scientific reports

OPEN

# Squid Game Optimizer (SGO): a novel metaheuristic algorithm

Mahdi Azizi[1,3,4], Milad Baghalzadeh Shishehgarkhaneh[2,4], Mahla Basiri[1,3] & Robert C. Moehler[2]

**In this paper, Squid Game Optimizer (SGO) is proposed as a novel metaheuristic algorithm inspired by the primary rules of a traditional Korean game. Squid game is a multiplayer game with two primary objectives: attackers aim to complete their goal while teams try to eliminate each other, and it is usually played on large, open fields with no set guidelines for size and dimensions. The playfield for this game is often shaped like a squid and, according to historical context, appears to be around half the size of a standard basketball court. The mathematical model of this algorithm is developed based on a population of solution candidates with a random initialization process in the first stage. The solution candidates are divided into two groups of offensive and defensive players while the offensive player goes among the defensive players to start a fight which is modeled through a random movement toward the defensive players. By considering the winning states of the players of both sides which is calculated based on the objective function, the position updating process is conducted and the new position vectors are produced. To evaluate the effectiveness of the proposed SGO algorithm, 25 unconstrained mathematical test functions with 100 dimensions are used, alongside six other commonly used metaheuristics for comparison. 100 independent optimization runs are conducted for both SGO and the other algorithms with a pre-determined stopping condition to ensure statistical significance of the results. Statistical metrics such as mean, standard deviation, and mean of required objective function evaluations are calculated. To provide a more comprehensive analysis, four prominent statistical tests including the Kolmogorov–Smirnov, Mann–Whitney, and Kruskal–Wallis tests are used. Meanwhile, the ability of the suggested SGOA is assessed through the cutting-edge real-world problems on the newest CEC like CEC 2020, while the SGO demonstrate outstanding performance in dealing with these complex optimization problems. The overall assessment of the SGO indicates that the proposed algorithm can provide competitive and remarkable outcomes in both benchmark and real-world problems.**

Real-world optimization problems are considered quite challenging tasks and intricate problems in almost every field, classifying them in miscellaneous categories, including constrained or unconstrained, single or multi-objective, continuous or discrete, and static or dynamic. In some of engineering and industrial applications, which can be formulated as optimization problems, researchers are attempting to optimize specific variables, whether to reduce costs and energy consumption or increase profit, production, efficiency, and performance. Consequently, it is imperative to have an efficient optimizer to guarantee that the best solutions are found. The central component of an optimizer is a search or optimization algorithm that is properly designed and executed to conduct the necessary exploration. For a long time, traditional search approaches have been used to solve optimization issues. Even though these strategies provide promising results in different real-world problems, they may face failure in more complicated optimization problems. Hence, to solve these complex problems, metaheuristic algorithms have been developed.

Generally, metaheuristic algorithms can be categorized into four primary groups. The first one is the evolutionary based metaheuristic algorithms, which simultaneously perform the search procedure using several initial points. Holland[1] introduced the Genetic Algorithm (GA), which is a popular population-based metaheuristic algorithm inspired by Darwinian evolution theory. Differential Evolution (DE) algorithm is another well-known stochastic population-based algorithm for global optimization[2,3]. Furthermore, with the popularity of the Imperialist Competitive Algorithm[4], there have been various more frequently used population-based algorithms, including Charged System Search[5–7], Intelligent Water Drops[8], Stochastic Paint Optimizer[9,10], Political

[1]Department of Civil Engineering, University of Tabriz, Tabriz, Iran. [2]Department of Civil Engineering, Monash University, Clayton, VIC 3800, Australia. [3]Department of Civil Engineering, Near East University, Nicosia, Cyprus. [4]These authors contributed equally: Mahdi Azizi and Milad Baghalzadeh Shishehgarkhaneh. email: mehdi.azizi875@gmail.com

Optimizer[11], Dynamic Virtual Bats Algorithm[12], Ali Baba and the forty thieves[13], Tiki-taka algorithm[14], and Coronavirus Optimization Algorithm[15]. The second category is swarm-based metaheuristic algorithms, which are based on the social behaviour of diverse species in natural groups like ants, bees, birds, fishes, and termites. Particle Swarm Optimization (PSO)[16] and Ant Colony Optimization (ACO)[17,18] are prominent swarm-based algorithms While these algorithms imitate bird and ant colonies' aggregation and foraging behaviors, respectively[19]. Artificial Bee Colony (ABC)[20,21], Al-Biruni Earth Radius[22], Border Collie Optimization[23], Stochastic Diffusion Search[24], Glowworm Swarm Optimization[25], Mountain Gazelle Optimizer[26], Cuckoo Search[27], Flower Pollination Algorithm[28], and Black Widow Optimization Algorithm[29] are other examples of this sort. Furthermore, physics-based metaheuristic algorithms are the third classification inspired by physics laws, such as heat transformation, gravitational force, particle motions, and wave propagation. Undoubtedly, the SA algorithm, inspired by annealing in solids' analogy to the statistical mechanics, has gained much popularity in this category[30]. Big-Bang Big-Crunch algorithm, which is inspired by the theories of the evolution of the universe, is another example of a physics-based optimization algorithm[31]. Additionally, it is noteworthy to state that there are still various metaheuristic methods that take inspiration from the laws of physics that have been documented in literature. Some of the most common algorithms include Atomic Orbital Search[32–34], Material Generation Algorithm[35–37], Cyber-physical Systems[38], Chaos Game Optimization[39,40], Archimedes Optimization Algorithm[41], Lichtenberg Algorithm[42], Energy Valley Optimizer[43], Crystal Structure Algorithm[44–46], Thermal Exchange Optimization Algorithm[47], Equilibrium Optimizer[48], Weighted Vertices Optimizer[49], and Lévy Flight Distribution[50]. Finally, the last group is human behavior-based metaheuristic algorithms. The development of certain human-based algorithms has been inspired by simulating various human behaviors, including Teaching–Learning-Based Optimization (TLBO)[51], Cultural Algorithm (CA)[52], and Harmony Search (HS)[53]. TLBO has been developed based on the teaching and learning activities between teachers and students. CA has been developed with an inspiration from human sociology. Similarly, HS has been developed by taking inspiration from the adjustments in tone made by musicians in a band. These algorithms aim to capture and replicate the effective aspects of human behavior in problem-solving and optimization tasks. By simulating these human behaviors, they provide new and effective approaches for solving complex problems in various fields.

The main contribution of this paper is to propose a novel search method for optimization purposes in which an intelligent procedure is conducted for finding the best optimal values of different optimization problems. The applicability of the proposed method in dealing with difficult optimum design problems is another key contribution of this paper so the capability of the novel algorithm will be assessed through different procedures. In this research, a novel population-based metaheuristic optimization algorithm, called Squid Game Optimizer (SGO), is proposed. The proposed algorithm is inspired by the strategy of the squid game. It finds the optimal solution from a population of candidate solutions; each solution is referred to as an offensive individual attempting to overcome the defensive population. As a result, an offensive player with a higher value for the objective function could beat others. Finally, the best offensive player will be the winner at the end of the iteration. This concept is being used to develop a metaheuristic algorithm for the first time. This study introduces a new and inspiring research approach by assessing the complexity level of test functions, which has not been previously attempted. Evaluating the performance of numerous algorithms under different scenarios is a challenging task, as it is not possible to confirm the superiority of any algorithm with complete certainty. To address this, the study employs 25 unconstrained mathematical test functions, each with 100 dimensions, to evaluate the proposed algorithm's effectiveness. To ensure the statistical validity of the results, this study conducts 100 independent optimization runs and computes measures such as the mean, standard deviation, and required objective function evaluations. A predefined stopping criterion is utilized, taking into account both the maximum number of objective function evaluations and the tolerance for the global best values of the problems being considered. Finally, a number of well-known statistical analyses such as the Kolmogorov–Smirnov, Wilcoxon, Mann–Whitney, and Kruskal–Wallis tests are employed for comparative purposes. In light of the inadequacy of evaluation test functions in newly developed metaheuristic algorithms, this study seeks to address this shortcoming by assessing the complexity level of two recent Competitions on Evolutionary Computation (CEC). Specifically, the CEC 2020 competition on bound constraint optimization[54] is used to assess the complexity of the SGO, while the CEC 2020 competition on real-world optimization[55] is used to compare the performance of the SGO against that of cutting-edge algorithms. By leveraging these competitions, the study aims to improve the evaluation of metaheuristic algorithms and enhance their effectiveness in solving real-world optimization problems.

It should be noted that the metaheuristic algorithms are some sorts of stochastic methods in which randomization procedures are the key aspect of these approaches while multiple random numbers are generated through the optimization procedure in order to reach the global optimal point. For this purpose, the main limitation of these algorithms including the SGO is the reliability of them in initial random solutions created by these algorithms. The initial solutions near to global optimal can make the algorithms converge to better solutions while the solutions far from the global optima can underestimate the capability of the algorithm. Hence, the statistical procedures by means of multiple optimization runs can overcome this drawback so the SGO is justified through adequate statistical analysis.

The main contribution of this paper is the proposition of a novel metaheuristic algorithm in which the strategies and rules of the Squid Game as one of the Korean ancient games are used as the inspirational concept. The applicability of this algorithm in dealing with different kinds of optimization problems is the main challenge of this population-based algorithm in which the solution candidates are divided into two groups of offensive and defensive players while the offensive player goes among the defensive players to start a fight which is modeled through a random movement toward the defensive players. The results of this algorithm in dealing with mathematical problems with 100 dimensions demonstrates the capability of this algorithm in dealing with large search domains while solving the engineering design problems by this algorithm can guarantee the possibility of using this algorithm in real-world applications which is a main concern of the researchers based on the recent challenges in developing intelligent techniques for engineering optimization purposes.

The rest of the paper is structured in the following manner. Section "Related works" and "Squid Game Optimizer Algorithm" describe the related works and SGO developed in this study, respectively. Mathematical test problems and statistical outcomes are provided and discussed in Sections "Mathematical test functions" and "Numerical results of the mathematical functions". The complete statistical analysis is given in Section "Statistical Analysis", while the complexity analysis is conducted in Section "Computational complexity and cost analysis". The real-world problems of CEC are demonstrated in Section "Real-world constrained optimization problems of CEC 2020" while Sect. 9 summaries this study's primary outcomes and propose future research directions.

## Related works

Nowadays, metaheuristic algorithms have been utilized for the optimization of intricate real-world scientific and engineering problems; since they do not need complicated mathematical expressions, they can deal with constraints much more simply than traditional methods, and they all try to favour the search for the global optimum solution rather than local ones. In health and medicine, Canayaz[56] used particle swarm and gray wolf optimization algorithms to discover Covid-19 at an early stage.; Basu et al.[57] employed Harris Hawks Optimization algorithm for identifying COVID-19 from radiological images; Bandyopadhyay et al.[58] employed a hybrid approach of this algorithm and Simulated Annealing algorithm to detect COVID-19 from CT scan images; and Hosseini et al.[59] used COVID-19 optimizer Algorithm for reducing the number of COVID-19-infected regions and thus slowing the spread of the disease. Moreover, metaheuristic algorithms have been considered for solving different problems in medicine, including healthcare systems[60], medical image segmentation[61–63], optimization of cognitive big data healthcare[64], diagnosis of brain tumors and breast cancer[65,66], diagnosis of Parkinson's disease[67], and Brain Cine-MRI[68]. Nonetheless, in chemistry, Chen et al.[69] used ranking-based differential evolution algorithms for dynamic optimization problems; Cheema et al.[70] employed a GA for optimization of glucose to gluconic acid fermentation; Mohd Zain et al.[71] used Backtracking Search Algorithm for optimization of fed-batch fermentation processes; Geem and Kim[72] used HS algorithm for wastewater treatment optimization for fish migration; and Piotrowski et al.[73] applied the ABC algorithm and the Direct Search Algorithm to optimize the biological procedures in wastewater sequencing batch reactors.

Similarly, researchers have used metaheuristic algorithms to solve and optimize a plethora of intricate engineering problems in recent years. In civil engineering, Çerçevik et al.[74] used three different algorithms, including Cuckoo Search Algorithm, Whale Optimization Algorithm (WOA), and Grey Wolf Optimizer to optimize the parameters of seismic isolated structures. Kaveh and Mahdavi[75] optimized arch dams' shape under earthquake loading with the Charged System Search and PSO algorithms. Azizi et al.[40] conducted truss structures' shape and size optimization employing the Chaos Game Optimization approach. Azizi et al.[76] optimized the design of engineering problems using the Atomic Orbital Search algorithm. Kaveh and Khosravian[77] used Vibrating Particles System algorithm to optimize truss structures' layout and size. Gandomi et al.[78] utilized the Cuckoo Search algorithm for five truss design optimization problems. Furthermore, Zhang et al.[79] utilized the PSO algorithm to optimise concrete mixture proportions. Sun et al.[80] applied the ABC algorithm to anticipate and optimise the concrete samples' compressive strength. Chou et al.[81] prognosticated the shear strength in reinforced concrete deep beams. In mechanical engineering, Muthu et al.[82] conducted an optimal tolerance design for assembly with the goal of minimizing quality loss and manufacturing cost, using both GA and PSO algorithms. Hassan et al.[83] applied the ACO algorithm to optimise pressure vessels' optimum design. Acharya et al.[84] optimized proportional–integral–derivative control scheme parameters to regulate the DC motor's speed. Pham et al.[85] used different metaheuristic optimization algorithms in the functionally graded sandwich porous beams' optimum design. Furthermore, other application of metaheuristic algorithms could be found in Refs.[86–114] including the Firefly Algorithm (FA).

