, cp = 0.3$	40,000
Source link:	https://www.researchgate.net/profile/Amir-Shabani-13	
BMO	$pl = 7$	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/74730-barnacles-mating-optimizer-bmo	
MRFO	Nill	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/73130-manta-ray-foraging-optimization-mrfo	
STSA	The range of ratioFEs is [0, 1], the range of k is [0, 2]	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/79802-stsa-algorithm-code	
AEFA	$k0 = 500, \alpha = 30$	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/71218-aefa-artificial-electric-field-algorithm	
MA	$a1 = 1, a2 = 1.5, \beta = 2, d = 0.1, fl = 0.1$, single point uniform crossover with the rate of 0.95	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/76902-a-mayfly-optimization-algorithm	
AOA	$C1 = 2, C2 = 6$ (standard), $C3 = 2, C4 = 0.5$ (cec or engineering)	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/79822-archimedes-optimization-algorithm	
GNDO	Nill	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/79526-the-source-code-for-gndo	
PPA	$r1 = 1, r2 = 0.1, r3 = 0.1, \alpha 1 = 0.2, \alpha 2 = 0.25, \beta 1 = 0.1, \beta 2 = 0.1, c1 = 0.1, c2 = 0.1, d1 = 0.01, d2 = 0.01$	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/74032-parasitism-predation-algorithm-ppa	
FBI	Nill	40,000
Source link:	https://www.mathworks.com/matlabcentral/fileexchange/76299-forensic-based-investigation	
BCMO	Nill	40,000
Source link:	https://github.com/ThangLe-duc/BCMO-Package	

All the algorithms are executed using MATLAB, and simulations are run on an Intel Core i7-5600U with 16 GB RAM. To achieve the best results, the parameters used for each algorithm are the values recommended by the authors in their corresponding papers. The parameter settings for each algorithm can be seen in Table 5. For a fair comparison, the population size for all algorithms is fixed at 40, and the maximum number of objective function evaluations (NFEs) is fixed at 40,000. Moreover, to ensure consistency and rule out the chance of accidentally obtaining better results, each algorithm is independently run 25 times for each benchmark function, and the mean of the obtained results along with the standard deviation are reported.

4.2. Comparison of exploitation capabilities

As previously stated, the investigation of the promising regions to reach the best position (optimum) of that region is called exploitation. Unimodal functions are considered the best way to evaluate the exploitation capability of algorithms because they do not have any local optimum. In this paper, 25 unimodal functions, including 16 variable-dimensional and nine fixed-dimensional functions, are used. The details of the functions are given in Table 2. The comparative results of the algorithms for unimodal functions are presented in Tables 6 and 7. The results are presented in two tables to improve the readability. The results include the average, standard deviation, and ranks of the algorithms against each function. However, to provide final rankings, their worst relative performances should also be considered, which is completed through a nonparametric statistical test called the Friedman mean rank test. The result of the Friedman test is presented in Fig. 3. It should be noted that a smaller bar or lower value on the bar indicates a better rank. Furthermore, the convergence curves of the top 5 algorithms, which are obtained through the Friedman mean rank test, for 16 selected unimodal benchmark functions are presented in Fig. 4.

4.3. Comparison of exploration capabilities

The goal of exploration is to find promising regions in the search space. Multimodal functions with complex landscapes have many local

Table 6
Comparison of the performance of HBO, PO, GPC, GBO, RDA, MPA, LFD, TSA, BWOA, ROA, ChOA, TSO, and EO for unimodal functions.

Fn	Stats	HBO	PO	GPC	GBO	RDA	MPA	LFD	TSA	BWOA	ROA	ChOA	TSO	EO
*F*1	Avg	8.8E−11	0	3.9E−22	9E−262	8.69979	1.1E−45	3.2E−07	5.3E−38	1.8E−07	94385.7	6.5E−11	2E−203	6.4E−80
	Std	8.5E−11	0	6.2E−22	0	3.18206	4.8E−45	7.7E−08	9.7E−38	8.6E−07	5960.94	2.2E−10	0	2E−79
	Rank	**16**	**1**	**14**	**6**	**24**	**12**	**18**	**13**	**17**	**26**	**15**	**9**	**10**
*F*2	Avg	0.03693	0.00022	1.9E−05	0.00029	0.26753	0.0006	2.80927	0.00741	0.0218	195.512	0.00094	5.9E−05	0.00063
	Std	0.00729	0.00021	1.8E−05	0.0002	0.07306	0.00032	1.38516	0.00316	0.03235	37.7612	0.00101	5.8E−05	0.00034
	Rank	**18**	**8**	**2**	**9**	**21**	**11**	**25**	**15**	**17**	**26**	**13**	**3**	**12**
*F*3	Avg	8.3E−34	0	5.1E−39	0	2.7E−06	4E−117	0.00057	4E−147	1.2E−12	0.12545	2.7E−32	3E−219	3E−298
	Std	4.1E−33	0	2.5E−38	0	3.9E−06	2E−116	0.00144	1E−146	6.1E−12	0.03496	1.3E−31	0	0
	Rank	**15**	**1**	**14**	**1**	**24**	**12**	**25**	**11**	**20**	**26**	**16**	**10**	**9**
*F*4	Avg	2.6E−07	0	2E−11	3E−132	3.33825	4.3E−25	0.00414	1.8E−22	0.00082	1748.9	2.3E−07	7E−104	1.4E−45
	Std	3.8E−07	0	1.6E−11	2E−131	1.25212	6.6E−25	0.00077	4.3E−22	0.00397	60.723	5.7E−07	2E−103	1.2E−45
	Rank	**16**	**1**	**14**	**7**	**22**	**12**	**19**	**13**	**17**	**26**	**15**	**9**	**10**
*F*5	Avg	19.3278	2E−281	1.2E−11	2E−120	52.7032	1.8E−17	0.00035	1.57708	21.4768	83.9443	0.12531	2E−104	8.3E−17
	Std	3.58858	0	1.1E−11	8E−120	6.24591	1.1E−17	4E−05	1.04552	6.14554	1.35114	0.21931	9E−104	1.7E−16
	Rank	**20**	**2**	**14**	**6**	**24**	**12**	**15**	**17**	**21**	**26**	**16**	**8**	**13**
*F*6	Avg	9.4E−11	0	8.96762	2.2E−07	10.5597	5E−08	0.33322	5.82439	3.5E−08	90766.7	6.81108	0.00061	1.3E−06
	Std	1E−10	0	0.30019	5.1E−07	6.13958	1.7E−08	0.12543	0.85172	1.5E−07	3533.2	0.43011	0.00086	1.3E−06
	Rank	**2**	**1**	**22**	**5**	**24**	**4**	**15**	**19**	**3**	**26**	**20**	**10**	**6**
*F*7	Avg	−275	−275	−63.28	−275	−224.52	−275	−190.52	−122.24	−149.6	−1932	−275	−275	−274.76
	Std	0	0	5.72655	0	4.86587	0	19.348	9.54935	16.9263	4086.37	0	0	0.52281
	Rank	**4**	**4**	**26**	**4**	**18**	**4**	**22**	**25**	**24**	**3**	**4**	**4**	**15**
*F*8	Avg	78617	0	1.2E−20	2E−216	63271.3	1.4E−08	2.7E−06	0.01703	1927.9	139987	101.852	8.4E−87	7.3E−11
	Std	13048.2	0	1.5E−20	0	8315.32	4.9E−08	7.3E−07	0.08151	665.145	19475.2	377.611	3.1E−86	2.7E−10
	Rank	**25**	**1**	**11**	**5**	**24**	**13**	**14**	**15**	**19**	**26**	**17**	**9**	**12**
*F*9	Avg	3E−07	0	2.7E−11	6E−132	2.2E+30	4.9E−25	0.00453	8E−23	2E−06	3.9E+58	1E−07	2.9E−93	3.1E−45
	Std	8.1E−07	0	2.8E−11	3E−131	1.1E+31	5.8E−25	0.00081	1.1E−22	4E−06	1.2E+59	2.1E−07	1.4E−92	4.6E−45
	Rank	**16**	**1**	**14**	**7**	**25**	**12**	**18**	**13**	**17**	**26**	**15**	**9**	**10**
*F*10	Avg	1E−17	0	2E−115	0	456660	1E−193	1E−44	3.3E−82	3.6E−21	6.4E+09	1.3E−25	0	6E−237
	Std	2.1E−17	0	1E−114	0	699110	0	1.8E−44	1.4E−81	1.8E−20	1.7E+09	3.3E−25	0	0
	Rank	**19**	**1**	**13**	**1**	**25**	**12**	**15**	**14**	**17**	**26**	**16**	**1**	**10**
*F*11	Avg	113.832	0	47.5457	39.7552	22725.4	43.8679	47.4354	48.2961	4213.05	3E+08	48.8069	0.01616	44.1268
	Std	48.3232	0	0.34859	2.34497	14963.9	0.45492	0.12113	0.8359	7778.33	3.1E+07	0.24011	0.01744	0.23069
	Rank	**18**	**1**	**13**	**5**	**24**	**7**	**12**	**15**	**23**	**26**	**16**	**2**	**8**
*F*12	Avg	6.9E−13	0	3.2E−25	2E−263	0.02889	1.1E−48	14.2723	9.2E−41	18.1527	104448	6.1E−14	2E−186	6.4E−83
	Std	1.6E−12	0	6.9E−25	0	0.0143	2E−48	9.55586	1.5E−40	9.21748	216182	2.1E−13	0	2E−82
	Rank	**15**	**1**	**13**	**6**	**19**	**11**	**23**	**12**	**24**	**26**	**14**	**9**	**10**
*F*13	Avg	1.78554	0.03973	0.66707	0.66667	254.29	0.66667	0.66756	0.8	96.0046	3256620	0.89336	0.28071	0.66667
	Std	1.76948	0.0422	0.00052	2.8E−07	126.761	5.3E−09	0.00108	0.16666	159.638	384850	0.15866	0.11727	9.8E−12
	Rank	**18**	**1**	**12**	**10**	**24**	**9**	**13**	**16**	**23**	**26**	**17**	**2**	**5**
*F*14	Avg	0.0999	0	3.9E−24	3E−250	297.228	8.7E−19	28.6491	0.00027	5.437	15682.2	1.1E−05	2E−201	5.7E−16
	Std	0.06422	0	6.2E−24	0	183.713	4.3E−18	41.5183	0.0004	4.30321	2087.82	2.7E−05	0	2.6E−15
	Rank	**16**	**1**	**10**	**5**	**25**	**11**	**22**	**14**	**21**	**26**	**13**	**8**	**12**
*F*15	Avg	387.014	0	1.5E−22	2E−191	487.627	3.8E−05	1.4E−06	2.1E−14	110.378	9.5E+07	22.7963	3.2E−43	5.1E−07
	Std	63.2533	0	1.9E−22	0	44.9877	2.7E−05	4E−06	4.8E−14	40.5353	4.3E+08	23.7844	1.2E−42	1.2E−06
	Rank	**22**	**1**	**10**	**5**	**24**	**14**	**13**	**11**	**19**	**26**	**17**	**9**	**12**
*F*16	Avg	0	−0.32	0	−0.1998	0	−0.1591	−0.04	2E−144	7E−240	5.3E−85	0	−0.48	0
	Std	0	0.4761	0	0.40784	0	0.37207	0.2	9E−144	0	2.6E−84	0	0.5099	0
	Rank	**11**	**7**	**11**	**8**	**11**	**9**	**10**	**23**	**21**	**26**	**11**	**6**	**11**
*F*17	Avg	1.88145	5.8874	3588.26	0.0194	0.4798	0.04488	27.9816	288.131	117.743	146.008	3037.82	339.506	0.21132
	Std	2.19508	12.6634	4002.02	0.02799	0.96281	0.089	48.3775	906.558	361.171	135.063	1970.91	489.423	0.31437
	Rank	**9**	**16**	**26**	**1**	**8**	**2**	**17**	**23**	**20**	**21**	**25**	**24**	**6**
*F*18	Avg	6E−197	0	1E−36	0	2.6E−29	3E−147	2.6E−12	2E−17	8.5E−94	0.00858	0	6E−205	0
	Std	0	0	4.3E−36	0	1.2E−28	1E−146	3.4E−12	1E−16	4.3E−93	0.00917	0	0	0
	Rank	**14**	**1**	**21**	**1**	**22**	**16**	**25**	**24**	**19**	**26**	**1**	**13**	**1**
*F*19	Avg	0	2.9E−27	0.32859	0	9.9E−27	0	7.2E−07	0.21338	0.08286	0.00235	0.03052	2.9E−08	0
	Std	0	1.4E−26	0.35597	0	4.2E−26	0	1.6E−06	0.34922	0.13137	0.00224	0.15241	4.4E−08	0
	Rank	**1**	**13**	**26**	**1**	**14**	**1**	**17**	**25**	**24**	**21**	**23**	**16**	**1**
*F*20	Avg	0	0	0.00014	0	1.6E−26	0	2.8E−13	0.288	0.00014	0.04058	0.00012	8.4E−05	0
	Std	0	0	0.00013	0	7.7E−26	0	8.3E−13	0.6735	0.00071	0.03842	0.00012	0.00011	0
	Rank	**1**	**1**	**23**	**1**	**15**	**1**	**16**	**26**	**24**	**25**	**22**	**21**	**1**
*F*21	Avg	1.4E−87	1.4E−87	1.4E−87	1.4E−87	1.4E−87	1.4E−87	1.4E−87	1.4E−87	2.22354	0.11435	1.4E−87	1.4E−87	1.4E−87
	Std	5E−103	5E−103	5E−103	5E−103	5E−103	5E−103	5E−103	5E−103	2.44935	0.08722	5E−103	5E−103	5E−103
	Rank	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**26**	**25**	**2**	**2**	**2**

(*continued on next page*)

Table 6 (*continued*).

Fn	Stats	HBO	PO	GPC	GBO	RDA	MPA	LFD	TSA	BWOA	ROA	ChOA	TSO	EO
*F*22	Avg	1.1E−70	0	3.1E−37	0	2.9E−29	2E−117	1.6E−15	1E−168	6E−05	0.00228	5E−152	3E−208	8E−301
	Std	5.3E−70	0	1.5E−36	0	1.4E−28	1E−116	1.8E−15	0	0.00026	0.00237	2E−151	0	0
	Rank	**19**	**1**	**21**	**1**	**22**	**16**	**24**	**13**	**25**	**26**	**14**	**12**	**9**
*F*23	Avg	0.29258	0.29425	0.29258	0.29258	0.29258	0.29258	0.29327	0.29258	0.2934	0.29699	0.29258	0.29259	0.29258
	Std	7.3E−17	0.00832	7.3E−07	1E−16	1.4E−16	1.3E−16	0.00171	2.8E−08	0.00264	0.00372	3E−07	1.8E−05	9E−17
	Rank	**2**	**24**	**16**	**2**	**7**	**9**	**22**	**13**	**23**	**25**	**15**	**17**	**2**
*F*24	Avg	19.1059	19.1059	19.1284	19.1059	19.1059	19.1059	19.1059	64.027	19.8577	3320.06	19.1095	19.1367	19.1059
	Std	1.1E−14	7.6E−15	0.0242	8.5E−15	7.9E−15	7.2E−15	2E−10	103.716	2.78949	4731.25	0.00275	0.03722	9.8E−15
	Rank	**2**	**8**	**22**	**3**	**8**	**8**	**17**	**25**	**24**	**26**	**21**	**23**	**3**
*F*25	Avg	3.9E−11	0	0.39206	0	4.2E−16	0	0.0003	0.09006	0.09494	0.0012	0.00022	3.9E−07	2.5E−17
	Std	1.5E−10	0	0.1993	0	1.4E−15	0	0.00021	0.21059	0.08809	0.00171	0.00022	1E−06	8.2E−17
	Rank	**14**	**1**	**26**	**1**	**12**	**1**	**19**	**24**	**25**	**20**	**18**	**16**	**10**

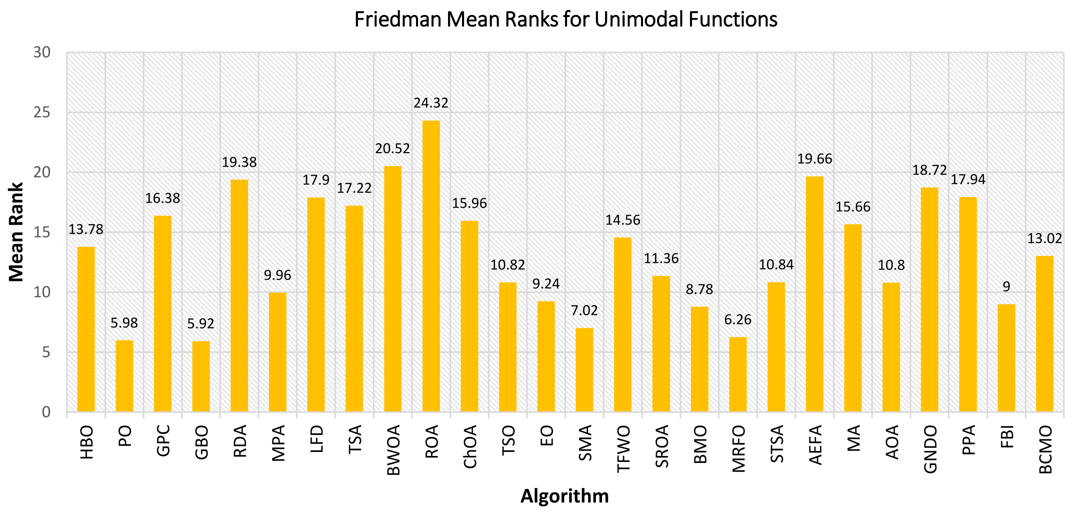

Fig. 3. Friedman mean ranks of all algorithms for unimodal benchmark functions.

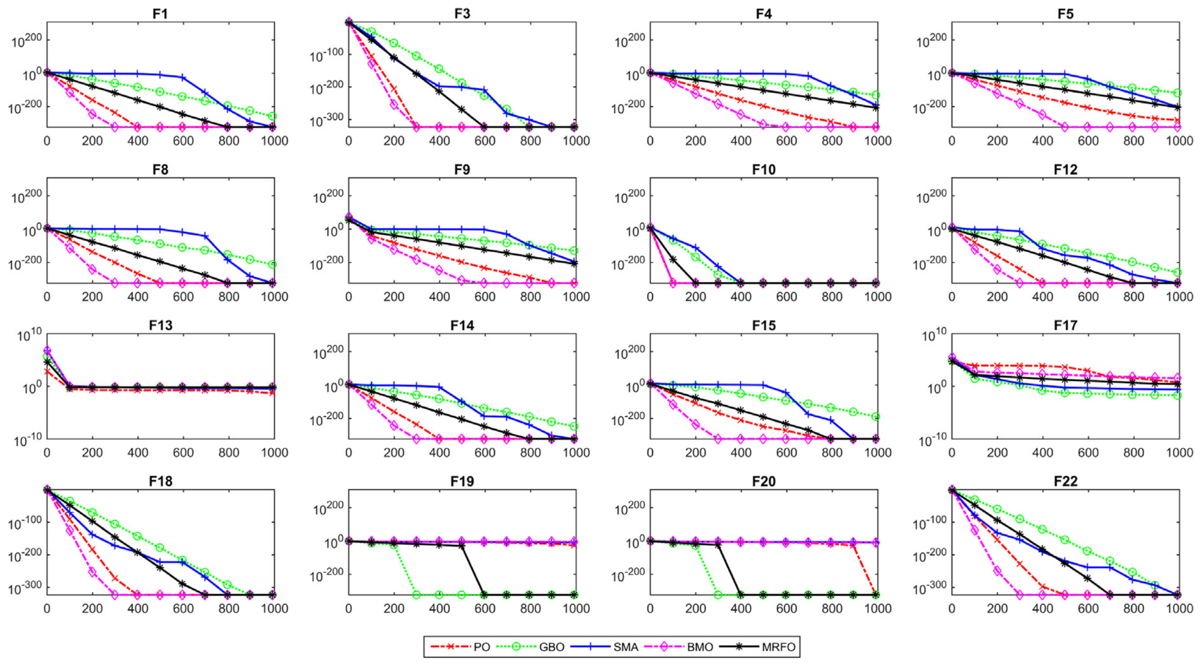

Fig. 4. Comparison of the convergence curves of the top five algorithms for selected unimodal functions.

