The Journal of Supercomputing (2022) 78:2125–2174
https://doi.org/10.1007/s11227-021-03943-w

Stock exchange trading optimization algorithm: a human-inspired method for global optimization

Hojjat Emami[1]

Accepted: 8 June 2021 / Published online: 25 June 2021

Abstract
In this paper, a human-inspired optimization algorithm called stock exchange trading optimization (SETO) for solving numerical and engineering problems is introduced. The inspiration source of this optimizer is the behavior of traders and stock price changes in the stock market. Traders use various fundamental and technical analysis methods to gain maximum profit. SETO mathematically models the technical trading strategy of traders to perform optimization. It contains three main actuators including rising, falling, and exchange. These operators navigate the search agents toward the global optimum. The proposed algorithm is compared with seven popular meta-heuristic optimizers on forty single-objective unconstraint numerical functions and four engineering design problems. The statistical results obtained on test problems show that SETO is capable of providing competitive and promising performances compared with counterpart algorithms in solving optimization problems of different dimensions, especially 1000-dimension problems. Out of 40 numerical functions, the SETO algorithm has achieved the global optimum on 36 functions, and out of 4 engineering problems, it has obtained the best results on 3 problems.

Keywords Human-inspired meta-heuristic · Numerical optimization · Engineering design problems · Stock exchange trading optimization (SETO) algorithm

1 Introduction

Optimization plays a crucial role in various domains, like industrial applications, business, engineering, social science, and transportation [1–3]. A lot of problems in science and engineering are generally constraint or unconstraint optimization problems. Generally speaking, optimization is the process of selecting the best possible

✉ Hojjat Emami
emami@ubonab.ac.ir

[1] University of Bonab, Bonab, Iran

solution for a given engineering/scientific problem [4]. An optimization problem P can be formulated mathematically as follows [5]:

$$P \doteq (Q, C, f) \tag{1}$$

where Q denotes the solution space defined over a finite set of optimization variables, C denotes a set of problem-dependent constraints, and f is an objective function that needs to be minimized or maximized. The goal is to find an optimum solution Q^* with minimum objective function value $f(q^*) \leq f(q)$, $\forall\, q \in Q$ in minimization problems. In maximization problems, the objective value of solution q^* is to be maximized. According to the structure measure, the optimization problems can be grouped into different classes: constraint or unconstraint, single- or multi-objective, and combinatorial problems [5]. Constraint optimization problems involve one or several certain restrictions that cannot be violated in the optimization process. On the contrary, unconstraint problems do not involve limitations or constraints. In single-objective problems, there is only one specific objective, while in multi-objective mode there is more than one objective to be maximized or minimized. In combinatorial optimization problems, the goal is to find or select a permutation of variables in a way that objective function is minimized or maximized.

The majority of real-world large-scale, multimodal, non-differentiable, and non-continuous optimization problems are difficult to solve with conventional mathematical and deterministic methods such as quasi-Newton and sequential quadratic programming. The classic deterministic and exact optimization methods often perform an exhaustive search by simple calculus rules and tend to utilize problem-specific information such as the gradients of the objective to guide the search process in solution space [6]. These methods may stick at the local optima, need to derivate the search space, and cannot efficiently balance between exploitation and exploration [4]. Meta-heuristic optimizers are efficient alternatives when dealing with large-scale and non-differentiable problems [7]. They have gained immense popularity amongst researchers due to their simplicity, durability, self-organization, coordination, easy implementation, robustness, and effectiveness in solving a variety of optimization problems [6].

Meta-heuristics can solve optimization problems with limited complexity [4]. However, the performance of most meta-heuristics depends on the tuning of user-defined parameters. Besides, meta-heuristics do not guarantee a global optimum solution is ever found but try to find a near-optimal solution within a reasonable time. Meta-heuristics are black-box optimizers that can be applied to a variety of optimization problems with only limited modifications. To solve an optimization problem, a meta-heuristic algorithm first creates one or multiple initial solutions. While stopping criteria are not satisfied, the algorithm explores and exploits the solution space with different actuators to generate new solutions. At each generation, the algorithm updates the solutions. Finally, a solution with the maximum fitness is considered the optimal solution for the given problem. The key factor to efficient search is the proper harmonization between exploitation (intensification) and exploration (diversification) [8].

The majority of meta-heuristic algorithms are inspired by the biological evolution, social behavior of humans, physics laws, and the survival and living systems of animals, insects, and birds [2, 4–6, 8, 9]. For example, particle swarm optimization (PSO) [9] models the social behavior of birds flocking. It starts the search process with a collection of solutions dubbed particles. Each particle navigates in the solution space using its local best and the global best knowledge found by other particles. Another example is ant colony optimization (ACO) [10], which simulates the searching of ants from the colony to the food source.

In recent years, we have witnessed human-inspired algorithms becoming increasingly one of the most important topics in the optimization field. Human-inspired algorithms simulate the approaches that humans use to solve problems. The presidential election algorithm (PEA) [11, 12] is a fundamental human behavior-inspired algorithm that models the interaction between voters and candidates in the election campaign. A few well-known human-inspired algorithms are football game algorithm (FGA) [13] inspired by the behavior of players to score a goal under the supervision of the coach; political optimizer (PO) [14] inspired by the multi-phased process of politics; heap-based optimizer (HBO) [15] inspired by the rank hierarchy in organizations, deer hunting optimization algorithm (DHOA) [1] simulates the hunting methods of the human toward deer; and nomadic people optimizer (NPO) [16] models the migration behavior of nomadic people in their searches and movement for sources of life including grass for grazing and water. Humans are the most intelligent creatures and always try to solve real-world problems in the best possible way; thus, modeling their behavior and actions can be a successful method to solve optimization problems. In the literature, human-inspired algorithms show outstanding performance in solving optimization problems.

Recently, tremendous progress has been made in the development of meta-heuristic algorithms. However, the field of meta-heuristics is far from maturity [5, 17]. According to the NFL theorem, it is possible that a certain meta-heuristic obtains better results on specific problems and not as good on others. In other words, no algorithm can solve all kinds of optimization problems at the same time [4, 18]. These reasons prove that there is still a need for introducing new algorithms or improving the existing ones. This paper presents a new human-inspired meta-heuristic for solving optimization problems. The proposed algorithm is referred to as stock exchange trading optimization (SETO). The procedure of SETO attempts to find the best share with maximum profit (optimal or near-optimal solution) in the stock market with the help of trading strategies. To the best of our knowledge, in the literature, there is no research, which simulates the stock trading strategies.

The stock exchange is a place where investors and traders can sell or buy their ownership of stocks. Equities or stocks represent fractional ownership in a company. The stock exchanges pursue two goals. The first objective is to provide capital to companies to expand their businesses. The secondary goal is to allow investors to share in the profits of publicly traded companies. Trade is the basic concept in the stock market that means the transfer of a share from a seller to a buyer based on an agreement on a price. Trading and investing are two different approaches to profit in the stock market. Both traders and investors seek profits through buying and selling shares. Investors buy shares and hold them for an extended period to earn large

returns. In contrast, traders attempt to make transactions that can help them profit quickly from price fluctuations over a shorter time frame. The objective of traders is to gain returns that outperform buy-and-hold investing. Traders often use different technical analysis tools such as stochastic oscillators and moving averages to find optimal share buy and sell points. They try to maximize profit by adopting optimal trading strategies and selecting the best shares. The procedure of the proposed SETO algorithm attempts to find the most profitable share in the stock exchange with the help of simple trading strategies. The best share corresponds to the optimal solution to the given optimization problem. The SETO algorithm first creates a population of candidate solutions. The algorithm improves the initial solutions using three operators including rising, falling, and exchange. The individuals in the population gradually converge to the optimal point.

Briefly speaking, the main contributions of this paper are as follows:

- A new meta-heuristic algorithm named stock exchange trading optimization (SETO) algorithm is proposed for solving numerical and engineering optimization problems. Most of the control parameters of SETO are already known and configured using the data drawn from the stock exchange and scientific resources about technical analysis. This issue turns the SETO into an optimizer quite easy to implement and execute.
- Forty single-objective numerical optimization functions from CEC competitions and four engineering design problems are used to evaluate the performance of the SETO and comparison algorithms. The experimental results confirm the superiority of the proposed SETO compared with counterparts.

The SETO algorithm is simple and easy to implement. It can be applied to all optimization problems that other optimizers can be applied for. SETO is an efficient choice to solve optimization problems in various disciplines such as physical science, mathematics, agricultural science, economics, computer science, communication, mechanical applications, civil engineering applications, manufacturing, and many other areas.

The remaining parts of this paper are structured as follows: Section 2 reviews the literature. Section 3 describes the inspiration source, mathematical model, and the working principle of the proposed SETO algorithm. Section 4 presents the experimental results obtained by the SETO and counterpart algorithms in solving single-objective numerical optimization problems. Section 5 evaluates the applicability of SETO and comparison algorithms on real-world engineering design problems. Finally, Sect. 6 concludes the paper and lists potential directions for future research.

2 Related work

According to the metaphor of the search procedures, the structure of the problem under consideration, and the search strategy, optimization meta-heuristics can be categorized into different classes. As shown in Fig. 1, two main groups of meta-heuristics are metaphor-based and non-metaphor-based algorithms [8]. The former

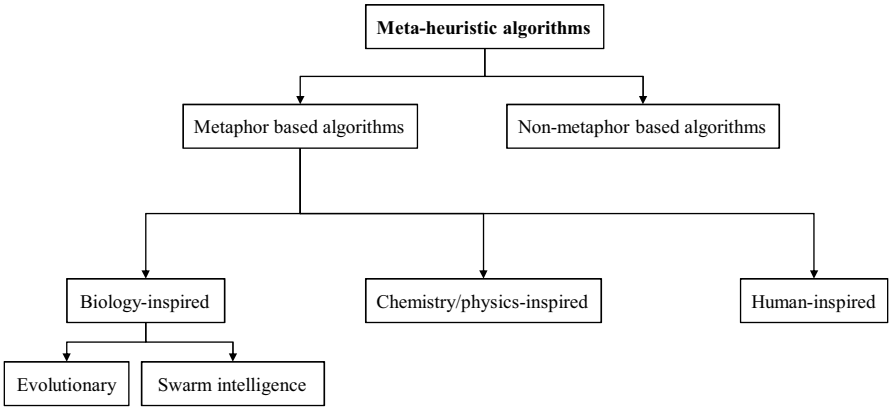


Fig. 1 Broad classification of optimization meta-heuristic algorithms

category consists of algorithms that model the natural evolution, collective or swarm intelligence of creatures, human actions in real life, chemistry or physical operations, etc. The latter category of algorithms did not simulate any natural phenomena or creatures' behavior for performing a search in the solution space of optimization problems.

The metaphor-based algorithms can be categorized into three main paradigms: biology-inspired, chemistry-/physics-inspired, and human-inspired algorithms. Biology-inspired algorithms simulate the evolution of living organisms or the collective intelligence of creatures such as ants, birds, and bees. Two classes of biology-inspired algorithms are evolutionary and swarm intelligence algorithms. Evolutionary algorithms are inspired by the laws of biological evolution in nature [15, 19]. The objective is to combine the best individuals to improve the survival and reproduction ability of individuals throughout generations. Since the fittest individuals have a higher chance to survive and reproduce, the individuals in the next generations may probably be better than previous ones [18]. This idea forms the search strategy of evolutionary algorithms, in which individuals will gradually reach the global optimum. The most popular evolutionary algorithm is the genetic algorithm (GA) [20] that follows Darwin's theory of evolution. In GA, first, a population of solutions is created randomly. The population evolves over iterations through selection, reproduction, combination, and mutation. Few popular evolutionary algorithms are fast evolutionary programming (FEP) [21], differential evolution (DE) [22], biogeography-based optimization (BBO) [23], forest optimization algorithm (FOA) [24], black widow optimization (BWO) [25], farmland fertility algorithm (FFA) [26], and seasons optimization algorithm (SOA) [18].

Swarm intelligence algorithms often model the interaction of living creatures in a community, herds, flocks, colonies, and schools [6]. The core idea of swarm intelligence algorithms is decentralization, in which the agents move toward the global optimum through simulated social and collective intelligence, and local interaction with their environment and with each other [8]. The algorithms in this category memorize the best solutions found at each generation to produce the optimal

solutions for the next generations. The most popular algorithms in this category are PSO [9], ACO [10], and artificial bee colony (ABC) [27]. Some recently developed swarm intelligence algorithms are firefly algorithm (FA) [28], krill herd (KH) [29], elephant herding optimization (EHO) [30], spider monkey optimization (SMO) [31], grey wolf optimizer (GWO) [32], whale optimization algorithm (WOA) [19, 33], butterfly optimization algorithm (BOA) [34], squirrel search algorithm (SSA) [35], grasshopper optimization algorithm (GOA) [36], seagull optimization algorithm (SOA) [37], normative fish swarm algorithm (NFSA) [38], red deer algorithm (RDA) [39], and Harris hawks optimization (HHO) [7]. For more detail and deep discussion about swarm intelligence algorithms, refer to the survey given in [6].

Chemistry- and physics-based algorithms simulate the chemistry and physical rules in the universe such as chemical reactions, gravitational force, inertia force, and magnetic force [25]. The search agents navigate and communicate through the search space following the chemistry and physical rules. Simulated annealing (SA) [40] is one of the founding algorithms in this category. SA models the annealing process in metallurgy. Other widely used chemistry- and physics-based algorithms are gravitational search algorithm (GSA) [41], big bang–big crunch (BB–BC) [42], artificial chemical reaction optimization algorithm (ACROA) [43], galaxy-based search algorithm (GbSA) [44], physarum-energy optimization algorithm (PEO) [45], thermal exchange optimization (TEO) [46], equilibrium optimizer (EO) [47], magnetic optimization algorithm (MOA) [48]. For a survey and discussion about physics-inspired algorithms, refer to [49].

Human-based algorithms are developed based on metaphors from human life, such as social relationships, political events, sports, music, and math. Since humans are considered the smartest creatures in solving real-world problems, human-inspired algorithms can also be more successful in solving optimization problems. Some human-inspired algorithms are harmony search (HS) [50], imperialist competitive algorithm (ICA) [51], teaching–learning-based optimization (TLBO) [52], league championship algorithm (LCA) [53], class topper optimization (CTO) [54], presidential election algorithm (PEA) [11], sine–cosine algorithm (SCA) [55], socio evolution & learning optimization algorithm (SELO) [56], team game algorithm (TGA) [57], ludo game-based swarm intelligence (LGSI) [58], heap-based optimizer (HBO) [15], coronavirus optimization algorithm (CVOA) [59], political optimizer (PO) [14], and Lévy flight distribution (LFD) [4].

Some algorithms are inspired by machine learning, reinforcement learning, and learning classifier systems [60–62]. For example, ActivO is an ensemble machine learning-based optimization algorithm [63]. ActivO combines strong and weak learner strategies to perform a search for optimal solutions. The weak learner is considered to explore the promising regions, and the strong learner is considered to identify the exact location of the optimum within promising areas. Another example is the molecule deep Q-networks (MolDQN) algorithm, which is developed by combining domain knowledge of chemistry and reinforcement learning techniques for molecule optimization. Researchers have proposed several methods for optimizing trading strategies in the stock exchange [64–68]. For example, Thakkar and Chaudhari [67] investigated the application of meta-heuristic algorithms for stock portfolio optimization, and trend and stock price prediction along with implications

of PSO. In other work, Kumar and Haider [68] proposed RNN–LSTM and improved its performance using PSO and flower pollination algorithm (FPA) for intraday stock market prediction. It is important to notice that this paper does not focus on the optimization or prediction in the stock exchange. To the best of our knowledge, in the literature, there is no research that simulates the stock trading strategies for developing numerical optimization meta-heuristics.

It should be noted that each of the meta-heuristic algorithms has been improved over the years, and several enhanced versions of them are available. The extended algorithms improve the basic operators or overcome the defections that exist in the conventional versions. For example, the chaotic election algorithm (CEA) [12] embeds the chaos-based advertisement operator to the conventional PEA algorithm [11] to improve its search capability and convergence speed. Some other algorithms that recently proposed and used in different applications are opposition-based learning firefly algorithm combined with dragonfly algorithm (OFADA) [69], random memory and elite memory equipped artificial bee colony (ABCWOA) algorithm [70], efficient binary symbiotic organisms search (EBSOS) [71, 72], efficient binary chaotic symbiotic organisms search (EBCSOS) [73], and binary farmland fertility algorithm (BFFA) [74].