The development of a new metaheuristic approach is crucial because it can efficiently and effectively explore the search space. A smart search process is needed to comprehensively evaluate all the search areas, as well as to branch out into previously unexplored regions with high-quality solutions. These two processes are known as intensification and diversification, respectively. Diversification refers to exploring regions that have not been adequately explored, while intensification involves utilizing the collected information by the metaheuristic at a particular point in time[115]. The exploration phase of a metaheuristic algorithm involves a systematic search for promising regions of the search space. During the exploration phase, the algorithm generates a diverse set of candidate solutions by using various strategies such as mutation, crossover, and local search. These techniques allow the algorithm to explore the search space and move towards promising regions that are likely to contain high-quality solutions. The exploration stage of a metaheuristic algorithm holds paramount importance as it plays a crucial role in preventing the algorithm from getting stuck in local optima and elevating the quality of solutions. By traversing diverse regions of the search space, the algorithm can discover superior solutions that are closer to the global optimum. Following the exploration phase, the algorithm usually transitions to an exploitation phase, where it focuses its search on the most favorable regions of the search space The balance between exploration and exploitation is an essential factor that determines the effectiveness of the algorithm in finding near-optimal solutions[116,117]. A proper balance of these two phases may guarantee that the global optimum is reached. Even though there are many metaheuristic algorithms, new algorithms are constantly necessary. As highlighted in No Free Lunch theory states, there remains no specific technique for obtaining the optimal answer for practically all optimization problems; developing new metaheuristic optimization algorithms remains an ongoing subject[118,119].

## Squid Game Optimizer Algorithm

**Inspiration.** Squid game, also known as *ojingeo*, is based on a Korean children's playground game and is basically a blend of tag and hopscotch. It is a multiplayer game with two primary goals: either for the attackers to complete the attack's goal or for the teams to eliminate each other. Squid game is often played on large and sandy fields, although players can play in vast, open, indoor, or outdoor areas. Meanwhile, there remain no recommendations or guidelines regarding the size and dimensions of the playground. The squid-shaped playfield seems to be approximately half the size of a typical basketball court, based on the history of the squid game.

As shown in Fig. 1a, squid game court resembles its namesake with a large body and tapering head. Begin by drawing the squid's head, which has a triangular tip and the square base of its body, indicated as section D. Section A is the big, rounded element, whereas section C is the triangle point within the circle. Section B is laid out at the exact centre of the horizontal line along the bottom of the squid, and it should be spacious enough for many participants to stand there, which is the main door or entry gate to the squid game court. Nevertheless, the bridge, which players may access via section E, is the exception to this rule. First and foremost, all players are divided into two groups: offensives and defensives (Fig. 1b). The offensive player must enter into section B as an offensive player while avoiding the other defensive players. The offensive team wins when any offensive player gets in section C with both feet, and to formally win the game, the offensive player must shout "Hurray." Defensive players can do whatever to get the offensive players off the field, whether it is pushing or pulling them over a line or knocking them down (Fig. 1c). If the defensive player can clear out all offensive players before reaching section C (Fig. 1d), they win. Only the defensive and offensive feet are permitted to contact the ground throughout the squid game. Players that are pushed, pulled, or tripped to the ground are ejected from the game.

This game begins by standing in section A. However, offensive players cannot bunny hop into section C and win the game immediately. They must exit section A and go to the other end of the court, where they may enter via section B. Subsequently, the offensive player in section D hop on 1 foot outside the court, making the game more equitable for the defensive team and equalizing the playing field for both teams. An offensive player could jump over the whole area F to play with both feet, which these parts are also called the bridge. Any offensive player who successfully leaps over this bridge on one foot is exempt from the 1-foot rule and may move on both feet for the remainder of the game. Offensive players may shove or force defensive players down if they get the chance, and reaching section C will be simpler if defensive players are removed.

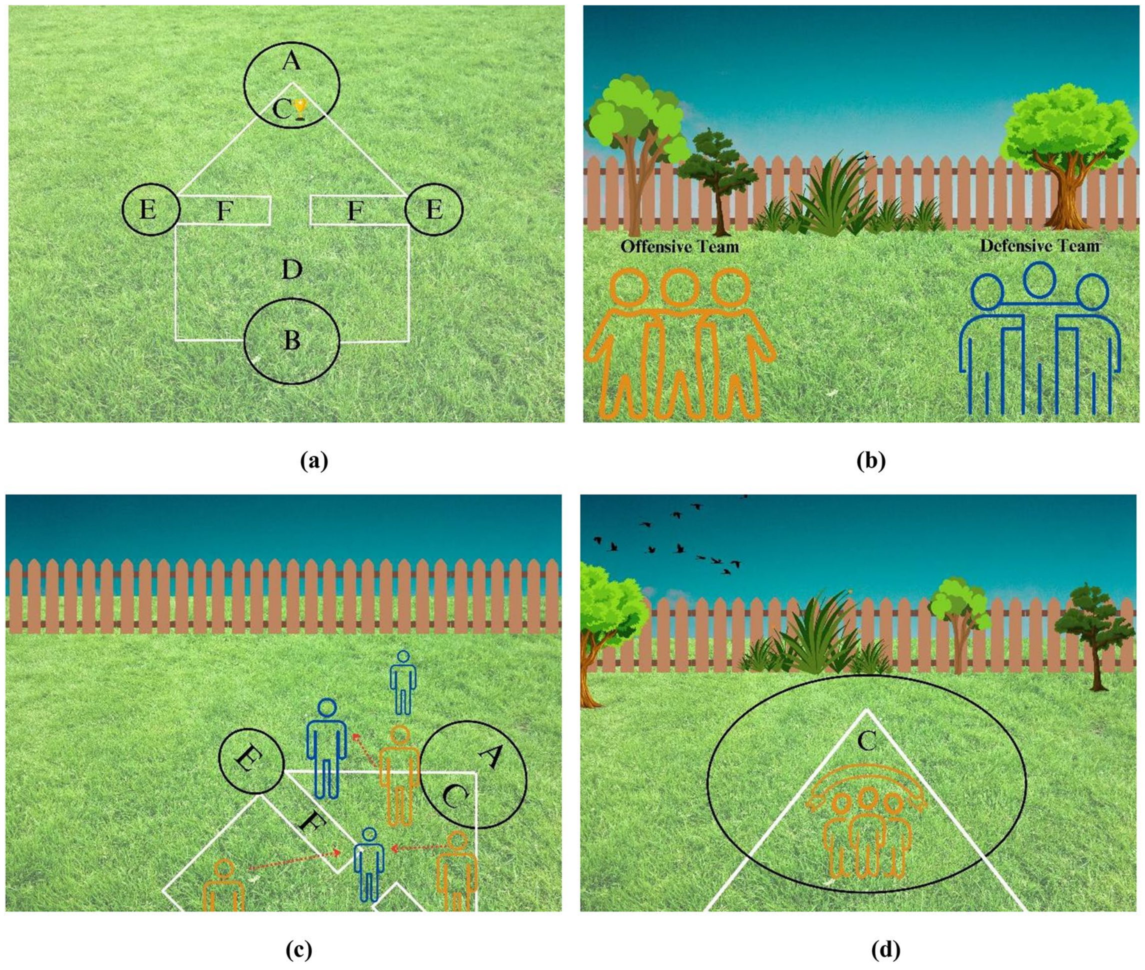

**Figure 1.** The schematic presentation of Squid Game.

Defensive players are allowed to move freely on both feet within the majority of the play area, which includes the circular sections. However, this defensive advantage comes to an end when any player exits sections D, A, B, or E. Similar to offensive players, defensive players must hop on one foot outside the court, and their team wins when they successfully eliminate all the offensive players.

**Mathematical model.** The mathematical presentation of the SGO as an optimization algorithm is explained in detail in this part using the strategy of Squid Game which has been discussed in detail in the earlier section. In the first stage, the initialization procedure is carried out as follows while the search space is considered as a particular part of the playground and the solution candidates ($X_i$) are assumed to be players:

$$X=\begin{bmatrix} X_1\\ X_2\\ \vdots\\ X_i\\ \vdots\\ X_n \end{bmatrix}=\begin{bmatrix} x_1^1x_1^2\cdots x_1^j\cdots x_1^d\\ x_2^1x_2^2\cdots x_2^j\cdots x_2^d\\ \vdots\vdots\vdots\ddots\vdots\\ x_i^1x_i^2\cdots x_i^j\cdots x_i^d\\ \vdots\vdots\vdots\ddots\vdots\\ x_n^1x_n^2\cdots x_n^j\cdots x_n^d \end{bmatrix},\begin{cases} i=1,2,\ldots,n.\\ j=1,2,\ldots,d. \end{cases} \tag{1}$$

$$x_i^j=x_{i,min}^j+rand.\left(x_{i,max}^j-x_{i,min}^j\right),\begin{cases} i=1,2,\ldots,n.\\ j=1,2,\ldots,d. \end{cases} \tag{2}$$

where $n$ indicates players' total number (solution candidates) in the playground (search space); $d$ is the considered problem's dimension; $x_i^j$ refers to the *jth* decision variable used to determine the *ith* candidate' initial position;$x_{i,max}^j$ and $x_{i,min}^j$ are the upper and lower bounds of the *jth* variable in the *ith* candidate; *rand* refers to a random number that is uniformly distributed within the range of [0, 1].

During the second stage of the algorithm, the players are separated into two equally sized groups: Offensives (Off) and Defensives (Def). The mathematical representation of these components is provided below:

$$X^{Off}=\begin{bmatrix} X_1^{Off}\\ X_2^{Off}\\ \vdots\\ X_i^{Off}\\ \vdots\\ X_m^{Off} \end{bmatrix}=\begin{bmatrix} x_1^1x_1^2\cdots x_1^j\cdots x_1^d\\ x_2^1x_2^2\cdots x_2^j\cdots x_2^d\\ \vdots\vdots\vdots\ddots\vdots\\ x_i^1x_i^2\cdots x_i^j\cdots x_i^d\\ \vdots\vdots\vdots\ddots\vdots\\ x_m^1x_m^2\cdots x_m^j\cdots x_m^d \end{bmatrix},\begin{cases} i=1,2,\ldots,m.\\ j=1,2,\ldots,d. \end{cases} \tag{3}$$

$$X^{Def}=\begin{bmatrix} X_1^{Def}\\ X_2^{Def}\\ \vdots\\ X_i^{Def}\\ \vdots\\ X_m^{Def} \end{bmatrix}=\begin{bmatrix} x_1^1x_1^2\cdots x_1^j\cdots x_1^d\\ x_2^1x_2^2\cdots x_2^j\cdots x_2^d\\ \vdots\vdots\vdots\ddots\vdots\\ x_i^1x_i^2\cdots x_i^j\cdots x_i^d\\ \vdots\vdots\vdots\ddots\vdots\\ x_m^1x_m^2\cdots x_m^j\cdots x_m^d \end{bmatrix},\begin{cases} i=1,2,\ldots,m.\\ j=1,2,\ldots,d. \end{cases} \tag{4}$$

where $m$ is players' total number in each group in the game; $X_i^{Off}$ is the *ith* offensive player; $X_i^{Def}$ is the *ith* defensive player.

Subsequently, after commencing the game, one offensive player goes among the defensive players to start a fight. It is worthwhile to mention that each offensive player has to move and fight with a single foot while the defensives are free to play with both feet. The mathematical representation of these components is provided below:

$$DG=\frac{\sum_{i=1}^{m}X_i^{Def}}{m},i=1,2,\ldots,m \tag{5}$$

$$X_i^{OffNew1}=\frac{X_i^{Off}+r_1\times DG-r_2\times X_{r_3}^{Def}}{2},i=1,2,\ldots,m \tag{6}$$

where $DG$ is the Defensive Group which mimics the crowd of defensive players; $X_i^{OffNew}$ is the upcoming *ith* offensive player' position vector ($X_i^{Off}$) in the playground;$r_1$ and $r_2$ are two random numbers within the bounds of [0, 1] which represents the capability of the offensive players in reaching any position between the *DG* and a randomly selected defensive player ($X_{r_3}^{Def}$); $r_3$ is a random integer number ranging from 1 to *m*.