Table 7
Comparison of the performance of SMA, TFWO, SROA, BMO, MRFO, STSA, AEFA, MA, AOA, GNDO, PPA, FBI, and BCMO for unimodal functions.

Fn	Stats	SMA	TFWO	SROA	BMO	MRFO	STSA	AEFA	MA	AOA	GNDO	PPA	FBI	BCMO
*F*1	Avg	0	1E−05	0.00218	0	0	3E−260	0.79194	0.0001	0	668.715	7.45428	4E−231	6.5E−55
	Std	0	1.9E−05	0.0013	0	0	0	1.95264	0.00011	0	501.24	5.58311	0	3.2E−54
	Rank	**1**	**19**	**21**	**1**	**1**	**7**	**22**	**20**	**1**	**25**	**23**	**8**	**11**
*F*2	Avg	0.00011	0.17171	0.00773	1.5E−05	0.00011	9.2E−05	0.55601	0.06792	0.00042	0.75479	0.66619	8.1E−05	0.00713
	Std	8.1E−05	0.05726	0.00566	1.7E−05	9.1E−05	4E−05	0.55206	0.02419	0.00026	0.34655	0.2229	6.9E−05	0.00477
	Rank	**7**	**20**	**16**	**1**	**6**	**5**	**22**	**19**	**10**	**24**	**23**	**4**	**14**
*F*3	Avg	0	3.1E−25	4.3E−25	0	0	0	6.2E−12	1.9E−21	0	2.4E−11	9.2E−11	0	8E−117
	Std	0	1.3E−24	1.4E−24	0	0	0	1.1E−11	9.2E−21	0	7.6E−11	1.6E−10	0	4E−116
	Rank	**1**	**17**	**18**	**1**	**1**	**1**	**21**	**19**	**1**	**22**	**23**	**1**	**13**
*F*4	Avg	5E−197	0.00107	0.15444	0	7E−210	1E−141	3.70473	0.11155	2E−164	95.1737	67.9468	1E−117	5E−26
	Std	0	0.00116	0.04609	0	0	1E−141	4.03607	0.28301	0	30.0356	27.3697	2E−117	2.5E−25
	Rank	**4**	**18**	**21**	**1**	**3**	**6**	**23**	**20**	**5**	**25**	**24**	**8**	**11**
*F*5	Avg	1E−201	36.9628	2.2E−18	0	7E−207	5.1E−92	5.38132	69.4154	3E−164	18.8861	42.6323	3E−108	7.9E−24
	Std	0	3.84653	8.1E−18	0	0	8.1E−92	1.72107	29.8765	0	2.09445	7.23238	3E−108	2.8E−23
	Rank	**4**	**22**	**11**	**1**	**3**	**9**	**18**	**25**	**5**	**19**	**23**	**7**	**10**
*F*6	Avg	0.00772	2.5E−05	0.00251	1.71412	7.7E−06	4.22112	2.41027	6.5E−05	10.169	636.734	7.56805	0.04535	0.26736
	Std	0.00536	6.3E−05	0.00129	0.44877	2.1E−05	0.24728	8.47957	3.1E−05	0.44454	337.691	4.83263	0.02456	0.17239
	Rank	**12**	**8**	**11**	**16**	**7**	**18**	**17**	**9**	**23**	**25**	**21**	**13**	**14**
*F*7	Avg	−275	−275	−1E+34	−275	−203.4	−178.84	−275	−275	−5985.3	−257.8	−210.12	−197.12	−272.04
	Std	0	0	4.7E+34	0	8.4113	3.97576	0	0	5067.93	13.1022	11.0277	2.83314	1.7673
	Rank	**4**	**4**	**1**	**4**	**20**	**23**	**4**	**4**	**2**	**17**	**19**	**21**	**16**
*F*8	Avg	0	788.222	70.0899	0	0	3E−106	3872.12	10644.5	1E−215	2474.85	2240.26	3E−184	1.4E−41
	Std	0	368.674	41.1135	0	0	9E−106	1042.18	2600.35	0	878.192	891.844	0	4.3E−41
	Rank	**1**	**18**	**16**	**1**	**1**	**8**	**22**	**23**	**6**	**21**	**20**	**7**	**10**
*F*9	Avg	2E−205	0.00603	0.2262	0	5E−210	8E−142	275.741	2.06611	4E−167	150.252	667.423	1E−117	2.3E−30
	Std	0	0.00494	0.06252	0	0	1E−141	45.836	3.5892	0	41.9746	153.314	2E−117	9.7E−30
	Rank	**4**	**19**	**20**	**1**	**3**	**6**	**23**	**21**	**5**	**22**	**24**	**8**	**11**
*F*10	Avg	0	8.4E−09	1.9E−18	0	0	0	1.17275	3.6E−15	0	40.7459	1.81308	0	1E−219
	Std	0	4.1E−08	6.3E−18	0	0	0	3.57092	7.8E−15	0	88.0868	2.80923	0	0
	Rank	**1**	**21**	**18**	**1**	**1**	**1**	**22**	**20**	**1**	**24**	**23**	**1**	**11**
*F*11	Avg	1.30367	201.593	25.3618	47.3877	43.1405	46.6645	955.411	189.818	48.8081	55320.2	1535.1	46.4251	47.9325
	Std	1.18767	153.101	18.7331	0.37527	0.46971	0.13928	785.987	220.148	0.09194	49346	1325.07	0.3413	0.21773
	Rank	**3**	**20**	**4**	**11**	**6**	**10**	**21**	**19**	**17**	**25**	**22**	**9**	**14**
*F*12	Avg	0	9.9E−08	2.8E−06	0	0	5E−263	0.08747	3.6E−05	0	36.731	0.57552	1E−233	0.15488
	Std	0	2E−07	1.6E−06	0	0	0	0.37429	1.7E−05	0	11.54	0.47479	0	0.07142
	Rank	**1**	**16**	**17**	**1**	**1**	**7**	**20**	**18**	**1**	**25**	**22**	**8**	**21**
*F*13	Avg	0.34168	7.27026	0.5525	0.66676	0.66667	0.66667	59.985	3.56058	0.7695	737.757	40.3642	0.66667	0.66853
	Std	0.18596	3.94809	0.18315	8.4E−05	9.7E−10	3.2E−11	63.61	3.55079	0.09944	853.032	25.7208	7.1E−09	0.00233
	Rank	**3**	**20**	**4**	**11**	**7**	**6**	**22**	**19**	**15**	**25**	**21**	**8**	**14**
*F*14	Avg	0	0.12961	0.0006	0	0	1E−213	91.2107	0.44567	6E−138	30.5828	3.39095	2E−228	1.35208
	Std	0	0.07059	0.00024	0	0	0	36.9807	0.40637	3E−137	31.2964	2.87872	0	0.41071
	Rank	**1**	**17**	**15**	**1**	**1**	**7**	**24**	**18**	**9**	**23**	**20**	**6**	**19**
*F*15	Avg	0	127.483	0.77826	0	0	1.6E−53	392.658	733.42	3.6E−46	12.1721	72.3116	1E−98	161.478
	Std	0	79.455	0.49452	0	0	7.9E−53	73.7516	118.762	1.8E−45	17.243	28.1132	5.1E−98	37.5916
	Rank	**1**	**20**	**15**	**1**	**1**	**7**	**23**	**25**	**8**	**16**	**18**	**6**	**21**
*F*16	Avg	−1	0	−0.7472	−1	−0.96	1E−288	3E−105	0	0	0	1E−107	1E−150	−0.9741
	Std	5.3E−16	0	0.36239	0	0.2	0	1E−104	0	0	0	6E−107	5E−150	0.06452
	Rank	**2**	**11**	**5**	**1**	**4**	**20**	**25**	**11**	**11**	**11**	**24**	**22**	**3**
*F*17	Avg	0.25399	0.05619	2.51238	36.7442	2.67916	3.58042	180.586	0.04962	75.1912	3.44683	0.12114	3.15732	4.17815
	Std	0.40741	0.09921	2.22408	99.9802	3.91765	4.38803	227.471	0.1035	59.2343	9.82014	0.26269	4.68471	10.6433
	Rank	**7**	**4**	**10**	**18**	**11**	**14**	**22**	**3**	**19**	**13**	**5**	**12**	**15**
*F*18	Avg	0	1.2E−86	4E−101	0	0	0	8.2E−28	4E−246	0	1E−288	2E−106	0	3E−155
	Std	0	5.9E−86	2E−100	0	0	0	9E−28	0	0	0	7E−106	0	1E−154
	Rank	**1**	**20**	**18**	**1**	**1**	**1**	**23**	**12**	**1**	**11**	**17**	**1**	**15**
*F*19	Avg	7.6E−10	1.1E−33	0	0.00016	0	7.8E−06	2E−27	0.03048	0.00126	2.8E−32	0	0	5.1E−33
	Std	1.5E−09	5.5E−33	0	0.00078	0	1.7E−05	2E−27	0.15241	0.00232	0	0	0	2.5E−32
	Rank	**15**	**9**	**1**	**19**	**1**	**18**	**12**	**22**	**20**	**11**	**1**	**1**	**10**
*F*20	Avg	5.1E−10	0	0	1.9E−10	0	4.4E−06	2.3E−27	0	2.7E−06	7.9E−31	0	0	0
	Std	1.4E−09	0	0	9.6E−10	0	4.9E−06	1.8E−27	0	1.1E−05	0	0	0	0
	Rank	**18**	**1**	**1**	**17**	**1**	**20**	**14**	**1**	**19**	**13**	**1**	**1**	**1**
*F*21	Avg	1.4E−87	1.4E−87	0	0.05712	1.4E−87	1.4E−87	1.4E−87	1.4E−87	1.4E−87	7.3E−17	1.4E−87	3E−23	1.4E−87
	Std	5E−103	5E−103	0	0.28562	5E−103	5E−103	5E−103	5E−103	5E−103	3.6E−16	5E−103	6.8E−23	5E−103
	Rank	**2**	**2**	**1**	**24**	**2**	**2**	**2**	**2**	**2**	**23**	**2**	**22**	**2**

(*continued on next page*)

Table 7 (*continued*).

Fn	Stats	SMA	TFWO	SROA	BMO	MRFO	STSA	AEFA	MA	AOA	GNDO	PPA	FBI	BCMO
*F*22	Avg	0	1.5E−78	3.9E−53	0	0	0	8.3E−29	8E−257	0	2E−236	2E−105	0	1E−133
	Std	0	7.3E−78	9.3E−53	0	0	0	8.6E−29	0	0	0	1E−104	0	4E−133
	Rank	**1**	**18**	**20**	**1**	**1**	**1**	**23**	**10**	**1**	**11**	**17**	**1**	**15**
*F*23	Avg	0.29258	0.29258	0	0.29261	0.29258	0.29258	0.32049	0.29263	0.29258	0.29258	0.29263	0.29258	0.29264
	Std	8.2E−15	9.3E−17	0	7.6E−05	1.2E−16	3.7E−10	0.0204	0.00026	2.3E−07	1.2E−11	0.00026	8.2E−17	0.00016
	Rank	**10**	**2**	**1**	**18**	**7**	**12**	**26**	**19**	**14**	**11**	**19**	**2**	**21**
*F*24	Avg	19.1059	19.1059	0	19.1059	19.1059	19.1059	19.1059	19.1059	19.1059	19.1059	19.1059	19.1059	19.1059
	Std	5.8E−09	1E−14	0	5.9E−15	7E−15	3.6E−05	9.9E−15	1E−14	1.1E−06	1.1E−14	8E−15	7.1E−15	7.1E−15
	Rank	**18**	**3**	**1**	**14**	**8**	**20**	**3**	**3**	**19**	**16**	**14**	**8**	**8**
*F*25	Avg	4.9E−08	8.3E−32	3.4E−29	0.05918	9.4E−12	9.1E−05	0.01758	0	0.0325	1.2E−32	0	2E−16	1.3E−30
	Std	9.6E−08	4.2E−31	1.7E−28	0.05409	3.6E−11	0.00015	0.02409	0	0.03487	0	0	6E−16	4.3E−30
	Rank	**15**	**7**	**9**	**23**	**13**	**17**	**21**	**1**	**22**	**6**	**1**	**11**	**8**

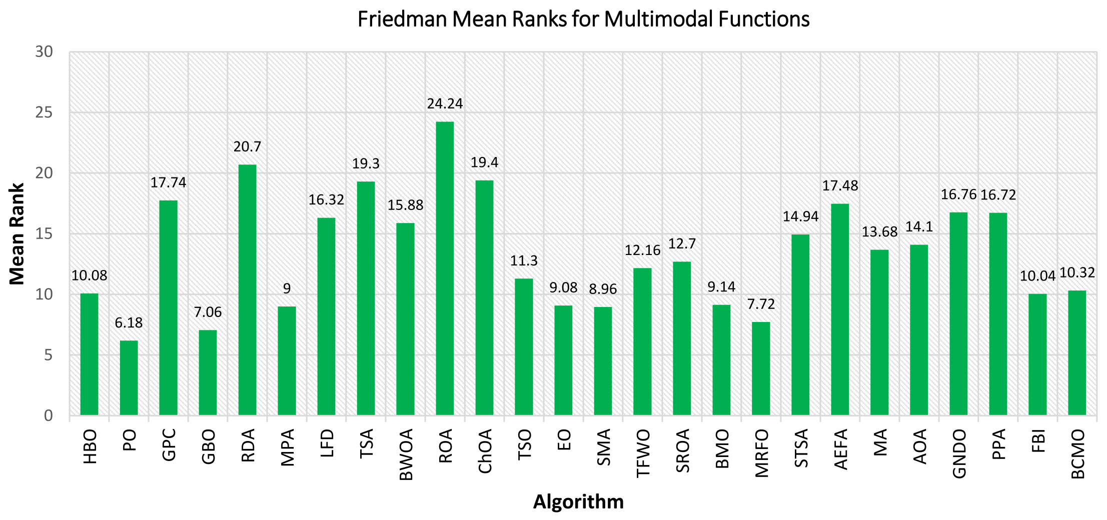

Fig. 5. Friedman mean ranks of all algorithms for multimodal benchmark functions.

optima and can be used to evaluate an algorithm's exploration capability because an algorithm may locate the most promising region if it can explore the search space well. To evaluate the exploration capability of algorithms, 25 multimodal functions, including 15 variable-dimensional and ten fixed-dimensional benchmark functions, are used. The names and characteristics of the multimodal functions are given in Table 3. The comparative results are presented in Tables 8 and 9. In addition, to statistically rank the algorithms, the Friedman mean rank test is performed, and the result is presented in Fig. 5. The convergence curves of the top 5 algorithms (according to the Friedman mean rank test) for eight selected multimodal benchmarks are presented in Fig. 6.

4.3.1. *Comparison of solving complex and challenging landscapes*

CEC-BC-2017 (Awad et al., 2017) is a collection of 29 challenging landscapes, including shifted, rotated, composite, and hybrid benchmarks. These artificial landscapes may help to further investigate the optimization capabilities in terms of balancing exploration and exploitation. The specifications of CEC-BC-2017 functions are presented in Table 4, and the results of all algorithms for these functions are shown in Tables 10 and 11. Moreover, to statistically rank these algorithms, the Friedman test is performed, and the output of the test is plotted in Fig. 7.

4.4. *Comparison of solving constrained engineering optimization problems*

Engineering optimization problems belong to the branch of constrained optimization. They have extensively been utilized as benchmarks in the literature to evaluate the applicability of algorithms to real-world optimization problems. In this section, the performance of all algorithms is compared by using four well-known engineering problems: the speed-reducer design problem, tension/compression spring design problem, rolling-element bearing design problem, and multiple-disk clutch brake design problem. The schematic views of these problems are given in Fig. 8. Since these problems belong to constrained optimization, the algorithms need some mechanism to satisfy the constraints. In following the literature on handling constraints, we used the death penalty method, in which the solution is penalized with a high cost for violating any constraint. For a fair comparison, all algorithms evaluate the objective function an equal number of times. The NFEs are fixed for all algorithms at 10,000 for the speed-reducer problem, 16,000 for the tension/compression problem, 8000 for the rolling-element bearing design problem, and 1000 for the multiple disk-clutch brake design problem. The results for engineering problems are compared in Table 12. Moreover, the Friedman mean rank test is performed to generate overall ranks, and the results are plotted in Fig. 7.

5. Discussion of results and evaluation of compared algorithms

In this section, based on the comparative results presented in the previous section, the performance of each algorithm is critically evaluated and discussed one by one. The weaknesses and strengths of all algorithms are highlighted based on their results, and possible future directions are also discussed.

5.1. *Performance of HBO*

When considering HBO on the unimodal benchmark functions, we find that its performance on approximately half of the unimodal functions is sufficient. In two cases, the exact global optimum is found. However, based on its comparative performance, HBO is part of the top

Table 8
Comparison of the performance of HBO, PO, GPC, GBO, RDA, MPA, LFD, TSA, BWOA, ROA, ChOA, TSO, and EO for multimodal functions.