After this short review, and from the experimental results reported in the literature, we can conclude that the obtained performances on most optimization problems are not perfect. This phenomenon clearly shows that a lot of effort is needed in the field. Each algorithm is suitable for solving certain types of problems. It seems that one of the interesting tasks in this field is to determine the best algorithms for each type of optimization problem. For deep analysis about meta-heuristic algorithms, refer to surveys given in [2, 8, 75]. Table 1 summarizes some of the recently proposed meta-heuristic algorithms.

3 Stock exchange trading optimization (SETO) algorithm

This section discusses the inspiration source and describes the mathematical model of the proposed stock exchange trading optimization (SETO) algorithm.

3.1 Inspiration

A stock exchange or bourse is an exchange where traders and investors sell and buy all types of securities such as shares of stock, bonds, and other financial instruments issued by listed companies [76]. The stock exchange often acts as a continuous auction market in which sellers and buyers perform transactions through electronic trading platforms and brokerages. People invest and trade with an efficient strategy in mind to make the most profit. Shares price never goes up in a straight line. They rise and fall on their way to higher prices. A rise occurs because more people want to buy a share than sell it. In the rising phase, the price of shares moves up. When the shares rise for a long period, correction may start. A correction and all types of market declines occur because investors or traders are more motivated to sell than buy.

Table 1 Some recently proposed meta-heuristics

Title	Inspiration source	Year	References
Genetic algorithm GA)	Darwin's theory of evolution	1992	[20]
Fast evolutionary programming (FEP)	Natural evolution	1999	[21]
Differential evolution (DE)	Natural evolution	2007	[22]
Biogeography-based optimization (BBO)	Geographical distribution of biological organisms	2008	[23]
Forest optimization algorithm	Growth of trees in forests	2014	[24]
Black widow optimization (BWO)	Unique mating behavior of black widow spiders	2020	[25]
Farmland fertility optimization (FFO)	Farmland fertility in nature	2018	[26]
Seasons optimization algorithm (SOA)	Trees growth behavior	2020	[18]
Particle swarm optimization (PSO)	Motion of bird flocks and schooling fish	1995	[9]
Ant colony optimization (ACO)	Foraging behavior of natural ants	2006	[10]
Artificial bee colony (ABC)	Intelligent behavior of bees	2007	[27]
Firefly algorithm (FA)	Flashing behavior of fireflies	2010	[28]
Krill herd (KH)	The herding behavior of krill communications	2012	[29]
Elephant herding optimization (EHO)	Herding behavior of elephant group	2016	[30]
Spider monkey optimization (SMO)	Fission–fusion social structure of spider monkeys in foraging	2014	[31]
Grey wolf optimizer (GWO)	Leadership hierarchy and hunting mechanism of grey wolves	2014	[32]
Whale optimization algorithm (WOA)	Humpback whales	2016	[19]
Butterfly optimization algorithm (BOA)	Food foraging behavior of the butterflies	2018	[34]
Squirrel search algorithm (SSA)	Dynamic behavior of flying squirrels	2019	[35]
Grasshopper optimization algorithm (GOA)	Foraging and swarming behavior of grasshoppers	2017	[36]
Seagull optimization algorithm (SOA)	Migration and attacking behaviors of a seagull in nature	2019	[37]
Normative fish swarm algorithm (NFSA)	Behavior of fish swarm in the real environment	2019	[38]
Red deer algorithm (RDA)	Unusual mating behavior of Scottish red deer	2020	[39]
Harris hawks optimization (HHO)	Cooperative behavior and chasing style of Harris' hawks	2019	[7]
Simulated annealing (SA)	Annealing procedure of the metal working	1983	[40]
Gravitational search algorithm (GSA)	Newtonian's law of gravity and the law of motion	2009	[41]

Table 1 (continued)

Title	Inspiration source	Year	References
Big bang–big crunch (BB–BC)	Big bang theory	2006	[42]
Artificial chemical reaction optimization algorithm (ACROA)	Natural chemical reaction	2011	[43]
Galaxy-based search algorithm (GBSA)	Spiral arm of spiral galaxies to search its surrounding	2011	[44]
Physarum-energy optimization algorithm (PEO)	Characteristic of ants' spatiotemporal variations	2017	[45]
Thermal exchange optimization (TEO)	Newton's law of cooling	2017	[46]
Equilibrium optimizer (EO)	Control volume mass balance	2019	[47]
Magnetic optimization algorithm (MOA)	Principles of magnetic field theory	2018	[48]
Harmony search (HS)	Music improvisation process	2001	[50]
Imperialist competitive algorithm (ICA)	Imperialistic competition	2007	[51]
Teaching–learning-based optimization (TLBO)	Teaching and learning process in a classroom	2012	[52]
League championship algorithm (LCA)	Sport championships	2014	[53]
Class topper optimization (CTO)	Learning intelligence of students in a class	2018	[54]
Presidential election algorithm (PEA)	Behavior of candidates in the presidential election	2015	[11]
Sine–cosine algorithm (SCA)	The mathematical form of the sine and cosine	2016	[55]
Socio evolution & learning optimization algorithm (SELO)	Social learning behavior of humans organized as families	2018	[56]
Team game algorithm (TGA)	Cooperation of individuals in a game	2018	[57]
Ludo game-based swarm intelligence (LGSI)	Ludo game playing strategies	2019	[58]
Heap-based optimizer (HBO)	Corporate rank hierarchy in organizations	2020	[15]
Coronavirus optimization algorithm (CVOA)	Coronavirus outbreak	2020	[59]
Political optimizer (PO)	Multi-phased process of politics	2020	[14]
Lévy flight distribution (LFD)	Lévy flight random walk for exploring unknown spaces	2020	[4]
Machine learning-based optimization algorithm (ActivO)	Machine learning strategies	2021	[71]

Fig. 2 A schematic view of RSI indicator http://forex-indicators.net/rsi

At this time, sellers will start lowering prices until buyers tend to buy the shares. Traders can sell their shares at any time they see fit or add to their number of shares. They use various indicators to obtain the selling and buying signals and maximize their gains through the analysis of stocks' momentum. Some of the most commonly used technical indicators are simple moving average (SMA), moving average convergence divergence (MACD), relative strength index (RSI), stochastic oscillator, and Bollinger bands among others [76].

The RSI [77] is a well-known momentum oscillator used in technical analysis. It measures the magnitude of recent price changes to investigate overbought or oversold conditions in the price of a share. It produces signals that tell traders to sell when the share is overbought and to buy when it is oversold. The RSI is often measured on a 14-day timeframe, and it oscillates between 0 and 100. The indicator has a lower line typically at 30 and an upper line at 70. A share is often considered oversold when the RSI is at or below 30 and overbought when it is around 70 [78]. RSI between the 30 and 70 levels is considered neutral. An oversold signal recommends that short-term declines are reaching maturity, and a share may be in for a rally. In contrast, an overbought signal could mean that short-term gains may be reaching a point of maturity, and a share may be in for a price correction. As shown in Fig. 2, RSI is often illustrated on a graph below the price chart.

In addition to the indicator signals, many investors use fundamental analysis especially price-to-earnings (P/E) ratio to find out if a share is correctly valued [79]. P/E shows how cheap or expensive the share is. If all things are equal (the lower the price, the higher the return), the lower P/E means the lower price of a share that is suitable for investors. However, if all things are not equal, a lower P/E may not indicate a good share for investing, because a share with a high P/E may provide a better

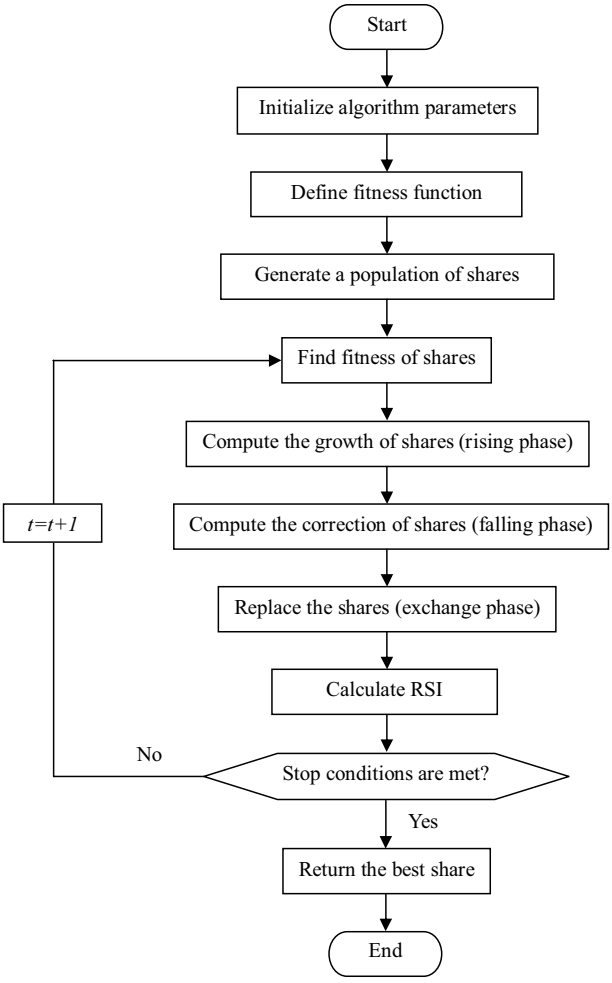


Fig. 3 Flowchart of the proposed SETO algorithm

return than a low P/E stock. Overall, in trading, it is better to compare the P/E of a share with its market peers to discover it is overvalued or undervalued.

Traders and shareholders try to maximize profits by looking for the best shares with the highest earning. The behavior of traders in the stock market is an adaptive optimization process.

3.2 Mathematical model

This section shows how the trading behavior of traders and changes in share prices is mathematically modeled to design the stock exchange trading optimization (SETO) algorithm. Figure 3 shows the flowchart of the SETO algorithm. The SETO is a population-based optimization algorithm, which starts its work with an initial

population of shares. Each share (stock) in the population is a potential solution to the problem. The objective is to find the most profitable share in the population, which corresponds to the optimal solution. The algorithm iteratively updates the population by three main operators including *rising*, *falling*, and *exchange*. Finally, the most profitable share is reported as the optimal solution. The rising phase models the growth of shares' prices in the stock exchange. The falling phase models the prices decline of shares. In the exchange phase, traders replace their shares with the lowest profit with the most profitable shares. In the following, the components of the algorithm are described in more detail.

3.2.1 Create initial population

To solve any optimization problem, the first step in the SETO algorithm is to create an initial population of candidate solutions. Each solution in the population is referred to as a *share* or *stock*. In this paper, the terms "share" and "stock" are utilized interchangeably in most cases. For an optimization problem $F(x)$ with D variables $\{x_1, x_2, \ \ldots, x_D\}$, the initial population is defined as

$$S = \left[S_1, S_2, \ldots, S_N\right]^T \tag{2}$$

where N is the population size. Each share $S_i \in S$ is a vector of D real-valued variables presented as follows:

$$S_i = \{s_{i1}, s_{i2}, \ldots, s_{iD}\} \tag{3}$$

where s_{ij} contains a possible value for the corresponding variable x_j of problem $F(x)$. The variable s_{ij} is initialized as

$$s_{i,j} = (u_{i,j} - l_{i,j}).\phi_{ij} + l_{i,j} \tag{4}$$

where ϕ_{ij} is a random number in the range [0, 1] generated by uniform distribution. l_i and u_i are the lower and upper bounds of t_{ij}, respectively.

The profitability of shares is evaluated using a fitness function f, which is related to the objective function of the problem. The fitness (profitability) of each share S_i is computed as follows:

$$f_i = f(S_i) = f\{s_{i1}, s_{i2}, \ldots, s_{iD}\} \tag{5}$$

where f_i is the fitness of share S_i according to the objective function of the problem. In minimization problems, the goal is to minimize the objective/cost function; however, in maximization problems, the goal is to maximize the objective function. In the terminology of SETO, for minimization problems, the fitness function equals the objective function, and for maximization problems, the fitness function has an inverse relation with the objective function. If a share is valuable, then its fitness will be greater and more traders will be attracted toward it. In this case, the share grows more and reaches higher prices. In other words, the share gradually converges to the optimal point.

At any given time, each share has a number of sellers and buyers. To identify the initial traders, we use a random initialization mechanism. To do this, first the normalized fitness (nf_i) of each share S_i is computed as follows:

$$nf_i = \frac{f_i - \min(M)}{\sum_{k=1}^{N} \left(f_k - \min(M)\right)}, \quad M = \{f_k | k = 1, 2, \ldots, N\} \tag{6}$$

The number of traders of S_i is computed as follows:

$$T_i = \lceil nf_i \times T \rceil \tag{7}$$

where T is the total number of traders, and T_i is the number of traders of share S_i. The number of traders can vary and change at any time. However, for simplicity, in the current implementation of SETO, the number of traders is considered constant, and during the running of the algorithm, the total number of traders does not increase or decrease. The initial number of buyers and sellers of share S_i is calculated as

$$\begin{aligned} b_i &= \lceil r \times T_i \rceil \\ s_i &= T_i - b_i \end{aligned} \tag{8}$$

where b_i and s_i are the number of buyers and sellers of S_i, respectively. The variable r is a random number in the range [0, 1], which is generated by the uniform distribution.

3.2.2 Rising

The rising operator simulates the growth of shares' prices in the market. In this phase, shares can move to higher prices. Here the highest price that shares can reach is considered as the optimal point. If the price of a share reaches its highest value, then the traders who have that stock will make the most profit. To mathematically model the rising phenomenon, we proposed the following equation:

$$S_i(t+1) = S_i(t) + R \times (S^g(t) - S_i(t)) \tag{9}$$

where $S_i(t)$ denotes the position of ith share at current iteration t, R is a $1 \times D$ vector of random numbers generated every iteration, and $S^g(t)$ is the best solution found until current iteration. The parameter R adds some amount of random deviations to the direction of movement in hope of escaping local optimums and more exploring solution space. Each element of vector $r_j \in R$ is defined as follows:

$$r_j = U(0, pc_i \times d_1) \tag{10}$$

where the function U generates a random number using uniform distribution in the range $[0, pc_i \times d_1]$. The variable pc_i is the ratio of buyers to sellers of S_i, and d_1 is the normalized distance between $S_i(t)$ and $S^g(t)$ defined as

$$d_1 = \frac{\sqrt{\sum_{j=1}^{D} (S_j^g(t) - S_{ij}(t))^2}}{ub - lb} \tag{11}$$

ub and lb are the upper and lower bound of the search space, respectively. The distance between shares is naturally related to the domain of the search space. Thus, the distance is normalized using $(ub - lb)$ in the denominator to avoid problem domain dependency. Supply and demand are two important factors in share growth. The higher the demand for a share, the more likely it is that the share will grow. For this purpose, the pc_i is considered in Eq. (10) to determine the impact of demand on share growth. Here, the demand for a share is indicated by the number of buyers. pc_i is simply defined as follows:

$$pc_i = \frac{b_i}{s_i + 1} \tag{12}$$

where b_i and s_i are the number of buyers and sellers of share S_i, respectively. To avoid the search boundary violation, the parameter pc_i is limited to a value in the range [0, 2]. So, Eq. (12) is revised as follows:

$$pc_i = \min(\frac{b_i}{s_i + 1}, 2) \tag{13}$$

In the rising phase, the demand for shares increases. To model this phenomenon, at each iteration of the algorithm and during rising, we remove a seller from the selling queue of S_i and add it to the buying queue as a buyer.

$$\begin{aligned} b_i &= b_i + 1; \\ s_i &= s_i - 1; \end{aligned} \tag{14}$$

In the implementation of SETO, it is assumed that any trader can buy or sell a share at any time. Therefore, the buying and selling queue of each share S_i are modeled as variables b_i and s_i.

In the rising phase, the algorithm spread the solutions far from the current area of search space to explore different areas of search space.