In the next step, after fighting between the *ith* offensive player ($X_i^{Off}$) and a specific defensive player ($X_{r_3}^{Def}$), the objective function evaluation for each player is carried out and identified as the Winning State (WS) of the players. If the wining state of the defensive player is lower than the winning state of the offensive player ($WS_i^{Def} \leq WS_i^{Off}$), the offensive player is assumed as the winner of the game and join the Successful Offensive Group (*SOG*) (point C in Fig. 2) regarding the primary rules of the squid game while the offensive player can use both feet for this purpose. The mathematical representation of these facets can be expressed as:

$$X^{SccOff} = \begin{bmatrix} X_1^{SccOff} \\ X_2^{SccOff} \\ \vdots \\ X_i^{SccOff} \\ \vdots \\ X_o^{SccOff} \end{bmatrix} = \begin{bmatrix} x_1^1 x_1^2 \cdots x_1^j \cdots x_1^d \\ x_2^1 x_2^2 \cdots x_2^j \cdots x_2^d \\ \vdots \vdots \vdots \ddots \vdots \\ x_i^1 x_i^2 \cdots x_i^j \cdots x_i^d \\ \vdots \vdots \vdots \ddots \vdots \\ x_o^1 x_o^2 \cdots x_o^j \cdots x_o^d \end{bmatrix}, \begin{cases} i = 1, 2, \ldots, o. \\ j = 1, 2, \ldots, d. \end{cases} \tag{7}$$

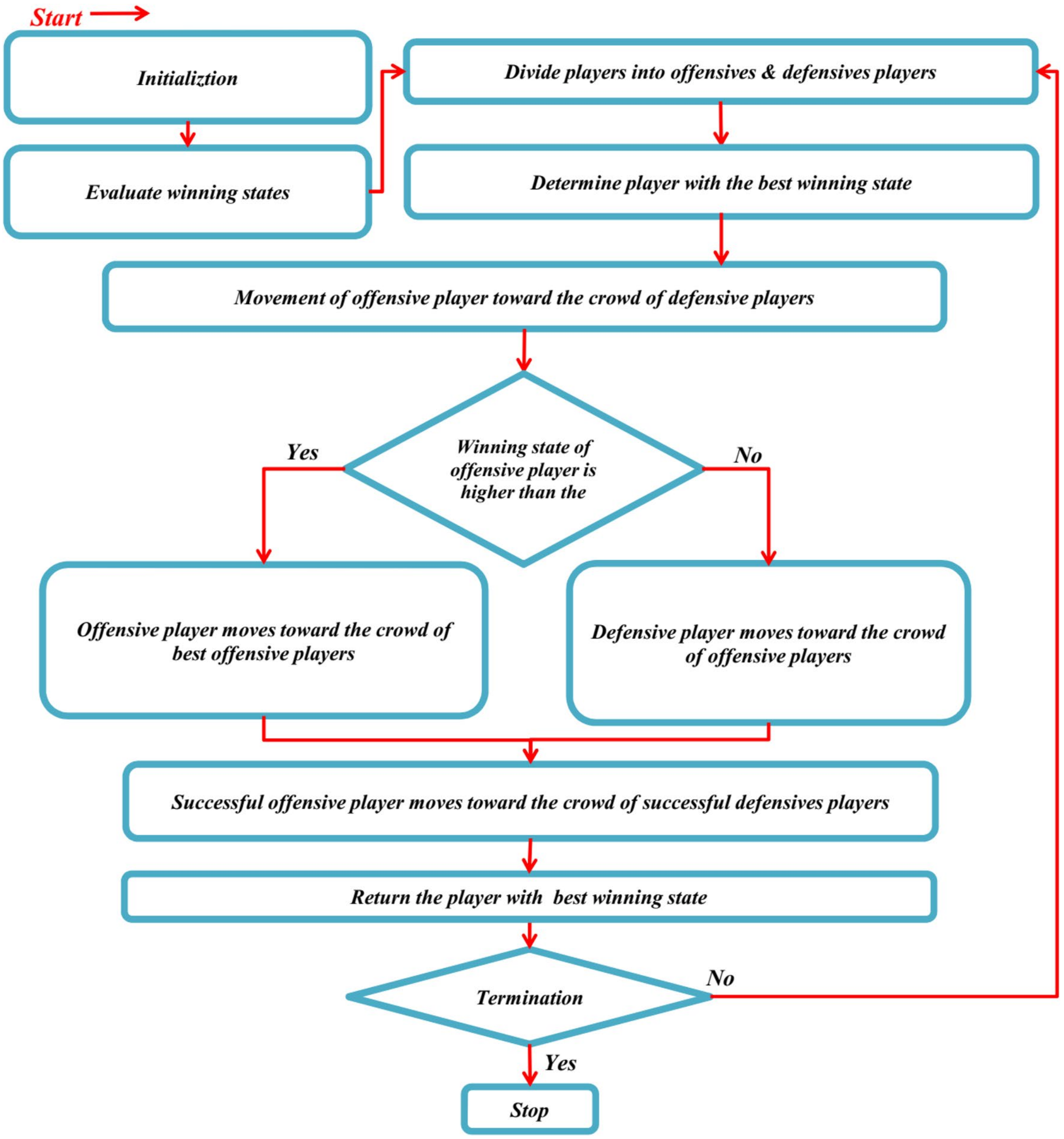
Start
Initializtion
Evaluate winning states
Divide players into offensives & defensives players
Determine player with the best winning state
Movement of offensive player toward the crowd of defensive players
Winning state of offensive player is higher than the
Yes
No
Offensive player moves toward the crowd of best offensive players
Defensive player moves toward the crowd of offensive players
Successful offensive player moves toward the crowd of successful defensives players
Return the player with best winning state
Termination
No
Yes
Stop

**Figure 2.** Flowchart of the SGO.

$$SOG = \frac{\sum_{i=1}^{o} X_i^{SccOff}}{o}, i = 1, 2, \ldots, o \tag{8}$$

$$X_i^{OffNew2} = X_i^{OffNew1} + r_1 \times SOG - r_2 \times BSi = 1, 2, \ldots, m \tag{9}$$

where $o$ is the number of successful offensive players in *SOG* which mimics the crowd of successful offensive players; $X_i^{OffNew2}$ shows the upcoming position vector of the *ith* offensive player ($X_i^{OffNew1}$); *BS* indicates the best solution candidate or the most successful offensive player in the *SOG*; $r_1$ and $r_2$ are two random numbers in the range of [0, 1].

If the winning state of the defensive player is higher than the winning state of the offensive player ($WS_i^{Def} > WS_i^{Off}$), the defensive player is considered as the winner of the game and join the Successful Defensive Group (*SDG*). The defensive players in this group are assumed to protect the critical point of the playground called the bridge (Point F in Fig. 2). Meanwhile, the successful defensive players go into the crowd of offensive players to be ready for starting a new fight. The mathematical presentation of these aspects are as follows:

$$SDG = \begin{bmatrix} X_1^{SccDef} \\ X_2^{SccDef} \\ \vdots \\ X_i^{SccDef} \\ \vdots \\ X_p^{SccDef} \end{bmatrix} = \begin{bmatrix} x_1^1 x_1^2 \cdots x_1^j \cdots x_1^d \\ x_2^1 x_2^2 \cdots x_2^j \cdots x_2^d \\ \vdots \vdots \vdots \ddots \vdots \\ x_i^1 x_i^2 \cdots x_i^j \cdots x_i^d \\ \vdots \vdots \vdots \ddots \vdots \\ x_p^1 x_p^2 \cdots x_p^j \cdots x_p^d \end{bmatrix}, \begin{cases} i = 1, 2, \ldots, p. \\ j = 1, 2, \ldots, d. \end{cases} \tag{10}$$

$$OG = \frac{\sum_{i=1}^{m} X_i^{Off}}{m}, i = 1, 2, \ldots, m) \tag{11}$$

$$X_i^{DefNew1} = X_i^{Def} + r_1 \times OG - r_2 \times X_{r_3}^{Off}\, i = 1, 2, \ldots, m \tag{12}$$

where *OG* is the Offensive Group which mimics the crowd of offensive players; $X_i^{DefNew1}$ is the upcoming *ith* defensive player' position vector ($X_i^{Def}$) in the playground (search space);$r_1$ and $r_2$ are two random numbers in the range of [0, 1] which represents the capability of the defensive players in reaching any position between the *OG* and a randomly selected offensive player ($X_{r_3}^{Off}$) to participate in a new fight; $r_3$ is a random integer number ranging from 1 to *m*.

To intelligently tune the exploration and exploitation phased of the proposed algorithm, another searching loop is implemented in the algorithm in which the offensive players in *SOG* try to pass the bridge which is protected by the defensive players in *SDG*. For this purpose, a position updating procedure is carried out for all of the offensive players in *SOG* by moving toward one specific defensive player in *SDG* (which mimics the bridge passing process) and the best so far found solution candidate (which mimics the reward for the offensive player to pass the bridge). The mathematical presentation of these aspects are as follows:

$$X_i^{OffNew3} = X_i^{SccOff} + r_1 \times BS - r_2 \times X_k^{SccDef} \begin{cases} i = 1, 2, \ldots, o. \\ k = 1, 2, \ldots, p. \end{cases} \tag{13}$$

where $o$ and $p$ is the number of successful offensive and deffensive players in *SOG* and *SDG* respectively; $X_i^{OffNew3}$ shows the upcoming position vector of the *ith* successful offensive player ($X_i^{OffNew1}$) which passes the bridge; *BS* indicates the best solution candidate or the most successful offensive player in the *SOG*; $r_1$ and $r_2$ are two random numbers in the range of [0, 1].

In order to handle the situation where the solution variables ($x_i^j$) do not satisfy the boundary conditions, a mathematical flag has been formulated. This flag directs the adjustment of the boundary for those variables which violate the range of variables. Besides, the termination criteria can be determined as a predefined value for the number of iterations or the number of function evaluations after which the optimization process is ended. The SGO method is described in detail below, while the flowchart and pseudo-code are shown in Figs. 2 and 3.

- Step 1: The initial positions of potential solutions, denoted as ($X_i$) or the "players" in the search space, are established through a random selection process.
- Step 2: The players are divided into two groups with equal populations, namely offensives (Off) and defensives (Def).
- Step 3: Each offensive player moves toward the crowd of a defensive player and one specific defensive player in order to start a fight.
- Step 4: The objective function for each offensive and defensive player are evaluated and indicated as the Winning State (WS).

```
        Evaluate fitness values for initial solution candidates as winning state of players ($WS_i$)
        while Iteration (Number of Function Evaluation) < Maximum number of iterations (Function Evaluations)
            Divide n players into two groups as offensives & defensives with m players in each group
            Determine the player with the best winning state (BS) in the playground
            for i=1:m
                The ith offensive player moves toward the crowd of defensive players (Eq. 6)
                Determine the winning state of ith offensive ($WS_i^{Off}$) and defensive players ($WS_i^{Def}$)
                if $WS_i^{Def} \leq WS_i^{Off}$
                    The ith offensive player moves toward the crowd of best offensive players (Eq. 9)
                    The ith offensive player joins the $SOG$
                else $WS_i^{Def} > WS_i^{Off}$
                    The ith defensive player moves toward the crowd of offensive players (Eq. 12)
                    The ith defensive player joins the $SDG$
                end if
            end for
            for i=1:o
                The ith successful offensive player moves toward the crowd of successful defensives (Eq. 13)
            end if
        end while
        Return the player with the best winning state (BS).
end Procedure
```

**Figure 3.** Pseudo code of the SGO.

- Step 5: The offensive players are able to join the Successful Offensive Group (*SOG*) if their WS is higher than that of the defensive player and moves toward the best offensive players in the playground.
- Step 6: The defensive players can join the Successful Defensive Group (*SDG*) if their WS is higher than that of the offensive player and moves toward the crowd of offensive players and one specific offensive player to be ready for another fight.
- Step 7: For each successful offensive player in the *SOG*, its position updating procedure is conducted by moving into the crowd of successful defensive players to pass the bridge.
- Step 8: The terminating criterion is checked.

## Mathematical test functions

To thoroughly examine the SGO algorithm, 25 unconstrained mathematical test functions, commonly used in global optimization, were selected, each with 100 dimensions. Table 1 presents a brief overview of these functions, and their full mathematical expressions have been documented by Jamil and Yang[120], Jamil et al.[121], Yang[122], Liang et al.[123], Talatahari and Azizi[39], and Talatahari et al.[44].

A total of 16 of the newest and most significant metaheuristic algorithms in optimization are used to assess the SGOA's overall performance compared to other metaheuristics, including the ABC[124], ACO[17], FA, GA[1], PSO[125], and WOA algorithms[126]. It is also worth mentioning that parameter tuning is imperative for some of these algorithms to have reasonable performance, extracting these parameters from the literature. Table 2 indicates the parameter presentation of the mentioned algorithms.

## Numerical results of the mathematical functions

In the current section, we present the outcomes of the SGO algorithm and six other metaheuristic algorithms on the 25 mathematical test functions that were investigated. The experiments were carried out by performing 150,000 objective function evaluations, and a stopping criterion of $1 \times 10^{-12}$ tolerance was employed for the SGO and the other metaheuristic algorithms. For statistical aims, 100 optimization runs are considered when calculating the mean and standard deviation of the optimization results. Furthermore, in dealing with the SGO and other alternatives, a fixed random state is considered to carry out a comparative examination under the same conditions. It is also noteworthy that the utilized tolerance in this paper is selected for 50 test functions while a lower tolerance such as $10^{-30}$ or $10^{-50}$ could be utilized for a smaller number of test functions.

The outcomes of the SGO algorithm and other 6 metaheuristics are given in Tables 3 and 4, which display the best values achieved in 100 optimization runs, along with the mean and standard deviation values for each of the 25 mathematical test functions. The analysis shows that the SGO algorithm generally outperforms the other metaheuristic algorithms.

The mean of objective function evaluations for each function considering the 100 optimization runs performed by the SGO and alternative algorithms demonstrate that the SGO outperforms a fast optimization process in most cases, whereas there is no need for completing the predefined 150,000 objective function evaluations for achieving the tolerance of $1 \times 10^{-12}$. The mean of objective function evaluations for SGO is 43,163.86, while WOA with 66,629.20, PSO with 135,698.30, GA with 141,524.40, ACO with 144,009.60, ABC with 144,011.90 and FA with 150,000 have the second to seventh ranks.