Fn	Stats	HBO	PO	GPC	GBO	RDA	MPA	LFD	TSA	BWOA	ROA	ChOA	TSO	EO
*F*26	Avg	23.9151	9.47508	269.463	116.764	218.429	103.424	303.062	238.585	40.4917	318.151	334.196	0.095	121.27
	Std	7.52135	32.7941	4.39371	21.8015	18.5572	12.361	9.93187	15.9142	28.5763	8.6702	9.01511	0.47374	16.7312
	Rank	**6**	**5**	**21**	**11**	**19**	**9**	**23**	**20**	**8**	**24**	**25**	**4**	**12**
*F*27	Avg	20.039	7.95967	0	0	372.85	0	0.01761	358.18	3.3E−05	295.859	4.97465	0	0
	Std	3.36203	18.614	0	0	16.0057	0	0.08789	62.6871	0.0001	19.8677	6.03572	0	0
	Rank	**19**	**17**	**1**	**1**	**26**	**1**	**14**	**25**	**13**	**24**	**16**	**1**	**1**
*F*28	Avg	4.22752	0.952	0.9	0.9	8.72494	1.00868	0.9	8.16204	1.00009	15.1545	13.2046	0.9	1.00087
	Std	0.96784	0.05099	3.4E−16	3.4E−16	0.52953	0.00501	2E−09	1.13818	0.00018	0.75391	0.49306	3.4E−16	0.00156
	Rank	**20**	**9**	**1**	**1**	**22**	**14**	**7**	**21**	**11**	**26**	**25**	**1**	**12**
*F*29	Avg	0.00259	14.4113	25959.5	0.00115	1.4E+07	110.652	17640.5	5357.7	1290.66	2.4E+11	27826.9	8744.32	0.07922
	Std	0.00581	48.7294	1361.73	0.0026	1.2E+07	297.079	3107.51	2930.47	3616.81	2.1E+10	1969.14	2102.1	0.19025
	Rank	**2**	**6**	**19**	**1**	**24**	**8**	**17**	**14**	**12**	**26**	**20**	**15**	**4**
*F*30	Avg	1.4E−05	0	1.8E−13	6E−135	34.7646	8.9E−27	5.2E−05	58.3093	0.00034	85.3887	0.00049	5E−104	4.2E−09
	Std	6.2E−05	0	1.7E−13	2E−134	3.19512	2.1E−26	6.7E−06	11.8057	0.0007	5.16176	0.00058	3E−103	2.1E−08
	Rank	**14**	**1**	**12**	**7**	**24**	**11**	**15**	**25**	**16**	**26**	**17**	**9**	**13**
*F*31	Avg	1.66267	0	2.2E−22	1.8E−50	107561	1E−23	3.3E−06	0.41851	0.00472	1.1E+19	2.7E−06	7.5E−19	2E−111
	Std	4.80318	0	8.7E−22	9.1E−50	170479	5E−23	1.2E−05	0.74614	0.01218	1.6E+19	1.3E−05	3.6E−18	8E−111
	Rank	**20**	**1**	**11**	**8**	**21**	**10**	**15**	**18**	**17**	**26**	**14**	**12**	**6**
*F*32	Avg	1.9E−06	−9E−16	2.6E−12	−9E−16	1.57051	2.5E−15	0.00011	1.09582	3.7E−06	20.4292	19.9646	−9E−16	4.4E−15
	Std	1.4E−06	0	3.1E−12	0	0.53309	7.1E−16	1.3E−05	1.49757	9E−06	0.08881	0.00047	0	1.8E−15
	Rank	**14**	**1**	**13**	**1**	**21**	**10**	**16**	**19**	**15**	**26**	**25**	**1**	**12**
*F*33	Avg	2.77595	1	133.057	43.3093	662.198	40.1966	71.1023	189.585	1.21432	2380319	141.125	1.00897	26.4082
	Std	1.8243	0	3.08368	11.0546	198.035	6.96597	47.6114	49.1734	1.06104	129495	10.0034	0.01505	6.65037
	Rank	**4**	**1**	**17**	**10**	**22**	**9**	**14**	**20**	**3**	**26**	**18**	**2**	**6**
*F*34	Avg	0.47995	0.01689	1.7E−12	1E−108	2.94145	0.14387	6.6E−05	0.41987	0.22387	30.7213	0.12285	4E−100	0.10387
	Std	0.04995	0.05094	1.3E−12	3E−108	0.48965	0.05066	1.2E−05	0.07638	0.06633	0.89208	0.04039	2E−99	0.02
	Rank	**19**	**9**	**6**	**4**	**22**	**15**	**7**	**17**	**16**	**26**	**14**	**5**	**13**
*F*35	Avg	−1957.2	−1958.3	−815.78	−1708.9	−1399.1	−1762.1	−1364.8	−1409.8	−1939.1	−914.74	−883.13	−1958.3	−1739.5
	Std	3.91428	4.6E−13	49.7182	64.5093	32.2977	41.6813	57.5066	75.5368	24.7464	38.9749	54.2499	0.00062	36.7555
	Rank	**5**	**1**	**26**	**13**	**20**	**9**	**21**	**19**	**7**	**24**	**25**	**2**	**10**
*F*36	Avg	1.3E−08	0	0	0	0.39612	0	1.9E−08	0.00338	0.07897	24.5007	0.00586	0	0
	Std	6.5E−08	0	0	0	0.19003	0	5.6E−09	0.00581	0.15771	1.25088	0.01438	0	0
	Rank	**14**	**1**	**1**	**1**	**24**	**1**	**15**	**17**	**22**	**26**	**18**	**1**	**1**
*F*37	Avg	8.1E−23	−0.76	1.8E−15	−1	3.6E−19	2.4E−27	−0.9995	4.4E−21	2.3E−21	3.9E−14	1.8E−16	−1	5.5E−25
	Std	2.9E−22	0.43589	1.8E−15	0	1.7E−19	8.6E−28	6.6E−05	2.5E−21	2.6E−22	2.1E−14	2.2E−16	0	4.3E−25
	Rank	**14**	**7**	**25**	**1**	**21**	**10**	**6**	**18**	**17**	**26**	**24**	**1**	**13**
*F*38	Avg	3.1E−19	1.2E−20	3.2E−09	1.4E−20	3.3E−17	1.7E−20	6.8E−20	8.1E−12	1.2E−20	6.4E−09	1.7E−19	1.2E−20	6E−20
	Std	1.1E−19	5.7E−35	4.9E−09	9.1E−21	4.5E−17	6.1E−21	4.4E−21	1.7E−11	3.1E−22	8.7E−09	1.9E−21	1.3E−23	2.1E−21
	Rank	**18**	**2**	**25**	**7**	**22**	**8**	**14**	**24**	**4**	**26**	**17**	**3**	**12**
*F*39	Avg	0.00088	1.3E−32	4.63657	0.00956	70694.6	0.01512	4.93584	5.27782	1.2E−05	8.9E+08	4.72757	0.00011	0.03874
	Std	0.00304	5.6E−48	0.10322	0.0127	111519	0.02033	0.0275	0.99835	5.4E−05	8.2E+07	0.16146	0.00013	0.05067
	Rank	**5**	**1**	**16**	**7**	**25**	**8**	**19**	**21**	**2**	**26**	**17**	**3**	**10**
*F*40	Avg	1.6E−11	9.4E−33	0.62066	1.4E−08	22174.5	2.3E−09	0.49981	9.61048	5.2E−07	3.6E+08	0.54467	6.4E−06	5.1E−08
	Std	3.3E−11	1.4E−48	0.03617	4.3E−08	52809.2	7.8E−10	0.0755	3.4102	2.6E−06	4.8E+07	0.14247	1.2E−05	5.7E−08
	Rank	**2**	**1**	**18**	**4**	**25**	**3**	**16**	**23**	**7**	**26**	**17**	**8**	**5**
*F*41	Avg	4E−227	0	1.8E−36	0	4E−35	2E−147	4.6E−11	2E−227	1.3E−07	0.15667	0	4E−215	0
	Std	0	0	5.9E−36	0	1.5E−34	9E−147	5.6E−11	0	6.6E−07	0.13191	0	0	0
	Rank	**14**	**1**	**21**	**1**	**22**	**17**	**24**	**13**	**25**	**26**	**1**	**15**	**1**
*F*42	Avg	−195.63	−195.63	−195.63	−195.63	−195.63	−195.63	−195.63	−195.63	−195.63	−195.14	−195.63	−195.63	−195.63
	Std	5.8E−14	5.8E−14	9.8E−06	5.8E−14	0.00039	5.8E−14	2.6E−12	1E−06	9.4E−05	0.81724	3.1E−05	4.5E−05	5.9E−14
	Rank	**1**	**1**	**19**	**1**	**23**	**1**	**14**	**18**	**21**	**25**	**22**	**20**	**1**
*F*43	Avg	−2.0218	−2.0218	−1.9496	−2.0218	−2.0218	−2.0218	−2.0218	−2.0218	−1.8805	−2.3282	−2.0218	−2.0217	−2.0218
	Std	1.4E−15	1.3E−15	0.04311	1.4E−15	3.7E−10	1.4E−15	1.4E−05	5.8E−12	0.09055	0.15221	1.1E−10	0.00025	1.3E−15
	Rank	**2**	**2**	**24**	**2**	**20**	**2**	**22**	**17**	**25**	**1**	**19**	**23**	**2**
*F*44	Avg	−106.76	−106.76	−106.74	−106.76	−106.76	−106.76	−106.76	−105.21	−106.25	−106.54	−106.75	−106.76	−106.76
	Std	2.9E−15	1.8E−14	0.02174	8.2E−15	0.00092	1.3E−14	5E−11	5.38641	1.77473	0.25766	0.01118	0.00116	7.1E−15
	Rank	**1**	**7**	**21**	**1**	**18**	**7**	**14**	**25**	**23**	**22**	**20**	**19**	**1**
*F*45	Avg	−1.0316	−1.0316	−1.0275	−1.0316	−1.0316	−1.0316	−1.0316	−1.0278	−1.0266	−1.0256	−1.0316	−1.0316	−1.0316
	Std	6.7E−16	6.7E−16	0.00563	6.8E−16	1.6E−05	6.4E−16	1.2E−11	0.01049	0.02297	0.00909	3.2E−06	1.3E−06	6.4E−16
	Rank	**1**	**1**	**23**	**1**	**21**	**1**	**16**	**22**	**24**	**25**	**20**	**19**	**1**

(*continued on next page*)

5 algorithms for 6 functions only. To better determine its comparative position, the Friedman mean rank test is performed. According to the Friedman test, HBO is the 14th best algorithm. Based on these individual and comparative performance analyses, we conclude that HBO has a weak exploitative capability and needs improvement. This finding may help researchers investigate this aspect further and improve HBO. In addition, the exploration capability of HBO is evaluated by using multimodal benchmark functions. Its performance on more than half of the functions is promising and could have been further improved with more iterations. Moreover, in 6 cases, the exact global optimum

Table 8 (*continued*).

Fn	Stats	HBO	PO	GPC	GBO	RDA	MPA	LFD	TSA	BWOA	ROA	ChOA	TSO	EO
*F*46	Avg	0.39789	0.39789	0.84735	0.39789	0.39789	0.39789	0.39789	0.39789	0.39791	0.40592	0.39805	0.39789	0.39789
	Std	0	0	0.73568	0	0	0	2.4E−11	3E−06	0.00012	0.01745	0.00017	2.8E−06	0
	Rank	**2**	**2**	**26**	**2**	**2**	**2**	**18**	**22**	**23**	**25**	**24**	**21**	**2**
*F*47	Avg	−3.8628	−3.8628	−3.5943	−3.8628	−3.8628	−3.8628	−3.8579	−3.8627	−3.8533	−3.8539	−3.8555	−3.8368	−3.8628
	Std	2.3E−15	2.3E−15	0.28255	2.3E−15	5.2E−06	2.3E−15	0.00342	3.1E−05	0.01071	0.00654	0.00231	0.05015	2.1E−15
	Rank	**1**	**1**	**26**	**1**	**18**	**1**	**21**	**19**	**24**	**23**	**22**	**25**	**1**
*F*48	Avg	−3.322	−3.3125	−2.1583	−3.2649	−3.3167	−3.322	−3.0819	−3.2585	−3.2585	−3.0236	−2.6098	−3.1135	−3.2504
	Std	4.5E−16	0.03292	0.40755	0.06062	0.00349	1.5E−15	0.06456	0.08008	0.07029	0.09529	0.46858	0.12692	0.05965
	Rank	**1**	**7**	**26**	**13**	**6**	**4**	**23**	**16**	**15**	**24**	**25**	**22**	**19**
*F*49	Avg	−2.0626	−2.0626	−2.0626	−2.0626	−2.0626	−2.0626	−2.0626	−2.0626	−2.0626	−2.0625	−2.0626	−2.0626	−2.0626
	Std	9.1E−16	9.1E−16	4.3E−07	9.1E−16	8.6E−07	9.1E−16	7.9E−15	8.8E−08	7E−06	0.00015	2E−06	3.7E−07	9.1E−16
	Rank	**1**	**1**	**21**	**1**	**22**	**1**	**15**	**19**	**23**	**25**	**24**	**20**	**1**
*F*50	Avg	1	1	1	1	1	1	1	1	1.00001	46.2722	1	1	1
	Std	0	0	4.5E−17	0	3.8E−14	0	4.4E−07	0	2.8E−05	61.7362	0	0	0
	Rank	**2**	**2**	**2**	**2**	**21**	**2**	**22**	**2**	**24**	**26**	**2**	**2**	**2**

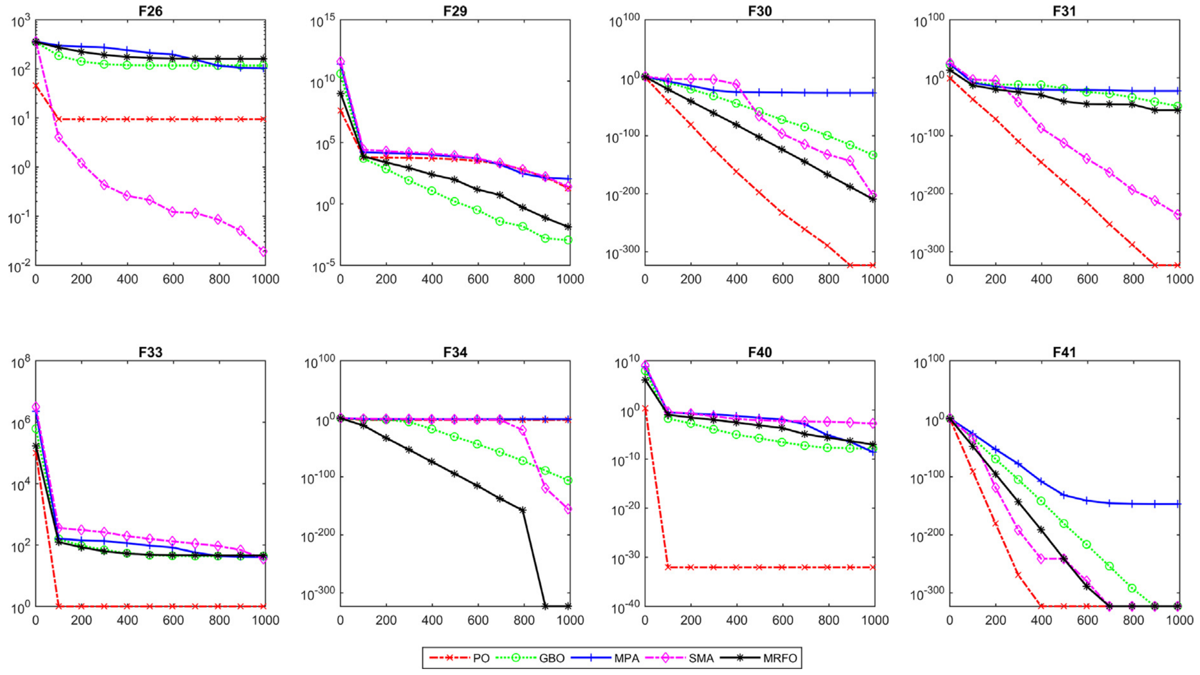

Fig. 6. Comparison of the convergence curves of the top five algorithms for selected multimodal functions.

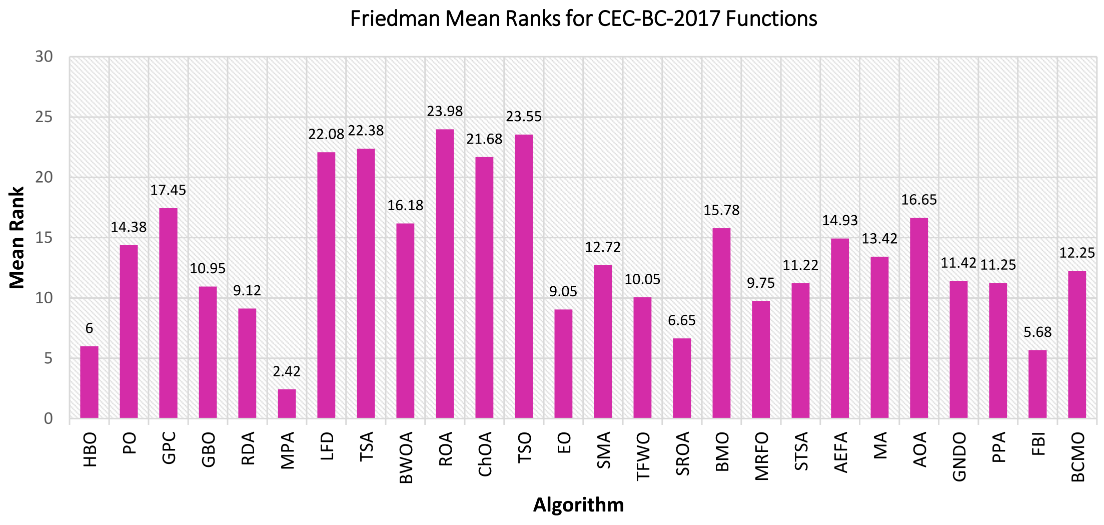

Fig. 7. Friedman mean ranks of all algorithms for CEC-BC-2017 benchmark functions.

Table 9
Comparison of the performance of SMA, TFWO, SROA, BMO, MRFO, STSA, AEFA, MA, AOA, GNDO, PPA, FBI, and BCMO for multimodal functions.

Fn	Stats	SMA	TFWO	SROA	BMO	MRFO	STSA	AEFA	MA	AOA	GNDO	PPA	FBI	BCMO
*F*26	Avg	0.0174	37.8801	−4E+26	181.824	159.737	287.8	349.826	115.962	−8E+10	216.288	157.27	161.698	142.506
	Std	0.02547	8.169	7.6E+26	25.7744	19.3397	6.4032	9.09031	14.4516	3.9E+11	46.7101	16.6006	10.2596	16.3283
	Rank	**3**	**7**	**1**	**17**	**15**	**22**	**26**	**10**	**2**	**18**	**14**	**16**	**13**
*F*27	Avg	0	16.6412	1.91382	0	0	0	57.5095	88.9901	0	52.8697	99.2446	0	0
	Std	0	5.93303	3.64394	0	0	0	8.87532	28.1514	0	15.6247	17.7303	0	0
	Rank	**1**	**18**	**15**	**1**	**1**	**1**	**21**	**22**	**1**	**20**	**23**	**1**	**1**
*F*28	Avg	0.9	1.10075	0.948	0.9	0.9	8.74962	1.00227	1.00003	12.0456	2.70021	1.03944	2.18983	1.54686
	Std	3.4E−16	0.50016	0.04336	3.4E−16	3.4E−16	4.54498	0.00784	1.2E−05	4.79047	1.64002	0.06975	1.00824	2.23909
	Rank	**1**	**16**	**8**	**1**	**1**	**23**	**13**	**10**	**24**	**19**	**15**	**18**	**17**
*F*29	Avg	21.337	0.3098	929.014	2275.51	0.01187	11447.6	5449511	402752	20372.7	3.8E+07	282016	857.03	340.055
	Std	11.3424	0.45311	197.547	1324.71	0.02057	805.249	3893192	1211749	2030.63	2.5E+07	430712	1094.91	358.226
	Rank	**7**	**5**	**11**	**13**	**3**	**16**	**23**	**22**	**18**	**25**	**21**	**10**	**9**
*F*30	Avg	1E−207	0.00592	0.02488	0	5E−212	6E−140	0.04612	0.01758	1E−166	3.70282	2.67474	1E−119	1.4E−34
	Std	0	0.02346	0.0086	0	0	1E−139	0.07451	0.02775	0	1.96345	2.01859	9E−120	6E−34
	Rank	**4**	**18**	**20**	**1**	**3**	**6**	**21**	**19**	**5**	**23**	**22**	**8**	**10**
*F*31	Avg	2E−236	2.2E+11	1.3E−05	0	1.4E−56	0	1.4E+09	200105	4.7E−07	1.50596	7.8E+07	2E−112	1.6E−31
	Std	0	1.1E+12	1E−05	0	7E−56	0	5.1E+09	972361	2.4E−06	1.7872	2.1E+08	1E−111	7E−31
	Rank	**4**	**25**	**16**	**1**	**7**	**1**	**24**	**22**	**13**	**19**	**23**	**5**	**9**
*F*32	Avg	−9E−16	0.76056	0.01019	−9E−16	−9E−16	2.7E−15	1.18934	4.87721	−9E−16	10.0252	12.6245	−9E−16	−9E−16
	Std	0	0.52986	0.00477	0	0	0	0.68366	3.50193	0	1.16351	1.37793	0	0
	Rank	**1**	**18**	**17**	**1**	**1**	**11**	**20**	**22**	**1**	**23**	**24**	**1**	**1**
*F*33	Avg	32.8359	13.4084	31.624	60.6184	45.7401	123.209	7869.37	286.922	149.146	32067.9	2413.04	104.63	70.3336
	Std	28.7961	7.7927	12.7859	6.65513	3.59908	4.38856	3728.91	93.3643	9.40305	13116.9	834.898	26.8541	21.0551
	Rank	**8**	**5**	**7**	**12**	**11**	**16**	**24**	**21**	**19**	**25**	**23**	**15**	**13**
*F*34	Avg	0	2.13192	0.4627	0	0	0.09987	2.29038	4.3119	0.09188	5.33987	11.0039	0.09987	0.00817
	Std	0	0.29827	0.26226	0	0	2.6E−10	0.50211	1.23703	0.02765	0.677	1.86066	8.1E−07	0.01833
	Rank	**1**	**20**	**18**	**1**	**1**	**11**	**21**	**23**	**10**	**24**	**25**	**12**	**8**
*F*35	Avg	−1958.3	−1953.2	−1958.3	−1549.3	−1696.5	−1109.9	−1784.2	−1682.4	−1066.9	−1614.7	−1728	−1698.4	−1724.7
	Std	0.01451	6.92555	0.00195	54.307	43.2115	28.2617	47.4589	56.5646	53.6557	54.1671	45.7018	63.7915	49.9529
	Rank	**4**	**6**	**3**	**18**	**15**	**22**	**8**	**16**	**23**	**17**	**11**	**14**	**12**
*F*36	Avg	0	0.05263	0.00036	0	0	0	0.05896	0.01133	0	1.11037	0.24219	0	0
	Std	0	0.05428	0.00038	0	0	0	0.07425	0.01209	0	0.0775	0.12193	0	0
	Rank	**1**	**20**	**16**	**1**	**1**	**1**	**21**	**19**	**1**	**25**	**23**	**1**	**1**
*F*37	Avg	−1	3.5E−25	−0.4423	−1	−1	2E−18	9.3E−22	9E−26	4.5E−19	5.6E−21	5.3E−22	6.1E−21	−0.1978
	Std	0	7E−25	0.25961	0	0	8.1E−19	5.8E−22	8.9E−26	3.5E−19	2.1E−21	3.6E−22	6E−21	0.22975
	Rank	**1**	**12**	**8**	**1**	**1**	**23**	**16**	**11**	**22**	**19**	**15**	**20**	**9**
*F*38	Avg	1.1E−21	7.3E−20	1.2E−20	1.7E−18	1.3E−20	4E−16	3.7E−19	4.3E−20	1.5E−19	5.2E−20	4E−20	1E−18	6.1E−20
	Std	3.4E−21	2.8E−20	4.6E−22	2.2E−18	7.9E−22	4.8E−16	4E−19	3.8E−21	1.7E−20	1.7E−20	4.1E−20	4.2E−19	1.1E−21
	Rank	**1**	**15**	**5**	**21**	**6**	**23**	**19**	**10**	**16**	**11**	**9**	**20**	**13**
*F*39	Avg	0.00286	0.03726	0.00042	4.95132	4.60789	2.62172	30.3653	0.71645	4.91386	14677.6	88.8541	1.84836	1.11029
	Std	0.00222	0.05623	0.00032	0.00291	0.95408	0.09543	10.3888	1.5448	0.05315	47860.7	17.3179	1.9048	0.46939
	Rank	**6**	**9**	**4**	**20**	**15**	**14**	**22**	**11**	**18**	**24**	**23**	**13**	**12**
*F*40	Avg	0.00167	0.01908	2.2E−05	0.04234	8.1E−08	0.18293	2.61205	1.65552	0.88226	7.58402	20.4445	0.00068	0.00328
	Std	0.00258	0.02857	3.4E−05	0.01584	1.6E−07	0.01685	0.92751	0.98017	0.09054	4.32026	9.50444	0.00033	0.00194
	Rank	**11**	**13**	**9**	**14**	**6**	**15**	**21**	**20**	**19**	**22**	**24**	**10**	**12**
*F*41	Avg	0	3E−101	2E−115	0	0	0	2.2E−26	0	0	0	2E−102	0	6E−153
	Std	0	1E−100	3E−115	0	0	0	2.4E−26	0	0	0	8E−102	0	2E−152
	Rank	**1**	**20**	**18**	**1**	**1**	**1**	**23**	**1**	**1**	**1**	**19**	**1**	**16**
*F*42	Avg	−195.63	−195.63	0	−195.63	−195.63	−195.63	−195.63	−195.63	−195.63	−195.63	−195.63	−195.63	−195.63
	Std	4.4E−11	5.8E−14	0	5.8E−14	5.8E−14	9.4E−09	0.00194	5.8E−14	1E−10	8.7E−14	5.8E−14	5.8E−14	5.8E−14
	Rank	**15**	**1**	**26**	**1**	**1**	**17**	**24**	**1**	**16**	**1**	**1**	**1**	**1**
*F*43	Avg	−2.0218	−2.0218	0	−2.0218	−2.0218	−2.0218	−2.0218	−2.0218	−2.0218	−2.0218	−2.0218	−2.0218	−2.0218
	Std	2.2E−15	1.4E−15	0	1.4E−15	1.3E−15	2.2E−12	1.4E−15	1.4E−15	3E−07	9.1E−16	4.1E−11	1.1E−15	1.3E−15
	Rank	**15**	**2**	**26**	**2**	**2**	**16**	**2**	**2**	**21**	**2**	**18**	**2**	**2**
*F*44	Avg	−106.76	−106.76	0	−106.76	−106.76	−106.76	−106.25	−106.76	−106.76	−106.76	−106.76	−106.76	−106.76
	Std	1.2E−07	9.2E−15	0	1.8E−14	2.2E−14	4.5E−06	1.428	2.3E−14	0.00016	1.2E−14	9.2E−15	8.7E−15	2E−14
	Rank	**15**	**1**	**26**	**7**	**7**	**16**	**24**	**7**	**17**	**7**	**1**	**1**	**7**
*F*45	Avg	−1.0316	−1.0316	0	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
	Std	9.5E−12	6.8E−16	0	6.8E−16	6.8E−16	2.7E−09	6.8E−16	6.8E−16	9.1E−09	4.5E−16	6.6E−16	6.8E−16	6.7E−16
	Rank	**15**	**1**	**26**	**1**	**1**	**17**	**1**	**1**	**18**	**1**	**1**	**1**	**1**