3.2.3 Falling

The falling phase simulates shares' prices decline. To mathematically model the falling, we propose the following equation:

$$S_i(t+1) = S_i(t) - W \times (S_i^l(t) - S_i(t)) \tag{15}$$

where $S_i^l(t)$ is the local best position the share S_i has ever found. The local search experience increases the convergence of the algorithm. W is a $1 \times D$ vector of uniform random numbers. Each element $w_j \in W$ is computed as follows:

$$w_j = U(0, nc_i \times d_2) \tag{16}$$

where function U generates a uniform random number in the range $[0, nc_i \times d_2]$. d_2 is the normalized distance between $S_i(t)$ and $S_i^l(t)$, which is calculated as follows:

$$d_2 = \frac{\sqrt{\sum_{j=1}^{D} (S_{ij}^l(t) - S_{ij}(t))^2}}{ub - lb} \tag{17}$$

nc_i is the ratio of sellers to buyers computed as

$$nc_i = \min(\frac{s_i}{b_i + 1}, 2) \tag{18}$$

In the case of falling prices, the share supply increases. To model this issue, at each iteration of the algorithm and during falling, we remove a buyer from buying queue of S_i and add it to the selling queue as a seller.

$$\begin{aligned} s_i &= s_i + 1; \\ b_i &= b_i - 1; \end{aligned} \tag{19}$$

At each iteration, the number of buyers and sellers of each share is controlled so that the total number of buyers and sellers does not exceed the total number of traders.

3.2.4 Exchange

In the exchange phase, traders replace their shares with the lowest profit with the most profitable shares. To do this, traders sell the lowest yielding shares and line up to buy the best shares. We implement this phenomenon by just picking one of the sellers from the sell queue of the worst share and assign it to the buy queue of the best share. The competition can be done among all shares to attract the traders; however, for simplicity, we assign the seller to the best share. To mathematically model this process, first, the worst share is identified. The share S_w with the lowest fitness is considered the worst if it obtains the lowest fitness.

$$\begin{aligned} S_{worst} = S_w \quad \text{where} \quad & f(S_w) < f(S_j) \\ & \forall\, j = 1, 2, \ldots, N, \quad w \neq j \end{aligned} \tag{20}$$

Then, one of the sellers is removed from the selling queue of the worst share S_{worst} and added to the buying queue of the best share. The best share S_{best} is determined as follows:

$$\begin{aligned} S_{best} = S_b \quad \text{where} \quad & f(S_b) > f(S_j) \\ & \forall\, j = 1, 2, \ldots, N, \quad b \neq j \end{aligned} \tag{21}$$

The exchange operator improves the population because it allows the best and worst shares eventually to grow. This reduces the number of sellers of the worst share and

increases the number of buyers of the best share. Therefore, the ratio of buyers to sellers increases, and in this case, the possibility of rising the shares increases.

3.2.5 RSI calculation

We use the RSI indicator to identify when the share rising or falling occurs. According to RSI value, SETO performs rising or falling as follows:

$$\begin{cases} \text{rising} & RSI \leq 30 \\ \text{falling} & RSI \geq 70 \\ p \times \text{rising} + (1-p) \times \text{falling} & 30 < RSI < 70 \end{cases} \tag{22}$$

where p is a binary random number with values 0 or 1 regenerated at every iteration. p is computed as follows:

$$p = \begin{cases} 1 & rand \geq 0.5 \\ 0 & else \end{cases} \tag{23}$$

where function *rand* generates a random number in the range [0, 1] using uniform distribution. For a share S_i, the RSI is calculated as follows [78]:

$$RSI = 100 - \frac{100}{1 + RS} \tag{24}$$

A simple moving average (SMA) method [76] is used to compute relative strength (RS) as follows:

$$RS = \sum_{i=1}^{K} P_i \Big/ \sum_{i=1}^{K} N_i \tag{25}$$

where P_i and N_i are the upward and downward price changes, respectively. K indicates the trading time frame of RSI. In the implementation of SETO, K is set to be 14 days (iterations). In the SETO algorithm, the price of shares is represented with their fitness. P_i and N_i are computed as follows:

$$P_i = \begin{cases} 1 & \text{if}\big(f_i(t) - f_i(t-1)\big) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

$$N_i = \begin{cases} 1 & \text{if}(f_i(t-1) - f_i(t)) > 0 \\ 0 & otherwise \end{cases} \tag{27}$$

where $f_i(t)$ and $f_i(t-1)$ are the fitness in the current and previous iterations, respectively. Here, the fitness corresponds to the close price of the share. If the previous fitness is the same as the last fitness, both P_i and N_i are set to be zero. The RSI will rise as the number of positive closes increase, and it will fall as the number of losses increase.

3.2.6 Stop condition

Until termination conditions are met, the algorithm iterates the rising, falling, and exchange phases on the population. Finally, the fittest share is returned as an optimal solution for the problem. The following termination conditions are considered to stop the algorithm:

- A predefined number of generations (G) is reached.
- A specified number of fitness function evaluations (FEs) is reached.
- The fitness of the best share is unchanged in successive iterations.

Algorithm 1 summarizes the pseudo-code of the proposed SETO algorithm.

Algorithm 1: Pseudo code of the proposed SETO algorithm

Input: The maximum number of iterations (G), number of shares (N), number of traders (T), RSI time frame (K)
Output: The fittest share S^g and its fitness

```
Initialize algorithm parameters;
Create an initial population of shares S_i, i = 1, 2, ..., N by Eqs. (2)-(4);
Calculate the fitness of shares by Eq. (5);
Determine the traders of shares;
Identify the best share S^g in the population;

t = 0;
while (t ≤ G) do

    for i=1 to N do
        if (t ≥ K and S_i(t).RSI ≤ 30) then
            [S]= Rising(S_i(t), S^g);
        else if (t ≥ K and S_i(t).RSI ≥ 70) then
            [S]= Falling (S);
        else
            r=rand;
            if (r > 0.5) then
                [S]= Rising(S);
            else
                [S]= Falling(S);
            end
        end

        // Exchange phase;
        [S]= Exchange(S);

        // RSI calculation;
        if (t >= K) then
            Compute P_i by Eq. (26);
            Compute N_i by Eq. (27);
            Calculated RSI by Eq.(24);
            S_i(t).RSI = RSI;
        end
        Update the best solution S^g;
    end

    t = t + 1;
end
Return the fittest share S^g and its fitness;
```

3.3 An example to show the functioning of SETO

To show the functioning of the SETO, it is benchmarked using the peak function. The purpose is to show how the shares move around the search space and gradually converge to the global optimum. The peak function is defined as follows:

$$f(x, y) = xe^{-(x^2+y^2)} \quad -2 \leq x, y \leq 2 \tag{28}$$

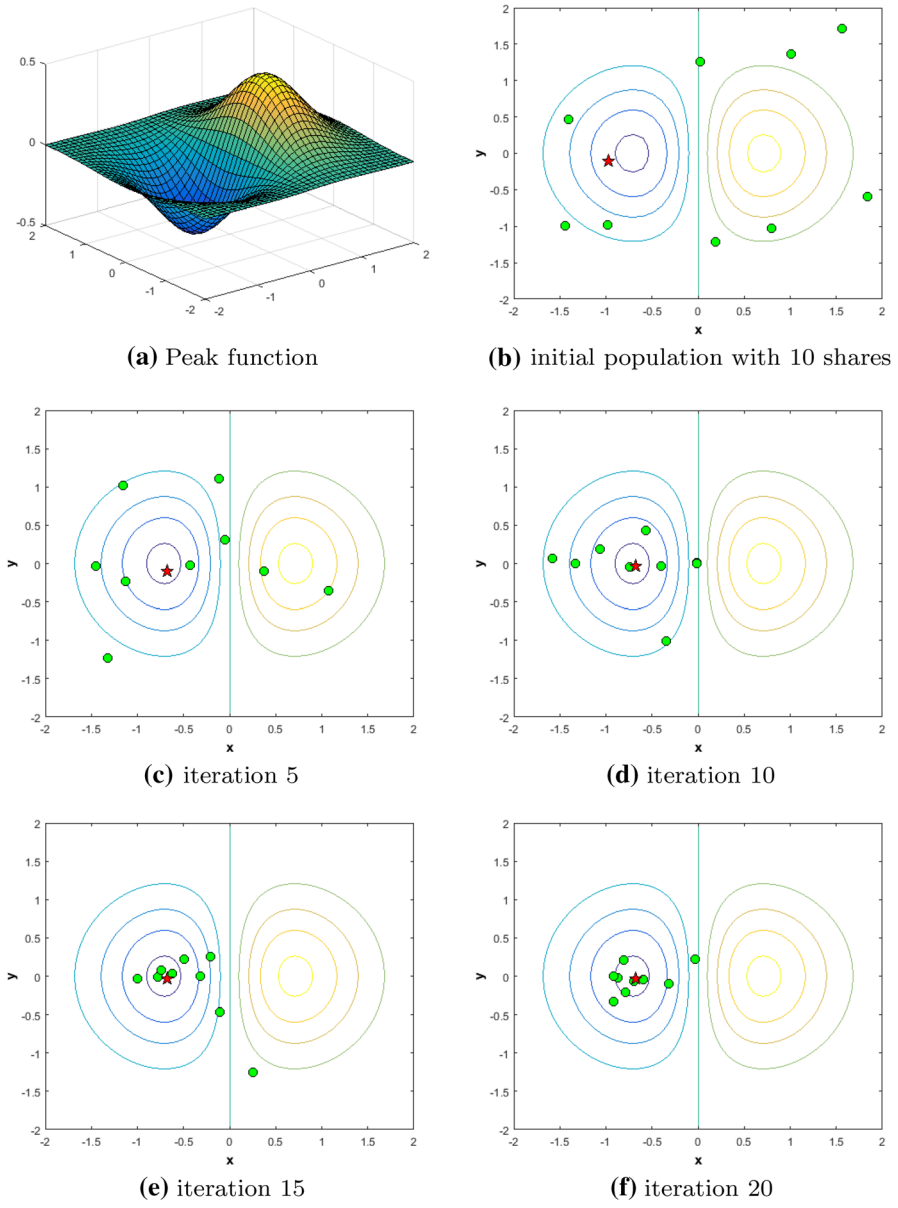


Fig. 4 The functioning of SETO on peak function

The global optimum of this problem is -0.4289 located at position $(x, y) = (-0.0708, 0.002)$. Figure 4a shows the graphical plot of the test function. Figure 4b shows the initial shares scattered throughout the search space. The shares are shown with a blue circle marker and the best share with a red star marker. Figure 4c, d shows the positions of shares at 5th, 10th, 15th, and 20th iterations,

respectively. Initially, the shares are scattered throughout the solution space and they are not in global optimum. In the 5th iteration, a share is close to the global optimal point, while other shares are placed at local optimums. In the 10th iteration, the best share is more close to the global optimum, and in the 15th iteration, most of the shares are more close to the global optimum. Finally, at the 20th iteration, the majority of trees converge to the global optimum.

4 Experiments

This section presents the performance evaluation of the proposed algorithm on a diverse set of unconstraint and single-objective numerical optimization functions. In the following, characteristics of test problems, performance metrics, parameter tuning, as well as numerical results are presented.

4.1 Test problems

To investigate the precision, convergence speed, and search capability of the proposed SETO and comparison algorithms, forty well-studied test problems are chosen from the literature [4, 7, 18, 25, 80, 81]. This test set covers four classes of functions as follows:

- **Group I** $F1-F10$ are fixed-dimension problems. This test set investigates the local optimum avoidance capacity of algorithms in solving problems with a fixed number of variables [18].
- **Group II** $F11-F22$ are single-objective unimodal functions. These test cases have a unique global best in their landscape. They are considered to measure the exploitation (intensification) ability of the algorithms [18, 80].
- **Group III** $F23-32$ are multimodal functions that consist of multiple local optimums in their landscape. The dimensionality and multiple local optima make multimodal functions more difficult and more complex to optimize. This group of functions is considered to reveal the local avoidance and exploration (diversification) capability of optimization algorithms [7].
- **Group IV** $F33-F40$ are shifted, rotated, hybrid and composite functions. This test set is drawn from CEC 2018 competition [81] on single-objective real-parameter numerical optimization problems. These functions evaluate the reliability, accuracy, and ability of the algorithms in providing a balance between exploration and exploitation.

The characteristics of test problems are summarized in Tables 2, 3, 4, 5, and 6. In the tables, the parameter f_{min} means the global optimum of the test function, *Vars* indicates the number of dimensions of the problem, and *Range* denotes the boundary of search space.

Table 2 Descriptions of fixed-dimension test functions

Function	Name	Range	Vars	f_{min}
F1	Adjiman	[– 1, 2]	2	– 2.02181
F2	Bartels Conn	[– 500, 500]	2	1
F3	Brent	[– 10, 10]	2	0
F4	Bukin 6	[(– 15, – 5), (– 5, – 3)]	2	180.3276
F5	Easom	[– 100, 100]	2	– 1
F6	Egg Crate	[– 5, 5]	2	0
F7	Matyas	[– 10, 10]	2	0
F8	Schaffer N. 4	[– 100, 100]	2	0.292579
F9	Three-Hump Camel	[– 5, 5]	2	0
F10	Zettle	[– 5, 10]	2	– 0.00379

Table 3 Descriptions of unimodal test functions

Function	Name	Range	Vars	f_{min}
F11	Brown	[– 1, 4]	30	0
F12	Dixon and Price	[– 10, 10]	30	0
F13	Powell Singular	[– 4, 5]	30	0
F14	Powell Sum	[– 1, 1]	30	0
F15	Rosenbrock	[– 30, 30]	30	0
F16	Schwefel's 2.20	[– 100, 100]	30	0
F17	Schwefel's 2.21	[– 100, 100]	30	0
F18	Schwefel's 2.22	[– 100, 100]	30	0
F19	Schwefel's 2.23	[– 10, 10]	30	0
F20	Sphere	[– 100, 100]	30	0
F21	Sum Squares	[– 10, 10]	30	0
F22	Xin-She Yang 1	[– 20, 20]	30	0

Table 4 Descriptions of multimodal test functions

Function	Name	Range	Vars	f_{min}
F23	Ackley	[– 32, 32]	30	0
F24	Alpine N. 1	[– 10, 10]	30	0
F25	Griewank	[– 100, 100]	30	0
F26	Periodic	[– 10, 10]	30	0.9
F27	Rastrigin	[– 5.12, 5.12]	30	0
F28	Salomon	[– 100, 100]	30	0
F29	Trignometric 2	[– 500, 500]	30	0
F30	Xin-She Yang 2	[– 5,5]	30	0
F31	Xin-She Yang N. 2	[– 2pi, 2pi]	30	0
F32	Xin-She Yang N. 4	[– 10, 10]	30	– 1

Table 5 Descriptions of group IV test functions

Function	Name	Range	Vars	f_{min}
F33	Shifted and Rotated Rastrigin's Function (CEC4)	[– 100, 100]	10	400
F34	Shifted and Rotated Lunacek BiRastrigin Function (CEC6)	[– 100, 100]	10	600
F35	Shifted and Rotated Non-Continuous Rastrigin's Function (CEC7)	[– 100, 100]	10	700
F36	Shifted and Rotated Schwefel's Function (CEC9)	[– 100, 100]	10	900
F37	Hybrid Function 1 (N = 3) (CEC10)	[– 100, 100]	10	1000
F38	Hybrid Function 6 (N=4) (CEC15)	[– 100, 100]	10	1500
F39	Composite Function 1 (N = 3) (CEC20)	[– 100, 100]	10	2000
F40	Composite Function 6 (N = 5) (CEC25)	[– 100, 100]	10	2500

Table 6 Control parameters of the algorithms used in the tests

Algorithm	Control parameters
GA [82]	$P_c = 0.67, P_m = 0.33$
PSO [83]	$c_1 = 2, c_2 = 2, \omega = 0.2$
GSA [41]	$G_0 = 100, \alpha = 20, k = [N \rightarrow 1]$
SCA [55]	$\alpha = 2, r_1 = 1 - t\frac{2}{G}, r_2 \in [0.2\pi], r_3 \in [0, 2], r_4 \in [0, 1]$
SELO [56]	$P = 2, O = 3, r_p = 0.999, r_k = 0.1$, follow_prob_factor_ownparent = 0.999 follow_prob_factor_otherkids = 0.9991, $r = [0.95000 \rightarrow 0.99995]$
HBO [15]	$C = \lfloor G/25 \rfloor, p_1 = 1 - (t/G), p_2 = p_1 + \frac{1-p_1}{2}$
LFD [4]	Threshold = 2, CSV = 0.5, $\beta = 1.5, \alpha_1 = 10, \alpha_2 = 0.00005, \alpha_3 = 0.005, \partial_1 = 0.9, \partial_2 = 0.1$
SETO	Initial number of traders (T)=100

4.2 Comparison algorithms

The proposed SETO is compared with seven well-established optimization metaheuristics such as GA [82], PSO [83], GSA [41], SCA [55], SELO [56], HBO [15], and LFD [4] algorithms. GA, PSO, and GSA are three well-studied algorithms in science and engineering. SCA, SELO, HBO, and LFD are recently proposed efficient human-inspired optimization algorithms that obtain competitive results on single-objective unconstraint numerical function and constraint real-world engineering problems. SCA is an iterative math-inspired algorithm that uses the sine and cosine relations to search the solution space. SELO is inspired by the social learning behavior of humans organized as families. HBO models the organization of people in a hierarchy called corporate rank hierarchy (CRH). It uses the heap data structure to map the concept of CRH. LFD is inspired by the Levy flight motions and the wireless sensor networks environment.