The convergence curves of different metaheuristic algorithms alongside the SGO are depicted in Fig. 2 in which the median run is used in each algorithm in dealing with each test function. It is obvious that the SGO has

| No | Name | Type | R | Min |
|---|---|---|---|---|
| $F_1$ | Ackley 1 | D, NS, C, Sc, M | [− 35, 35] | 0 |
| $F_2$ | Alpine 1 | ND, S, NSc, U, C | [− 10, 10] | 0 |
| $F_3$ | Brown | NS, Sc, D, C, U | [− 1, 4] | 0 |
| $F_4$ | Chung Reynolds | D, PS, Sc, C, U | [− 100, 100] | 0 |
| $F_5$ | Csendes | M, Sc, S, D, C | [− 1, 1] | 0 |
| $F_6$ | Deb 1 | S, D, C, Sc, M | [− 1, 1] | − 1 |
| $F_7$ | Dixon & Price | NS, Sc, C, D, U | [− 10, 10] | 0 |
| $F_8$ | Extended Easom | M, NSc, C, D, S | [− 2π, 2π] | − 1 |
| $F_9$ | Exponential | M, NS, Sc, C, D | [− 1, 1] | − 1 |
| $F_{10}$ | Griewank | M, Sc, NS, D, C | [− 100,100] | 0 |
| $F_{11}$ | Holzman 2 | S | [− 10, 10] | 0 |
| $F_{12}$ | Hyper-ellipsoid | U, C | [− 500, 500] | 0 |
| $F_{13}$ | Inverted cosine wave | NS | [− 10, 10] | − 99 |
| $F_{14}$ | Levy 8 | NS | [− 10, 10] | 0 |
| $F_{15}$ | Mishra 1 | M, Sc, NS, C, D | [0, 1] | 2 |
| $F_{16}$ | Pathological | M, NSc, NS, C, D | [− 100, 100] | 0 |
| $F_{17}$ | Pint´er | M, Sc, NS, C, D | [− 10, 10] | 0 |
| $F_{18}$ | Powell Singular | U, Sc, NS, C, D | [− 4, 5] | 0 |
| $F_{19}$ | Powell Sum | U, Sc, S, C, D | [− 1, 1] | 0 |
| $F_{20}$ | Rastrigin | M, S, D, C | [− 5.12, 5.12] | 0 |
| $F_{21}$ | Qing | M, C, D, Sc, S | [− 500, 500] | 0 |
| $F_{22}$ | Quintic | M, NSc, S, C, D | [− 10, 10] | 0 |
| $F_{23}$ | Rosenbrock | U, C, D, NS, Sc | [− 30, 30] | 0 |
| $F_{24}$ | Salomon | NS, Sc, M, C, D | [− 100, 100] | 0 |
| $F_{25}$ | Schumer Steiglitz | U, S, Sc, C, D | [− 100, 100] | 0 |

**Table 1.** The mathematical test functions used in this study possess certain fundamental characteristics.

| Algorithms | Parameter | Definition | Value |
|---|---|---|---|
| *FA* | $N_{pop}$ | Number of fireflies | 50 |
| | $\gamma$ | Light absorption coefficient | 1 |
| | $\beta$ | Attraction coefficient base value | 2 |
| | $\alpha$ | Mutation coefficient | 0.2 |
| | $\alpha_{damp}$ | Mutation coefficient damping ratio | 0.98 |
| | $\delta$ | Uniform mutation range | ± 0.05 |
| *GA* | $N_{pop}$ | Number of population | 50 |
| | $p_c$ | Crossover percentage | 0.8 |
| | $p_m$ | Mutation percentage | 0.3 |
| | $\mu$ | Mutation rate | 0.02 |
| | $\beta$ | Roulette wheel selection pressure | 1 |
| *ABC* | $N_{pop}$ | Colony size | 50 |
| | $\alpha$ | Acceleration coefficient upper bound | 1 |
| *ACO* | $N_{Ant}$ | Number of ants | 50 |
| | $\alpha$ | Pheromone exponential weight | 1 |
| | $\beta$ | Heuristic exponential weight | 1 |
| | $\rho$ | Evaporation rate | 0.05 |
| *PSO* | $NB$ | Swarm size | 50 |
| | $W$ | Initial inertia | 0.5 |
| | $C_1$ | Cognitive coefficient | 1.5 |
| | $C_2$ | Social coefficient | 1.5 |
| *WOA* | $N_{pop}$ | Number of search agents | 50 |

**Table 2.** Parameter presentation of the alternative metaheuristic algorithms.

| No | Alternative metaheuristic algorithms | | | | | | |
|---|---|---|---|---|---|---|---|
| | ABC | ACO | FA | GA | PSO | WOA | SGO |
| $F_1$ | 2.09E+01 | 2.09E+01 | 2.06E+01 | 2.17E+00 | 2.35E+00 | 0.00E+00 | 0.00E+00 |
| $F_2$ | 2.10E+02 | 2.21E+02 | 5.04E+01 | 9.68E−01 | 3.35E−01 | 0.00E+00 | 0.00E+00 |
| $F_3$ | 1.30E+07 | 2.48E+08 | 4.77E−01 | 1.74E−04 | 2.86E−07 | 0.00E+00 | 0.00E+00 |
| $F_4$ | 3.77E+10 | 4.99E+10 | 1.78E+10 | 1.73E−02 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| $F_5$ | 1.03E+01 | 1.03E+01 | 6.10E−06 | 0.00E+00 | 6.36E−08 | 0.00E+00 | 0.00E+00 |
| $F_6$ | − 5.28E−01 | − 5.25E−01 | − 9.66E−01 | − 1.00E+00 | − 1.00E+00 | − 1.00E+00 | − 1.00E+00 |
| $F_7$ | 1.63E+07 | 1.82E+07 | 2.99E+04 | 2.95E+01 | 6.67E−01 | 6.67E−01 | 9.99E−01 |
| $F_8$ | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | − 9.92E−01 | 0.00E+00 |
| $F_9$ | − 7.82E−05 | − 1.41E−05 | − 9.96E−01 | − 1.00E+00 | − 1.00E+00 | − 1.00E+00 | − 1.00E+00 |
| $F_{10}$ | 4.61E+01 | 5.68E+01 | 3.44E+01 | 1.05E−02 | 3.02E−08 | 0.00E+00 | 0.00E+00 |
| $F_{11}$ | 4.51E+06 | 4.51E+06 | 6.06E+03 | 1.77E−05 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| $F_{12}$ | 1.18E+30 | 4.34E+31 | 2.58E+33 | 4.63E+23 | 2.08E+08 | 0.00E+00 | 0.00E+00 |
| $F_{13}$ | − 6.98E+00 | − 5.39E+00 | − 1.54E+01 | − 4.14E+01 | − 4.97E+01 | − 9.90E+01 | − 9.90E+01 |
| $F_{14}$ | 6.79E+02 | 7.93E+02 | 3.54E+01 | 1.98E+01 | 4.36E+00 | 1.84E−02 | 6.45E−12 |
| $F_{15}$ | 2.00E+00 | 2.00E+00 | 2.10E+00 | 2.00E+00 | 2.00E+00 | 2.00E+00 | 2.00E+00 |
| $F_{16}$ | 4.60E+01 | 4.64E+01 | 2.79E+01 | 2.70E+01 | 2.89E+01 | 0.00E+00 | 0.00E+00 |
| $F_{17}$ | 1.42E+05 | 1.74E+05 | 4.22E+04 | 2.35E+04 | 1.15E+04 | 0.00E+00 | 0.00E+00 |
| $F_{18}$ | 2.70E+04 | 3.70E+04 | 1.18E+02 | 3.20E+00 | 1.87E−02 | 0.00E+00 | 0.00E+00 |
| $F_{19}$ | 4.51E−01 | 4.51E−01 | 3.73E−08 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| $F_{20}$ | 1.44E+03 | 1.49E+03 | 5.22E+02 | 3.82E+01 | 1.10E+02 | 0.00E+00 | 0.00E+00 |
| $F_{21}$ | 6.57E+11 | 6.57E+11 | 6.57E+11 | 1.45E+03 | 2.03E−02 | 3.03E+03 | 7.93E+04 |
| $F_{22}$ | 8.85E+05 | 9.13E+05 | 6.13E+02 | 4.37E+01 | 2.82E+01 | 1.31E+01 | 2.30E−04 |
| $F_{23}$ | 8.47E+08 | 8.47E+08 | 3.61E+07 | 4.87E+02 | 8.57E+01 | 9.51E+01 | 2.40E−06 |
| $F_{24}$ | 4.54E+01 | 4.91E+01 | 3.57E+01 | 2.34E+00 | 1.70E+00 | 0.00E+00 | 0.00E+00 |
| $F_{25}$ | 1.05E+09 | 1.05E+09 | 5.38E+08 | 1.90E−03 | 0.00E+00 | 0.00E+00 | 0.00E+00 |

**Table 3.** The best results of metaheuristic algorithms considering mathematical test functions.

better convergence behavior alongside the WOA comparing to other methods. Based on the results, the SGO is capable of converging to global best minimum by considering the $1 \times 10^{-12}$ tolerance in a faster way in dealing with 21 test functions while for the other 4, the results of other algorithms are competitive. Based on the results of Table 4 and Fig. 2, the WOA shows better convergence behavior in dealing with F7 and F14. Regarding F21, the results of GA, PSO and WOA are better than the SGO while for F6, the results of GA, PSO, FA and WOA are more competitive than the results of SGO (Fig. 4).

## Statistical analysis

In addition to presenting the mean and standard deviation of the outcomes, the authors recognize the limitation of these measures in providing a comprehensive evaluation of algorithm performance when dealing with test functions. As such, a comprehensive statistical analysis has been conducted to address this issue, and the most relevant statistical tests have been utilized to evaluate the results. This approach provides a more in-depth analysis of the algorithms' capabilities and can yield more accurate and reliable conclusions regarding their performance. By conducting a comprehensive statistical analysis, the study aims to provide a more robust evaluation of the SGO algorithm and its performance compared to other metaheuristics. The paper employs various statistical tests to analyze and evaluate the performance of the miscellaneous metaheuristic algorithms used in dealing with the test functions. These tests include the Kolmogorov Smirnov (KS) test, which assesses the normality of results, the Mann Whitney (MW) test, which compares the ranks of different metaheuristics two by two, and the Kruskal Wallis (KW) test, which compares the mean ranks of the metaheuristic algorithms to determine the overall rankings. By using these statistical tests, the paper provides a more comprehensive evaluation of the algorithms, beyond just the mean and standard deviation of results.

**Kolmogorov Smirnov (KS) test.** Table 5 shows the results of the Kolmogorov–Smirnov test used to identify whether to use non-parametric or parametric statistical tests for the collected data set, comprising the collection of mathematical test functions and other metaheuristic algorithms as alternatives. If the p-value of this test is less than 0.05, non-parametric tests are necessary, whereas if the p-value is higher than 0.05, parametric tests can be used. For future studies, the MW and KW tests, which are non-parametric statistical tests, should be selected based on the findings of the KS test.