(*continued on next page*)

is found. To determine its comparative performance, the Friedman test is performed. According to the Friedman test, HBO is the 9th best algorithm, which is better than its position for unimodal functions. By comparing its performance on both types of functions, we may conclude that HBO has better exploration capability than exploitation capability.

To check its balance, the performance of HBO is also evaluated on CEC-BC-2017 benchmarks. Out of 29 benchmarks, HBO is a top 5 algorithm for 14 of these benchmarks, and its position according to the Friedman mean rank test is 3rd. Only MPA and FBI performed better than HBO on the CEC-BC-2017 benchmarks. This shows that HBO has

Table 9 (*continued*).

Fn	Stats	SMA	TFWO	SROA	BMO	MRFO	STSA	AEFA	MA	AOA	GNDO	PPA	FBI	BCMO
$F46$	Avg	0.39789	0.39789	0	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789
	Std	5.8E−10	0	0	0	0	1E−07	0	0	3.6E−16	0	0	0	0
	Rank	**19**	**2**	**1**	**2**	**2**	**20**	**2**	**2**	**16**	**17**	**2**	**2**	**2**
$F47$	Avg	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628	−3.8622	−3.8628	−3.8628
	Std	3.8E−08	2.3E−15	2.3E−15	2.2E−15	2.2E−15	6.8E−07	2.3E−15	2.3E−15	9.4E−14	1.8E−15	0.00173	2.3E−15	2.3E−15
	Rank	**16**	**1**	**1**	**1**	**1**	**17**	**1**	**1**	**15**	**1**	**20**	**1**	**1**
$F48$	Avg	−3.2363	−3.2507	−3.322	−3.2882	−3.2744	−3.2505	−3.322	−3.2839	−3.3071	−3.2649	−3.295	−3.322	−3.2411
	Std	0.05451	0.05945	4.5E−16	0.06325	0.05945	0.05941	4.4E−16	0.0566	0.04104	0.06062	0.05027	5E−06	0.0566
	Rank	**21**	**17**	**1**	**10**	**12**	**18**	**1**	**11**	**8**	**14**	**9**	**5**	**20**
$F49$	Avg	−2.0626	−2.0626	0	−2.0626	−2.0626	−2.0626	−2.0626	−2.0626	−2.0626	−2.0626	−2.0626	−2.0626	−2.0626
	Std	1.5E−11	9.1E−16	0	9.1E−16	9.1E−16	1.1E−09	9.1E−16	9.1E−16	1E−09	1.4E−15	9.1E−16	9.1E−16	9.1E−16
	Rank	**16**	**1**	**26**	**1**	**1**	**18**	**1**	**1**	**17**	**1**	**1**	**1**	**1**
$F50$	Avg	1	1	0	1	1	1	1.00002	1	1	1	1.00001	1	1
	Std	0	0	0	0	0	0	3.5E−05	0	0	0	3.2E−05	0	0
	Rank	**2**	**2**	**1**	**2**	**2**	**2**	**25**	**2**	**2**	**20**	**23**	**2**	**2**

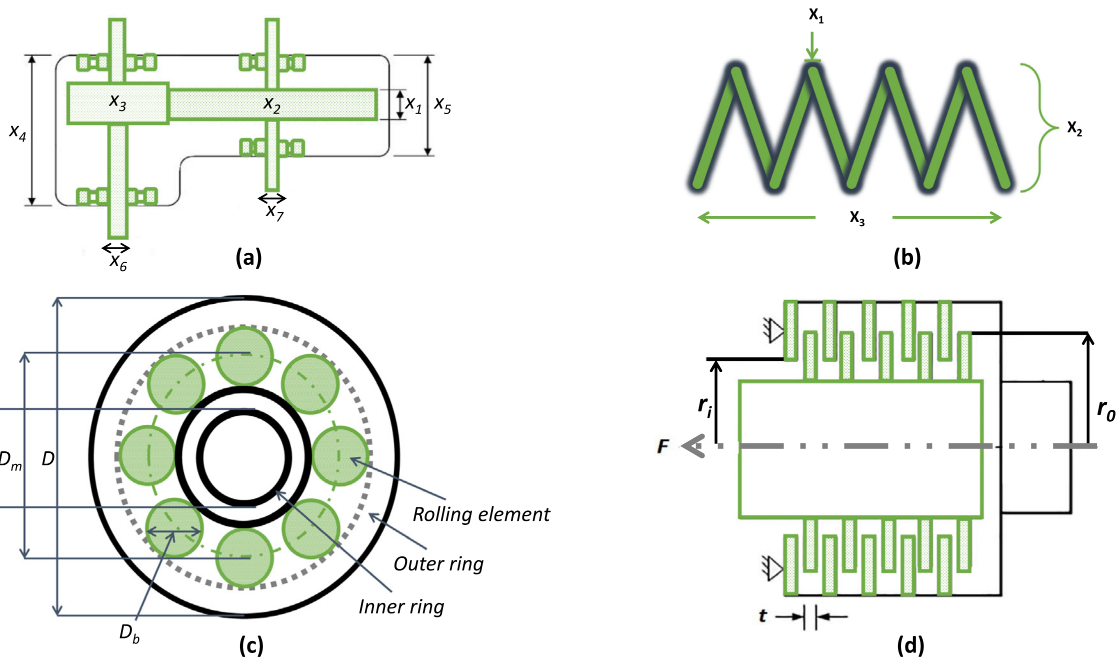

Fig. 8. Schematic views of engineering problems: (a) speed-reducer design problem, (b) tension/compression spring design, (c) rolling-element bearing design, and (d) multiple-disk clutch brake design.

very good balance mechanism and perhaps this is the reason behind its average exploitation capabilities. Finally, its applicability to real-world problems is evaluated using 4 engineering problems. HBO outperforms all other algorithm on two problems and is 2nd and 3rd best on the other two problems. Moreover, HBO ranks first in the list according to the Friedman test, thus demonstrating its excellent optimization skills on real-word constrained problems. Based on the comparative results and our findings, we recommend HBO for complex and real-world constrained problems because it has excellent balance; however, for problems where better exploitation is required, HBO may not be an ideal option. Our findings give direction to the research community on the aspects of this algorithm require further improvement.

5.2. Performance of PO

The performance of PO on unimodal functions is excellent, as presented in Table 6. For 16 of 25 unimodal functions, the global optimum is found using PO. In analysing its comparative performance, there are only 3 unimodal functions for which PO is not in top 10. Based on the results of the Friedman test presented in Fig. 3, PO ranks second. Only GBO is better than PO in this list. Considering its excellent performance, we also analyse the convergence behaviour of PO on unimodal function, as shown in Fig. 4. As seen, PO has an excellent convergence speed, and the best positions are reached in half of the iterations. However, for $F19$, PO converges prematurely. We can conclude from the comparative analysis that PO has excellent exploitation capability and is recommended for the problems having no or fewer local optima. Furthermore, to analyse the exploration capability, the results on multimodal functions are presented in Table 8. PO outperforms all other algorithms on 13 multimodal benchmark functions, and there is only one benchmark for which PO is not included in top 10. According to the Friedman test presented in Fig. 5, PO is the best and is ranked first. Its convergence behaviour is presented in Figure Fig. 6. According to the results, PO shows outstanding convergence speed and finds the most promising region in fewer iterations than the other algorithms. However, it is evident from $F26$, $F29$, and $F34$ that PO gets stuck in some local optimum, which is due to not providing sufficient time for exploration. Based on its excellent performance, we highly recommend PO for problems with local minima.

Moreover, the performance on complex functions is checked on CEC-BC-2017 benchmarks, as shown in Table 10. The performance of PO on complex functions drastically decreases. The Friedman test shows that PO is the 16th best algorithm for CEC-BC-2017 benchmark

Table 10
Comparison of the performance of HBO, PO, GPC, GBO, RDA, MPA, LFD, TSA, BWOA, ROA, ChOA, TSO, and EO for CEC-BC-2017 functions.

Fn	Stats	HBO	PO	GPC	GBO	RDA	MPA	LFD	TSA	BWOA	ROA	ChOA	TSO	EO
*F*51	Avg	490.1832	6284.225	6.56E+08	1928.807	3895.069	100.0001	4.13E+09	2.98E+09	1.05E+09	5.75E+09	1.53E+09	5.73E+09	2835.628
	Std	406.4005	4490.46	2.22E+08	1822.998	4086.156	9.95E−05	1.15E+09	2.82E+09	5.45E+08	1.33E+09	1.32E+09	3.37E+09	2836.865
	Rank	**4**	**16**	**20**	**8**	**14**	**1**	**24**	**23**	**21**	**26**	**22**	**25**	**10**
*F*52	Avg	301.9884	496.4448	3910.75	300	300.121	300	13179.94	10070.35	5685.21	20198.01	2322.536	12281.34	300
	Std	3.133711	299.0156	1762.088	6.86E−14	0.271763	1.48E−09	3944.375	5879.9	2775	4865.122	730.5671	4372.856	3.8E−05
	Rank	**13**	**16**	**19**	**2**	**12**	**4**	**25**	**22**	**21**	**26**	**18**	**23**	**8**
*F*53	Avg	405.0895	405.6096	462.8461	400.0867	405.8066	400	661.3159	516.531	472.9854	801.3793	618.3492	707.7565	404.5132
	Std	0.899104	0.759631	11.48758	0.192641	0.445851	1.18E−07	98.5527	112.1954	36.50607	145.1241	165.7755	178.1216	1.186692
	Rank	**11**	**12**	**20**	**3**	**13**	**1**	**24**	**22**	**21**	**26**	**23**	**25**	**9**
*F*54	Avg	512.3064	512.0532	538.319	527.7405	516.741	508.7288	569.598	561.0994	524.294	588.2685	556.8031	569.9563	510.3766
	Std	3.662879	6.567678	4.687924	11.59847	3.197425	3.258824	10.49854	20.83961	6.903424	8.722404	8.216864	15.79674	3.828501
	Rank	**7**	**6**	**20**	**17**	**8**	**2**	**24**	**23**	**15**	**26**	**22**	**25**	**5**
*F*55	Avg	600	609.5841	620.5379	600.0696	600	600.0011	636.1653	629.7209	610.1235	654.9035	626.4831	640.9433	600.0336
	Std	2.84E−07	16.22112	4.962298	0.161223	2.05E−05	0.003836	8.147344	14.7908	3.327668	5.440641	9.244737	14.79662	0.167621
	Rank	**1**	**18**	**21**	**9**	**2**	**6**	**24**	**23**	**19**	**26**	**22**	**25**	**7**
*F*56	Avg	724.9839	719.0217	770.3638	737.6339	731.6104	721.2798	822.2314	780.4222	721.3173	933.754	794.0629	807.3814	723.0608
	Std	3.797462	3.943396	8.491221	10.34231	5.575558	4.178397	11.68376	24.4371	4.770791	44.51272	18.99426	20.93126	5.39879
	Rank	**7**	**2**	**21**	**14**	**13**	**4**	**25**	**22**	**5**	**26**	**23**	**24**	**6**
*F*57	Avg	812.2694	834.1004	825.2125	822.3268	814.8331	808.477	865.8275	844.5482	812.1495	883.8291	841.852	846.3078	812.4967
	Std	4.756199	4.585033	3.458119	7.059269	4.700751	3.326294	7.549954	18.93194	4.011316	9.903009	7.611581	10.42799	5.178555
	Rank	**6**	**21**	**18**	**17**	**9**	**3**	**25**	**23**	**5**	**26**	**22**	**24**	**7**
*F*58	Avg	900	1562.54	1099.068	907.4759	900	900	1595.363	1396.554	938.6425	2105.081	1351.554	1546.42	900.0763
	Std	0	334.5499	151.3889	25.07853	6.03E−09	3.1E−07	205.7607	437.7011	26.92796	269.7936	246.8689	262.9735	0.169339
	Rank	**1**	**24**	**20**	**11**	**3**	**4**	**25**	**22**	**17**	**26**	**21**	**23**	**7**
*F*59	Avg	1700.123	1863.125	1616.704	1912.783	1396.441	1365.543	2404.377	2118.773	1772.122	2771.599	2811.991	2436.181	1401.825
	Std	143.9461	293.3702	163.1048	332.1394	197.7961	155.6582	190.5343	282.4205	182.9994	179.6725	201.5914	371.9854	298.8309
	Rank	**13**	**17**	**7**	**18**	**3**	**2**	**23**	**20**	**14**	**25**	**26**	**24**	**4**
*F*60	Avg	1102.972	1121.961	1155.29	1113.102	1103.715	1101.964	1541.127	2607.228	1304.689	1988.988	1308.626	3164.058	1106.23
	Std	1.154806	37.87039	22.19321	15.14266	1.960145	0.979516	204.9977	2440.073	355.4252	401.3736	97.72729	2811.522	3.882114
	Rank	**2**	**14**	**17**	**8**	**4**	**1**	**23**	**25**	**19**	**24**	**20**	**26**	**5**
*F*61	Avg	97816.42	28195.66	314385	19590.56	15185.61	1212.518	92924297	1729484	1188993	1.57E+08	7811606	56286583	10819.58
	Std	108255.4	24606.49	340278.6	16558.59	11713.33	26.32236	57539605	2186432	959491.5	81132243	7183716	1.93E+08	6545.101
	Rank	**15**	**11**	**17**	**10**	**7**	**1**	**25**	**21**	**20**	**26**	**23**	**24**	**4**
*F*62	Avg	2222.283	12374.34	9390.342	1583.617	13274.4	1304.898	23029.57	15837.85	9178.518	602818.4	25813.03	17922.27	8436.373
	Std	1037.519	14057.79	8782.869	275.8515	11081.08	2.077764	43239.73	8421.645	3226.963	817878.8	14503.83	12949.83	7768.092
	Rank	**6**	**19**	**13**	**4**	**20**	**1**	**24**	**22**	**12**	**26**	**25**	**23**	**11**
*F*63	Avg	1461.153	1559.144	3998.492	1493.959	1492.468	1403.463	1754.114	3972.605	3133.318	1931.545	5665.297	1565.059	1461.125
	Std	58.61203	57.81456	1153.372	38.46783	98.97256	2.012688	270.1925	1899.785	1931.127	646.8213	890.5835	78.57097	21.8247
	Rank	**7**	**13**	**23**	**11**	**10**	**1**	**16**	**22**	**20**	**18**	**25**	**14**	**6**
*F*64	Avg	1562.54	6094.563	2841.052	1592.413	2072.207	1500.512	6284.979	7792.69	2785.067	11807.64	11860.34	8458.244	1629.668
	Std	46.5559	1252.925	993.2569	55.79981	1315.139	0.421037	3097.121	7043.54	1104.804	8207.902	8471.706	3698.167	85.7494
	Rank	**6**	**20**	**15**	**7**	**12**	**1**	**21**	**22**	**14**	**24**	**25**	**23**	**8**
*F*65	Avg	1602.442	1712.282	1758.216	1709.923	1636.62	1601.296	1899.213	1917.992	1813.3	2020.382	1976.301	2024.136	1668.53
	Std	3.142476	150.3255	110.2967	99.61594	44.70945	0.842881	135.7121	160.252	108.4786	109.4418	102.137	136.9766	73.09599
	Rank	**2**	**11**	**17**	**10**	**6**	**1**	**21**	**22**	**19**	**25**	**23**	**26**	**7**
*F*66	Avg	1707.251	1957.639	1749.019	1733.49	1737.904	1717.77	1849.138	1832.811	1748.738	1906.686	1784.253	1808.884	1734.156
	Std	8.661983	71.84736	6.178304	16.94806	29.60472	8.696369	43.92178	101.0715	18.87741	66.87943	15.75313	44.15852	13.08396
	Rank	**1**	**26**	**15**	**8**	**11**	**4**	**24**	**23**	**14**	**25**	**20**	**21**	**9**
*F*67	Avg	4424.523	23812.27	34106.74	6945.01	19313.25	1801.084	308577.7	24435.67	5117.987	1519106	57818.63	15763.01	13518.29
	Std	2029.338	7006.224	12371.98	8674.125	11792.16	0.814174	586590.2	15809.44	2587.268	2414654	37842.21	16440.24	11324.63
	Rank	**5**	**20**	**23**	**10**	**19**	**1**	**25**	**21**	**6**	**26**	**24**	**18**	**16**
*F*68	Avg	1991.14	3747.232	9447.225	1967.164	8506.953	1900.853	20358.96	174013.6	3789.908	13840.45	20035.89	56658.12	1941.614
	Std	156.6173	1896.778	4356.331	45.9652	9117.455	0.410787	30692.83	400558	1460.758	3862.236	4772.783	102549.5	19.96705
	Rank	**8**	**12**	**20**	**7**	**19**	**1**	**24**	**26**	**13**	**21**	**23**	**25**	**6**
*F*69	Avg	2000.025	2120.341	2074.603	2060.087	2001.526	2013.537	2141.312	2187.768	2055.538	2197.985	2237.948	2204.124	2032.153
	Std	0.086437	94.20655	55.70844	52.9381	2.645725	10.11572	32.56463	99.07093	44.0393	42.57419	81.01238	56.1896	39.40424
	Rank	**1**	**20**	**17**	**13**	**2**	**6**	**21**	**23**	**12**	**24**	**26**	**25**	**9**
*F*70	Avg	2263.182	2311.066	2273.989	2273.922	2278.852	2200	2263.375	2325.811	2281.853	2288.708	2317.86	2322.966	2298.302
	Std	55.54827	9.882577	59.26274	61.37039	53.75626	1.13E−05	27.81748	52.57447	49.38467	31.52931	57.2391	46.8159	37.30676
	Rank	**5**	**21**	**9**	**8**	**10**	**1**	**6**	**26**	**11**	**14**	**24**	**25**	**17**

(*continued on next page*)

functions. The reason behind such degradation is perhaps its premature convergence on complex benchmarks. It is recommended to enhance the duration of the exploration phase and better control the transition between exploration and exploitation. Finally, the performance of PO on engineering problems can be seen in Table 12. As can be seen, the performance of PO on all these problems is average. According to the results of the Friedman test presented in Fig. 9, PO ranks eight, which is above average. Conclusively, based on our findings, we highly recommend PO for problems with simple to average level complexity; however, problems having very challenging landscapes, PO may not be

Table 10 (*continued*).