4.3 Experimental setting

The experiments were performed using MATLAB 2016b on a Laptop machine with 8GB main memory and 64-bit i7 Intel (R) Core (TM) 2.2GHz processor. The population size (N), the maximum iteration number (G), and the maximum number of fitness function evaluations (FEs) for all the algorithms were set to be 25 and $10^3 \times D$, respectively. D indicates the dimension of problems. The configuration of control parameters for comparison algorithms is summarized in Table 6. The parameters are tuned as recommended in the corresponding literature. Most of the control parameters of SETO are already known and configured using the data drawn from the stock exchange and scientific resources about technical analysis. This issue turns the SETO into an optimizer quite easy to implement and execute. In the current implementation of the SETO algorithm, the only parameter that needs to be adjusted is the initial number of traders (T). As given in Table 6, the parameter T is set to 100. Different values of the variable T do not affect the performance of the algorithm. The parameter T is used to calculate the ratio of buyers to sellers (pc) and the ratio of sellers to buyers (nc). The values of pc and nc do not change significantly as the total number of traders increases or decreases. These parameters are limited to a value in the range [0, 2]. Regarding population size, it is obvious that with increasing population size, the performance of optimization algorithms improves, but also the execution time of the algorithms increases. However, the population size is considered the same for all algorithms. To fair comparison, the basic standard versions of the algorithms are used for tests. We used the source codes published by the authors and customized them to be compatible with our experimental configuration. The quality of solutions reported by the algorithms is calculated by the *Mean* and the standard deviation (*Std*) measures. In an ideal state, the *Mean* is equal to the global optimum of the problem, and the *std* is 0. As the *std* increases, the reliability of the algorithm decreases. To obtain the statistical results, the algorithms were executed 30 times on each test problem following the experimental instructions provided in [18, 84]. The results at each run are recorded to calculate the mean and the standard deviation of the best solutions found in 30 independent runs.

4.4 Numerical results and discussion

Tables 7, 8, 9, and 10 summarize the statistical results obtained by the proposed SETO and comparison algorithms. The main objective is to evaluate the performance of the comparison algorithms in finding the optimal solutions and measure the quality of the found solutions. In the tables, the symbol $\ominus$ means that SETO performs better than the counterpart algorithm on the specified test function, $\oplus$ indicates that the competing algorithm has performed better on the specified function than SETO, and $\odot$ indicates that both the competing algorithm and SETO have attained the same results. The best results are illustrated in boldface. Overall, SETO outperforms its counterparts in terms of statistical tests on most benchmark problems. Inspecting the results reported in Tables 7, 8, 9, and 10, we have the following observations:

Table 7 Statistical results on 30D fixed-dimension functions

Function	GA	PSO	GSA	SCA
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
F1	**– 2.01E–01 ± 4.30E–03** ⊙	**– 2.02E–01 ± 9.11E–16** ⊙	**– 2.02E–01 ± 4.45E–04** ⊙	**– 2.02E–01 ± 3.35E–10** ⊙
F2	**1.00E+00 ± 4.17E–02** ⊙	**1.00E + 00 ± 0.00E + 00**⊙	**1.00E+00 ± 1.19E–04** ⊙	**1.00E + 00 ± 0.00E + 00**⊙
F3	2.25E–05 ± 1.68E–04 ⊖	**1.38E–87 ± 5.17E–139** ⊙	9.11E–06 ± 1.69E–05 ⊖	**1.38E–87 ± 3.11E–105** ⊙
F4	**1.80E + 02 ± 0.00E + 00**⊙	**1.80E + 02 ± 0.00E + 00**⊙	**1.80E+02 ± 4.38E–04** ⊙	**1.80E + 02 ± 0.00E + 00**⊙
F5	**– 1.00E+00 ± 0.00E+00** ⊕	**– 1.00E+00 ± 0.00E+00** ⊕	**– 1.00E+00 ± 0.00E+00** ⊕	**– 1.00E+00 ± 3.51E–04** ⊕
F6	8.03E–1 ± 9.30E–18 ⊖	1.50E–130 ± 7.93E–129 ⊖	4.92E–21 ± 3.16E–18 ⊖	6.29E–160 ± 5.19E–159 ⊖
F7	2.06E–21 ± 5.32E–19 ⊖	1.53E–65 ± 2.78E–61 ⊖	2.39E–29 ± 1.22E–31 ⊖	2.41E–85 ± 2.50E–86 ⊖
F8	2.17E–01 ± 1.62E–02 ⊖	**2.93E–01 ± 4.01E–17** ⊙	2.01E–01 ± 1.50E–02 ⊖	**2.93E–01 ± 3.79E–09** ⊙
F9	2.70E–94 ± 3.17E–95 ⊖	3.50E–68 ± 4.58E–64 ⊖	1.44E–22 ± 9.60E–22 ⊖	2.13E–140 ± 3.60E–139 ⊖
F10	**– 3.79E–03 ± 1.91E–02** ⊙	**– 3.79E–03 ± 6.47E–15** ⊙	**– 3.79E–03 ± 5.02E–17** ⊙	**– 3.79E–03 ± 3.97E–10** ⊙
⊖	4	3	4	3
⊕	1	1	1	1
⊙	5	6	5	6
	SELO	HBO	LFD	SETO
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
F1	**– 2.02E–01 ± 2.30E–04** ⊙	**– 2.02E–01 ± 2.30E–04** ⊙	**– 2.02E–01 ± 2.05E–09** ⊙	**– 2.02E–01 ± 0.00E+00**
F2	**1.00E + 00 ± 0.00E + 00**⊙	**1.00E + 00 ± 0.00E + 00**⊙	**1.00E + 00 ± 0.00E + 00**⊙	**1.00E+00 ± 0.00E+00**
F3	**1.38E–87 ± 5.10E–95** ⊙	**1.38E–87 ± 2.60E–105** ⊙	**1.38E–87 ± 0.00E+00** ⊙	**1.38E–87 ± 0.00E+00**
F4	**1.80E + 02 ± 0.00E + 00**⊙	**1.80E + 02 ± 0.00E + 00**⊙	2.29E+02 ± 2.84E–21 ⊖	**1.80E+02 ± 0.00E+00**
F5	5.97E–03 ± 2.60E–03 ⊕	**– 1.00E+00 ± 0.00E+00** ⊕	**– 1.00E+00 ± 0.00E+00** ⊕	– 9.61E–01 ± 4.50E–01
F6	2.71E–100 ± 4.91E–100 ⊖	**0.00E + 00 ± 0.00E + 00**⊙	1.86E–12 ± 6.87E–13 ⊖	**0.00E+00 ± 0.00E+00**
F7	4.62E–43 ± 3.09E–41 ⊖	2.19E–109 ± 7.24E–138 ⊖	3.35E–19 ± 1.28E–18 ⊖	**0.00E+00 ± 0.00E+00**
F8	**2.93E–01 ± 3.41E–05** ⊙	**2.93E–01 ± 5.49E–19** ⊙	**2.93E–01 ± 1.28E–15** ⊙	**2.93E–01 ± 6.23E–21**
F9	6.02E–74 ± 1.55E–73 ⊖	**0.00E + 00 ± 0.00E + 00**⊙	8.27E–14 ± 2.69E–13 ⊖	**0.00E+00 ± 0.00E+00**

Table 7 (continued)

	SELO Mean ± Std	HBO Mean ± Std	LFD Mean ± Std	SETO Mean ± Std
F10	**– 3.79E–03 ± 5.53E–11 ⊙**	**– 3.79E–03 ± 2.03E–20 ⊙**	**– 3.79E–03 ± 4.31E–16 ⊙**	**– 3.79E–03 ± 6.14E–21**
⊖	4	1	4	–
⊕	1	1	1	–
⊙	5	8	5	–

Best results are illustrated in boldface

Table 8 Statistical results of 30D unimodal functions

Function	GA	PSO	GSA	SCA
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
F11	3.20E+01 ± 1.17E+00 ⊖	2.74E+01 ± 1.02E+01 ⊖	6.10E–18 ± 4.59E–19 ⊖	4.18E–07 ± 2.98E–06 ⊖
F12	7.93E+01 ± 2.94E+01 ⊖	4.31E+01 ± 9.22E+00 ⊖	6.73E–01 ± 3.07E–02 ⊖	2.19E+00 ± 1.42E+00 ⊖
F13	9.01E+03 ± 9.77E+02 ⊖	8.44E+02 ± 3.50E+02 ⊖	4.02E–03 ± 8.90E–02 ⊖	2.60E+00 ± 6.32E+01 ⊖
F14	3.88E–09 ± ⊖7.66E–10 ⊖	5.17E–11 ± 3.08E–12 ⊖	3.06E–18 ± 8.05E–19 ⊖	3.91E–10 ± 5.66E–10 ⊖
F15	1.44E+04 ± 2.56E+04 ⊖	2.36E+03 ± 1.08E+04 ⊖	2.89E+01 ± 1.40E+01 ⊖	6.92E+01 ± 8.01E+01 ⊖
F16	4.71E–05 ± 2.70E–05 ⊖	7.33E–07 ± 2.96E–06 ⊖	3.16E–09 ± 9.25E–09 ⊖	3.11E–05 ± 2.98E–05 ⊖
F17	2.05E+01 ± 7.30E+00 ⊖	3.30E–02 ± 7.92E–01 ⊖	6.41E–03 ± 3.77E–02 ⊖	2.11E+01 ± 1.13E+01 ⊖
F18	6.12E–01 ± 4.47E–01 ⊖	2.17E–02 ± 3.99E–02 ⊖	4.70E+01 ± 2.08E+01 ⊖	9.11E–06 ± 3.55E–07 ⊖
F19	1.54E–03 ± 4.39E–02 ⊖	4.17E–15 ± 2.78E–14 ⊖	9.15E–87 ± 3.08E–88 ⊖	2.19E+03 ± 7.33E+03 ⊖
F20	4.15E–20 ± 3.91E–20 ⊖	3.07E–13 ± 4.79E–14 ⊖	5.13E–16 ± 9.70E–17 ⊖	8.53E–04 ± 1.78E–05 ⊖
F21	5.35E–01 ± 2.70E+02 ⊖	7.15E–01 ± 2.68E+01 ⊖	9.66E–17 ± 4.37E–18 ⊖	6.91E–04 ± 2.73E–04 ⊖
F22	4.14E–55 ± 2.17E–48 ⊖	6.09E–102 ± 2.81E–107 ⊖	1.56E–41 ± 2.40E–42 ⊖	5.32E–190 ± 1.91E–184 ⊖
⊖	12	12	12	12
⊕	0	0	0	0
⊙	0	0	0	0
	SELO	HBO	LFD	SETO
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
F11	5.22E–78 ± 9.40E–81 ⊖	**0.00E + 00 ± 0.00E + 00⊙**	9.22E–01 ± 1.61E–01 ⊖	**0.00E+00 ± 0.00E+00**
F12	9.73E–01 ± 5.02E–04 ⊖	6.67E–01 ± 3.17E–04 ⊖	9.99E–01 ± 4.87E–02 ⊖	**6.66E–01 ± 4.61E–90**
F13	7.38E–02 ± 5.49E–02 ⊖	5.67E–07 ± 1.04E–07 ⊖	4.52E–06 ± 2.33E–08 ⊖	**0.00E+00 ± 0.00E+00**
F14	4.70E–25 ± 3.76E–24 ⊖	**0.00E + 00 ± 0.00E + 00⊙**	1.61E–05 ± 1.14E–04 ⊖	**0.00E+00 ± 0.00E+00**
F15	2.90E+01 ± 1.90E+01 ⊖	7.76E+01 ± 4.02E+01 ⊖	2.94E–02 ± 6.32E–01 ⊖	**2.86E+01 ± 0.00E+00**
F16	3.66E–20 ± 7.70E–20 ⊖	**0.00E + 00 ± 0.00E + 00⊙**	1.46E–01 ± 2.04E–01 ⊖	**0.00E+000.00E+00**
F17	1.94E–32 ± 5.44E–35 ⊖	7.27E–47 ± 2.85E–55 ⊖	7.08E–04 ± 6.13E–04 ⊖	**0.00E+00 ± 0.00E+00**

Table 8 (continued)

	SELO Mean ± Std	HBO Mean ± Std	LFD Mean ± Std	SETO Mean ± Std
F18	5.24E–63 ± 6.14E–62 ⊖	**0.00E + 00 ± 0.00E + 00**⊙	6.40E–32 ± 5.28E–33 ⊖	**0.00E+00 ± 0.00E+00**
F19	4.85E–101 ± 1.16E–102 ⊖	**0.00E + 00 ± 0.00E + 00**⊙	1.11E–40 ± 8.76E–41 ⊖	**0.00E+00 ± 0.00E+00**
F20	3.54E–55 ± 1.67E–57 ⊖	**0.00E + 00 ± 0.00E + 00**⊙	7.25E–05 ± 4.75E–25 ⊖	**0.00E+00 ± 0.00E+00**
F21	**0.00E + 00 ± 0.00E + 00**⊙	**0.00E + 00 ± 0.00E + 00**⊙	5.00E–04 ± 2.71E–04 ⊖	**0.00E+00 ± 0.00E+00**
F22	1.26E–94 ± 7.19E–96 ⊖	4.34E–232 ± 0.00E+00 ⊖	**0.00E + 00 ± 0.00E + 00**⊙	**0.00E+00 ± 0.00E+00**
⊖	11	5	11	–
⊕	0	0	0	–
⊙	1	7	1	–