| | | Alternative metaheuristic algorithms | | | | | | |
|---|---|---|---|---|---|---|---|---|
| No | Data | ABC | ACO | FA | GA | PSO | WOA | SGO |
| *F1* | *Mean* | 2.11E+01 | 2.11E+01 | 2.07E+01 | 2.62E+00 | 3.86E+00 | 0.00E+00 | 0.00E+00 |
| | *Std* | 6.28E−02 | 5.82E−02 | 3.40E−02 | 2.12E−01 | 5.88E−01 | 0.00E+00 | 0.00E+00 |
| *F2* | *Mean* | 2.33E+02 | 2.49E+02 | 6.41E+01 | 5.40E+00 | 2.26E+00 | 0.00E+00 | 6.15E−08 |
| | *Std* | 8.76E+00 | 1.01E+01 | 5.93E+00 | 2.90E+00 | 1.36E+00 | 0.00E+00 | 6.15E−07 |
| *F3* | *Mean* | 3.70E+10 | 5.79E+12 | 6.80E−01 | 8.11E−04 | 5.76E−03 | 0.00E+00 | 0.00E+00 |
| | *Std* | 7.69E+10 | 1.46E+13 | 1.01E−01 | 4.48E−04 | 1.35E−02 | 0.00E+00 | 0.00E+00 |
| *F4* | *Mean* | 5.73E+10 | 7.20E+10 | 2.29E+10 | 3.53E−01 | 7.18E+02 | 0.00E+00 | 0.00E+00 |
| | *Std* | 7.96E+09 | 7.13E+09 | 2.03E+09 | 3.22E−01 | 3.84E+03 | 0.00E+00 | 0.00E+00 |
| *F5* | *Mean* | 1.67E+01 | 1.75E+01 | 3.20E−05 | 3.90E−14 | 1.13E−06 | 0.00E+00 | 0.00E+00 |
| | *Std* | 1.99E+00 | 2.02E+00 | 2.69E−05 | 3.90E−13 | 7.12E−07 | 0.00E+00 | 0.00E+00 |
| *F6* | *Mean* | − 4.80E−01 | − 4.79E−01 | − 9.50E−01 | − 9.44E−01 | − 9.97E−01 | − 8.22E−01 | − 6.00E−01 |
| | *Std* | 1.04E−02 | 1.04E−02 | 6.90E−03 | 2.17E−02 | 4.61E−03 | 1.02E−01 | 2.26E−01 |
| *F7* | *Mean* | 2.60E+07 | 2.72E+07 | 5.59E+04 | 7.60E+01 | 2.06E+01 | 6.67E−01 | 1.00E+00 |
| | *Std* | 3.06E+06 | 3.05E+06 | 1.32E+04 | 3.63E+01 | 3.26E+01 | 1.84E−06 | 2.07E−04 |
| *F8* | *Mean* | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | − 9.58E−01 | 0.00E+00 |
| | *Std* | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 2.58E−02 | 0.00E+00 |
| *F9* | *Mean* | − 6.23E−06 | − 1.96E−06 | − 9.94E−01 | − 1.00E+00 | − 1.00E+00 | − 1.00E+00 | − 1.00E+00 |
| | *Std* | 8.73E−06 | 1.94E−06 | 1.11E−03 | 1.18E−05 | 1.03E−03 | 0.00E+00 | 0.00E+00 |
| *F10* | *Mean* | 6.08E+01 | 6.80E+01 | 3.88E+01 | 3.35E−02 | 8.19E−02 | 5.93E−04 | 0.00E+00 |
| | *Std* | 4.38E+00 | 3.37E+00 | 1.69E+00 | 1.40E−02 | 1.44E−01 | 4.22E−03 | 0.00E+00 |
| *F11* | *Mean* | 6.52E+06 | 6.78E+06 | 1.37E+04 | 1.81E−04 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| | *Std* | 6.98E+05 | 7.54E+05 | 3.88E+03 | 1.46E−04 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| *F12* | *Mean* | 2.01E+30 | 7.09E+31 | 1.03E+34 | 5.27E+26 | 1.52E+20 | 0.00E+00 | 0.00E+00 |
| | *Std* | 4.45E+29 | 1.64E+31 | 2.80E+33 | 1.21E+27 | 1.46E+21 | 0.00E+00 | 0.00E+00 |
| *F13* | *Mean* | − 5.40E+00 | −3.68E+00 | − 1.07E+01 | − 3.53E+01 | − 3.43E+01 | − 9.87E+01 | − 9.90E+01 |
| | *Std* | 5.05E−01 | 5.51E−01 | 1.50E+00 | 2.50E+00 | 5.38E+00 | 3.33E+00 | 0.00E+00 |
| *F14* | *Mean* | 8.44E+02 | 9.51E+02 | 5.79E+01 | 4.16E+01 | 1.50E+01 | 1.09E−01 | 4.88E+00 |
| | *Std* | 5.90E+01 | 6.19E+01 | 1.15E+01 | 1.32E+01 | 4.88E+00 | 1.49E−01 | 4.81E+00 |
| *F15* | *Mean* | 2.00E+00 | 2.00E+00 | 2.12E+00 | 2.00E+00 | 2.00E+00 | 2.00E+00 | 2.00E+00 |
| | *Std* | 0.00E+00 | 0.00E+00 | 6.66E−03 | 0.00E+00 | 3.79E−12 | 0.00E+00 | 0.00E+00 |
| *F16* | *Mean* | 4.68E+01 | 4.73E+01 | 3.03E+01 | 3.16E+01 | 3.33E+01 | 1.47E−02 | 3.62E−08 |
| | *Std* | 2.51E−01 | 2.52E−01 | 7.15E−01 | 1.28E+00 | 1.61E+00 | 1.01E−01 | 1.42E−07 |
| *F17* | *Mean* | 1.61E+05 | 2.01E+05 | 5.01E+04 | 3.87E+04 | 2.39E+04 | 0.00E+00 | 0.00E+00 |
| | *Std* | 7.99E+03 | 9.00E+03 | 2.91E+03 | 6.21E+03 | 5.00E+03 | 0.00E+00 | 0.00E+00 |
| *F18* | *Mean* | 4.97E+04 | 6.13E+04 | 2.06E+02 | 9.24E+00 | 4.80E−01 | 0.00E+00 | 0.00E+00 |
| | *Std* | 6.98E+03 | 8.93E+03 | 4.32E+01 | 2.50E+00 | 9.22E−01 | 0.00E+00 | 0.00E+00 |
| *F19* | *Mean* | 1.18E+00 | 1.30E+00 | 1.50E−07 | 1.11E−09 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| | *Std* | 3.11E−01 | 3.37E−01 | 8.03E−08 | 3.94E−09 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| *F20* | *Mean* | 1.57E+03 | 1.62E+03 | 6.11E+02 | 5.49E+01 | 1.78E+02 | 0.00E+00 | 0.00E+00 |
| | *Std* | 4.58E+01 | 4.78E+01 | 3.56E+01 | 5.48E+00 | 3.66E+01 | 0.00E+00 | 0.00E+00 |
| *F21* | *Mean* | 8.60E+11 | 8.87E+11 | 8.87E+11 | 4.17E+03 | 2.55E+04 | 1.34E+04 | 1.52E+05 |
| | *Std* | 6.21E+10 | 6.97E+10 | 6.97E+10 | 2.18E+03 | 1.47E+05 | 5.30E+03 | 6.45E+04 |
| *F22* | *Mean* | 1.14E+06 | 1.16E+06 | 1.01E+03 | 8.81E+01 | 6.74E+01 | 3.13E+01 | 2.36E+01 |
| | *Std* | 9.85E+04 | 1.04E+05 | 2.98E+02 | 1.86E+01 | 2.04E+01 | 1.44E+01 | 9.40E+01 |
| *F23* | *Mean* | 1.12E+09 | 1.14E+09 | 6.15E+07 | 8.83E+02 | 3.58E+02 | 9.57E+01 | 9.61E+01 |
| | *Std* | 8.38E+07 | 8.69E+07 | 1.12E+07 | 5.02E+02 | 6.16E+02 | 3.11E−01 | 1.38E+01 |
| *F24* | *Mean* | 4.98E+01 | 5.25E+01 | 3.96E+01 | 2.99E+00 | 3.33E+00 | 1.32E−01 | 0.00E+00 |
| | *Std* | 1.67E+00 | 1.30E+00 | 9.87E−01 | 2.59E−01 | 7.63E−01 | 6.17E−02 | 0.00E+00 |
| *F25* | *Mean* | 1.39E+09 | 1.42E+09 | 6.99E+08 | 1.81E−02 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| | *Std* | 9.79E+07 | 1.12E+08 | 5.84E+07 | 1.69E−02 | 0.00E+00 | 0.00E+00 | 0.00E+00 |

**Table 4.** The outcomes' mean and standard deviation for metaheuristic algorithms considering mathematical test functions.

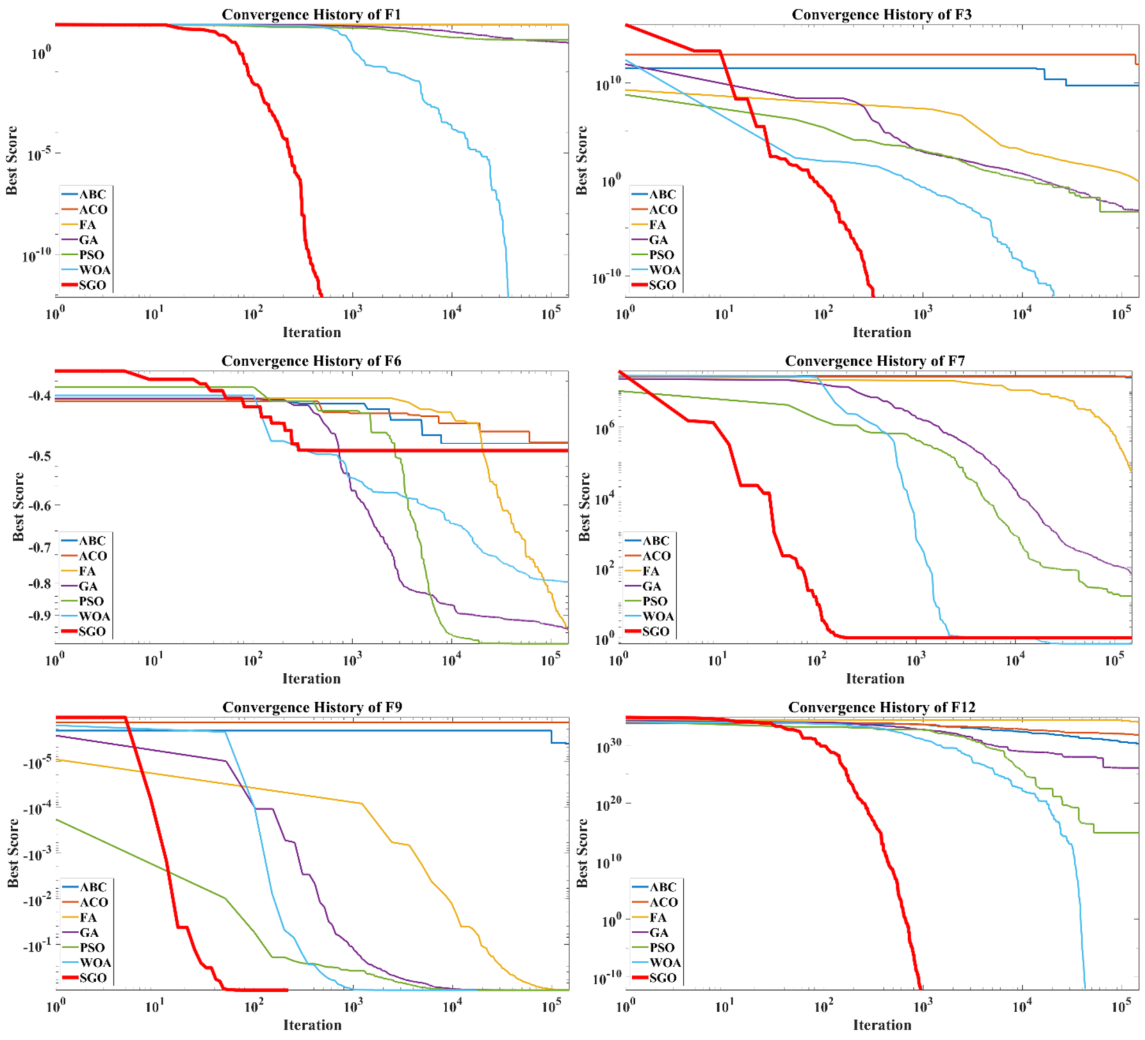

**Figure 4.** Convergence curves of metaheuristics regarding the median runs.

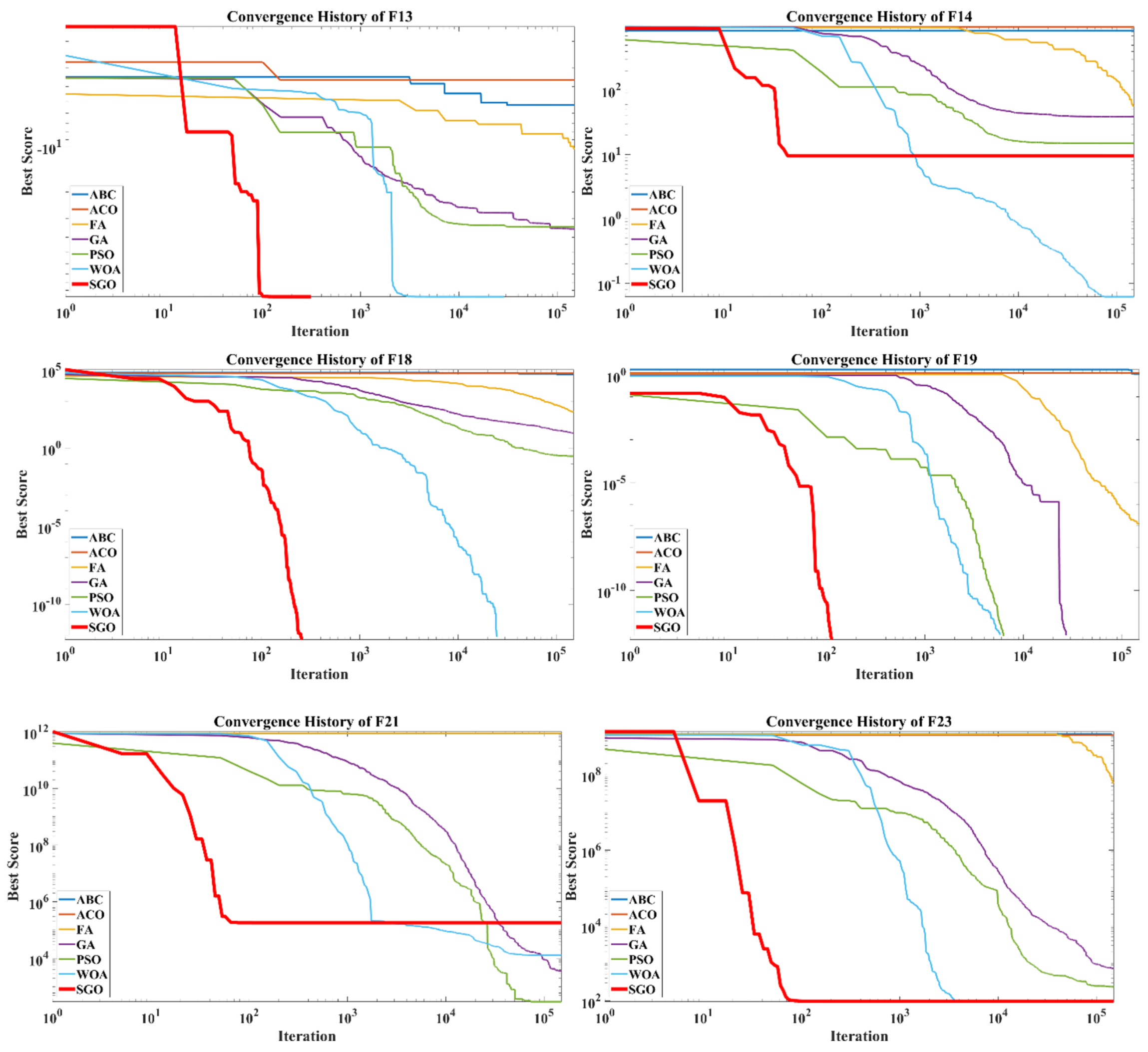

**Figure 4.** (continued)

| Main algorithm | Data type | Alternative Metaheuristic Algorithms | | | | | |
|---|---|---|---|---|---|---|---|
| | | ABC | ACO | FA | GA | PSO | WOA |
| SGO | Min | 9.36E−09 | 9.36E−09 | 5.57E−08 | 1.52E−06 | 1.12E−04 | 9.90E−01 |
| | Mean | 1.52E−06 | 1.52E−06 | 1.52E−06 | 1.12E−04 | 3.97E−04 | 8.77E−01 |
| | Std | 6.95E−06 | 6.95E−06 | 6.95E−06 | 1.12E−04 | 3.97E−04 | 6.49E−01 |
| | Fun. Evl | 6.95E−06 | 6.95E−06 | 1.52E−06 | 6.95E−06 | 6.95E−06 | 2.92E−05 |

**Table 5.** The KS test outcomes (p-values) of different metaheuristic algorithms.

**Mann Whitney (MW) test.** The MW test is conducted to compare the effectiveness of a wide variety of metaheuristics by considering the sum of their ranks. This test's null hypothesis is that the two chosen variables from different data sets have the same statistical behavior. The results of the MW test, which are given in Table 6, show the sum of ranks for various metaheuristics in dealing with the test functions. The bold values in the table show that the metaheuristics with lower sum of ranks have better statistical behavior. The majority of the outcomes indicate that the proposed SGO algorithm has a lower sum of ranks, indicating its superiority over other algorithms. It is important to mention that the MW test can be a valuable tool for comparing the statistical performance of various algorithms, particularly in terms of their capacity to identify the global optimum.