Fn	Stats	HBO	PO	GPC	GBO	RDA	MPA	LFD	TSA	BWOA	ROA	ChOA	TSO	EO
*F*71	Avg	2297.96	2300.425	2396.995	2302.177	2296.873	2278.482	2642.62	2572.047	2350.045	2823.891	3115.422	2735.962	2300.808
	Std	12.90143	0.195162	27.21597	1.236683	17.85329	40.13945	165.0893	359.8853	25.91404	166.1585	707.624	358.0728	0.370325
	Rank	**6**	**10**	**21**	**13**	**5**	**2**	**23**	**22**	**18**	**25**	**26**	**24**	**11**
*F*72	Avg	2613.785	2615.945	2646.967	2625.883	2612.147	2602.277	2687.492	2696.26	2645.8	2701.122	2656.887	2688.181	2614.094
	Std	3.429981	9.607836	3.527403	11.17619	4.455717	42.30506	10.62747	29.25329	12.1264	16.50191	6.596816	38.94117	4.785548
	Rank	**7**	**9**	**21**	**15**	**5**	**1**	**23**	**25**	**20**	**26**	**22**	**24**	**8**
*F*73	Avg	2666.326	2744.902	2778.816	2720.113	2731.727	2496.042	2763.954	2825.166	2752.475	2807.444	2786.724	2828.938	2731.774
	Std	103.0306	10.01287	46.46992	88.92833	69.41559	35.24447	54.9513	29.3312	34.89563	44.08796	24.58464	63.6196	48.69148
	Rank	**4**	**16**	**22**	**10**	**13**	**1**	**21**	**25**	**17**	**24**	**23**	**26**	**14**
*F*74	Avg	2918.958	2935.728	2973.192	2927.37	2931.35	2885.835	3159.408	3025.027	2962.916	3300.85	3018.828	3307.842	2924.203
	Std	22.60715	19.42267	41.10646	23.46776	22.9192	59.54181	62.36042	109.2656	15.0015	85.69573	71.5899	205.0728	69.71974
	Rank	**5**	**15**	**21**	**10**	**12**	**2**	**24**	**23**	**20**	**25**	**22**	**26**	**8**
*F*75	Avg	2896.601	2866.686	3107.439	2960.232	2935.075	2784.002	3465.199	3781.462	3273.271	3577.879	3960.879	3963.486	3013.578
	Std	45.92145	64.98878	38.32045	75.26882	49.7426	114.3089	130.3795	491.5128	176.7573	144.089	331.5911	445.8474	244.829
	Rank	**5**	**4**	**18**	**10**	**7**	**2**	**22**	**24**	**21**	**23**	**25**	**26**	**13**
*F*76	Avg	3091.521	3091.878	3099.694	3102.468	3090.936	3089.002	3131.713	3165.256	3156.212	3169.679	3099.198	3207.521	3093.33
	Std	2.186673	2.510503	1.160699	17.47848	1.62159	0.420016	18.1191	39.72729	26.48638	21.61066	3.519101	85.10597	3.88895
	Rank	**4**	**5**	**12**	**14**	**2**	**1**	**21**	**23**	**22**	**24**	**11**	**25**	**6**
*F*77	Avg	3175.441	3403.789	3310.086	3342.671	3249.93	3088	3450.537	3485.996	3513.346	3551.503	3244.486	3615.625	3369.545
	Std	60.1842	28.33745	80.44659	154.704	122.5476	59.99946	73.21388	200.5359	191.5404	76.45879	6.855988	198.6262	102.1119
	Rank	**3**	**20**	**15**	**17**	**8**	**2**	**21**	**22**	**23**	**24**	**6**	**25**	**18**
*F*78	Avg	3187.746	3159.188	3203.782	3225.015	3162.871	3143.532	3337.106	3310.839	3217.667	3411.19	3355.304	3444.17	3176.761
	Std	13.85019	22.86039	33.38168	59.82597	12.97373	8.602191	61.72347	85.48284	37.71558	73.99094	63.24614	119.3953	30.85855
	Rank	**8**	**2**	**12**	**16**	**3**	**1**	**22**	**21**	**14**	**24**	**23**	**25**	**5**
*F*79	Avg	52273.1	644183.4	556078.7	463298.1	273232.2	3397.648	3990463	2508076	857367.1	10166346	3793386	8442589	403991.7
	Std	62770.51	452738.5	307908.6	625252.6	373689.5	3.855551	2969231	4205999	1321808	4063393	3507908	10133554	588613.9
	Rank	**3**	**18**	**16**	**15**	**8**	**1**	**23**	**20**	**19**	**25**	**22**	**24**	**14**

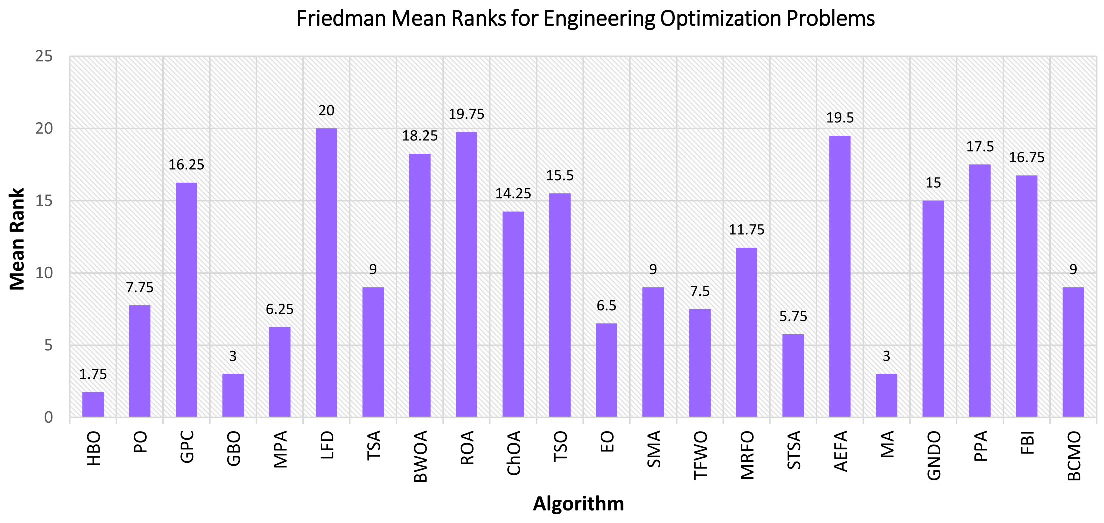

Fig. 9. Friedman mean ranks of all algorithms for engineering optimization problems.

an ideal option. we suggest that the research community work on its balance to utilize this algorithm for highly challenging problems.

5.3. GPC performance

As in Table 6, the global optimum could not be obtained for any unimodal function using GPC. There are only 2 functions for which GPC could ranks among the top 5 algorithms. According to the Friedman test, its rank is 17th on the list, which is below average. Based on these facts, we can say GPC does not have very good exploitative capabilities. However, for 3 multimodal functions, the global optimum is found. GPC ranked in the top ten for only 8 functions. The Friedman rank of GPC is 22nd, which is truly very poor. Hence, we may conclude that the exploration capability of GPC is in need of serious improvement. Furthermore, the performance of GPC on the CEC-BC-2017 benchmarks is also very poor and below average. There are only two functions for which GPC ranked among the top ten. The Friedman rank of GPC for the CEC-BC-2017 benchmarks is 21st, which is also very poor. However, for engineering problems, GPC ranks 20th.

Based on the above findings, we conclude that GPC has an overall weak exploitation and exploration capabilities. Furthermore, its balance between exploitation and exploitation is also very weak. Working to improve the exploration and exploitation capabilities of this algorithm could be a good future direction.

5.4. Performance of GBO

The results show that a global optimum is found for 8 unimodal functions using GBO, and there are no unimodal functions for which GBO does not rank among the top 10. This is also evident from the Friedman mean rank test, where GBO ranked first on the list. The convergence comparison presented in Fig. 4 shows that GBO is a

Table 11
Comparison of the performance of SMA, TFWO, SROA, BMO, MRFO, STSA, AEFA, MA, AOA, GNDO, PPA, FBI, and BCMO for CEC-BC-2017 functions.

Fn	Stats	SMA	TFWO	SROA	BMO	MRFO	STSA	AEFA	MA	AOA	GNDO	PPA	FBI	BCMO
*F*51	Avg	7606.598	3298.899	100.0009	3344.493	2825.001	30316362	643.1859	4189.729	47596410	1582.072	300.0749	3169.954	1032.073
	Std	4321.063	3161.986	0.000477	3572.838	3313.91	23175043	632.738	4257.973	46088698	1900.742	220.8147	3183.31	1167.093
	Rank	**17**	**12**	**2**	**13**	**9**	**18**	**5**	**15**	**19**	**7**	**3**	**11**	**6**
*F*52	Avg	300.0039	300	300	613.2301	300	386.7291	12860.49	300	4275.476	300.0142	300	300.0031	367.1514
	Std	0.004465	5.68E−14	3.29E−06	386.7642	1.2E−11	33.44695	4505.818	2.59E−07	1757.771	0.071024	2.36E−05	0.008194	149.4064
	Rank	**10**	**1**	**7**	**17**	**3**	**15**	**24**	**5**	**20**	**11**	**6**	**9**	**14**
*F*53	Avg	407.6406	402.4963	400.1038	407.3567	401.3236	409.3788	407.1461	400.0531	421.143	410.3759	400.1419	400.4197	404.6225
	Std	10.26746	1.235106	0.146191	12.22129	0.683112	0.813251	0.185356	0.030619	18.90428	15.96927	0.318736	0.262025	2.221617
	Rank	**16**	**8**	**4**	**15**	**7**	**17**	**14**	**2**	**19**	**18**	**5**	**6**	**10**
*F*54	Avg	517.9304	516.9271	510.0694	531.0426	521.4911	526.1825	504.7758	519.5187	541.3666	529.3711	523.9984	509.9561	523.1626
	Std	8.226363	7.089799	2.433923	12.82868	8.324389	5.621679	2.584978	13.04707	5.49411	11.11694	10.22481	2.011018	9.229165
	Rank	**10**	**9**	**4**	**19**	**12**	**16**	**1**	**11**	**21**	**18**	**14**	**3**	**13**
*F*55	Avg	600.0837	600	600.0003	609.119	601.9358	603.7734	600.0632	600.1336	601.7969	607.6605	610.8827	600.0002	600.1117
	Std	0.031481	4.59E−05	0.000199	11.16591	3.78755	1.071607	0.164181	0.167772	0.988059	8.396949	5.897178	0.000315	0.286284
	Rank	**10**	**3**	**5**	**17**	**14**	**15**	**8**	**12**	**13**	**16**	**20**	**4**	**11**
*F*56	Avg	725.5457	726.9392	720.4998	764.4812	746.611	746.2572	713.3456	727.5141	743.5818	740.1235	743.1525	727.7597	731.2314
	Std	7.701321	7.777355	2.637299	19.66585	15.5968	4.951313	1.497994	9.773385	6.521981	10.34752	16.8999	3.580526	9.793137
	Rank	**8**	**9**	**3**	**20**	**19**	**18**	**1**	**10**	**17**	**15**	**16**	**11**	**12**
*F*57	Avg	814.3685	816.6755	807.9951	825.7495	819.5352	815.8705	804.0992	818.8989	831.9665	819.9787	821.5309	809.4219	815.3224
	Std	6.180948	7.077947	1.91638	6.195672	7.487354	4.035123	1.706966	9.079606	5.298126	6.311762	8.254118	3.178191	6.079285
	Rank	**8**	**12**	**2**	**19**	**14**	**11**	**1**	**13**	**20**	**15**	**16**	**4**	**10**
*F*58	Avg	900.0195	903.8817	900.0001	999.8752	912.7819	902.8538	900	914.5807	915.2008	928.9462	1011.677	900.1198	909.8064
	Std	0.09091	10.25001	0.000262	97.02065	26.76818	0.920001	0	22.31907	15.8286	26.89929	136.2516	0.202605	11.15957
	Rank	**6**	**10**	**5**	**18**	**13**	**9**	**1**	**14**	**15**	**16**	**19**	**8**	**12**
*F*59	Avg	1623.421	1575.597	1294.895	2120.418	1830.829	1673.736	2023.857	1631.836	2275.314	1652.146	1842.374	1469.379	1664.884
	Std	268.9201	206.3863	94.68404	305.596	342.168	214.9122	349.8226	288.4642	238.8184	284.0637	282.9398	109.1845	298.5812
	Rank	**8**	**6**	**1**	**21**	**15**	**12**	**19**	**9**	**22**	**10**	**16**	**5**	**11**
*F*60	Avg	1194.287	1112.292	1103.071	1119.06	1114.437	1115.826	1327.282	1118.399	1489.445	1135.722	1130.357	1106.786	1117.522
	Std	82.83907	7.66638	1.204941	13.85217	11.0854	5.260174	298.3339	16.22883	342.0761	21.88504	22.96259	3.574882	12.55076
	Rank	**18**	**7**	**3**	**13**	**9**	**10**	**21**	**12**	**22**	**16**	**15**	**6**	**11**
*F*61	Avg	62444.13	17107.4	1565.02	239022.3	14732.74	86820.85	1132383	15715.13	2563928	2399.172	75238.09	12480.87	378036.2
	Std	53354.94	14369.13	111.902	357410.5	12809.53	78655.71	1269509	15064.36	1710120	1626.524	335984	6983.591	540261
	Rank	**12**	**9**	**2**	**16**	**6**	**14**	**19**	**8**	**22**	**3**	**13**	**5**	**18**
*F*62	Avg	10569.79	4793.303	1311.011	13802.75	2938.143	10739.98	10827.2	5174.324	11248.51	1347.409	1665.766	4312.033	10545.17
	Std	11650.1	2755.747	2.046288	9337.524	1391.718	8995.853	3819.171	4525.821	2958.699	75.64731	662.619	2488.008	5857.996
	Rank	**15**	**9**	**2**	**21**	**7**	**16**	**17**	**10**	**18**	**3**	**5**	**8**	**14**
*F*63	Avg	1530.499	1423.729	1419.137	5149.824	1470.673	1759.822	6272.002	3631.739	1709.358	1416.366	1442.692	1465.031	2290.916
	Std	237.886	13.56827	5.20533	3794.42	21.10761	874.7007	3103.304	2541.138	295.7538	9.864172	18.60594	22.43568	2263.383
	Rank	**12**	**4**	**3**	**24**	**9**	**17**	**26**	**21**	**15**	**2**	**5**	**8**	**19**
*F*64	Avg	2500.385	1519.375	1502.346	4394.685	1666.342	1926.283	15650.27	3166.502	3455.548	1510.332	1539.068	1631.487	4041.162
	Std	1540.392	18.31846	0.673629	2786.728	122.5051	841.3384	8648.41	3199.511	2411.295	18.1888	23.4764	88.46547	3852.233
	Rank	**13**	**4**	**2**	**19**	**10**	**11**	**26**	**16**	**17**	**3**	**5**	**9**	**18**
*F*65	Avg	1673.861	1729.304	1609.976	1837.152	1685.951	1626.912	2018.107	1773.354	1744.394	1712.379	1731.055	1614.692	1718.296
	Std	84.38173	108.9607	14.41311	148.3391	111.1619	33.58275	113.5745	127.0501	91.78311	119.9915	82.16766	35.2002	122.5069
	Rank	**8**	**14**	**3**	**20**	**9**	**5**	**24**	**18**	**16**	**12**	**15**	**4**	**13**
*F*66	Avg	1754.199	1733.397	1711.154	1744.136	1736.947	1727.169	1817.574	1757.463	1776.373	1749.288	1746.181	1717.577	1732.755
	Std	42.91564	27.08818	6.691305	14.95226	19.38154	12.85099	74.60866	52.12585	17.85783	19.00683	19.61334	6.605743	24.64731
	Rank	**17**	**7**	**2**	**12**	**10**	**5**	**22**	**18**	**19**	**16**	**13**	**3**	**6**
*F*67	Avg	24657.55	5293.945	1821.141	14930.46	5773.317	11449.91	12152.03	8374.326	10432.07	1835.227	1999.234	5392.407	13225.5
	Std	15175.39	5075.86	0.372846	11380.95	4499.6	7809.646	7029.241	9156.106	4901.003	26.92495	353.8455	2596.009	10786.92
	Rank	**22**	**7**	**2**	**17**	**9**	**13**	**14**	**11**	**12**	**3**	**4**	**8**	**15**
*F*68	Avg	8214.177	1909.161	1902.02	7024.125	2040.777	4970.598	19797.36	3944.018	3731.652	1902.747	1927.017	2073.684	7241.917
	Std	7865.397	8.271678	0.283544	6345.979	158.7669	4659.91	20128.94	4059.141	1231.881	1.293865	21.57575	127.1339	5226.907
	Rank	**18**	**4**	**2**	**16**	**9**	**15**	**22**	**14**	**11**	**3**	**5**	**10**	**17**
*F*69	Avg	2024.611	2007.965	2003.626	2114.928	2037.091	2021.61	2178.582	2105.876	2065.81	2064.08	2061.557	2005.748	2035.954
	Std	10.35314	8.917933	3.518017	64.36423	52.05721	6.398866	84.11337	91.24301	17.88163	53.15708	36.95318	5.406635	42.70083
	Rank	**8**	**5**	**3**	**19**	**11**	**7**	**22**	**18**	**16**	**15**	**14**	**4**	**10**
*F*70	Avg	2306.558	2312.655	2287.522	2307.546	2231.547	2216.94	2306.7	2317.129	2264.08	2293.435	2287.934	2205.099	2290.298
	Std	46.54527	35.46436	46.42583	47.99783	52.75725	38.34587	1.956324	20.54887	38.32121	53.84695	55.84746	22.81764	51.78405
	Rank	**18**	**22**	**12**	**20**	**4**	**3**	**19**	**23**	**7**	**16**	**13**	**2**	**15**

(*continued on next page*)

little slower than the other algorithms but very good in avoiding the premature convergence. Based on these facts, we recommend GBO for problems requiring exploitative capabilities. Similarly, the performance of GBO on multimodal functions is also promising. GBO manages to find global optimum for 13 multimodal benchmarks, and its Friedman mean rank is 2nd on the list, as can be seen in Fig. 5. Considering the good performance of GBO, we also compare its convergence curve in Fig. 6. It is found that GBO has better convergence speeds, as seen for $F29$, $F30$, $F34$, and $F41$. However, there is no significant improvement after a certain point for $F26$, $F31$, $F33$, and $F40$, which indicates the

Table 11 (*continued*).