Best results are illustrated in boldface

Table 9 Statistical results of 30D multimodal functions

Function	GA	PSO	GSA	SCA
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
F23	1.70E+01 ± 2.57E+00 ⊖	2.30E+00 ± 5.54E+01 ⊖	7.20E–10 ± 6.31E–10 ⊖	1.25E+01 ± 7.42E+00 ⊖
F24	5.10E–03 ± 8.11E–02 ⊖	7.90E–02 ± 8.30E–01 ⊖	3.22E–10 ± 1.55E–11 ⊖	3.50E–02 ± 7.80E–02 ⊖
F25	1.16E+01 ± 4.33E+00 ⊖	1.16E–02 ± 5.33E–03 ⊖	5.42E–04 ± 8.12E–04 ⊖	3.81E–01 ± 5.12E–01 ⊖
F26	6.80E–01 ± 2.05E–03 ⊖	1.00E+00 ± 3.59E–09 ⊖	1.00E+00 ± 7.36E–15 ⊖	2.17E+00 ± 5.02E+00 ⊖
F27	8.37E+00 ± 4.68E+00 ⊖	1.46E+01 ± 2.07E+01 ⊖	1.63E+01 ± 5.18E+00 ⊖	1.04E+01 ± 1.53E+01 ⊖
F28	9.12E–01 ± 4.29E–02 ⊖	4.35E–01 ± 3.09E–02 ⊖	1.40E+00 ± 3.30E–01 ⊖	2.10E–01 ± 4.87E–02 ⊖
F29	**4.32E+00 ± 2.25E+00** ⊕	**7.51E+00 ± 2.41E+00** ⊕	6.56E+03 ± 7.13E+02 ⊖	9.39E+01 ± 1.70E+01 ⊖
F30	3.75E+03 ± 6.56E+01 ⊖	5.67E+04 ± 3.87E+02 ⊖	7.13E–04 ± 6.64E–03 ⊖	3.64E–04 ± 9.25E–05 ⊖
F31	5.52E–11 ± 3.14E+11 ⊖	1.04E–09 ± 4.97E–12 ⊖	9.14E–12 ± 3.32E–15 ⊖	7.16E–11 ± 1.90E–12 ⊖
F32	7.41E–09 ± 6.63E–08 ⊖	6.18E–11 ± 2.57E–10 ⊖	4.60E–31 ± 2.77E–30 ⊖	7.40E–11 ± 1.27E–10 ⊖
⊖	9	9	10	10
⊕	1	1	0	0
⊙	0	0	0	0
	SELO	HBO	LFD	SETO
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
F23	4.67E–17 ± 9.13E–16 ⊖	6.22E–15 ± 6.22E–15 ⊖	2.87E–21 ± 3.41E–37 ⊖	**0.00E+00 ± 0.00E+00**
F24	4.50E–02 ± 3.17E–03 ⊖	8.09E–28 ± 7.13E–30 ⊖	1.99E–07 ± 2.45E–06 ⊖	**0.00E+00 ± 0.00E+00**
F25	6.23E–25 ± 8.70E–26 ⊖	**0.00E + 00 ± 0.00E + 00**⊙	3.86E–06 ± 1.80E–06 ⊖	**0.00E+00 ± 0.00E+00**
F26	**9.00E–01 ± 4.13E–05** ⊙	1.00E+00 ± 1.06E–06 ⊖	**9.00E–01 ± 0.00E+00** ⊙	**9.00E–01 ± 0.00E+00**
F27	9.12E–27 ± 1.45E–30 ⊖	**0.00E + 00 ± 0.00E + 00**⊙	6.73E–07 ± 4.97E–07 ⊖	**0.00E+00 ± 0.00E+00**
F28	1.03E+00 ± 3.46E–01 ⊖	9.99E–02 ± 0.00E+00 ⊖	3.40E–06 ± 8.22E–07 ⊖	**0.00E+00 ± 0.00E+00**
F29	3.97E+02 ± 1.04E+02 ⊖	**1.00E+00 ± 5.60E–18** ⊕	6.34E+01 ± 7.16E+01 ⊖	1.13E+01 ± 3.71E+01
F30	2.88E–15 ± 1.64E–17 ⊖	8.10E–26 ± 5.20E–254 ⊖	2.39E–10 ± 5.90E+17 ⊖	**0.00E+00 ± 0.00E+00**
F31	8.16E–04 ± 3.55E–05 ⊖	3.51E–12 ± 1.82E–15 ⊖	5.38E–12 ± 2.17E–36	**0.00E+00 ± 5.10E–234**

Table 9 (continued)

	SELO Mean ± Std	HBO Mean ± Std	LFD Mean ± Std	SETO Mean ± Std
F32	2.55E–06 ± 3.71E–06 ⊖	7.35E–30 ± 4.37E–29 ⊖	**– 1.00E+00 ± 0.00E+00** ⊙	**– 1.00E+00 ± 0.00E+00**
⊖	9	7	8	–
⊕	0	1	0	–
⊙	1	2	2	–

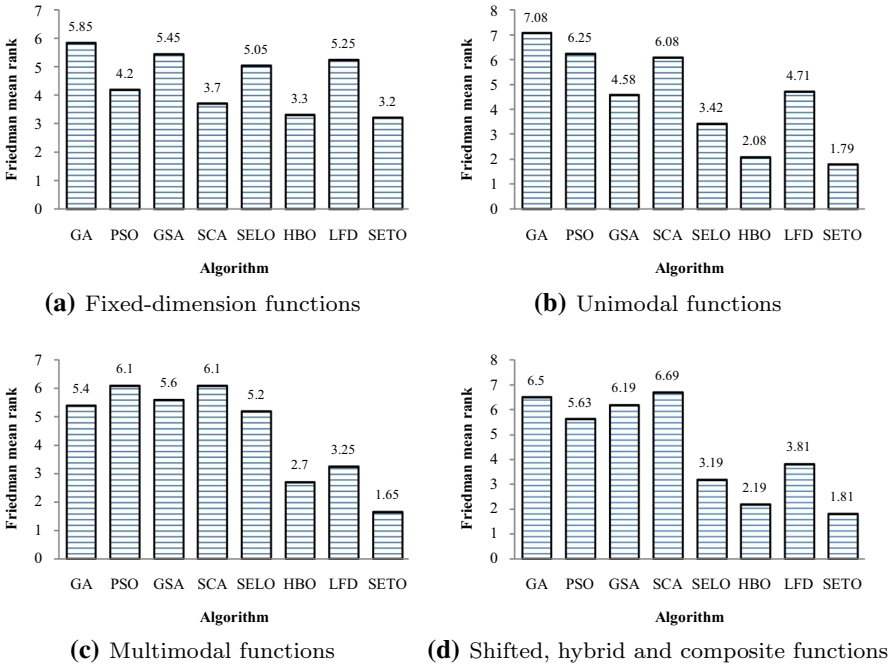
Best results are illustrated in boldface

Table 10 Statistical results on 10D group IV test function

Function	GA	PSO	GSA	SCA
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
F33	4.08E+02 ± 5.18E+00 ⊖	4.05E+02 ± 7.95E+00 ⊖	4.05E+02 ± 4.71E+00 ⊖	4.29E+02 ± 1.51E+01 ⊖
F34	6.41E+02 ± 1.25E+01 ⊖	6.31E+02 ± 3.40E+00 ⊖	6.19E+02 ± 6.03E+00 ⊖	6.18E+02 ± 5.52E+00 ⊖
F35	7.27E+02 ± 3.59E+00 ⊖	7.42E+02 ± 6.63E+00 ⊖	7.39E+02 ± 2.92E+00 ⊖	7.63E+02 ± 4.92E+00 ⊖
F36	9.03E+02 ± 4.70E+00 ⊖	**9.00E + 02 ± 0.00E + 00** ⊙	**9.00E + 02 ± 0.00E + 00** ⊙	9.56E+02 ± 3.97E+01 ⊖
F37	1.68E+03 ± 9.80E+01 ⊖	1.63E+03 ± 1.89E+02 ⊖	2.53E+03 ± 2.18E+02 ⊖	2.24E+03 ± 1.07E+02 ⊖
F38	1.76E+03 ± 1.28E+02 ⊖	1.80E+03 ± 1.99E+02 ⊖	9.82E+03 ± 1.32E+03 ⊖	2.13E+03 ± 2.16E+02 ⊖
F39	2.10E+03 ± 1.76E+01 ⊖	2.16E+03 ± 4.91E+01 ⊖	2.19E+03 ± 4.17E+01 ⊖	2.07E+03 ± 1.94E+01 ⊖
F40	2.63E+03 ± 6.23E+01 ⊖	2.65E+03 ± 9.24E+01 ⊖	2.67E+03 ± 4.30E+00 ⊖	2.62E+03 ± 9.71E+00 ⊖
⊖	8	7	7	8
⊕	0	0	0	0
⊙	0	1	1	0
	SELO	HBO	LFD	SETO
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
F33	4.02E+02 ± 1.32E+00 ⊖	**4.00E + 02 ± 0.00E + 00** ⊙	**4.02E + 02 ± 2.15E + 00** ⊙	**4.00E+02 ± 0.00E+00**
F34	6.02E+02 ± 1.43E+01 ⊖	**6.00E + 02 ± 0.00E + 00** ⊙	6.05E+02 ± 6.22E+00 ⊖	**6.00E+02 ± 0.00E+00**
F35	7.13E+02 ± 6.90E+00 ⊖	7.10E+02 ± 3.79E−17 ⊖	7.26E+02 ± 3.50E+01 ⊖	**7.00E+02 ± 2.41E−15**
F36	**9.00E + 02 ± 0.00E + 00** ⊙	**9.00E + 02 ± 0.00E + 00** ⊙	**9.00E + 02 ± 0.00E + 00** ⊙	**9.00E+02 ± 0.00E+00**
F37	1.62E+03 ± 5.20E+01 ⊖	**1.00E + 03 ± 0.00E + 00** ⊙	1.34E+03 ± 6.76E+01 ⊖	**1.00E+03 ± 0.00E+00**
F38	1.68E+03 ± 4.30E+01 ⊖	1.60E+03 ± 1.15E+02 ⊖	1.84E+03 ± 4.30E+01 ⊖	**1.58E+03 ± 5.60E+01**
F39	2.04E+03 ± 2.60E+01 ⊖	**2.00E+03 ± 6.40E−01** ⊕	2.05E+03 ± 1.13E+02 ⊖	2.01E+03 ± 7.22E+01
F40	2.63E+03 ± 3.90E+01 ⊖	2.56E+03 ± 8.30E+01 ⊕	**2.55E+03 ± 6.70E+01** ⊕	2.58E+03 ± 7.64E+01
⊖	7	2	5	–
⊕	0	2	1	–
⊙	1	4	2	–

Best results are illustrated in boldface

- In the case of fixed-dimension test cases, the SETO and HBO take 1st rank for all test functions in terms of best mean results. However, in terms of *std*, the first position belongs to SETO, which shows its stable convergence behavior in solving fixed-dimension problems. Both SCA and PSO attain third rank among others. GA, PSO, GSA, SCA, SELO, HBO, LFD, and SETO, respectively, generate 5, 7, 5, 7, 6, 9, 6, and 9 best mean results out of the total 10 functions. From the results given in Table 7, it is evident that both SETO and HBO have excellent exploitation ability; however, SETO is more stable than HBO. The high exploitation power of SETO is due to two reasons. First, the algorithm updates the position of shares in the search space if the next positions are better than precedent positions. Second, shares move toward the best solution from different directions at each generation that helps them jump out of local optima. Figure 5a illustrates the results of the Friedman mean rank test [85] on fixed-dimension functions. The Friedman mean rank value of SETO is minimum, which shows that it obtains 1st rank compared with other algorithms.
- The results reported by SETO in solving unimodal functions are superior. It generates the best mean results in all test functions. The second rank belongs to HBO with 7 best mean results out of the total 12. This confirms that SETO has superior exploitation power and convergence speed in solving unimodal functions. GA, PSO, GSA, SCA, SELO, HBO, LFD, and SETO, respectively, generate 0, 0, 0, 0, 1, 7, 1, and 12 best mean results out of the total 12 functions. Inspecting the *std* values shows that SETO attains the best standard deviations among other algorithms, which confirms its stability in the searching process. Figure 5b shows the results of the Friedman test on unimodal functions. As shown in the plot, SETO obtains the best mean rank among others.
- As shown in Table 9, SETO is very powerful in solving multimodal functions. It generates the best mean results for all test functions except F29. Inspecting the results, we conclude that SETO significantly outperforms its counterparts due to its high exploration power. The reason for this success lies in the position updating mechanism in the rising phase, in which the shares jump out of the local optima and move toward the best solution from different directions. GA, PSO, GSA, SCA, SELO, HBO, LFD, and SETO, respectively, generate 1, 1, 0, 0, 1, 3, 2, and 9 best mean results out of the total 10 functions. As shown in Fig. 5c, the SETO attains 1st position and HBO 2nd rank among all algorithms on multimodal functions.
- The performance of SETO in solving group IV shifted and rotated, hybrid and composite functions is superior, and it outperformed other algorithms on F33–F38 functions. For F39 and F40, HBO and LFD generate the best mean results, respectively. The mean results for F39 and F40, where SETO is not the top performer algorithm, are still very comparable and competitive to the best results attained by HBO and LFD. As illustrated in Fig. 5d, SETO attains the best mean rank among others in solving group IV functions. This confirms that SETO can provide a proper balance between exploitation and exploration mechanisms in solving complex and difficult problems. GA, PSO, GSA, SCA, SELO, HBO, LFD, and SETO, respectively, generate 0, 1, 1, 0, 1, 5, 3, and 6 best mean results out of the total 8 functions.



(a) Fixed-dimension functions **(b)** Unimodal functions

(c) Multimodal functions **(d)** Shifted, hybrid and composite functions

Fig. 5 Statistical result of Friedman mean rank test for benchmark functions

The key factor to efficient search is the proper harmonization between exploration (diversification) and exploitation (intensification). In the SETO algorithm, the rising operator is responsible for exploring the search space, and the falling operator is responsible for exploiting the promising areas. The rising operator directs the search agents (shares) in the solution space to explore unvisited areas and finds the promising areas, whilst the falling operator tries to carefully examine the inside of the promising areas via accumulated local knowledge. The falling operator moves the solutions far from the current area of search so that explorative move should reach all the regions within search space accessed at least once. On the other hand, using local experience, the falling operator forces the solutions to converge quickly without wasting too many moves. The results confirm that SETO can provide a proper balance between exploitation and exploration mechanisms in the search and optimization process.

Figure 6 presents the mean and overall ranks of comparison algorithms computed by the nonparametric Friedman test [85] on all benchmark functions. The results reveal that SETO obtains 1st overall rank and HBO obtains 2nd rank among all algorithms. The third and fourth ranks belong to SELO and LFD, respectively. The difference between the LFD and SELO is insignificant and minute. GA is ranked last. This phenomenon suggests that the introduction of new algorithms or the improvement of existing ones is needed to solve classic and modern optimization problems.

Table 11 presents the results of the multi-problem-based Wilcoxon signed-rank test [85] at significant level $\alpha = 0.05$ for benchmark functions. This test is performed

to determine the significant differences between the reported results by comparison algorithms. In Table 11, the SETO is the control algorithm. The results show that the SETO is statistically successful than its counterparts in solving test functions.

To show the quantitative differences between the results of SETO and those attained by comparison algorithms on benchmark functions, we perform a contrast estimation [85]. The objective is to determine by how far the SETO outperforms its counterparts. As shown in Table 12, SETO has a significant difference with other algorithms that shows its good optimization ability on different test problems.

4.5 Scalability analysis

The convergence speed and optimization ability of algorithms will decrease as the dimension of problems increases. To investigate this issue, we performed a series of tests on 1000-dimension benchmark functions to evaluate the scalability of algorithms. The experiments are performed on scalable unimodal functions F11–F22 and multimodal functions F23–F32. The algorithms terminate when they reach the global optimum point, or they have failed to find a better solution than the existing solution during the last 50,000 FEs. The results are listed in Tables 13 and 14. From the results, it can be concluded that SETO attains all the best mean results in 1000 dimension problems except F15. However, the best mean result for F15 is very competitive to the best result. SELO, LFD, and HBO generate good performances; however, their difference with SETO is not minute. The results confirm the superior scalability of SETO compared with its counterparts. Figure 7 illustrates the execution time consumed by algorithms to reach the global optimum. From the figure, we observe that SETO takes less execution time than other algorithms in most test functions. SETO performs exploration and exploration at the same time and converges faster. Therefore, SETO has less search time than other algorithms. After reaching the global optimum, the solutions do not change, and according to the termination conditions mentioned in Sect. 3.2.6, the algorithm stops.

4.6 Convergence test

To investigate the searching performance of the algorithms, we perform a convergence test on five candidate functions F6, F11, F28, F35, and F40 as representatives from each benchmark group of fixed-dimension, unimodal, multimodal, shifted and rotated, and composite test functions, respectively. Figure 8 illustrates the graphical representation, convergence plot, and distribution of solutions for test functions. The convergence plots confirm that the SETO avoids premature convergence; however, it converges relatively faster than other optimizers in solving different types of test functions. This is due to the efficient exploitation, exploration, and local avoidance ability of SETO compared with others. As shown in the box-and-whisker plots, SETO generates solutions with the minimum dispersion, which proves its stability in the search process.