**Kruskal Wallis (KW) test.** The KW test is a statistical test that is frequently employed to compare the rankings of multiple variables across various datasets. In contrast to the MW and W tests, which are executed in a pairwise manner based on the summation and average of ranks, the KW test compares the average of ranks across datasets simultaneously. Lower mean ranks in a dataset are indicative of better statistical behavior in this

| Main algorithm | Data type | Alternative metaheuristic algorithms | | | | | |
|---|---|---|---|---|---|---|---|
| | | ABC | ACO | FA | GA | PSO | WOA |
| SGO | Min | 914.50 | 914.50 | 914.50 | 861.50 | 822.50 | 646.00 |
| | | 360.50 | 360.50 | 360.50 | 413.50 | 452.50 | 629.00 |
| | Mean | 895.50 | 895.50 | 883.50 | 832.50 | 815.50 | 652.00 |
| | | 379.50 | 379.50 | 391.50 | 442.50 | 459.50 | 623.00 |
| | Std | 868.00 | 868.00 | 864.50 | 827.00 | 815.00 | 672.00 |
| | | 407.00 | 407.00 | 410.50 | 448.00 | 460.00 | 603.00 |
| | Fun. Evl | 849.00 | 847.00 | 862.50 | 851.00 | 844.00 | 780.50 |
| | | 426.00 | 428.00 | 412.50 | 424.00 | 431.00 | 494.50 |

**Table 6.** The MW test outcomes (summation of ranks) of different metaheuristic algorithms.

test. Table 7 presents the statistical analysis of the performance of different metaheuristics on mathematical test functions using the KW test. The algorithms with the lowest mean ranks are bolded, highlighting their superior performance compared to other algorithms. The SGO algorithm consistently exhibited lower mean ranks in all cases, indicating its superiority over other algorithms.

## Computational complexity and cost analysis

Recent research has focused on developing algorithms that can optimize complex problems in a time- and computationally-efficient manner. Researchers have employed the computational complexity methods of the CEC 2020 benchmark suite on problems that have boundaries to overcome this challenge. In this suite, $T_0$ measures the specific mathematical process' run time, while $T_1$ measures the computational time required for 200,000 function evaluations of the G1 function. $T_2$ measures the mentioned algorithms' computational time, including SGO in this study, for 200,000 function evaluations of the $G_1$ function, and $\widehat{T}_2$ is the mean value of $T_2$ calculated five times[117]. Table 8 presents the outcomes of the computational time complexity of SGO and different algorithms using these procedures, demonstrating the superiority of the SGO algorithm regarding computational efficiency.

The "Big O notation" is a well-established mathematical notation that has widespread application in the scientific and mathematical domains, including in computational complexity studies for metaheuristic algorithms. To evaluate and contrast the effectiveness of different algorithms, it is common practice to assess their memory usage and execution time. However, while it is easy to set numerical values for the complexity of an algorithm, analyzing runtime concerns is a more complex issue that requires careful consideration. To eliminate the influence of computer and hardware constraints, other complexity procedures should be utilized for determining algorithmic complexity. The "Big O notation" is a widely-used term in computer science to quantify the required memory and run time of algorithms for comparative analysis. To compute the computational complexity of the SGO algorithm, the number of initial solution candidates (NP) and the problem dimension ($D$) are first calculated. The initialization phase of SGO has a computational complexity of $O(NP \times D)$, while the complexity of evaluating the objective function is $O(NP) \times O(F(x))$, in which $F(x)$ represents the objective function of the problem being considered. Each iteration in the main search loop of SGOA has a computational complexity equivalent to the number of iterations (*MxIter*). Updating the positions of each solution candidate in the search space during offensive and defensive player movements on the playground has a computational complexity of

| Rankings | Mathematical functions | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Min | | Mean | | Std | | Fun. Evl | |
| | Algorithms | Mean of ranks | Algorithms | Mean of ranks | Algorithms | Mean of ranks | Algorithms | Mean of ranks |
| 1 | **SGO** | **38.06** | **SGO** | **42.02** | **SGO** | **44.42** | **SGO** | **39.64** |
| 2 | WOA | 42.40 | WOA | 43.88 | WOA | 49.72 | WOA | 52.92 |
| 3 | PSO | 76.80 | PSO | 84.04 | GA | 89.32 | PSO | 96.36 |
| 4 | GA | 87.84 | GA | 85.72 | PSO | 90.24 | GA | 102.20 |
| 5 | FA | 116.74 | FA | 111.80 | FA | 108.24 | ACO | 106.88 |
| 6 | ABC | 126.46 | ABC | 123.56 | ABC | 116.84 | ABC | 107.00 |
| 7 | ACO | 127.70 | ACO | 124.98 | ACO | 117.22 | FA | 111.00 |
| *Chi-sq* | 85.12 | | 71.54 | | 53.99 | | 82.66 | |
| *Prob > Chi-sq* | 3.11E−16 | | 1.98E−13 | | 7.41E−10 | | 1.01E−15 | |

**Table 7.** The KW test outcomes, including the mean of the ranks regarding mathematical test functions. Significant values are in bold.

| Metaheuristics | Properties | Results (s) |
|---|---|---|
| *IMODE* | *T0* | 0.01117 |
| | *T1* | 0.2235 |
| | $\widehat{T}_2$ | 0.3330 |
| | $(\widehat{T}_2 - T1)/T0$ | 0.9780 |
| *j2020* | *T0* | 0 |
| | *T1* | 0.0465 |
| | $\widehat{T}_2$ | 0.1818 |
| | $(\widehat{T}_2 - T1)/T0$ | Inf |
| *GSK* | *T0* | 0.0411 |
| | *T1* | 1.12E−05 |
| | $\widehat{T}_2$ | 1.76E−05 |
| | $(\widehat{T}_2 - T1)/T0$ | 1.58E−04 |
| *SGO (Present Study)* | *T0* | 0.0226 |
| | *T1* | 0.0142 |
| | $\widehat{T}_2$ | 3.9279 |
| | $(\widehat{T}_2 - T1)/T0$ | 173.1725 |

**Table 8.** Different algorithms' computational time complexity concerning the CEC 2020 complexity process.

*O(MxIter × NP × D × 3)*. Finally, the objective function evaluation within the main search loop of SGOA has a computational complexity of *O(MxIter × NP × D × 3) × O(F(x))*.

## Real-world constrained optimization problems of CEC 2020

In academic research, metaheuristic algorithms are typically assessed for their effectiveness in solving real-world optimization problems that involve both bound and design constraints. In this study, the CEC 2020 benchmark suite of real-world constraint optimization problems is utilized[55], with Table 9 providing a summary of the engineering design problems and their mathematical formulations found in the literature. Figures 5, 6 and 7 present schematic representations of these problems. To ensure statistical validity, 30 independent optimization runs were carried out, each with 200,000 function evaluations. In order to handle the constraints of these problems, a prominent penalty technique with a static coefficient is employed in the current study.

The results of the speed reducer problem for SGO and other algorithms have been presented in Tables 10 and 11, which include optimal design variables and design constraints. The best optimization runs carried out by various techniques show that SGO can achieve a result of 2994.42, which is superior to other metaheuristics. In addition, the means and worst outcomes obtained by SGO are 2994.45 and 2994.48, respectively, which are more favourable than those of other methods. These statistical outcomes suggest that SGO is a highly effective and efficient algorithm for solving the speed reducer problem. The successful performance of SGO can be attributed to its capability to strike a balance between exploitation and exploration of the search space, as well as its efficient handling of the complex constraints inherent in this problem.

Tables 12 and 13 present the best and statistical outcomes of various algorithms, including the proposed SGO, for the hydro-static thrust bearing design problem. The tables also display the optimal design variables and design constraints. The SGO algorithm performed the best among all the approaches by achieving 1618.95, while the best result from other approaches was obtained by CGO at 1621.24. Moreover, SGO's means and worst results were 1778 and 1910, respectively, which is superior to the results of other methods.

Tables 14 and 15 present the statistical outcomes of the proposed SGO algorithm and other alternative algorithms in handling the maximization problem of rolling element bearing design, alongside the optimal design variables and constraints. The best optimization runs performed by various methods reveal that SGO achieves a TLBO result as 81,859.74, whereas the maximum value of the objective function is obtained by ALO with 85,546.63, taking into account that this particular problem is a maximization problem.

| No. (CEC No.) | Name | D | g | h |
|---|---|---|---|---|
| $H_1$ (RC15) | Speed reducer | 7 | 11 | 0 |
| $H_2$ (RC25) | Hydro-static thrust bearing | 4 | 7 | 0 |
| $H_3$ (RC28) | Rolling element bearing | 10 | 9 | 0 |

**Table 9.** Real-world constrained optimization problems. *D* dimensions, *g* number of inequality constraints, *h* number of equality constraints.