Fn	Stats	SMA	TFWO	SROA	BMO	MRFO	STSA	AEFA	MA	AOA	GNDO	PPA	FBI	BCMO
*F*71	Avg	2358.252	2348.082	2261.413	2296.823	2299.733	2310.974	2300.014	2299.956	2375.416	2302.155	2308.487	2281.122	2304.174
	Std	277.3053	220.7666	41.85207	17.05483	11.66904	17.91979	0.069578	15.48488	145.0803	22.55254	19.70045	26.64845	8.630459
	Rank	**19**	**17**	**1**	**4**	**7**	**16**	**9**	**8**	**20**	**12**	**15**	**3**	**14**
*F*72	Avg	2620.066	2628.964	2613.499	2623.92	2620.126	2634.795	2609.808	2623.625	2641.453	2626.668	2610.152	2605.564	2623.215
	Std	6.34414	10.56336	2.901299	10.46538	10.44512	5.867474	4.516451	8.245977	7.78314	11.60617	94.36833	45.85793	9.724722
	Rank	**10**	**17**	**6**	**14**	**11**	**18**	**3**	**13**	**19**	**16**	**4**	**2**	**12**
*F*73	Avg	2758.052	2762.607	2552.969	2717.308	2695.411	2674.986	2726.19	2753.65	2730.695	2738.595	2710.111	2603.523	2716.201
	Std	10.04007	12.65155	82.21613	87.65126	110.8951	119.034	41.40668	9.166319	56.67153	80.00159	108.7195	107.6309	96.87489
	Rank	**19**	**20**	**2**	**9**	**6**	**5**	**11**	**18**	**12**	**15**	**7**	**3**	**8**
*F*74	Avg	2935.518	2937.496	2897.8	2930.322	2922.728	2918.435	2946.968	2942.657	2952.638	2925.003	2932.144	2877.411	2921.915
	Std	30.41462	35.56216	0.276127	21.80103	23.58915	19.61096	7.38688	16.60653	8.334292	27.25783	22.87149	67.63731	24.02159
	Rank	**14**	**16**	**3**	**11**	**7**	**4**	**18**	**17**	**19**	**9**	**13**	**1**	**6**
*F*75	Avg	3106.959	3081.656	2731.123	2920.356	2936.977	2997.269	2937.559	3082.838	3239.804	3167.757	3042.16	2836.786	3013.46
	Std	374.4171	332.8232	98.18498	92.68205	83.06238	14.18849	290.7551	318.7536	221.5012	287.4697	186.4978	88.93525	108.1582
	Rank	**17**	**15**	**1**	**6**	**8**	**11**	**9**	**16**	**20**	**19**	**14**	**3**	**12**
*F*76	Avg	3091.083	3096.832	NA	3099.144	3104.352	3094.885	3109.091	3103.61	3126.966	3101.396	3114.938	3094.697	3106.775
	Std	1.679054	2.89576	NA	8.784461	14.79162	0.948936	17.84627	11.29683	23.33072	7.852265	19.08049	2.986285	14.49388
	Rank	**3**	**9**	NA	**10**	**16**	**8**	**18**	**15**	**20**	**13**	**19**	**7**	**17**
*F*77	Avg	3320.414	3272.96	NA	3304.373	3232.552	3227.357	3395.9	3301.824	3297.503	3269.79	3244.509	3067.869	3250.528
	Std	170.6187	174.8919	NA	128.0299	143.0811	57.60398	23.62254	139.3344	4.739379	139.7099	140.6487	114.1577	131.7452
	Rank	**16**	**11**	NA	**14**	**5**	**4**	**19**	**13**	**12**	**10**	**7**	**1**	**9**
*F*78	Avg	3188.048	3218.736	NA	3279.103	3194.944	3181.361	3300.69	3251.31	3253.096	3171.512	3214.621	3183.755	3191.985
	Std	47.04296	52.78331	NA	60.60623	32.28143	6.799227	107.0045	74.85839	13.37302	27.31913	58.81435	12.71354	24.74181
	Rank	**9**	**15**	NA	**19**	**11**	**6**	**20**	**17**	**18**	**4**	**13**	**7**	**10**
*F*79	Avg	254590.4	239203.3	NA	563419.1	282140.4	54133.74	2602977	362337.6	84781.42	381071.1	348602.1	17465.09	357497.1
	Std	463442.3	356492.5	NA	915014	438302.4	30431.26	2555397	502094.9	122715.9	707056.5	563569.1	18628.79	536664.7
	Rank	**7**	**6**	NA	**17**	**9**	**4**	**21**	**12**	**5**	**13**	**10**	**2**	**11**

Table 12
Comparison of the performance of all metaheuristics for engineering optimization problems.

Speed-reducer design problem

	HBO	**PO**	**GPC**	**GBO**	**MPA**	**LFD**	**TSA**	**BWOA**	**ROA**	**ChOA**	**TSO**
Avg	2994.471	2997.263	9.08E+12	2994.471	3003.7	1.46E+17	3066.283	1.17E+11	9.32E+10	3207.645	3.59E+12
Std	1.83E−07	9.730802	1.47E+13	0.000249	4.900553	6.39E+15	20.45511	2.86E+11	2.06E+11	38.28001	9.57E+12
Rank	1	6	21	2	9	22	12	19	18	14	20
	EO	**SMA**	**TFWO**	**MRFO**	**STSA**	**AEFA**	**MA**	**GNDO**	**PPA**	**FBI**	**BCMO**
Avg	3002.978	2995.331	2996.163	3062.041	3008.595	3600.423	2994.472	3079.201	6.76E+09	3407.515	3001.449
Std	9.248234	0.772008	7.902031	16.85486	5.745024	631.2908	0.002891	37.41254	3.38E+10	283.801	3.500226
Rank	8	4	5	11	10	16	3	13	17	15	7

Tension/compression spring design problem

	HBO	**PO**	**GPC**	**GBO**	**MPA**	**LFD**	**TSA**	**BWOA**	**ROA**	**ChOA**	**TSO**
Avg	0.012846	0.013732	3.58E+18	0.012743	0.01269	6.38E+15	0.013381	8.44E+18	0.021939	0.014448	0.01395
Std	0.000186	0.001399	6.49E+18	7.11E−05	3.44E−05	3.19E+16	0.000653	2.37E+19	0.005192	0.001668	0.000982
Rank	3	12	20	2	1	19	8	22	18	14	13
	EO	**SMA**	**TFWO**	**MRFO**	**STSA**	**AEFA**	**MA**	**GNDO**	**PPA**	**FBI**	**BCMO**
Avg	0.013093	0.015481	0.013453	0.013536	0.012993	4.42E+18	0.012863	0.013539	0.014695	0.014466	0.013304
Std	0.000626	0.001886	0.001416	0.000616	0.000613	7.64E+18	0.000203	0.000918	0.002535	0.000608	0.000924
Rank	6	17	9	10	5	21	4	11	16	15	7

Rolling-element bearing design problem

	HBO	**PO**	**GPC**	**GBO**	**MPA**	**LFD**	**TSA**	**BWOA**	**ROA**	**ChOA**	**TSO**
Avg	−85533.2	−82111.9	−81147.2	−85454.4	−85417.7	65535	−83341.8	−26407.8	4E+20	−73648.8	−79930.4
Std	0.071121	6534.939	2943.444	248.1417	209.3727		1316.682	13120.36	7.78E+20	7406.883	3140.749
Rank	1	9	11	3	4	18	8	17	21	15	13
	EO	**SMA**	**TFWO**	**MRFO**	**STSA**	**AEFA**	**MA**	**GNDO**	**PPA**	**FBI**	**BCMO**
Avg	−85106.5	−84753.7	−81460	−73795.3	−84638.2	1.53E+21	−85471.4	−52848.4	2.26E+20	4.57E+18	−80283.5
Std	380.3849	2430.003	4311.156	3331.534	317.4404	2.12E+21	222.9589	14448.35	5.93E+20	1.56E+19	3469.081
Rank	5	6	10	14	7	22	2	16	20	19	12

Multiple disk-clutch brake design problem

	HBO	**PO**	**GPC**	**GBO**	**MPA**	**LFD**	**TSA**	**BWOA**	**ROA**	**ChOA**	**TSO**
Avg	0.314911	0.319327	0.374623	0.320425	0.350633	3.7E+13	0.331565	0.404669	NA	0.395021	0.451962
Std	0.003529	0.010346	0.04115	0.017349	0.04523	2.21E+12	0.036396	0.063555	NA	0.034318	0.08019
Rank	2	4	13	5	11	21	8	15	NA	14	16
	EO	**SMA**	**TFWO**	**MRFO**	**STSA**	**AEFA**	**MA**	**GNDO**	**PPA**	**FBI**	**BCMO**
Avg	0.328355	0.333863	0.324307	0.374201	0.313657	0.600283	0.319302	0.695313	0.50994	0.561302	0.33544
Std	0.018846	0.02554	0.012727	0.037894	1.7E−16	0.115077	0.00872	0.138565	0.109947	0.161422	0.022122
Rank	7	9	6	12	1	19	3	20	17	18	10

unbalanced behaviour of GBO. It is also evident from its performance on the CEC-BC-2017 complex benchmarks. According to the Friedman test presented in Fig. 7, GBO ranks 9th. A very good performance on unimodal and multimodal benchmarks and comparatively low performance on complex benchmarks indicates that GBO does not have very good balance among exploration and exploitation. We encourage the research community to take this as a future direction and improve GBO from this perspective. Finally, the performance of GBO on engineering problems is also very promising and according Friedman test, it is ranked as 2nd best algorithm after HBO.

Based on the above facts, GBO is highly recommended for engineering and other optimization problems; however, its balance on complex problems needs to be improved. Hence, the balance enhancement of GBO may be considered as a future direction.

5.5. Performance of RDA

As shown from the results of the unimodal functions in Table 2, RDA is one of the worst performing algorithms. It could hardly manage to reach the top 10 for any of the benchmarks. According to the Friedman test, its rank is 23rd on the list. Furthermore, its performance on multimodal functions is one of the worst. According to the Friedman test, its rank is 25th in the list. However, the performance of RDA is remarkably better on the complex benchmarks of CEC-2017. Its position, according to Friedman mean rank test is 6th in the list, which is opposite of its previous performances.

Based on the above findings, we conclude that RDA has a very good balance among exploration and exploitation but needs serious improvements on its exploitation and exploration capabilities. This may be the limitation of underlaying mathematical model; however, we recommend this as open direction for researchers to investigate.

5.6. Performance analysis of MPA

The performance of MPA on unimodal functions is above average and ranks in top ten in most of the cases. Thus, it secures the 8th position in the ranking list generated using the Friedman test. Please see Table 6 and Fig. 3 for the reference. Its performance on multimodal functions is even better, and according to Friedman test, its position is 5th on the list. Although, the performance of MPA on unimodal and multimodal functions is above average, it is not extraordinary; however, its results on CEC-BC-2017 are superior because it outperforms all competitors and secures 1st position in list generated by Friedman test. Please see Table 10 and Fig. 7 for the reference. Finally, the performance of MPA on engineering problems is also above average. On the tension/compression spring design problem, it outperforms all other algorithms.

Based on the above discussion, we highly recommend MPA for complex real-world problems requiring good balance among exploration and exploitation; however, we also suggest that researchers work on enhancing its exploration and exploitation capabilities.

5.7. Performance analysis of LFD

The exploitation capability of LFD, as demonstrated in Table 6, is found to be very poor. Except for $F16$ and $F21$, LFD does not rank among the top 10 positions. According to the Friedman test, the rank of LFD is 20th in the list. Its performance on multimodal functions is somewhat better but still below average. The algorithm is in the 18th position on the rank list generated by the Friedman test. Hence, we can conclude the overall exploration and exploitation capabilities of LFD need serious improvement. When we consider the case of complex functions, the performance of LFD is worse. The position of LFD in rank list generated by the Friedman test is 23rd, which is 3 positions above the last position. For the case of the engineering problem, LFD is found to have the worst performance. Based on this discussion, we conclude that LFD is truly very weak from all aspects.

5.8. Performance analysis of TSA

The performance of TSA on the individual level is sufficient for many unimodal functions, but its comparative performance is below average. According to the Friedman test, TSA ranks 19th position for unimodal functions. The results and ranking both can be seen in Table 6 and Fig. 3, respectively. On the other hand, its performance on multimodal functions is much worse. The position of TSA on the rank list generated by the Friedman test is 23rd. Furthermore, its poor performance is also noted on CEC-BC-2017 benchmarks, where the rank of TSA is 24 of 26. However, the performance of TSA on engineering problems can be rated as average because it is ranked 11th. Based on these results, we declare that TSA is a below-average algorithm with poor exploration and exploitation capabilities. Furthermore, its balance between exploration and exploitation requires serious improvement.

5.9. Performance analysis of BWOA

The performance of BWOA is found to be below average compared to the other algorithms regardless the underlying benchmark set used for the evaluation. According to the Friedman mean rank test, BWOA ranks 25th in the case of unimodal functions, 17th in the case of multimodal functions, 19th in the case of CEC-BC-2017 benchmarks, and 19th on engineering optimization problems. It is all evident from the results presented in previous section. Based on these results, we conclude that BWOA has comparatively poor exploitative and exploratory capabilities. Furthermore, its balance between exploration and exploitation is weak. There are many improvement potentials for this algorithm. This algorithm can be chosen for future study with an ultimate goal of improving its exploration and exploitation capabilities.

5.10. Performance analysis of ROA

In all experiments, regardless of the benchmark set used for evaluation, the worst performance is achieved using ROA. The rank of ROA on unimodal functions is 26th (last), on multimodal functions is 26th (last), on CEC-BC-2017 benchmarks is 26th (last), and on engineering problems is second to last. The reason behind its poor performance is obviously its poor optimization capabilities. A very good future direction will be to critically analyse the behaviour and mathematical model of ROA and address the root cause of this behaviour.

5.11. Performance analysis of ChOA

The comparative performance of ChOA is also not very promising in most of the cases. ChOA attains the 17th position on unimodal functions, 24th position on multimodal functions, 22nd position on complex functions of CEC-BC-2017, and 13th position on engineering problems. The overall performance is rated as below average; however, on engineering problems the performance ChOA is average level. Improving ChOA from the perspectives of exploration and exploitation can be a potential future direction for researchers.

5.12. Performance analysis of TSO

The performance of TSO on unimodal as well as multimodal functions mostly falls among the top ten algorithms. The rank of this algorithm on unimodal functions is 10th and on multimodal functions is 11th. It should be noted that the performance of TSO at the individual level is good; however, in comparison to the other algorithms, its performance is a little lower. We can conclude from this that the exploration and exploitation capabilities of TSO are sufficient, but there is still potential for improvement. In contrast, the performance drastically falls on the CEC-BC-2017 benchmarks. The rank of TSO on CEC functions is 25th, which shows that TSO has very poor balance between exploration and exploitation. Finding the cause of this behaviour and improving the balance between exploration and exploitation can be future directions for the research community.

5.13. Performance analysis of EO

The performance EO in part of the top 10 for most of the functions regardless of the type of benchmarks. This indicates that this algorithm has some good qualities in terms of its exploration and exploitation capabilities. It attains 7th, 5th, 5th, and 6th positions against unimodal functions, multimodal functions, CEC-BC-2017 benchmarks, and engineering problems, respectively. Its consistent performance and above average rank in all cases show that EO has comparatively better balance than many other algorithms. However, EO does not outperform other algorithms in any case, which shows its potential for improvement. Based on its consistent and above average performance, we recommend EO for different types of problems; however, we also recommend that researchers investigate the areas that can be improved to increase the performance of this algorithm.

5.14. Performance analysis of SMA

As presented in Tables 7 and 9, the overall performance of SMA on unimodal and multimodal functions is very good, and it is ranked 4th for unimodal as well as multimodal functions. In both cases, SMA is among the top 10 algorithms for most functions. However, its performance on CEC-BC-2017 drops significantly, and SMA is ranked 14th on the rank table generated by the Friedman test. Similarly, its performance on engineering problems is also of an average level, as SMA is ranked 9th on the list. We may conclude from this discussion that SMA can be a very good option for simple problems; however, it struggles for complex problems, which may be due to its unbalanced behaviour. SMA can be improved further by working on its balance between exploration and exploitation. In addition, the convergence behaviour of SMA is also assessed, and it is found that SMA has some capability to escape local optima but converges prematurely in few cases, as presented in Figs. 4 and 6.

5.15. Performance analysis of TFWO

The performance of TFWO is average for unimodal and multimodal functions, as shown in Tables 7 and 9. For 8 unimodal and 13 multimodal functions, TFWO ranks among the top ten algorithms. According to the Friedman mean rank test, TFWO attains 15th position for unimodal functions and 12th position for multimodal functions. This means that compared with the other algorithms, TFWO has an average level of exploration and exploitation capabilities. In contrast, its performance on CEC-BC-2017 and engineering problems is much improved, and the rank of TFWO is 8th on CEC benchmarks and 7th on engineering problems. Hence, we can conclude that TFWO has better balance between exploration and exploitation and can be used for complex or engineering problems. We suggest working on its exploration and exploitation capabilities to improve the balance between exploration and exploitation.

5.16. Performance analysis of SROA

As presented in Tables 7, 9, and 11, SROA has average performance on unimodal and multimodal functions, while its performance on CEC-BC-2017 benchmarks is truly very good. According to the Friedman test, it attains the 12th, 13th, and 4th positions for unimodal, multimodal, and CEC benchmarks, respectively. The clear difference among the ranks for CEC benchmarks and the other two types of functions shows that SROA has balanced exploration and exploitation but both capabilities individually need some improvements to compete with other available algorithms. This analysis open up a research direction for the research community to work on mathematical model of SROA to further improve its exploration and exploitation capabilities at the individual level.

5.17. Performance analysis of BMO

For most unimodal and multimodal benchmark functions, BMO ranks among the top algorithms. Its Friedman mean rank for unimodal and multimodal functions are 5th and 7th, respectively. It is evident from the results that the exploration and exploitation capabilities at the individual level are much better; however, the performance deteriorates for CEC-BC-2017 benchmarks, as shown in Table 11 and Fig. 7. Its rank of 18 for CEC-BC2017 benchmarks indicates that BMO has comparatively poor balance between exploration and exploitation. The algorithm may perform even better if the balance between exploration and exploitation is improved.

5.18. Performance analysis of MRFO

In terms of overall performance, MRFO is one of the best algorithms. There is hardly a unimodal or multimodal function for which MRFO is not among the top ten algorithms. According to the Friedman test, MRFO attains 3rd position for unimodal functions, 3rd position for multimodal functions, and 7th position for CEC-BC-2017 benchmark functions; however, its performance on engineering problems is average, and the rank attained by this algorithm for engineering problems is 12th. Considering that MRFO yields good performance, the convergence curves are also plotted, as shown in Figs. 4 and 6. In both figures, it can be seen that MRFO has a good convergence speed in most cases; however, it falls in local optima in few multimodal cases. Based on our findings, we recommend MRFO for diverse-natured optimization problems.