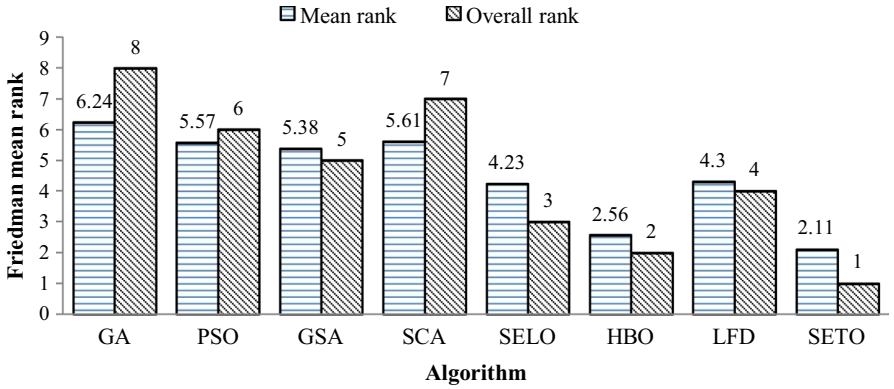


Fig. 6 The mean and overall ranks of optimization algorithms computed by Friedman test for all benchmark functions

Table 11 Results of multi-problem-based two-sided Wilcoxon signed-rank test at 0.05 significant level for SETO against counterpart algorithms on benchmark functions

SETO vs.	T+	T–	*p* value	winner
GA	466	30	0.00001	SETO
PSO	358	20	0.00001	SETO
GSA	281	19	0.00001	SETO
SCA	427	8	0.00001	SETO
SELO	168	22	0.00001	SETO
HBO	65.5	25.5	0.00298	SETO
LFD	220	56	0.00016	SETO

4.7 Computational complexity

4.7.1 Time complexity

The time complexity of SETO is calculated as follows:

- The population initialization phase costs $O(ND)$.
- Calculating the initial fitness of all shares needs $O(NC)$, where C indicates the cost of the objective function.
- The time complexity of the rising phase bounded by $O(ND + NC)$.
- The falling phase costs $O(ND + NC)$.
- The exchange phase costs $O(N)$.
- The time complexity of the RSI calculation phase is $O(N)$.

The overall time complexity of SETO within one iteration in the worst case can be calculated as

Table 12 Contrast estimation between optimization algorithms on all test problems

	GA	PSO	GSA	SCA	SELO	HBO	LFD	SETO
GA	0	– 0.2255	– 0.2400	– 0.1660	– 0.2435	– 0.3139	– 0.3520	– 0.3792
PSO	0.2255	0	– 0.0145	0.0595	– 0.0181	– 0.0884	– 0.1265	– 0.1538
GSA	0.2400	0.0145	0	0.0740	– 0.0036	– 0.0739	– 0.1120	– 0.1393
SCA	0.1660	– 0.0595	– 0.0740	0	– 0.0775	– 0.1479	– 0.1860	– 0.2132
SELO	0.2435	0.0181	0.0036	0.0775	0	– 0.0704	– 0.1085	– 0.1357
HBO	0.3139	0.0884	0.0739	0.1479	0.0704	0	– 0.0381	– 0.0654
LFD	0.3520	0.1265	0.1120	0.1860	0.1085	0.0381	0	– 0.0273
SETO	0.3792	0.1538	0.1393	0.2132	0.1357	0.0654	0.0273	0

$$O(ND) + O(NC) + O(ND + NC) + O(ND + NC) + O(N) + O(N) = \\ O(ND) + O(NC) + 2O(ND + NC) + 2O(N) \tag{29}$$

Since the cost of computing objective function varies for each optimization problem, Eq. (29) can be revised as follows:

$$\begin{cases} O(ND) + O(ND) + 2O(ND + ND) + 2O(N) \approx O(ND) & \text{if}(D > C) \\ O(NC) + O(NC) + 2O(NC + NC) + 2O(N) \approx O(NC) & \text{otherwise} \end{cases} \tag{30}$$

The overall time complexity of SETO is $O(GND)$ or $O(GNC)$ when the algorithm iterates for G iterations. The overall time complexity of GA, PSO, GSA, SCA, SELO, HBO, and LFD is $O(GND)$ in the worst case. The time complexity of the SETO is asymptotically equivalent to its counterparts. This proves that the SETO is computationally efficient compared with other algorithms.

4.7.2 Space complexity

The proposed SETO needs $O(N \times D)$ space to store population at each generation, where N denotes the population size, and D is the number of dimensions of problems. Besides, the algorithm uses $O(N)$ space to store the fitness of shares. The overall space complexity of the SETO is $O(ND)$.

5 Engineering problems

To show the applicability of the SETO algorithm on real-world problems, we applied it to four well-studied engineering problems including three-bar truss design, rolling element bearing design, pressure vessel design, and speed reducer design. Since the engineering problems consist of several constraints, the SETO is equipped with a constraint handling method to handle the design constraints. In this way, if each of the solutions violates the constraints, the algorithm ignores that solution and regenerates a valid one instead.

Table 13 Statistical results of 1000D unimodal functions

Function	GA	PSO	GSA	SCA
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
F11	8.66E+15 ± 1.52E+06 ⊖	4.40E+02 ± 7.51E+00 ⊖	8.10E+05 ± 3.86E+00 ⊖	4.86E+03 ± 6.84E+02 ⊖
F12	2.53E+09 ± 8.96E+05 ⊖	2.06E+05 ± 2.75E+02 ⊖	1.07E+07 ± 5.54E+03 ⊖	4.28E+08 ± 6.55E+06 ⊖
F13	9.31E+05 ± 8.64E+02 ⊖	9.44E+03 ± 9.80E+01 ⊖	2.08E+04 ± 9.34E+02 ⊖	5.23E–02 ± 1.70E–03 ⊖
F14	2.97E–02 ± 6.00E–04 ⊖	2.92E+00 ± 5.00E–03 ⊖	1.45E–17 ± 6.35E–18 ⊖	3.19E+05 ± 3.25E+03 ⊖
F15	1.12E+10 ± 6.47E+02 ⊖	1.95E+04 ± 3.65E+02 ⊖	1.51E+07 ± 9.17E+05 ⊖	2.16E+09 ± 6.32E+07 ⊖
F16	4.28E+04 ± 1.28E+02 ⊖	4.76E+02 ± 8.47E+01 ⊖	5.71E+03 ± 6.02E+02 ⊖	1.06E+03 ± 3.27E+02 ⊖
F17	9.78E+01 ± 1.65E+00 ⊖	9.95E–01 ± 5.00E–03 ⊖	3.18E+01 ± 1.52E+01 ⊖	9.91E+01 ± 1.60E+01 ⊖
F18	inf ⊖	4.83E+02 ± 1.39E+01 ⊖	6.31E+01 ± 3.12E+00 ⊖	inf ⊖
F19	3.54E+03 ± 1.56E+01 ⊖	8.27E+01 ± 5.00E–02 ⊖	4.62E+06 ± 6.15E+03 ⊖	1.51E+11 ± 3.65E+06 ⊖
F20	2.64E+06 ± 5.61E+02 ⊖	3.11E+02 ± 3.50E+01 ⊖	**0.00E + 00 ± 0.00E + 00⊙**	3.08E+05 ± 2.49E+04 ⊖
F21	1.23E+07 ± 6.12E+03 ⊖	1.56E+05 ± 1.20E–02 ⊖	4.23E+05 ± 3.31E+02 ⊖	1.30E+06 ± 2.14E+04 ⊖
F22	**0.00E + 00 ± 0.00E + 00⊙**	**0.00E + 00 ± 0.00E + 00⊙**	**0.00E + 00 ± 0.00E + 00⊙**	**0.00E + 00 ± 0.00E + 00⊙**
⊖	11	11	10	11
⊕	0	0	0	0
⊙	1	1	2	1
	SELO	HBO	LFD	SETO
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
F11	5.12E–04 ± 9.45E–05 ⊖	1.45E+01 ± 5.02E+00 ⊖	1.66E–06 ± 8.55E–08 ⊖	**0.00E+00 ± 0.00E+00**
F12	1.00E+00 ± 6.00E–03 ⊖	1.56E+05 ± 1.26E+03 ⊖	1.00E+00 ± 4.15E–28 ⊖	**6.67E–01 ± 0.00E+00**
F13	2.58E–05 ± 7.00E–04 ⊖	8.61E+02 ± 7.64E+00 ⊖	2.53E–06 ± 1.19E–08 ⊖	**0.00E+00 ± 0.00E+00**
F14	5.66E–06 ± 6.48E–08 ⊖	7.19E–09 ± 1.75E–10 ⊖	1.77E–03 ± 2.56E–05 ⊖	**0.00E+00 ± 0.00E+00**
F15	9.97E+02 ± 6.03E+01 ⊕	1.76E+03 ± 1.42E+02 ⊖	**9.89E+02 ± 3.26E–02 ⊕**	9.99E+02 ± 0.00E+00
F16	2.47E–04 ± 3.60E–04 ⊖	2.05E+01 ± 6.56E+00 ⊖	1.98E–01 ± 3.90E–04 ⊖	**0.00E+00 ± 0.00E+00**
F17	7.97E+01 ± 7.55E–01 ⊖	9.87E+01 ± 8.05E+00 ⊖	6.94E–04 ± 6.31E–07 ⊖	**0.00E+00 ± 0.00E+00**
F18	1.30E+193 ± 6.64E+85 ⊖	inf ⊖	inf ⊖	**0.00E+00 ± 0.00E+00**

Table 13 (continued)

Function	GA Mean ± Std	PSO Mean ± Std	GSA Mean ± Std	SCA Mean ± Std
F19	1.15E–03 ± 4.70E–05 ⊖	6.64E+06 ± 5.64E+03 ⊖	7.01E–43 ± 3.95E–45 ⊖	**0.00E+00 ± 0.00E+00**
F20	2.60E–08 ± 6.00E–06 ⊖	6.27E–03 ± 3.40E–04 ⊖	6.32E–05 ± 8.32E–07 ⊖	**0.00E+00 ± 0.00E+00**
F21	1.03E–07 ± 5.60E–04 ⊖	1.39E+03 ± 9.66E+01 ⊖	3.01E-04 ± 1.70E–05 ⊖	**0.00E+00 ± 0.00E+00**
F22	**0.00E + 00 ± 0.00E + 00**⊙	**0.00E + 00 ± 0.00E + 00**⊙	**0.00E + 00 ± 0.00E + 00**⊙	**0.00E+00 ± 0.00E+00**
⊖	10	11	10	–
⊕	1	1	1	–
⊙	1	1	1	–

Best results are illustrated in boldface

Table 14 Statistical results of 1000D multimodal functions

Function	GA	PSO	GSA	SCA
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
F23	2.09E+01 ± 5.19E+00 ⊖	3.83E+00 ± 1.70E–01 ⊖	9.41E+00 ± 2.15E+00 ⊖	2.08E+01 ± 2.16E+00 ⊖
F24	2.39E+03 ± 1.26E+01 ⊖	2.95E+02 ± 2.30E+01 ⊖	3.79E+02 ± 5.56E+01 ⊖	2.82E+02 ± 2.37E+00 ⊖
F25	2.29E+04 ± 7.52E+01 ⊖	7.00E–01 ± 0.00E+00 ⊖	6.64E+01 ± 1.53E+01 ⊖	5.78E+01 ± 7.01E+00 ⊖
F26	3.80E+02 ± 5.80E+01 ⊖	2.43E+02 ± 1.30E–01 ⊖	1.69E+02 ± 1.08E+01 ⊖	3.61E+02 ± 2.60E+01 ⊖
F27	1.56E+04 ± 8.24E+02 ⊖	4.64E+03 ± 6.70E+01 ⊖	3.78E+01 ± 2.40E+01 ⊖	2.66E+03 ± 9.60E+01 ⊖
F28	1.62E+02 ± 1.14E+01 ⊖	1.80E+00 ± 6.00E–10 ⊖	3.61E+01 ± 7.68E+00 ⊖	6.15E+01 ± 6.06E+00 ⊖
F29	6.31E+07 ± 9.52E+03 ⊖	2.84E+03 ± 4.60E+01 ⊖	3.13E+06 ± 3.55E+04 ⊖	7.57E+06 ± 5.82E+04 ⊖
F30	inf ⊖	1.46E+00 ± 3.20E–03 ⊖	1.32E+102 ± 1.56E+36 ⊖	inf ⊖
F31	4.10E–182 ± 6.30E–185 ⊖	**0.00E + 00 ± 0.00E + 00⊙**	**0.00E+00 ± 0.00E+00 ⊙**	**0.00E + 00 ± 0.00E + 00⊙**
F32	**0.00E + 00 ± 0.00E + 00⊙**	**0.00E+00 ± 0.00E+00 ⊙**	**0.00E + 00 ± 0.00E + 00⊙**	**0.00E + 00 ± 0.00E + 00⊙**
⊖	9	8	8	8
⊕	0	0	0	0
⊙	1	2	2	2
	SELO	HBO	LFD	SETO
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
F23	3.11E–04 ± 3.16E–07 ⊖	2.45E+00 ± 1.08E+00 ⊖	3.24E–04 ± 2.61E–04 ⊖	**– 8.88E–16 ± 0.00E+00**
F24	1.99E–38 ± 4.74E–39 ⊖	5.82E–03 ± 3.64E–05 ⊖	2.58E–03 ± 3.20E–02 ⊖	**0.00E+00 ± 0.00E+00**
F25	2.22E–16 ± 2.38E–17 ⊖	4.39E–05 ± 7.00E–06 ⊖	1.34E–07 ± 6.80E–08 ⊖	**0.00E+00 ± 0.00E+00**
F26	**9.00E–01 ± 5.53E–24 ⊙**	8.74E+00 ± 2.10E+00 ⊖	**9.00E–01 ± 0.00E+00 ⊙**	**9.00E–01 ± 0.00E+00**
F27	6.36E–04 ± 3.16E–05 ⊖	1.09E+03 ± 6.52E+02 ⊖	8.24E–06 ± 1.20E–05 ⊖	**0.00E+00 ± 0.00E+00**
F28	3.00E–01 ± 5.60E–03 ⊖	1.44E+01 ± 2.50E+00 ⊖	1.21E–03 ± 4.80E–04 ⊖	**0.00E+00 ± 0.00E+00**
F29	3.09E+03 ± 9.90E+01 ⊖	6.32E+03 ± 5.53E+02 ⊖	3.59E+03 ± 9.75E+02 ⊖	**3.59E+01 ± 1.13E+01**
F30	1.56E–53 ± 3.51E–55 ⊖	inf ⊖	inf ⊖	**0.00E+00 ± 0.00E+00**

Table 14 (continued)

Function	GA Mean ± Std	PSO Mean ± Std	GSA Mean ± Std	SCA Mean ± Std
F31	1.75E–143 ± 3.07E–148 ⊖	**0.00E + 00 ± 0.00E + 00⊙**	**0.00E + 00 ± 0.00E + 00⊙**	**0.00E+00 ± 0.00E+00**
F32	**0.00E + 00 ± 0.00E + 00⊙**	**0.00E + 00 ± 0.00E + 00⊙**	**0.00E + 00 ± 0.00E + 00⊙**	**0.00E+00 ± 0.00E+00**
⊖	8	8	7	–
⊕	0	0	0	–
⊙	2	2	3	–

Best results are illustrated in boldface

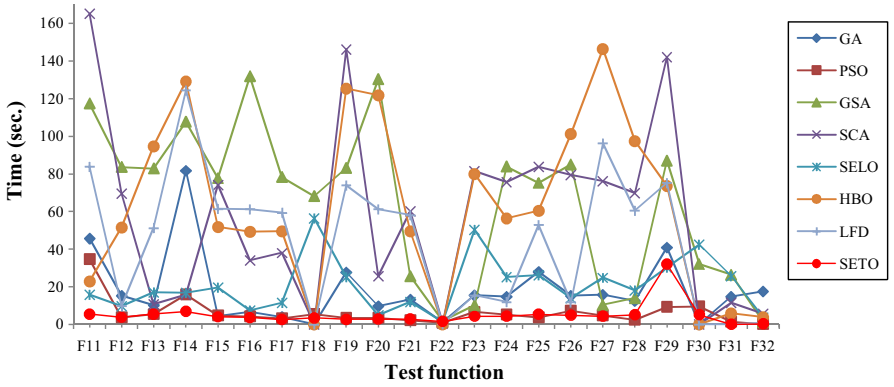


Fig. 7 Comparison of the execution time of algorithms in 1000D unimodal and multimodal functions

5.1 Three-bar truss design problem

Figure 9a shows the structure of the three-bar truss design problem. This problem is one of the most studied test cases used in the literature [7, 36]. The objective is to design a truss with three bars so that its weight to be minimal. The problem has two parameters including the area of bars 1 and 3 and the area of bar 2. To design the truss, three constraints should be considered: stress, deflection, and buckling. The problem is mathematically defined as follows:

$$\begin{aligned}
&\text{let} && \vec{X} = [x_1, x_2] = [A_1, A_2] \\
&\text{minimize} && f(\vec{X}) = l \times (2\sqrt{2x_1} + x_2)\ 0 \le \mathrm{x}_1, x_2 \le 1 \\
&\text{subject to} && g_1(\vec{X}) = \frac{\sqrt{2x_1 + x_2}}{\sqrt{2x_1^2 + 2x_1x_2)}} P - \sigma \le 0 \\
& && g_2(\vec{X}) = \frac{x_2}{\sqrt{2x_1^2 + 2x_1x_2)}} P - \sigma \le 0 \\
& && g_3(\vec{X}) = \frac{1}{\sqrt{2x_2 + x_1)}} P - \sigma \le 0 \\
&\text{where} && l = 100cm,\ \ P = 2KN/cm^2, \text{and}\ \ \sigma = 2\mathrm{KN/cm}^2
\end{aligned} \tag{31}$$

Table 15 compares the optimization results obtained by algorithms on the three-bar truss design problem. To generate results, the algorithms were iterated 30 times, each time with a different initial population. The population size and the *FEs* are set to 25 and 100,000, respectively. The results confirm that SETO outperforms other algorithms in finding the optimal parameters and the weight of the truss.