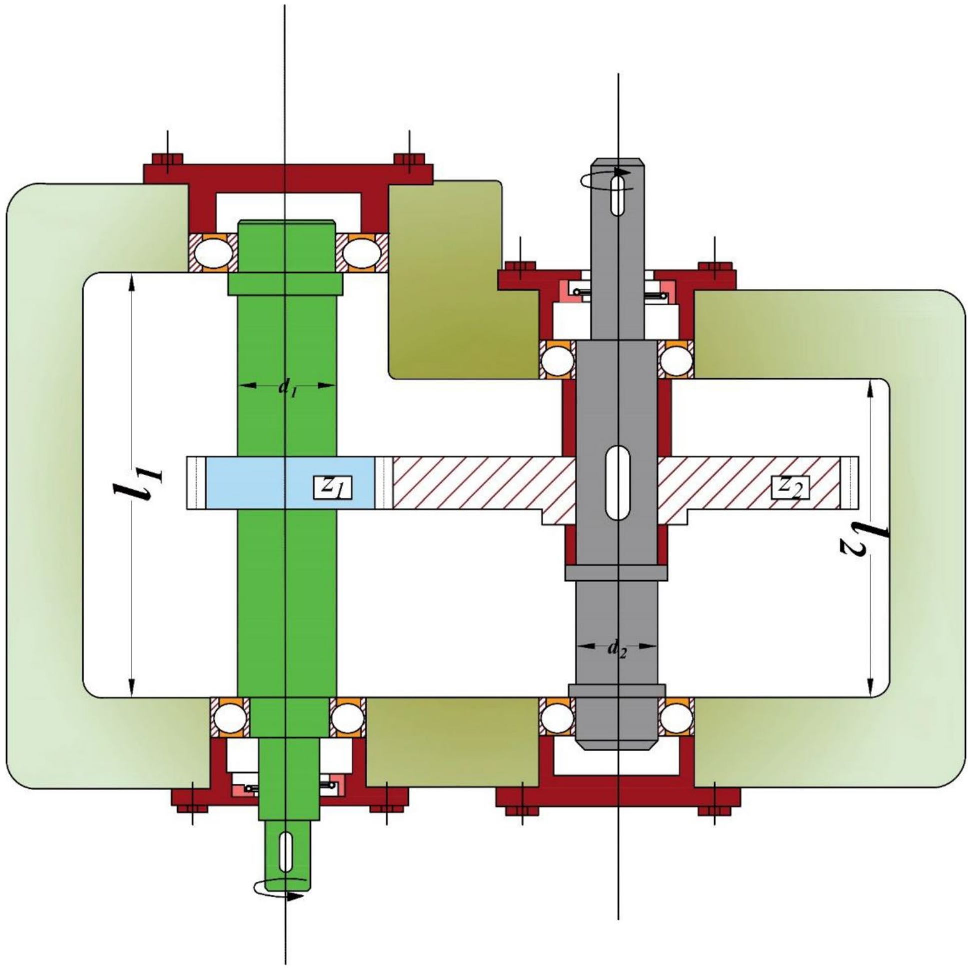

**Figure 5.** Schematic representation of the speed reducer problem.

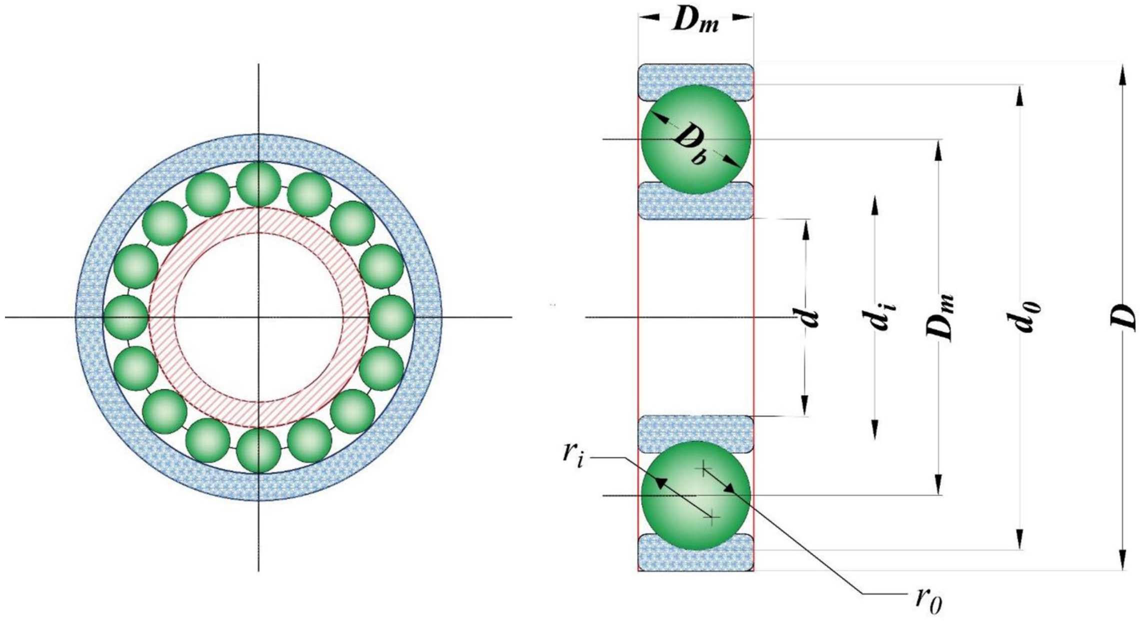

**Figure 6.** Schematic representation of the hydro-static thrust bearing.

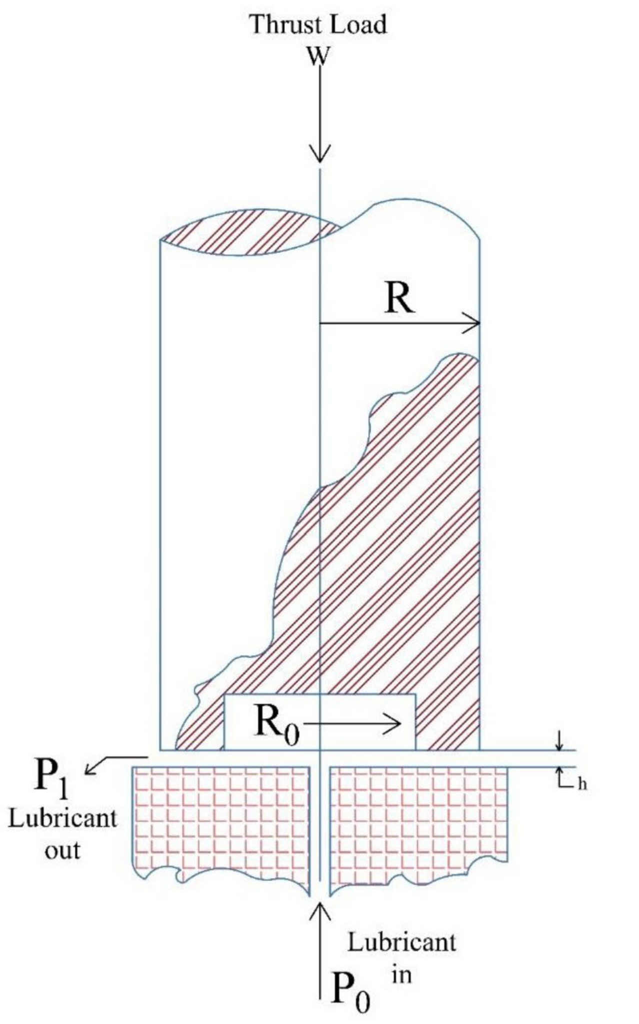

**Figure 7.** Schematic representation of the rolling element bearing problem.

## Discussion

Based on the provided results in the previous section, the capability of the proposed SGO algorithm have been investigated through different optimization problems. From mathematical functions point of view, the SGO can provide better competitive results including the best, mean, works and standard deviation while the SGO can provide lower mean of objective function evaluations. For 20 of the considered 25 mathematical test function, the SGO is capable of reaching to the global optimum point while the second best algorithm (WOA) is capable of reaching to global optima of 18 test functions. Regarding statistical test, the SGO can calculate lower mean and summation of ranks in competing with other alternative algorithms while the results of the WOA is also very close to the SGO's outcomes. The minimum difference between the SGO and other approaches regarding MW test is for minimum values in which the summation of ranks for SGO and WOA are 646 and 629 respectively. Regarding the mean of ranks of different approaches, the maximum difference is between SGO and ACO for minimum values with mean ranks of 38.06 and 127.70 respectively. For the considered engineering optimization problem, the SGO can reach better optimum design in two of them by providing better statistical results too while for one of them, the SGO provides very competitive results. SGO can reach 2994.42 for the speed reducer problem which is the best among other approaches while the DE algorithm can calculate better standard deviation for this case. SGO is capable of calculating 1618.98 for the hydro-static thrust bearing design problem as the best among other methods while the MGA with 0.11 as standard deviation has better outcome than other methods.

## Conclusion and future directions

The Squid Game Optimizer (SGO) algorithm is a novel proposed metaheuristic algorithm inspired by the primary rules of a traditional Korean game. Metaheuristic algorithms are powerful optimization techniques that are widely used to solve complex optimization problems with various constraints and objectives. The SGO algorithm is a population-based optimization algorithm that mimics the behaviours of the players in the Squid Game by dividing the population into different groups based on their fitness values and applying different search strategies

| | ES[127] | SBS[128] | CSA[129] | DE[130] | MGA[35] | CGO[131] | Present study (SGO) |
|---|---|---|---|---|---|---|---|
| Best | 3025.0050000000 | 3008.0800000000 | 3000.9810000000 | 2994.4710660000 | 2994.4436490000 | 2994.4388690000 | **2994.4248154921** |
| $b$ | 3.5061630000 | 3.5061220000 | 3.5015000000 | 3.5000000000 | 3.5000066840 | 3.5000079560 | 3.5000001594 |
| $m$ | 0.7008310000 | 0.7000060000 | 0.7000000000 | 0.7000000000 | 0.7000000000 | 0.7000006560 | 0.7000000038 |
| $z$ | 17.0000000000 | 17.0000000000 | 17.0000000000 | 17.0000000000 | 17.0000000000 | 17.0000008100 | 17.0000000080 |
| $l_1$ | 7.4601810000 | 7.5491260000 | 7.6050000000 | 7.3000000000 | 7.3000000000 | 7.3005419270 | 7.3000048291 |
| $l_2$ | 7.9621430000 | 7.8593300000 | 7.8181000000 | 7.7153199100 | 7.7153272500 | 7.7153576930 | 7.7153232372 |
| $d_1$ | 3.3629000000 | 3.3655760000 | 3.3520000000 | 3.3502146700 | 3.3505953620 | 3.3505423910 | 3.3505412814 |
| $d_2$ | 5.3090000000 | 5.2897730000 | 5.2875000000 | 5.2866544700 | 5.2866584470 | 5.2866579300 | 5.2866545726 |
| $g_1(x)$ | – 0.0777000000 | – 0.0755000000 | – 0.0743000000 | – 0.0739152 | – 2.1550556750 | – 2.1551222770 | – 2.1550016575 |
| $g_2(x)$ | – 0.2013000000 | – 0.1994000000 | – 0.1983000000 | – 0.1979985 | – 98.1359464800 | – 98.1371022200 | – 98.1350284098 |
| $g_3(x)$ | – 0.4741000000 | – 0.4562000000 | – 0.4349000000 | – 0.9999967 | – 1.9253722120 | – 1.9242737610 | – 1.9251156798 |
| $g_4(x)$ | – 0.8971000000 | – 0.8994000000 | – 0.9008000000 | – 0.9999995 | – 18.3099259200 | – 18.3096983400 | – 18.3098982853 |
| $g_5(x)$ | – 0.0110000000 | – 0.0132000000 | – 0.0011000000 | – 0.6668526000 | – 0.0535900310 | – 0.0004371520 | – 0.0003184915 |
| $g_6(x)$ | – 0.0125000000 | – 0.0017000000 | – 0.0004000000 | – 0.0000000 | – 0.0019196020 | – 0.0016664740 | – 0.0000513607 |
| $g_7(x)$ | – 0.7022000000 | – 0.7025000000 | – 0.7025000000 | – 0.7025000000 | – 28.1000000000 | – 28.0999882900 | – 28.0999999300 |
| $g_8(x)$ | – 0.0006000000 | – 0.0017000000 | – 0.0004000000 | – 0.0000000 | – 0.0000095500 | – 0.0000066800 | – 0.0000002007 |
| $g_9(x)$ | – 0.5831000000 | – 0.5826000000 | – 0.5832000000 | – 0.5833333 | – 6.9999904520 | – 6.9999933180 | – 6.9999997993 |
| $g_{10}(x)$ | – 0.0691000000 | – 0.0796000000 | – 0.0890000000 | – 0.0513257 | – 0.3741069580 | – 0.3747283410 | – 0.3741929070 |
| $g_{11}(x)$ | – 0.0279000000 | – 0.0179000000 | – 0.0130000000 | – 0.0000000 | – 0.0000029600 | – 0.0000340000 | – 0.0000032074 |

**Table 10.** Best outcomes of miscellaneous approaches for the speed reducer problem. Teeth module (m), face width (b), length of the first shaft between bearings (l1), the diameter of the first shaft (d1), number of teeth on pinion (z), length of the second shaft between bearings (l2), the diameter of the second shaft (d2). Significant values are in bold and italics.

| Approaches | Best | Mean | Worst | Std-Dev |
|---|---|---|---|---|
| ES[127] | 3025.0050000000 | 3088.7778000000 | 3078.5918000000 | NA |
| SBS[128] | 3008.0800000000 | 3012.1200000000 | 3028.2800000000 | NA |
| CSA[129] | 3000.9810000000 | 3007.1997000000 | 3.0090000000 | 4.9634000000 |
| DE[130] | 2994.4710660000 | 2994.4710660000 | 2994.4710660000 | **0.0000000000** |
| MGA[35] | 2994.4388690000 | 2994.4706500000 | 2996.5582370000 | 4.72E−16 |
| CGO[131] | 2994.4436490000 | 2994.4653970000 | 2995.5049330000 | 0.1102820000 |
| Present Study (SGO) | **2994.4248154921** | **2994.4553460550** | **2994.4899883644** | 0.0202512078 |

**Table 11.** Statistical results for the speed reducer problem considering different approaches. Significant values are in bold and italics.

| | COM[132] | CGS[133] | EA[134] | TLBO[51] | MGA[35] | CGO[131] | Present Study (SGO) |
|---|---|---|---|---|---|---|---|
| Best | 2288.2268000000 | 2161.4215000000 | 1950.2860000000 | 1625.4427600000 | 1623.9809380000 | 1621.2461750000 | **1618.9878101725** |
| $R$ | 7.1550000000 | 6.7780000000 | 6.2710000000 | 5.9557805026 | 5.9632415160 | 5.9634400230 | 5.9560709112 |
| $R_0$ | 6.6890000000 | 6.2340000000 | 12.9010000000 | 5.3890130519 | 5.3959079890 | 5.3955878610 | 5.3889322344 |
| $\mu$ | 0.0000083210 | 6.096 E−06 | 0.0000056050 | 0.0000053586 | 0.0000053800 | 0.0000053600 | 0.0000053785 |
| Q | 9.1680000000 | 3.8090000000 | 2.9380000000 | 2.2696559728 | 2.2822425050 | 2.2648221880 | 2.2726159184 |
| $g_1(x)$ | – 11,086.7430000000 | – 8329.7681000000 | – 2126.8673400000 | – 0.0001374735 | – 144.9586796000 | – 9.0788651780 | – 3.3136822083 |
| $g_2(x)$ | – 402.4493000000 | – 177.3527000000 | – 68.0396000000 | – 0.0000010103 | – 1.1948020210 | – 2.5136231960 | – 0.1030173878 |
| $g_3(x)$ | – 35.0571960000 | – 10.6845430000 | – 3.7051910000 | – 0.0000000210 | – 0.3724500270 | – 0.0021106440 | – 0.3149154198 |
| $g_4(x)$ | – 0.0015420000 | – 0.0006520000 | – 0.0005590000 | – 0.0003243625 | – 0.0003291500 | – 0.0003248340 | – 0.0003268968 |
| $g_5(x)$ | – 0.4660000000 | – 0.5440000000 | – 0.6660000000 | – 0.5667674507 | – 0.5673335270 | – 0.5678521610 | – 0.5671386768 |
| $g_6(x)$ | – 0.0001440000 | – 0.0007170000 | – 0.0008050000 | – 0.0009963614 | – 0.0009963550 | – 0.0009963660 | – 0.0009963634 |
| $g_7(x)$ | – 563.6444010000 | – 83.6182210000 | – 849.7186830000 | – 0.0000090762 | – 4.1442588760 | – 15.3591184600 | – 3.2036645069 |

**Table 12.** Comparison of the best solutions for the hydro-static thrust bearing design problem. Bearing step radius (R), recess radius (R0), oil viscosity (μ), flow rate (Q). Significant values are in bold and italics.

| Approaches | Best | Mean | Worst | Std-Dev |
|---|---|---|---|---|
| EGWO[135] | 1625.4646700000 | 1627.7441980000 | 1650.6987470000 | 3.8155469730 |
| JA[136] | 1625.4427100000 | 1796.8936700000 | 2104.3776000000 | 0.2100000000 |
| TLBO[138] | 1625.4427600000 | 1797.7079800000 | 2096.8012000000 | 0.1900000000 |
| MGA[35] | 1621.2461750000 | 1739.1567290000 | 1992.9613050000 | ***0.1100000000*** |
| CGO[131] | 1621.2461750000 | 1706.0414310000 | 1981.1732950000 | 64.4989571200 |
| Present Study (SGO) | **1618.9878101725** | **1778.2800314929** | **1910.7840110922** | 83.0282071657 |