5.19. Performance analysis of STSA

The performance of STSA is average. On unimodal functions, it is ranked 11th. On multimodal function, it ranks 16th, which may also be considered average performance. However, on the CEC-BC-2017 benchmarks, its performance is above average, and it attains the 9th position on the list. Similarly, its performance on engineering problems is also promising, as it attains 4th position in the rank list generated using the Friedman test. Based on these comparative results, STSA may be considered a good candidate for engineering problems, but the algorithm needs some improvements in its mathematical model to perform well on problems from different domains.

5.20. Performance analysis of AEFA

AEFA is found to be one of the worst performers in most cases. The Friedman ranks of this algorithm for unimodal, multimodal, CEC-BC-2017 functions, and engineering problems are 24th, 21st, 17th, and 24th, respectively. We can conclude from these results that the exploration and exploitation capabilities along with the required balance between exploration and exploitation are very weak, and serious changes are required for improvement.

5.21. Performance analysis of MA

Based on the results presented in Tables 7, 9, and 11, we can conclude that the performance of MA is average. This algorithms attains 16th, 17th, and 15th ranks on the unimodal, multimodal, and CEC-BC-2017 benchmark functions. However, the performance of MA on engineering problems is remarkably better, and it attains the 2nd position according to the ranks generated by the Friedman mean rank test. Based on these results, we recommend MA for engineering constrained optimization problems; however, to obtain the best results in diverse areas, the algorithm needs improvements in its exploration and exploitation capabilities, which may be considered a future direction.

5.22. Performance analysis of AOA

On unimodal functions, AOA shows some good performance and attains the 9th position in the ranking list of all 26 algorithms. However, on multimodal functions, the performance of AOA is average, and it attains the 15th position according to the Friedman test. We can conclude that this algorithm has better exploitation capabilities, but its exploration capabilities need to be improved. Furthermore, its exploration and exploitation balancing capabilities are below average, as it attains the 20th position on the list. This means that AOA does not have a good balance between exploration and exploitation. A potential research direction may be to address these issues to obtain better performance when using this algorithm.

5.23. Performance analysis of GNDO

The performance of GNDO is found to be poor on unimodal and multimodal functions. It ranks 22nd and 20th on unimodal and multimodal functions, respectively. These results show that GNDO needs some serious revisions in its mathematical model to perform well on these benchmarks. However, on the CEC-BC-2017 benchmarks and engineering problems, the performance of GNDO is of average level, and it ranks 12th and 13th on these functions and problems, respectively.

5.24. Performance analysis of PPA

According to the rank list generated by the Friedman test, PPA attains the 21st, 20th, 11th, and 22nd positions on the unimodal, multimodal, CEC-BC-2017 benchmarks, and engineering problems, respectively. These ranks show that PPA has comparatively poor exploration and exploitation capabilities. The balance between exploration and exploitation is not that bad, which is evident from the results of CEC-BC-2017 benchmark functions.

5.25. Performance analysis of BCMO

The overall performance of this algorithm is found to be average. According to the Friedman test, BCMO attains the 13th, 10th, 13th, and 9th positions on the unimodal, multimodal, CEC-BC-2017 benchmarks, and engineering constrained optimization problems, respectively. Based on these results, we suggest improvements in the mathematical model of this algorithm to improve its exploration and exploitation capabilities.

5.26. Performance analysis of FBI

According to the results, FBI is found to be one of the best algorithms compared in this study. As shown in Tables 7, 9, 11, FBI is among the top 10 algorithms for most of the unimodal, multimodal, and CEC-BC-2017 benchmark functions. According to the Friedman mean rank test, it attains the 6th, 8th, and 2nd positions on these sets of benchmarks, respectively. The algorithm demonstrates good exploitation and exploration capabilities, especially the 2nd position on CEC-BC-2017 benchmarks shows that algorithm has very good balance between exploration and exploitation. However, the weakness of this algorithm is shown on constrained engineering optimization problems, where it attains the 17th position, which represents a below average performance. Based on these findings, we recommend this algorithm for unconstrained problems; however, we also recommend that the research community makes appropriate changes in this algorithm to make it suitable for constrained optimization problems, which may be considered an open research direction.

5.27. Summary of the discussion

In this section, different types of benchmark functions and engineering problems are used to evaluate the optimization capabilities of 26 algorithms. The Friedman mean rank test is performed to rank the algorithms in each category. Moreover, the convergence behaviour is analysed by comparing the convergence curves of the top 5 algorithms for selected unimodal and multimodal benchmarks. The top five algorithms from each category, along with their ranks in each category, are presented in Table 13. By comparing the ranks of algorithms in all categories, GBO is declared to be the best optimizer because its rank 1st in one category and 2nd in two categories. MPA, which ranks 2nd overall, ranks 1st in one category and 5th in two categories. The overall third best algorithm is PO, which ranks 1st in one category and 2nd in another category. The fourth best algorithm is HBO, which ranks 1st in one category and 3rd in another category. Finally, MRFO may be declared the fifth-best algorithm because it ranks 3rd in two categories.

Table 13
Comparison of the ranking of the top 5 algorithms for each type of benchmark.

Algorithm	Unimodal functions	Multimodal functions	CEC-BC-2017 functions	Engineering problems
GBO	1	2		2
MPA		5	1	5
PO	2	1		
HBO			3	1
MRFO	3	3		
SMA	4	4		
FBI			2	
MA				2
SROA			4	
STSA				4
BMO	5			
EO			5	

Based on the overall rankings and category-based performances, we make the following observations and recommendations:

- GBO is an algorithm with excellent exploitation and exploration capabilities but slightly unbalanced behaviour for complex landscapes. We recommend improving GBO by improving the balance between exploration and exploitation.
- MPA has an excellent balance between exploration and exploitation at the expense of exploitation capability and convergence speed. We recommend enhancing the convergence speed and exploitation capability of MPA by making appropriate changes in the position updating mechanism of this algorithm.
- PO has an excellent exploitation capability and outstanding convergence speed. However, it may suffer from premature convergence for complex landscapes due to an insufficient time for exploration. It is recommended to enhance the duration of the exploration phase and better control the transition between exploration and exploitation.
- HBO demonstrates a good balance between exploration and exploitation, which is evident from its performance for CEC-BC-2017 functions and engineering problems, but it lacks a fast convergence speed in an attempt to attain balance. Although HBO has some mechanism to incorporate exploitation, the convergence speed needs to be improved by making appropriate changes in the position updating mechanism of this algorithm.
- MRFO has good exploration and exploitation capabilities, but compared to GBO and PO, it is weaker. Moreover, its performance is also affected by complex landscapes, which raises questions about its capability to balance exploration and exploitation. We recommend that its capability to balance exploration and exploitation be improved.

Based on the above findings and observations, we highly recommend GBO, MPA, PO, and HBO to solve different real-world optimization problems.

6. Conclusion

In this paper, a review of recent developments in the field of metaheuristics is presented. The optimization capability of 26 recently published novel metaheuristics is analysed. Based on this analysis, the top five algorithms with better exploitation capability, top five algorithms with better exploration capability, top five algorithms with better balance among exploration and exploitation, and top algorithms with better capability to solve constrained engineering problems are identified. Ultimately, a few observations in terms of the weaknesses and strengths of the top five algorithms from each category are stated, and based on these observations, some recommendations to further improve these algorithms are made.

By thoroughly searching the literature and narrowing the relevant articles, 57 novel metaheuristics are identified. Based on the availability of the source code, the performance of 26 metaheuristics is evaluated by using 79 benchmarks, including 25 unimodal, 25 multimodal, and 29 CEC-BC-2017 benchmarks. Moreover, four constrained engineering problems are also used to evaluate the applicability of the algorithms to real-world problems. By using the Friedman's mean rank test, GBO, PO, MRFO, and SMA are found to be the top five performers for unimodal functions, and PO, GBO, MRFO, SMA, and MPA are found to be the top five performers for multimodal functions. Moreover, MPA, FBI, HBO, SROA, and EO show more balanced behaviour, which is evident from the results for CEC-BC-2017. Finally, HBO, GBO, MA, STSA, and MPA demonstrate the best performance for engineering problems.

Based on the accumulative performance, GBO, MPA, PO, and HBO are found to be the top four algorithms and are recommended as potential candidates for real-world optimization problems. Moreover, it is recommended that the performance of GBO and PO be further improved by improving the balance between exploration and exploitation. In addition, the performance of MPA and HBO may be improved by enhancing convergence speed and improving exploitation capability.

Declaration of competing interest

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

Data availability

I will publish my code and data on my GitHub account

Acknowledgements

The researcher(s) would like to thank the Deanship of Scientific Research, Qassim University, Saudi Arabia for funding the publication of this project. In addition, researchers would like to thank the Deanship of Scientific Research, Qassim University for the support of this project.

Appendix

See Table A.14.

Table A.14
List of abbreviations used in the paper.

Abbreviation	Definition
ACO	Ant colony optimization
SA	Simulated annealing
GA	Genetic algorithm
EP	Evolutionary programming
ES	Evolutionary strategy
GP	Genetic programming
AA	Ant algorithm
DE	Differential evolution
PSO	Particle swarm optimization
ABC	Artificial bee colony
FA	Firefly algorithm
CS	Cuckoo search
BA	Bat algorithm
HS	Harmony search
SCOA	Social cognitive optimization algorithm
SC	Society and civilization
BBBC	Big bang-big crunch
ICA	Imperialist competitive algorithm
GSA	Gravitational search algorithm
TLBO	Teaching learning-based optimization
FPA	Flower pollination algorithm
SLCA	Soccer league competition algorithm
GWO	Grey wolf optimizer
WOA	Whale optimization algorithm
SCA	Sine cosine algorithm
VPSA	Vibrating particles system algorithm
TGA	Tree growth algorithm
CPA	Cyclical parthenogenesis algorithm
WEA	Water evaporation algorithm
TEOA	Thermal exchange optimization algorithm
SHO	Spotted hyena optimizer
HMSA	Human mental search algorithm
FFA	Farmland fertility algorithm
EOA	Earthworm optimisation algorithm
RHA	Rhino herd algorithm
QSA	Queuing search algorithm
CTOA	Car tracking optimization algorithm
SDMP	Self-defense mechanism of the plants
COA	Coyote optimization algorithm
SqSA	Squirrel search algorithm
SOA	Seagull optimization algorithm
HBIA	Hitchcock bird-inspired algorithm
SLnO	Sea lion optimization
EPC	Emperor penguins colony
SFO	Sailfish optimizer
BES	Bald eagle search
NMR	Naked mole-rat
BOA	Butterfly optimization algorithm
HHO	Harris hawks optimization
LGSI	Ludo game-based swarm intelligence
PRO	Poor and rich optimization
ExA	Expectation algorithm
SMIA	Social media inspired algorithm
SDO	Supply-demand-based optimization
NPO	Nomadic people optimizer
SMO	Social mimic optimization
F3EA	Find-fix-finish-exploit-analyze
DHOA	Deer hunting optimization algorithm
BMO	Barnacles mating optimizer
BSSA	Bear smell search algorithm
BWOA	Black widow optimization algorithm
CCLA	Caledonian crow learning algorithm
ChOA	Chimp optimization algorithm
CvOA	Coronavirus optimization algorithm
DGBCO	Dynamic group-based cooperative optimization

(*continued on next page*)

Table A.14 (*continued*).

Abbreviation	Definition
MRFO	Manta ray foraging optimization
MPA	Marine predators algorithm
MA	Mayfly algorithm
PFDIBI	Parallel fully dynamic iterative bio-inspired
PPA	Parasitism predation algorithm
RDA	Red deer algorithm
SSOM	Shuffled shepherd optimization method
STSA	Sine tree-seed algorithm
SMA	Slime mould algorithm
TSA	Tunicate swarm algorithm
WSA	Water strider algorithm
BIOA	Billiards-inspired optimization algorithm
DGO	Darts game optimizer
KKOA	Kho-kho optimization algorithm
HOGO	Hide objects game optimization
AOA	Archimedes optimization algorithm
AEFA	Artificial electric field algorithm
BCMO	Balancing composite motion optimization
BHMO	Black hole mechanics optimization
DDAO	Dynamic differential annealed optimization
EO	Equilibrium optimizer
FNATRM	Filter nonmonotone adaptive trust region method
GNDO	Generalized normal distribution optimization
GBO	Gradient-based optimizer
GPE	Grey prediction evolution
LMQA	Limited memory Q-BFGS algorithm
MSA	Momentum search algorithm
NMA	Newton metaheuristic algorithm
PSA	Photon search algorithm
QIA	Quantum-inspired algorithm
TSO	Transient search optimization
TFWO	Turbulent flow of water-based optimization
VLE	Vapor-liquid equilibrium
VEA	Volcano eruption algorithm
AISA	Adolescent identity search algorithm
FBI	Forensic-based investigation
GPC	Giza pyramids construction
GTOA	Group teaching optimization algorithm
HBO	Heap-based optimizer
HDOA	Human dynasties optimization algorithm
HUA	Human urbanization algorithm
IAS	Interactive autodidactic school
PO	Political optimizer
SROA	Search and rescue optimization algorithm
GO	Group optimization
LFD	Lévy flight distribution
CHA	Color harmony algorithm
OHDO	Opposition-based high dimensional optimization
OAOP	Optimization algorithm based on OCM and PCM
ROA	Rain optimization algorithm
CSO	Cat swarm optimization
CSA	Crow search algorithm
SE	Social effect
MU	Maximum unsuccessful search number
WSNs	Wireless sensor networks

References

Abdollahzadeh, B., Gharehchopogh, F.S., Mirjalili, S., 2021. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. Comput. Ind. Eng. 158, 107408.

Abdollahzadeh, B., Soleimanian Gharehchopogh, F., Mirjalili, S., 2021b. Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. Int. J. Intell. Syst. 36 (10), 5887–5958.

Abualigah, L., Abd Elaziz, M., Sumari, P., Geem, Z.W., Gandomi, A.H., 2022. Reptile search algorithm (RSA): A nature-inspired meta-heuristic optimizer. Expert Syst. Appl. 191, 116158.

Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., Gandomi, A.H., 2021a. The arithmetic optimization algorithm. Comput. Methods Appl. Mech. Engrg. 376, 113609.

Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A.A., Al-Qaness, M.A., Gandomi, A.H., 2021b. Aquila optimizer: A novel meta-heuristic optimization algorithm. Comput. Ind. Eng. 157, 107250.

Agushaka, J.O., Ezugwu, A.E., Abualigah, L., 2022. Dwarf mongoose optimization algorithm. Comput. Methods Appl. Mech. Engrg. 391, 114570.

Ahmadianfar, I., Bozorg-Haddad, O., Chu, X., 2020. Gradient-based optimizer: A new metaheuristic optimization algorithm. Inform. Sci. 540, 131–159.

Al-Sorori, W., Mohsen, A.M., 2020. New caledonian crow learning algorithm: A new metaheuristic algorithm for solving continuous optimization problems. Appl. Soft Comput. 106325.

Alsattar, H.A., Zaidan, A.A., Zaidan, B.B., 2019. Novel meta-heuristic bald eagle search optimisation algorithm. Artif. Intell. Rev..

Arora, S., Singh, S., 2018. Butterfly optimization algorithm: A novel approach for global optimization. Soft Comput. 23 (3), 715–734.

Arslan, H., 2020. A parallel fully dynamic iterative bio-inspired shortest path algorithm. Arab. J. Sci. Eng..

Askari, Q., Saeed, M., Younas, I., 2020a. Heap-based optimizer inspired by corporate rank hierarchy for global optimization. Expert Syst. Appl. 161, 113702.

Askari, Q., Younas, I., Saeed, M., 2020b. Political optimizer: A novel socio-inspired meta-heuristic for global optimization. Knowl.-Based Syst. 105709.

Askarzadeh, A., 2016. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Comput. Struct. 169, 1–12.

Atashpaz-Gargari, E., Lucas, C., 2007. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In: 2007 IEEE Congress on Evolutionary Computation. IEEE.

Attik, M., Bougrain, L., Alexandre, F., 2005. Neural network topology optimization. In: International Conference on Artificial Neural Networks. Springer, pp. 53–58.

Awad, N., Ali, M., Liang, J., Qu, B., Suganthan, P., 2017. Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization. In: 2017 IEEE Congress on Evolutionary Computation. CEC, IEEE Congr. Evol. Comput.

Azcárate, C., Mallor, F., Gafaro, A., 2008. Multiobjective optimization in health care management. A metaheuristic and simulation approach. Algorithmic Oper. Res. 3 (2).

Bäck, T., Schwefel, H.-P., 1993. An overview of evolutionary algorithms for parameter optimization. Evol. Comput. 1 (1), 1–23.

Balochian, S., Baloochian, H., 2019. Social mimic optimization algorithm and engineering applications. Expert Syst. Appl. 134, 178–191.

Berwick, R., 2003. An idiot's guide to support vector machines (SVMs). Retrieved on October 21, 2011.

Bianchi, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J., 2008. A survey on metaheuristics for stochastic combinatorial optimization. Nat. Comput. 8 (2), 239–287.

Biegler, L.T., Lang, Y.-d., Lin, W., 2014. Multi-scale optimization for process systems engineering. Comput. Chem. Eng. 60, 17–30.

Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35 (3), 268–308.

Bogar, E., Beyhan, S., 2020. Adolescent identity search algorithm (AISA): A novel metaheuristic approach for solving optimization problems. Appl. Soft Comput. 95, 106503.

Boussaïd, I., Lepagnot, J., Siarry, P., 2013. A survey on optimization metaheuristics. Inform. Sci. 237, 82–117.

Brammya, G., Praveena, S., Preetha, N.S.N., Ramya, R., Rajakumar, B.R., Binu, D., 2019. Deer hunting optimization algorithm: A new nature-inspired meta-heuristic paradigm. In: Rosaci, D. (Ed.), Comput. J..

Brand, M., Masuda, M., Wehner, N., Yu, X.-H., 2010. Ant colony optimization algorithm for robot path planning. In: 2010 International Conference on Computer Design and Applications, vol. 3. IEEE, pp. V3–436.

Caraveo, C., Valdez, F., Castillo, O., 2018. A new optimization meta-heuristic algorithm based on self-defense mechanism of the plants with three reproduction operators. Soft Comput. 22 (15), 4907–4920.

Chen, J., Cai, H., Wang, W., 2018. A new metaheuristic algorithm: Car tracking optimization algorithm. Soft Comput. 22 (12), 3857–3878.

Cheraghalipour, A., Hajiaghaei-Keshteli, M., 2017. Tree growth algorithm (TGA): An effective metaheuristic algorithm inspired by trees behavior. In: 13th International Conference on Industrial Engineering, vol. 13. Scientific Information Databases Babolsar.

Chou, J.-S., Nguyen, N.-M., 2020. FBI inspired meta-optimization. Appl. Soft Comput. 106339.

Chu, S.-C., Tsai, P.-W., Pan, J.-S., 2006. Cat swarm optimization. In: Pacific Rim International Conference on Artificial Intelligence. Springer, pp. 854–858.

Crawford, B., Soto, R., Cabrera, G., Salas-Fernández, A., Paredes, F., 2019. Using a social media inspired optimization algorithm to solve the set covering problem. In: International Conference on Human-Computer Interaction. Springer, pp. 43–52.

Deb, K., 1991. Optimal design of a welded beam via genetic algorithms. AIAA J. 29 (11), 2013–2015.

Dehghani, M., Montazeri, Z., Dehghani, A., Malik, O.P., 2020. GO: Group optimization. Gazi Univ. J. Sci. 33 (2), 381–392.

Dehghani, M., Montazeri, Z., Givi, H., Guerrero, J.M., Dhiman, G., 2020a. Darts game optimizer: A new optimization technique based on darts game. Int. J. Intell. Eng. Syst 13, 286–294.

Dehghani, M., Montazeri, Z., Saremi, S., Dehghani, A., Malik, O.P., Al-Haddad, K., Guerrero, J.M., 2020b. HOGO: Hide objects game optimization. Int. J. Intell. Eng. Syst. 13 (10).

Dehghani, M., Samet, H., 2020. Momentum search algorithm: A new meta-heuristic optimization algorithm inspired by momentum conservation law. SN Appl. Sci. 2 (10), 1–15.

Dhiman, G., Kumar, V., 2017. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. Adv. Eng. Softw. 114, 48–70.

Dhiman, G., Kumar, V., 2019. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. Knowl.-Based Syst. 165, 169–196.