5.2 Rolling element bearing design problem

Figure 9b shows the schematic view of the rolling element bearing design problem. It is a maximization problem, which contains ten geometric variables and nine design constraints to control the assembly and geometric-based restrictions [15].

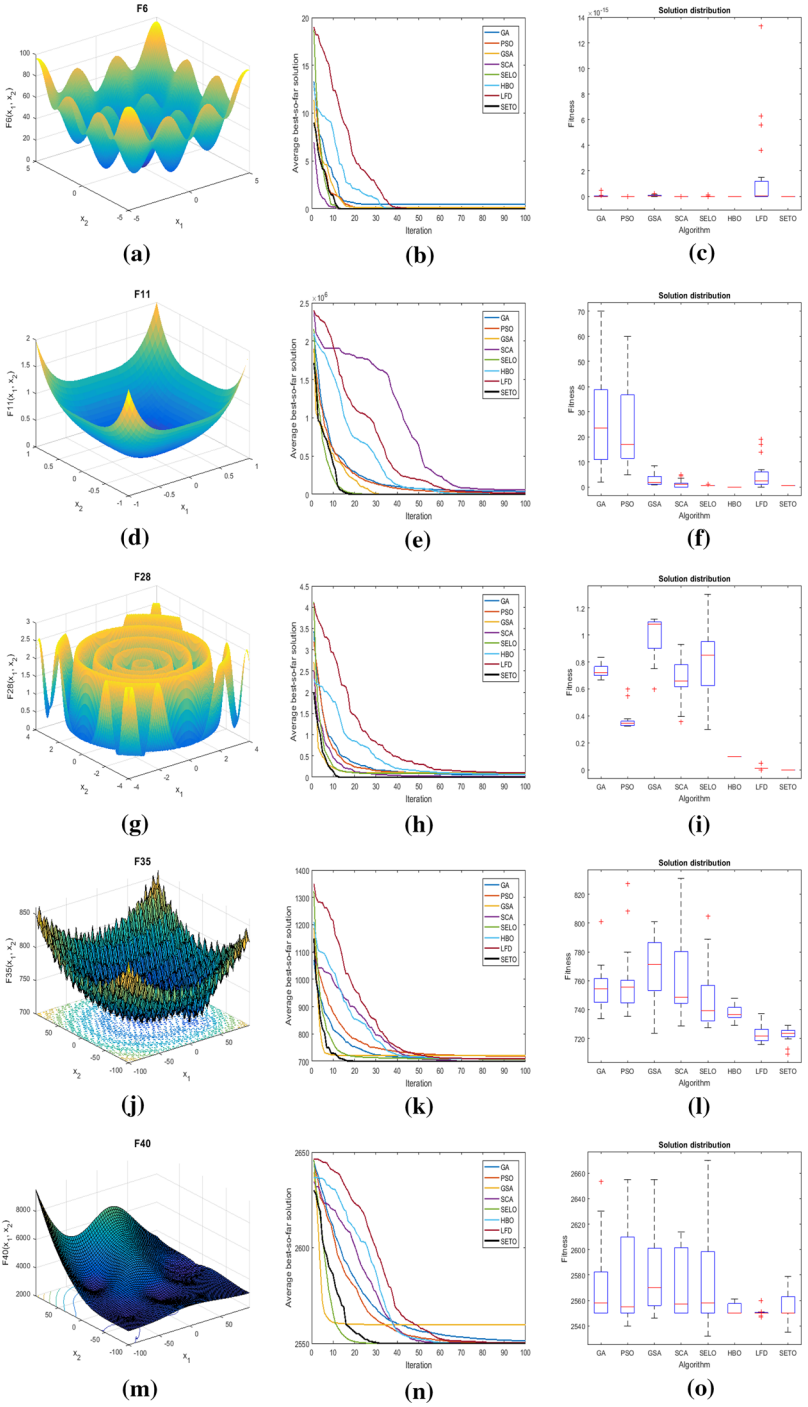


Fig. 8 Convergence graphs and solution distributions of comparison algorithms on F6, F11, F28, F35, and F40 test functions

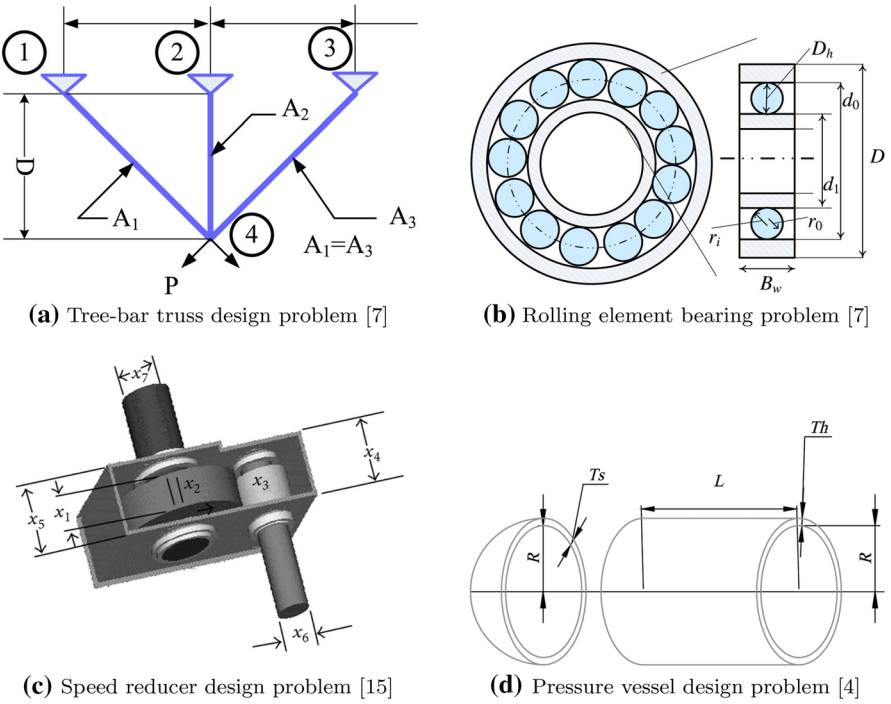
The objective is to maximize the dynamic load-carrying capacity of a rolling element bearing. This problem is mathematically formulated as follows [15]:

$$
\begin{aligned}
&\text{maximize}\\
&\quad C_d = f_c Z^{2/3} D_b^{1.8} \qquad \text{if}(D \le 25.4mm)\\
&\quad C_d = 3.647 f_c Z^{2/3} D_b^{1.4} \quad \text{if}(D > 25.4mm)\\
&\text{subject to}\\
&\quad g_1(\vec{z}) = \frac{\phi_0}{2\sin^{-1}(D_b/D_m)} - Z + 1 \le 0, \qquad g_2(\vec{z}) = 2D_b - K_{D\min}(D-d) > 0,\\
&\quad g_3(\vec{z}) = K_{D\max}(D-d) - 2D_b \ge 0, \qquad g_4(\vec{z}) = \xi B_w - D_b \le 0,\\
&\quad g_5(\vec{z}) = D_{\mathrm{m}} - 0.5(D+d) \ge 0, \qquad g_6(\vec{z}) = (0.5+e)(D+d) - D_m \ge 0,\\
&\quad g_7(\vec{z}) = 0.5(D - D_m - D_b) - \varepsilon D_b \ge 0 \quad g_8(\vec{z}) = f_i \ge 0.515,\\
&\quad g_9(\vec{z}) = f_o \ge 0.515\\
&\text{where}\\
&\quad f_c = 37.91\left[1 + \left\{1.04\left(\frac{1-\gamma}{1+\gamma}\right)^{1.72}\left(\frac{f_i(2f_0-1)}{f_o(2f_i-1)}\right)^{0.41}\right\}^{10/3}\right]^{-0.3} \times \left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1-\gamma)^{1/3}}\right]\left[\frac{2f_i}{2f_i-1}\right]^{0.41}\\
&\quad x = \left[\{(D-2)/2 - 3(T/4)\}^2 + \{D/2 - T/4 - D_b\}^2 - \{d/2 + T/4\}^2\right]\\
&\quad y = 2\{(D-d)/2 - 3(T/4)\}\{(D/d) - T/4 - D_b\}\\
&\quad \phi_o = 2\prod -\cos^{-1}\left(\frac{x}{y}\right)\\
&\quad \gamma = \frac{D_b}{D_m}, f_i = \frac{r_i}{D_b}, f_o = \frac{r_i}{D_b}, T = D - d - 2D_b\\
&\quad D = 160, d = 90\\
&\quad B_w = 30, r_i = r_o = 11.0330.5(\mathrm{D} + \mathrm{d}) \le \mathrm{D}_m \le 0.6(D+d),\\
&\quad 0.15(\mathrm{D} - \mathrm{d}) \le \mathrm{D}_b \le 0.45(D-d), 4 \le Z \le 50, 0.515 \le f_i \text{and } \mathrm{f}_o \le 0.6,\\
&\quad 0.4 \le K_{D\min} \le 0.5,\\
&\quad 0.6 \le K_{D\max} \le 0.7, 0.3 \le e \le 0.4, 0.02 \le \mathrm{e} \le 0.1,\\
&\quad 0.6 \le \xi \le 0.85
\end{aligned}
\tag{32}
$$

Table 16 summarizes the solutions obtained by the proposed SETO and comparison algorithms for the rolling element bearing design problem. Inspecting the results in Table 16, we conclude that the SETO obtains superior results compared with other optimizers and exposes the best design.

5.3 Speed reducer design problem

Figure 9c shows a schematic view of the speed reducer design problem. The objective is to design a simple gearbox with the minimum weight that is embedded between the propeller and the engine in light aircraft [15]. The problem consists of constraints on surface stress, bending stress of the gear teeth, stresses in the shafts, and transverse deflections of the shafts. The mathematical formulation of the problem is as follows [15]:



(a) Tree-bar truss design problem [7]

(b) Rolling element bearing problem [7]

(c) Speed reducer design problem [15]

(d) Pressure vessel design problem [4]

Fig. 9 Engineering design problems used in the tests

Table 15 Comparison of result obtained by algorithms for tree-bar truss design

Algorithm	Problem parameters		Optimum weight
	x_1	x_2	
GA	0.792	0.399	263.9037
PSO	0.7901	0.4042	263.8974
GSA	0.7898	0.4052	263.8967
SCA	0.7875	0.4117	263.8989
SELO	0.7878	0.4108	263.8964
HBO	0.7887	0.4082	263.8959
LFD	0.7879	0.4106	263.8963
SETO	0.7886	0.4083	**263.8958**

Best results are illustrated in boldface

Table 16 Comparison of results for rolling element bearing design problem

Algorithms	GA	PSO	GSA	SCA	SELO	HBO	LFD	SETO
Dm	127.4083	127.5557	125	125.812	126.3521	125.7189	126.3999	125.7227
Db	20.3698	20.2762	20.6628	20.8214	21.0299	21.4233	21	21.4233
Z	11	11	11	1100	1100	1100	11	11
fi	0.515	0.515	0.515	0.515	0.515	0.515	0.515	0.515
fo	0.515	0.515	0.5333	0.5182	0.515	0.515	0.5251	0.515
Kdmin	0.4	0.5	0.5	50	0.4	0.4	0.5	0.4
Kdmax	0.6	0.6	60	0.63	0.6011	0.7	0.6	0.7
ε	0.3	0.7	0.3469	0.3003	0.3	0.3	0.3	0.3
e	0.1	0.3	0.02	0.0669	0.1	0.0998	0.1	0.1
ξ	0.6	0.0956	0.6884	0.6001	0.6004	60	0.6	0.6
C_d	80863.22	80433.47	81373.29	81256.51	83805.29	85537.48	83670.78	**85539.19**

Best results are illustrated in boldface

$$
\begin{aligned}
\text{minimize} \quad & f(\vec{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - \\
& 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2) \\
\text{subject to} \quad & g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3} - 1 \le 0, \quad g_2(\vec{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0, \\
& g_3(\vec{x}) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0, \quad g_4(\vec{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0, \\
& g_5(\vec{x}) = \frac{1}{110x_6^3}\sqrt{\left(\frac{745x_4}{110x_2x_3}\right)^2 + 16.9 \times 10^6} - 1 \le 0 \\
& g_6(\vec{x}) = \frac{1}{85x_7^3}\sqrt{\left(\frac{745x_5}{110x_2x_3}\right)^2 + 157.5 \times 10^6} - 1 \le 0 \\
& g_7(\vec{x}) = \frac{x_2x_3}{40} - 1 \le 0, \quad g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \le 0 \\
& g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \le 0, \quad g_{10}(\vec{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0 \\
& g_{11}(\vec{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0 \\
\text{where} \quad & 2.6 \le x_1 \le 3.6,\ 0.7 \le x_2 \le 0.8,\ 17 \le x_3 \le 28, \\
& 7.3 \le x_4 \le 8.3,\ 7.3 \le x_5 \le 8.3,\ 2.9 \le x_6 \le 3.9, \\
& 5.0 \le x_7 \le 5.5
\end{aligned}
\tag{33}
$$

As shown in Table 17, HBO obtains the best results. With a slight difference from HBO, the proposed SETO takes the second rank. Except for HBO, the proposed SETO attains the best results compared to other optimizers, which confirms that it can be a suitable choice for designing the speed reducer.

5.4 Pressure vessel design problem

Pressure vessels are widely used in industry structures such as gas tanks and champagne bottles. The goal is to design a cylindrical vessel with the minimum fabrication cost. The problem consists of four design parameters including the thickness of

Table 17 Comparison of results for speed reducer design problem

Algorithm	Problem parameters							Optimal cost
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
GA	3.599614	0.7	17	7.300000	7.715320	3.350238	5.286655	3033.6028
PSO	3.600000	0.7	17	8.299999	7.715358	3.352207	5.286655	3043.0812
GSA	3.500252	0.7	17	7.750236	7.715629	3.351082	5.286725	2998.8137
SCA	3.564661	0.7	17	7.300000	7.858052	3.356281	5.288056	3025.4368
SELO	3.500195	0.70002	17	7.307411	7.918465	3.350301	5.286724	2999.2274
HBO	3.500000	0.7	17	7.300000	7.715320	3.350210	5.286650	**2994.4711**
LFD	3.500006	0.7	17	7.304732	7.715321	3.350224	5.286655	2994.5173
SETO	3.500013	0.700001	17	7.300330	7.715996	3.350216	5.286655	2994.4991

Best results are illustrated in boldface

Table 18 Comparison of results for pressure vessel design problem

Algorithm	Problem parameters				Cost
	T_s	T_h	R	L	
GA	0.87591	0.43296	45.38407	139.77943	6074.4540
PSO	0.86919	0.43221	45.03562	143.43608	6071.4145
GSA	1.12500	0.62500	55.98870	84.45420	8538.8359
SCA	0.86390	0.42703	44.76160	146.21210	6048.6169
SELO	0.87871	0.43435	45.52902	138.30634	6080.3521
HBO	0.84186	0.41600	43.60000	159.00000	6003.3650
LFD	0.84308	0.41674	43.68298	157.94655	6005.8844
SETO	0.81268	0.40171	42.10791	176.53302	**5947.3050**

Best results are illustrated in boldface

the head (T_s), the thickness of the body (T_h), the inner radius (R), and the length of the cylindrical section (L). Figure 9d shows the overall structure of the pressure vessel design problem. The problem is mathematically defined as follows [4]:

$$\text{let} \quad \vec{x} = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L] \quad \text{where} \quad \begin{array}{l} 0 \le x_1, x_2 \le 99, \\ 10 \le x_3, x_4 \le 200, \end{array}$$

$$\begin{array}{ll} \text{minimize} & f(\vec{x}) = 0.6224 x_1 x_2 x_4 + 1.778 x_2 x_2^3 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3 \\ \text{subject to} & c_1(\vec{x}) = -x_1 + 0.0193 x_3 \le 0, \\ & c_2(\vec{x}) = -x_3 + 0.00954 x_3 \le 0, \\ & \mathrm{c}_3(\vec{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \le 0, \\ & c_4(\vec{x}) = x_4 - 240 \le 0 \end{array} \tag{34}$$

Table 18 reports the results attained by SETO and comparison optimizers. The parameters and costs of SETO are very competitive to those obtained by other algorithms. This confirms that the SETO is able to deal with the constrained search space of pressure vessel design problem.

In the current implementation of the SETO algorithm, it faces three challenges:

- Tuning some of the control parameters with optimal values for different applications. Most of the control parameters of SETO are already known and configured using the data drawn from the stock exchange and scientific resources about technical analysis. This issue turns the SETO into an optimizer quite easy to implement and execute. However, in some applications, different values for the parameters can increase the performance of the algorithm. Parameter setting is not specific to the SETO algorithms and exists in all algorithms.
- Increasing the execution time of the algorithm due to the calculation of the Euclidean distance between shares in rising and falling phases. As the dimension of the problem increases, the execution time of the algorithm also increases,
- The algorithm still traps in local optima on some benchmark functions and cannot converge to the global optimum, as we can see in speed reducer design problem and some numerical functions such as F39, F40. This suggests that increasing the exploitation and exploration power of the genetic algorithm is needed.

To summarize, the advantages of the SETO algorithm are as follows:

- It can be used for both continuous and discrete problems with some easy modifications.
- It is simple and efficient. It achieves superior results on different groups of numerical functions and engineering optimization problems.
- It can be applied to all problems that other algorithms can be applied for.
- It converges to the global optimum of the optimization problems faster than its counterparts.
- It outperformed other algorithms on most benchmark functions. Out of 40 numerical optimizatin functions, SETO has achieved the global optimum on 36 functions, and out of 4 engineering complex problems, it obtained the best results on 4 cases.

6 Conclusion

This paper presents a novel stock exchange trading optimization (SETO) algorithm to solve numerical and engineering optimization problems. The algorithm is based on technical-based trading strategies in the stock market. Rising, falling, and exchange are the three main phases of the algorithm that hopefully causes the solutions to converge to the global optimum of the cost function. SETO is easy to implement and conceptually simple. To test the performance of SETO, it is compared with several state-of-the-art optimizers in solving a wide variety of numerical global optimization and real-world problems. The results confirm that SETO attained outstanding performance compared with its counterparts in most test cases. This issue is demonstrated with the experiments and the statistics of results. There remain several directions for future research. One of the interesting works is to apply the SETO to a variety of real-world applications to precisely determine the advantages and

weaknesses of the algorithm. Another work is to develop a multi-objective version of the SETO to employ it for solving multi-objective problems. Finally, modeling various indicators and phenomena in the stock exchange such as options and share portfolio, and improving the potential of algorithm operators can be helpful to guide the search process and further improve the performance of the algorithm.

References

1. Brammya G, Praveena S, Ninu Preetha NS, Ramya R, Rajakumar BR, Binu D (2019) Deer hunting optimization algorithm: a new nature-inspired meta-heuristic paradigm. Comput J
2. Molina D, Poyatos J, Del Ser J, García S, Hussain A, Herrera F (2020) Comprehensive taxonomies of nature- and bio-inspired optimization: inspiration versus algorithmic behavior, critical analysis and recommendations. Cognit Comput 12(5):897–939
3. Abbasi M, Yaghoobikia M, Rafiee M, Jolfaei A, Khosravi MR (2020) Energy-efficient workload allocation in fog-cloud based services of intelligent transportation systems using a learning classifier system. IET Intell Transp Syst 14(11):1484–1490
4. Houssein EH, Saad MR, Hashim FA, Shaban H, Hassaballah H (2020) Lévy flight distribution: a new metaheuristic algorithm for solving engineering optimization problems. Eng Appl Artif Intell 94:103731
5. Hussain K, Salleh M, Cheng S, Shi Y (2018) Metaheuristic research: a comprehensive survey. Artif Intell Rev 52:2191–2233
6. Yang XS, Deb S, Zhao YX, Fong S, He X (2018) Swarm intelligence: past, present and future. Soft Comput 22(18):5923–5933
7. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Futur Gener Comput Syst 97:849–872
8. Abdel-Basset M, Abdel-Fatah L, Sangaiah AK (2018) Meta-heuristic algorithms: a comprehensive review. In: Computational intelligence for multimedia big data on the cloud with engineering applications. Elsevier Inc
9. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, pp 1942–1948
10. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1:28–39
11. Emami H, Derakhshan F (2015) Election algorithm: a new socio-politically inspired strategy. AI Commun 28(3):591–603
12. Emami H (2019) Chaotic election algorithm. Comput Inform 38:1444–1478
13. Fadakar F, Ebrahimi M (2016) A new metaheuristic football game inspired algorithm. In: 1st Conference on Swarm Intelligence and Evolutionary Computation CSIEC 2016—Proceedings, pp 6–11
14. Askari Q, Younas I, Saeed M (2020) Political optimizer: a novel socio-inspired meta-heuristic for global optimization. Knowl Based Syst 195:105709
15. Askari Q, Saeed M, Younas I (2020) Heap-based optimizer inspired by corporate rank hierarchy for global optimization. Expert Syst Appl 161:113702
16. Salih SQ, Alsewari ARA (2020) A new algorithm for normal and large-scale optimization problems: Nomadic People Optimizer. Neural Comput Appl 32(14):10359–10386
17. Sörensen K, Sevaux M, Glover F (2017) A history of metaheuristics. In: ORBEL29-29th Belgian Conference on Operations Research, pp 791–808
18. Emami H (2020) Seasons optimization algorithm. Eng Comput 123456789:1–21
19. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
20. Holland JH (1992) Genetic algorithms—computer programs that 'evolve' in ways that resemble natural selection can solve complex problems even their creators do not fully understand. Sci Am 66–72
21. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102
22. Huang F, Wang L, He Q (2007) An effective co-evolutionary differential evolution for constrained optimization. Appl Math Comput 186:340–356
23. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12(6):702–713

24. Ghaemia M, Feizi-Derakhshi MR (2014) Forest optimization algorithm. Expert Syst Appl 41(15):6676–6687
25. Hayyolalam V, Pourhaji Kazem AA (2020) Black widow optimization algorithm: a novel meta-heuristic approach for solving engineering optimization problems. Eng Appl Artif Intell 87:103249
26. Shayanfar H, Gharehchopogh FS (2018) Farmland fertility: a new metaheuristic algorithm for solving continuous optimization problems. Appl Soft Comput J 71:728–746
27. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39(3):459–471
28. Yang X (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspir Comput 2(2):78–84
29. Gandomia AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845
30. Wang GG, Deb S, Coelho LDS (2016) Elephant herding optimization. In: Proceedings of 2015 3rd International Symposium on Computational and Business Intelligence ISCBI, pp 1–5
31. Bansal JC, Sharma H, Jadon SS, Clerc M (2014) Spider monkey optimization algorithm for numerical optimization. Memet Comput 16(1):31–47
32. Mirjalili S, Mohammad S, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
33. Soleimanian F, Gholizadeh H (2019) A comprehensive survey: whale optimization algorithm and its applications. Swarm Evol Comput 48:1–24
34. Arora S, Singh S (2018) Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput 23(3):715–734
35. Jain M, Singh V, Rani A (2019) A novel nature-inspired algorithm for optimization: squirrel search algorithm. Swarm Evol Comput 44:148–175
36. Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. Adv Eng Softw 105:30–47
37. Dhiman G, Kumar V (2019) Seagull optimization algorithm: theory and its applications for large scale industrial engineering problems. Knowl Based Syst 165:169–196
38. Mohamad-saleh WTJ, Tan W (2019) Normative fish swarm algorithm (NFSA) for optimization. Soft Comput 24(3):2083–2099
39. Fathollahi-Fard ANM, Hajiaghaei-Keshteli M, Tavakkoli-Moghaddam R (2020) Red deer algorithm (RDA): a new nature-inspired meta-heuristic. Soft Comput 24(19):14637–14665
40. Kirkpatrick S, Vecchi GCD, Science MP (1983) Optimization by simulated annealing. Science 220:671–680
41. Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248
42. Erol OK, Eksin I (2006) A new optimization method: big bang–big crunch. Adv Eng Softw 37:106–111
43. Alatas B (2011) ACROA: artificial chemical reaction optimization algorithm for global optimization. Expert Syst Appl 38(10):13170–13180
44. Shah-hosseini H (2011) Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. Int J Comput Sci Eng 6(2):132–140
45. Feng X, Liu Y, Yu H, Luo F (2017) Physarum-energy optimization algorithm. Soft Comput 23(3):871–888
46. Kaveh A, Dadras A (2017) A novel meta-heuristic optimization algorithm: thermal exchange optimization. Adv Eng Softw 110:69–84
47. Faramarzi A, Heidarinejad M, Stephens B, Mirjalili S (2019) Equilibrium optimizer: a novel optimization algorithm. Knowl Based Syst 191:105190
48. Kushwaha N, Pant M, Kant S, Jain VK (2018) Magnetic optimization algorithm for data clustering. Pattern Recognit Lett 115:59–65
49. Alexandros GD (2017) Nature inspired optimization algorithms related to physical phenomena and laws of science: a survey. Int J Artif Intell Tools 26(6):1–25
50. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
51. Atashpaz-Gargari E, Lucas C (2007) Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: 2007 IEEE Congress on Evolutionary Computation, CEC2007, Singapore, pp 4661–4667
52. Rao RV, Savsani VJ, Vakharia DP (2012) Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems. Inf Sci 183(1):1–15

53. Husseinzadeh Kashan A (2014) League championship algorithm (LCA): an algorithm for global optimization inspired by sport championships. Appl Soft Comput J 16:171–200
54. Das P, Das DK, Dey S (2018) A new class topper optimization algorithm with an application to data clustering. IEEE Trans Emerg Top Comput 6750:1–11
55. Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl Based Syst 96:120–133
56. Kumar M, Kulkarni AJ, Satapathy SC (2018) Socio evolution & learning optimization algorithm: a socio-inspired optimization methodology. Futur Gener Comput Syst 81:252–272
57. Mahmoodabadi MJ, Rasekh M, Zohari T (2018) TGA: team game algorithm. Future Comput Inform J 3(2):191–199
58. Singh PR, Elaziz MA, Xiong S (2019) Ludo game-based metaheuristics for global and engineering optimization. Appl Soft Comput J 84:105723
59. Martinez-Alvarez F et al (2020) Coronavirus optimization algorithm: a bio-inspired meta-heuristic based on the COVID-19 propagation model. Big Data 8(4):308–322
60. Abbasi M, Yaghoobikia M, Rafiee M, Jolfaei A, Khosravi MR (2020) Energy-efficient workload allocation in fog-cloud based services of intelligent transportation systems using a learning classifier system. IET Intell Transp Syst 14(11):1484–1490
61. Zhou Z, Kearnes S, Li L, Zare RN, Riley P (2019) Optimization of molecules via deep reinforcement learning. Sci Rep 9(1):1–10
62. Talbi EG (2019) Machine learning for metaheuristics—state of the art and perspectives. In: 11th International Conference on Knowledge and Smart Technology (KST), pp XXIII–XXIII
63. Owoyele O, Pal P (2021) A novel machine learning-based optimization algorithm (ActivO) for accelerating simulation-driven engine design. Appl Energy 285:116455
64. Nabipour M, Nayyeri P, Jabani H, Mosavi A, Salwana E, Shahab S (2020) Deep learning for stock market prediction. Entropy 22(8):840
65. Das SR, Mishra D, Rout M (2019) Stock market prediction using Firefly algorithm with evolutionary framework optimized feature reduction for OSELM method. Expert Syst Appl 4:100016
66. Kelotra A, Pandey P (2020) Stock market prediction using optimized deep-ConvLSTM model. Big Data 8(1):5–24
67. Thakkar A, Chaudhari K (2020) A comprehensive survey on portfolio optimization, stock price and trend prediction using particle swarm optimization. Springer, pp 1–32
68. Kumar K, Haider MT (2021) Enhanced prediction of intra-day stock market using metaheuristic optimization on RNN-LSTM network. New Gener Comput 39(1):231–272
69. Abedi M, Gharehchopogh FS (2020) An improved opposition based learning firefly algorithm with dragonfly algorithm for solving continuous optimization problems. Intell Data Anal 24(2):309–338
70. Rahnema N, Gharehchopogh FS (2020) An improved artificial bee colony algorithm based on whale optimization algorithm for data clustering. Multimed Tools Appl 79(44):32169–32194
71. Mohammadzadeh H, Soleimanian F (2021) Feature selection with binary symbiotic organisms search algorithm for email spam detection. Int J Inf Technol Decis Mak 20(1):469–515
72. Soleimanian F, Shayanfar H, Gholizadeh H (2020) A comprehensive survey on symbiotic organisms search algorithms. Artif Intell Rev 53:2265–2312
73. Mohmmadzadeh H, Soleimanian F (2021) An efficient binary chaotic symbiotic organisms search algorithm approaches for feature selection problems. J Supercomput
74. Hosseinalipour A, Soleimanian F, Masdari M, Khademi A (2021) A novel binary farmland fertility algorithm for feature selection in analysis of the text psychology. Appl Intell 1–36
75. Darwish A (2018) Bio-inspired computing: algorithms review, deep analysis, and the scope of applications. Future Comput Inform J 3(2):231–246
76. Murphy JJ (1999) Technical analysis of the financial markets: a comprehensive guide to trading methods and applications. Penguin
77. Wilder JW (1978) New concepts in technical trading systems. Trend Research
78. Anderson B, Li S (2015) An investigation of the relative strength index. Banks Bank Syst 10(1):92–96
79. Wafi AS, Hassan H, Mabrouk A (2015) Fundamental analysis models in financial markets—review study. Procedia Econ Finance 30(15):939–947
80. Civicioglu P (2013) Backtracking search optimization algorithm for numerical optimization problems. Appl Math Comput 219(15):8121–8144
81. Suganthan P, Ali M, Wu G, Mallipeddi R (2018) Special session & competitions on real-parameter single objective optimization. In: CEC2018, Rio de Janeiro, Brazil

82. Haupt RL, SE H (2004) Practical genetic algorithms. Wiley
83. Thangaraj R, Pant M, Abraham A, Bouvry P (2011) Particle swarm optimization: hybridization perspectives and experimental illustrations. Appl Math Comput 217(12):5208–5226
84. Mirjalili S, Gandomi AH, Zahra S, Saremi S (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:1–29
85. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18