Dorigo, M., 1992. Optimization, Learning and Natural Algorithms (Ph.D. thesis). Politecnico di Milano, Italy.

Dorigo, M., Caro, G.D., 1999. Ant colony optimization: A new meta-heuristic. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). IEEE.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: Optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. B 26 (1), 29–41.

Erol, O.K., Eksin, I., 2006. A new optimization method: Big bang–big crunch. Adv. Eng. Softw. 37 (2), 106–111.

Faramarzi, A., Heidarinejad, M., Mirjalili, S., Gandomi, A.H., 2020a. Marine predators algorithm: A nature-inspired metaheuristic. Expert Syst. Appl. 113377.

Faramarzi, A., Heidarinejad, M., Stephens, B., Mirjalili, S., 2020b. Equilibrium optimizer: A novel optimization algorithm. Knowl.-Based Syst. 191, 105190.

Fathollahi-Fard, A.M., Hajiaghaei-Keshteli, M., Tavakkoli-Moghaddam, R., 2020. Red deer algorithm (RDA): A new nature-inspired meta-heuristic. Soft Comput. 1–29.

Fogel, L.J., 1962. Autonomous automata. Ind. Res. 4, 14–19.

Fogel, L.J., 1964. On the Organization of Intellect (Ph.D. thesis). University of California, Los Angeles–Engineering.

Fouad, M.M., El-Desouky, A.I., Al-Hajj, R., El-Kenawy, E.-S.M., 2020. Dynamic group-based cooperative optimization algorithm. IEEE Access 8, 148378–148403.

Fraser, A.S., 1957a. Simulation of genetic systems by automatic digital computers I. introduction. Aust. J. Biol. Sci. 10 (4), 484–491.

Fraser, A.S., 1957b. Simulation of genetic systems by automatic digital computers II. effects of linkage on rates of advance under selection. Aust. J. Biol. Sci. 10 (4), 492–500.

Gandomi, A.H., Yang, X.-S., Talatahari, S., Alavi, A.H., 2013. Metaheuristic algorithms in modeling and optimization. Metaheuristic Appl. Struct. Infrastruct. 1–24.

Gao, G.-G.W.X.-Z., Zenger, K., Coelho, L.d.S., 2018. A novel metaheuristic algorithm inspired by rhino herd behavior.

Geem, Z.W., Kim, J.H., Loganathan, G.V., 2001. A new heuristic optimization algorithm: Harmony search. Simulation 76 (2), 60–68.

GhaemiDizaji, M., Dadkhah, C., Leung, H., 2020. OHDA: An opposition based high dimensional optimization algorithm. Appl. Soft Comput. 106185.

Ghafil, H.N., Jármai, K., 2020. Dynamic differential annealed optimization: New metaheuristic optimization algorithm for engineering applications. Appl. Soft Comput. 106392.

Ghasemi, M., Davoudkhani, I.F., Akbari, E., Rahimnejad, A., Ghavidel, S., Li, L., 2020. A novel and effective optimization algorithm for global optimization and its engineering applications: Turbulent flow of water-based optimization (TFWO). Eng. Appl. Artif. Intell. 92, 103666.

Ghasemi-Marzbali, A., 2020. A novel nature-inspired meta-heuristic algorithm for optimization: Bear smell search algorithm. Soft Comput. 1–33.

Ghasemian, H., Ghasemian, F., Vahdat-Nejad, H., 2020. Human urbanization algorithm: A novel metaheuristic approach. Math. Comput. Simulation.

Gholizadeh, S., Danesh, M., Gheyratmand, C., 2020. A new Newton metaheuristic algorithm for discrete performance-based design optimization of steel moment frames. Comput. Struct. 234, 106250.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. 13 (5), 533–549.

Glover, F., Laguna, M., 1998. Tabu search. In: Handbook of Combinatorial Optimization. Springer, pp. 2093–2229.

Golinski, J., 1973. An adaptive optimization system applied to machine synthesis. Mech. Mach. Theory 8 (4), 419–436.

Gomes, C.P., Williams, R., 2005. Approximation algorithms. In: Search Methodologies. Springer, pp. 557–585.

Gong, W., razmjooy, N., 2020. A new optimisation algorithm based on OCM and PCM solution through energy reserve. Int. J. Ambient Energy 1–14.

Gonzalez, T.F., 2007. Handbook of Approximation Algorithms and Metaheuristics. Chapman and Hall/CRC.

Handl, J., Kell, D.B., Knowles, J., 2007. Multiobjective optimization in bioinformatics and computational biology. IEEE/ACM Trans. Comput. Biol. Bioinform. 4 (2), 279–292.

Harifi, S., Khalilian, M., Mohammadzadeh, J., Ebrahimnejad, S., 2019. Emperor penguins colony: A new metaheuristic algorithm for optimization. Evol. Intell. 12 (2), 211–226.

Harifi, S., Mohammadzadeh, J., Khalilian, M., Ebrahimnejad, S., 2020. Giza pyramids construction: An ancient-inspired metaheuristic algorithm for optimization. Evol. Intell. 1–19.

Hashim, F.A., Hussain, K., Houssein, E.H., Mabrouk, M.S., Al-Atabany, W., 2020. Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems. Appl. Intell. 1–21.

Hayyolalam, V., Kazem, A.A.P., 2020. Black widow optimization algorithm: A novel meta-heuristic approach for solving engineering optimization problems. Eng. Appl. Artif. Intell. 87, 103249.

Hegazy, T., Kassab, M., 2003. Resource optimization using combined simulation and genetic algorithms. J. Constr. Eng. Manag. 129 (6), 698–705.

Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H., 2019. Harris hawks optimization: Algorithm and applications. Future Gener. Comput. Syst. 97, 849–872.

Hochba, D.S., 1997. Approximation algorithms for NP-hard problems. ACM Sigact News 28 (2), 40–52.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. Univ. of Mich. Press, Ann Arbor.

Holland, J.H., 1992. Genetic algorithms. Sci. Am. 267 (1), 66–73.

Hosseini, E., Sadiq, A.S., Ghafoor, K.Z., Rawat, D.B., Saif, M., Yang, X., 2020. Volcano eruption algorithm for solving optimization problems. Neural Comput. Appl. 1–17.

Houssein, E.H., Saad, M.R., Hashim, F.A., Shaban, H., Hassaballah, M., 2020. Lévy flight distribution: A new metaheuristic algorithm for solving engineering optimization problems. Eng. Appl. Artif. Intell. 94, 103731.

Hu, Z., Xu, X., Su, Q., Zhu, H., Guo, J., 2020. Grey prediction evolution algorithm for global optimization. Appl. Math. Model. 79, 145–160.

Jahangiri, M., Hadianfard, M.A., Najafgholipour, M.A., Jahangiri, M., Gerami, M.R., 2020. Interactive autodidactic school: A new metaheuristic optimization algorithm for solving mathematical and structural design optimization problems. Comput. Struct. 235, 106268.

Jain, M., Singh, V., Rani, A., 2019. A novel nature-inspired algorithm for optimization: Squirrel search algorithm. Swarm Evol. Comput. 44, 148–175.

Jiang, J., Xu, M., Meng, X., Li, K., 2020. STSA: A sine tree-seed algorithm for complex continuous optimization problems. Physica A 537, 122802.

Jin, S., Zhou, M., Wu, A.S., 2003. Sensor network optimization using a genetic algorithm. In: Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics. pp. 109–116.

Karaboga, D., 2005. An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report, Technical report-TR06, Erciyes university, engineering faculty, computer engineering department.

Kashan, A.H., Tavakkoli-Moghaddam, R., Gen, M., 2019. Find-fix-finish-exploit-analyze (F3EA) meta-heuristic algorithm: An effective algorithm with new evolutionary operators for global optimization. Comput. Ind. Eng. 128, 192–218.

Kaur, S., Awasthi, L.K., Sangal, A.L., Dhiman, G., 2020. Tunicate swarm algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. Eng. Appl. Artif. Intell. 90, 103541.

Kaveh, A., Dadras, A., 2017. A novel meta-heuristic optimization algorithm: Thermal exchange optimization. Adv. Eng. Softw. 110, 69–84.

Kaveh, A., Eslamlou, A.D., 2020. Water strider algorithm: A new metaheuristic and applications. In: Structures, vol. 25. Elsevier, pp. 520–541.

Kaveh, A., Ghazaan, M.I., 2017. A new meta-heuristic algorithm: Vibrating particles system. Sci. Iran. Trans. A 24 (2), 551.

Kaveh, A., Khanzadi, M., Moghaddam, M.R., 2020a. Billiards-inspired optimization algorithm; A new meta-heuristic method. In: Structures, vol. 27. Elsevier, pp. 1722–1739.

Kaveh, A., Seddighian, M.R., Ghanadpour, E., 2020b. Black hole mechanics optimization: A novel meta-heuristic algorithm. Asian J. Civ. Eng. 21 (7), 1129–1149.

Kaveh, A., Zaerreza, A., 2020. Shuffled shepherd optimization method: A new meta-heuristic algorithm. Eng. Comput..

Kaveh, A., Zolghadr, A., 2017. Cyclical parthenogenesis algorithm: A new meta-heuristic algorithm. Asian J. Civ. Eng. Build. Hous..

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of ICNN95 - International Conference on Neural Networks. IEEE.

Khishe, M., Mosavi, M.R., 2020. Chimp optimization algorithm. Expert Syst. Appl. 113338.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220 (4598), 671–680.

Koza, J.R., 1989. Hierarchical genetic algorithms operating on populations of computer programs. In: IJCAI, vol. 89. pp. 768–774.

Koza, J.R., 1990. A paradigm for genetically breeding populations of computer programs to solve problems. Computer Science Dept., Stanford Univ., Margaret Jacks Hall, Stanford, Calif.

Lai, K.K., Mishra, S.K., Panda, G., Chakraborty, S.K., Samei, M.E., Ram, B., 2020. A limited memory q-BFGS algorithm for unconstrained optimization problems. J. Appl. Math. Comput. 1–20.

Le-Duc, T., Nguyen, Q.-H., Nguyen-Xuan, H., 2020. Balancing composite motion optimization. Inform. Sci. 520, 250–270.

Li, S., Chen, H., Wang, M., Heidari, A.A., Mirjalili, S., 2020. Slime mould algorithm: A new method for stochastic optimization. Future Gener. Comput. Syst..

Liu, Y., Li, R., 2020. PSA: A photon search algorithm.. J. Information Processing Systems 16 (2).

Martínez-Álvarez, F., Asencio-Cortés, G., Torres, J., Gutiérrez-Avilés, D., Melgar-García, L., Pérez-Chacón, R., Rubio-Escudero, C., Riquelme, J.C., Troncoso, A., 2020. Coronavirus optimization algorithm: A bioinspired metaheuristic based on the COVID-19 propagation model. arXiv preprint arXiv:2003.13633.

Masadeh, R., A., B., Sharieh, A., 2019. Sea lion optimization algorithm. Int. J. Adv. Comput. Sci. Appl. 10 (5).

Mirjalili, S., 2016. SCA: A Sine Cosine algorithm for solving optimization problems. Knowl.-Based Syst. 96, 120–133.

Mirjalili, S., Lewis, A., 2016. The whale optimization algorithm. Adv. Eng. Softw. 95, 51–67.

Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61.

Moazzeni, A.R., Khamehchi, E., 2020. Rain optimization algorithm (ROA): A new metaheuristic method for drilling optimization solutions. J. Pet. Sci. Eng. 195, 107512.

Mohamed, A.-A.A., Hassan, S., Hemeida, A., Alkhalaf, S., Mahmoud, M., Eldin, A.M.B., 2020. Parasitism–predation algorithm (PPA): A novel approach for feature selection. Ain Shams Eng. J. 11 (2), 293–308.

Moosavi, S.H.S., Bardsiri, V.K., 2019. Poor and rich optimization algorithm: A new human-based and multi populations algorithm. Eng. Appl. Artif. Intell. 86, 165–181.

Moosavian, N., Roodsari, B.K., 2014. Soccer league competition algorithm: A novel meta-heuristic algorithm for optimal design of water distribution networks. Swarm Evol. Comput. 17, 14–24.

Morais, R.G., Mourelle, L.M., Nedjah, N., 2018. Hitchcock birds inspired algorithm. In: Computational Collective Intelligence. Springer International Publishing, pp. 169–180.

Mousavirad, S.J., Ebrahimpour-Komleh, H., 2017. Human mental search: A new population-based metaheuristic optimization algorithm. Appl. Intell. 47 (3), 850–887.

Mu, L., Wang, P., Xin, G., 2020. Quantum-inspired algorithm with fitness landscape approximation in reduced dimensional spaces for numerical function optimization. Inform. Sci..

Nadaraya, E.A., 1964. On estimating regression. Theory Probab. Appl. 9 (1), 141–142.

Ow, P.S., Morton, T.E., 1988. Filtered beam search in scheduling. Int. J. Prod. Res. 26 (1), 35–62.

Oyelade, O.N., Ezugwu, A.E.-S., Mohamed, T.I., Abualigah, L., 2022. Ebola optimization search algorithm: A new nature-inspired metaheuristic optimization algorithm. IEEE Access 10, 16150–16177.

Pardalos, P.M., Romeijn, H.E., 2013. Handbook of Global Optimization, vol. 2. Springer Science & Business Media.

Pierezan, J., Coelho, L.D.S., 2018. Coyote optimization algorithm: A new metaheuristic for global optimization problems. In: 2018 IEEE Congress on Evolutionary Computation. CEC, IEEE, pp. 1–8.

Price, K., Storn, R.M., Lampinen, J.A., 2006. Differential Evolution: A Practical Approach to Global Optimization. Springer Science & Business Media.

Qais, M.H., Hasanien, H.M., Alghuwainem, S., 2020. Transient search optimization: A new meta-heuristic optimization algorithm. Appl. Intell. 1–16.

Rao, R.V., Savsani, V.J., Vakharia, D.P., 2011. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. Comput. Aided Des. 43 (3), 303–315.

Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2009. GSA: A gravitational search algorithm. Inform. Sci. 179 (13), 2232–2248.

Ray, T., Liew, K.M., 2003. Society and civilization: An optimization algorithm based on the simulation of social behavior. IEEE Trans. Evol. Comput. 7 (4), 386–396.

Rechenberg, I., 1965. Cybernetic solution path of an experimental problem. Royal Aircraft Establishment Library Translation 1122.

Rechenberg, I., 1978. Evolutionsstrategien, simulationsmethoden in der medizin und biologie. In: Workshop.(Sonderdruck Medizinische Informatik Und Statistik, 8.). Springer, Berlin.

Saha, A., Das, P., Chakraborty, A.K., 2017. Water evaporation algorithm: A new metaheuristic algorithm towards the solution of optimal power flow. Eng. Sci. Technol. 20 (6), 1540–1552.

Salgotra, R., Singh, U., 2019. The naked mole-rat algorithm. Neural Comput. Appl. 31 (12), 8837–8857.

Salih, S.Q., Alsewari, A.A., 2019. A new algorithm for normal and large-scale optimization problems: Nomadic people optimizer. Neural Comput. Appl..

Sandgren, E., 1990. Nonlinear integer and discrete programming in mechanical design optimization.

Shabani, A., Asgarian, B., Salido, M., Gharebaghi, S.A., 2020. Search and rescue optimization algorithm: A new optimization method for solving constrained engineering optimization problems. Expert Syst. Appl. 161, 113698.

Shadravan, S., Naji, H.R., Bardsiri, V.K., 2019. The sailfish optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. Eng. Appl. Artif. Intell. 80, 20–34.

Shastri, A.S., Jagetia, A., Sehgal, A., Patel, M., Kulkarni, A.J., 2019. Expectation algorithm (ExA): A socio-inspired optimization methodology. In: Socio-Cultural Inspired Metaheuristics. Springer, pp. 193–214.

Shayanfar, H., Gharehchopogh, F.S., 2018. Farmland fertility: A new metaheuristic algorithm for solving continuous optimization problems. Appl. Soft Comput. 71, 728–746.

Singh, P.R., Elaziz, M.A., Xiong, S., 2019. Ludo game-based metaheuristics for global and engineering optimization. Appl. Soft Comput. 84, 105723.

Srivastava, A., Das, D.K., 2020. A new Kho-Kho optimization algorithm: An application to solve combined emission economic dispatch and combined heat and power economic dispatch problem. Eng. Appl. Artif. Intell. 94, 103763.

Storn, R., Price, K., 1997. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 11 (4), 341–359.

Sulaiman, M.H., Mustaffa, Z., Saari, M.M., Daniyal, H., 2020. Barnacles mating optimizer: A new bio-inspired algorithm for solving engineering optimization problems. Eng. Appl. Artif. Intell. 87, 103330.

Taramsco, C., Crawford, B., Soto, R., Cortés-Toro, E.M., Olivares, R., 2020. A new metaheuristic based on vapor-liquid equilibrium for solving a new patient bed assignment problem. Expert Syst. Appl. 158, 113506.

Wagan, A.I., Shaikh, M.M., et al., 2020. A new metaheuristic optimization algorithm inspired by human dynasties with an application to the wind turbine micrositing problem. Appl. Soft Comput. 90, 106176.

Wang, G.-G., Deb, S., Coelho, L.D.S., 2018. Earthworm optimisation algorithm: A bio-inspired metaheuristic algorithm for global optimisation problems. Int. J. Bio-Inspired Comput. 12 (1), 1–22.

Wang, X., Ding, X., Qu, Q., 2020. A new filter nonmonotone adaptive trust region method for unconstrained optimization. Symmetry 12 (2), 208.

Xie, X.-F., Zhang, W.-J., Yang, Z.-L., 2002. Social cognitive optimization for nonlinear programming problems. In: Proceedings. International Conference on Machine Learning and Cybernetics, vol. 2. IEEE, pp. 779–783.

Yadav, A., Kumar, N., et al., 2020. Artificial electric field algorithm for engineering optimization problems. Expert Syst. Appl. 149, 113308.

Yang, X.-S., 2010a. Nature-Inspired Metaheuristic Algorithms. Luniver Press.

Yang, X.-S., 2010b. Firefly algorithm, Levy flights and global optimization. In: Research and Development in Intelligent Systems XXVI. Springer, pp. 209–218.

Yang, X.-S., 2010c. A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization. NICSO, Springer Berlin Heidelberg, pp. 65–74.

Yang, X.-S., 2012. Flower pollination algorithm for global optimization. In: International Conference on Unconventional Computing and Natural Computation. Springer, pp. 240–249.

Yang, X.-S., Deb, S., 2009. Cuckoo search via Levy flights. In: 2009 World Congress on Nature & Biologically Inspired Computing. NaBIC, IEEE.

Yazdani, D., Meybodi, M., 2015. A modified gravitational search algorithm and its application. In: 2015 7th Conference on Information and Knowledge Technology. IKT, IEEE, pp. 1–6.

Yu, B., Yang, Z.-Z., Yao, B., 2009. An improved ant colony optimization for vehicle routing problem. European J. Oper. Res. 196 (1), 171–176.

Zaeimi, M., Ghoddosian, A., 2020. Color harmony algorithm: An art-inspired metaheuristic for mathematical function optimization. Soft Comput. 1–40.

Zervoudakis, K., Tsafarakis, S., 2020. A mayfly optimization algorithm. Comput. Ind. Eng. 106559.

Zhang, Y., Jin, Z., 2020. Group teaching optimization algorithm: A novel metaheuristic method for solving global optimization problems. Expert Syst. Appl. 148, 113246.

Zhang, Y., Jin, Z., Mirjalili, S., 2020. Generalized normal distribution optimization and its applications in parameter extraction of photovoltaic models. Energy Convers. Manage. 224, 113301.

Zhang, J., Xiao, M., Gao, L., Pan, Q., 2018. Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems. Appl. Math. Model. 63, 464–490.

Zhao, W., Wang, L., Zhang, Z., 2019. Supply-demand-based optimization: A novel economics-inspired algorithm for global optimization. IEEE Access 7, 73182–73206.

Zhao, W., Zhang, Z., Wang, L., 2020. Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. Eng. Appl. Artif. Intell. 87, 103300.

Zibulevsky, M., Elad, M., 2010. L1-L2 optimization in signal and image processing. IEEE Signal Process. Mag. 27 (3), 76–88.