**Table 13.** Statistical results of different approaches for the hydro-static thrust bearing design problem. Significant values are in bold and italics.

| | TLBO[137] | ABC[138] | GWO[138] | ALO[138] | MGA[35] | CGO[131] | Present Study (SGO) |
|---|---|---|---|---|---|---|---|
| Best | 81,859.7400000000 | 85,428.2495000000 | 85,529.0830000000 | ***85,546.6377000000*** | 83,912.8798300000 | 83,918.4925300000 | 81,859.7402667138 |
| $D_m$ | 21.4255900000 | 125.6599000000 | 125.7090000000 | 125.7180000000 | 125.0002787000 | 125.0000000000 | 125.7190555843 |
| $D_b$ | 125.7191000000 | 21.4086200000 | 21.4231600000 | 21.4252420000 | 21.8745119200 | 21.8750000000 | 21.4255902292 |
| $Z$ | 11.0000000000 | 11.0000000000 | 11.0000000000 | 11.0000000000 | 10.7770658300 | 10.7770090500 | 10.7258834098 |
| $f_i$ | 0.5150000000 | 0.5150000000 | 0.5150000000 | 0.5150000000 | 0.5150008220 | 0.5150000000 | 0.5150000007 |
| $f_0$ | 0.5150000000 | 0.5150000000 | 0.5293220000 | 0.5157018000 | 0.5150029930 | 0.5150000000 | 0.5150000000 |
| $K_{Dmin}$ | 0.4242660000 | 0.4271660000 | 0.4208670000 | 0.4541646000 | 0.4059083530 | 0.4000000000 | 0.4240900558 |
| $K_{Dmax}$ | 0.6339480000 | 0.6688490000 | 0.6332960000 | 0.6464928000 | 0.6555880200 | 0.6462005260 | 0.6974482305 |
| $\varepsilon$ | 0.3000000000 | 0.3000000000 | 0.3002240000 | 0.3000122000 | 0.3000041550 | 0.3000000000 | 0.3000000000 |
| $e$ | 0.0688580000 | 0.0713860000 | 0.0200000000 | 0.0638003000 | 0.0775449260 | 0.0501524450 | 0.0449615191 |
| $\zeta$ | 0.7994980000 | 0.6000000000 | 0.6194320000 | 0.6107592000 | 0.6000000000 | 0.6000000000 | 0.6071352958 |

**Table 14.** Comparison of the best solutions for the rolling element bearing design problem. Pitch diameter (Dm), ball diameter (Db), total number of balls (Z), inner raceway curvature coefficient (fi), and the outer raceway curvature coefficient (f0). Significant values are in bold and italics.

| Approaches | Best | Mean | Worst | Std-Dev |
|---|---|---|---|---|
| TLBO[137] | 81,859.7400000000 | 81,438.9870000000 | 80,807.8551000000 | ***0.6600000000*** |
| ABC[138] | 85,428.2495000000 | ***85,121.7544000000*** | ***83,859.0851000000*** | 362.5700000000 |
| GWO[138] | 85,529.0830000000 | 83,395.0849000000 | 43,543.4508000000 | 8224.5000000000 |
| ALO[138] | ***85,546.6377000000*** | 84,032.8636000000 | 73,872.8164000000 | 3121.8000000000 |
| MGA[35] | 83,912.8798300000 | 83,892.2564700000 | 83,711.2131700000 | 23.6584100000 |
| CGO[131] | 83,918.4925300000 | 83,916.5974900000 | 83,829.8000000000 | 10.5358000000 |
| Present Study (SGO) | 81,859.7402667138 | 80,404.8554383897 | 79,156.2618531067 | 719.1899843849 |

**Table 15.** Statistical results of different approaches for the rolling element bearing design problem. Significant values are in bold and italics.

to each group. This methodology enables the SGO algorithm to effectively explore the search space and prevent being trapped in local optima. To assess the performance of the SGO algorithm, 25 unconstrained mathematical test functions were employed in this study. To ensure the statistical significance of the results, 100 independent optimization runs were executed. Additionally, the SGO algorithm was evaluated on two of the most recent CEC, specifically the CEC 2020 for bound constraint optimization and the CEC 2020 for real-world optimization. The CEC benchmarks are considered a widely recognized standard evaluation platform for optimization algorithms, which ensures a fair and unbiased comparison among different algorithms. The experimental results indicate that the SGO algorithm is highly competitive and exhibits superior performance compared to other well-known metaheuristics on a diverse set of optimization problems. The SGO algorithm can discover high-quality solutions with faster convergence rates, better solution accuracy, and higher diversity in the solution space. The SGO algorithm is also shown to be robust and scalable, and can effectively handle optimization problems with various degrees of complexity and dimensions. SGO has shown superior performance compared to other metaheuristic algorithms in dealing with unconstrained mathematical test functions and can converge to the global best solutions in most situations. Here are the main outcomes and key findings of the research:

- SGO was observed to have high computational efficiency, requiring the lowest number of objective function evaluations to reach the global optimum solution. This suggests that SGO has a promising potential to be used in various optimization problems that require efficient algorithms.
- By comparing various algorithms in a two-by-two way, the SGO could offer superior outcomes with smaller means of rankings in most cases based on the MW statistical test findings.
- KW statistical test's outcomes, which includes the mean rank, suggest that in all evaluated datasets, the SGO algorithm outperforms other algorithms.
- The SGO algorithm was able to produce superior solutions than other approaches in the literature for the CEC 2020 on real-world problems' constrained design examples.
- The optimal solution obtained by SGO in the speed reducer problem, based on the best optimization runs among various algorithms, is 2994.4248, which outperforms the results of other methods.
- SGO outperforms other algorithms in solving the hydro-static thrust bearing design problem, achieving the best result of 1618.9878. The closest result from other algorithms is for CGO, which achieved 1621.24.
- Considering the rolling element bearing design problem, SGO could provide 81,859.74, while the TLBO provides the best maximum value of the objective function in this case.

The experimental results clearly demonstrate that the SGO algorithm outperforms other well-established metaheuristic algorithms in terms of three critical factors: parameter-free, quick convergence, and the lowest possible objective function evaluation. The parameter-free nature of the SGO algorithm makes it more user-friendly and easier to implement in practical applications, eliminating the need for fine-tuning of parameters to achieve optimal performance. The quick convergence behaviour of the SGO algorithm enables it to discover high-quality solutions more efficiently, thereby reducing the overall computational time and cost of optimization. Finally, the lowest possible objective function evaluation of the SGO algorithm results in faster and more accurate optimization, making it a highly efficient and effective optimization technique. These three advantages of the SGO algorithm make it a promising tool for solving complex optimization problems in various fields, including real-size engineering design problems. The robustness and scalability of the SGO algorithm, as shown in the experimental results, suggest that it can effectively handle optimization problems with various degrees of complexity and dimensions, making it suitable for real-world applications.

The findings of this study have significant implications for future research in the field of evolutionary computation and optimization. The proposed SGO algorithm provides a novel and effective approach to solving complex optimization problems, and its three main advantages over other metaheuristic algorithms open up new avenues for research on optimization techniques with similar features. Moreover, the proposed SGO algorithm can be extended and customized for different applications by incorporating specific problem constraints and objectives, providing a highly adaptable optimization framework. Therefore, future studies should test the SGO algorithm on a wider range of optimization problems and further explore its potential in various real-world applications.

Received: 22 December 2022; Accepted: 28 March 2023
Published online: 01 April 2023

## References

1. Holland, J. H. In *Adaptive Control of Ill-Defined Systems* (eds Oliver, G. S. *et al.*) 317–333 (Springer, 1984).
2. Storn, R. & Price, K. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **11**, 341–359. https://doi.org/10.1023/A:1008202821328 (1997).
3. Chakraborty, U. K. *Advances in Differential Evolution* Vol. 143 (Springer, 2008).
4. Atashpaz-Gargari, E. & Lucas, C. In *2007 IEEE Congress on Evolutionary Computation* 4661–4667 (IEEE).
5. Kaveh, A. & Talatahari, S. A novel heuristic optimization method: Charged system search. *Acta Mech.* **213**, 267–289. https://doi.org/10.1007/s00707-009-0270-4 (2010).
6. Talatahari, S. & Azizi, M. In *Nature-Inspired Metaheuristic Algorithms for Engineering Optimization Applications* (eds Serdar, C. *et al.*) 309–334 (Springer, 2021).
7. Talatahari, S., Azizi, M., Toloo, M. & BaghalzadehShishehgarkhaneh, M. Optimization of large-scale frame structures using fuzzy adaptive quantum inspired charged system search. *Int. J. Steel Struct.* **22**, 686–707. https://doi.org/10.1007/s13296-022-00598-y (2022).
8. Hosseini, H. S. In *2007 IEEE Congress on Evolutionary Computation* 3226–3231 (IEEE).
9. Kaveh, A., Talatahari, S. & Khodadadi, N. Stochastic paint optimizer: theory and application in civil engineering. *Eng. Comput.* **2020**, 1–32 (2020).
10. Khodadadi, N., Abualigah, L. & Mirjalili, S. Multi-objective stochastic paint optimizer (MOSPO). *Neural Comput. Appl.* **34**, 18035–18058 (2022).
11. Askari, Q., Younas, I. & Saeed, M. Political optimizer: A novel socio-inspired meta-heuristic for global optimization. *Knowl.-Based Syst.* **195**, 105709. https://doi.org/10.1016/j.knosys.2020.105709 (2020).
12. Topal, A. O. & Altun, O. A novel meta-heuristic algorithm: Dynamic virtual bats algorithm. *Inf. Sci.* **354**, 222–235. https://doi.org/10.1016/j.ins.2016.03.025 (2016).
13. Braik, M., Ryalat, M. H. & Al-Zoubi, H. A novel meta-heuristic algorithm for solving numerical optimization problems: Ali Baba and the forty thieves. *Neural Comput. Appl.* **34**, 409–455. https://doi.org/10.1007/s00521-021-06392-x (2022).
14. Ab Rashid, M. F. F. Tiki-taka algorithm: A novel metaheuristic inspired by football playing style. *Eng. Comput.* **38**, 313–343. https://doi.org/10.1108/EC-03-2020-0137 (2020).
15. Martínez-Álvarez, F. *et al.* Coronavirus optimization algorithm: A bioinspired metaheuristic based on the COVID-19 propagation model. *Big Data* **8**, 308–322 (2020).
16. Kennedy, J. & Eberhart, R. In *Proceedings of ICNN'95-International Conference on Neural Networks* 1942–1948 (IEEE).
17. Dorigo, M., Maniezzo, V. & Colorni, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man. Cybernet. Part B* **26**, 29–41. https://doi.org/10.1109/3477.484436 (1996).
18. Dorigo, M., Maniezzo, V. & Colorni, A. *The Ant System: An Autocatalytic Optimizing Process* (1991).

19. Xie, L. *et al.* Tuna swarm optimization: A novel swarm-based metaheuristic algorithm for global optimization. *Comput. Intell. Neurosci.* **2021**, 9210050. https://doi.org/10.1155/2021/9210050 (2021).
20. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization* (Citeseer, 2005).
21. Karaboga, D. & Basturk, B. In *Foundations of Fuzzy Logic and Soft Computing* (eds Patricia, M. *et al.*) 789–798 (Springer, 2007).
22. El Sayed, M. *et al.* Al-Biruni Earth Radius (BER) metaheuristic search optimization algorithm. *Comput. Syst. Sci. Eng.* **45**, 1917–1934 (2023).
23. Dutta, T., Bhattacharyya, S., Dey, S. & Platos, J. Border Collie optimization. *IEEE Access* https://doi.org/10.1109/ACCESS.2020.2999540 (2020).
24. Nasuto, S. & Bishop, J. Vol. 129 115–123 (2008).
25. Krishnanand, K. N. & Ghose, D. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005* 84–91 (IEEE).
26. Abdollahzadeh, B., Gharehchopogh, F. S., Khodadadi, N. & Mirjalili, S. Mountain gazelle optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems. *Adv. Eng. Softw.* **174**, 103282 (2022).
27. Yang, X. S. & Suash, D. In *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)* 210–214.
28. Yang, X.-S. 240–249 (Springer Berlin Heidelberg).
29. Hayyolalam, V. & Pourhaji Kazem, A. A. Black widow optimization algorithm: A novel meta-heuristic approach for solving engineering optimization problems. *Eng. Appl. Artif. Intell.* **87**, 103249. https://doi.org/10.1016/j.engappai.2019.103249 (2020).
30. Kirkpatrick, S., Gelatt, C. D. Jr. & Vecchi, M. P. Optimization by simulated annealing. *Science* **220**, 671–680. https://doi.org/10.1126/science.220.4598.671 (1983).
31. Erol, O. K. & Eksin, I. A new optimization method: Big Bang-Big Crunch. *Adv. Eng. Softw.* **37**, 106–111. https://doi.org/10.1016/j.advengsoft.2005.04.005 (2006).
32. Azizi, M. Atomic orbital search: A novel metaheuristic algorithm. *Appl. Math. Model.* **93**, 657–683. https://doi.org/10.1016/j.apm.2020.12.021 (2021).
33. Azizi, M., Talatahari, S., Khodadadi, N. & Sareh, P. Multiobjective atomic orbital search (MOAOS) for global and engineering design optimization. *IEEE Access* **10**, 67727–67746 (2022).
34. Azizi, M., Mohamed, A. W. & Shishehgarkhaneh, M. B. In *Handbook of Nature-Inspired Optimization Algorithms: The State of the Art: Volume II: Solving Constrained Single Objective Real-Parameter Optimization Problems* 189–214 (Springer, 2022).
35. Talatahari, S., Azizi, M. & Gandomi, A. H. Material generation algorithm: A novel metaheuristic algorithm for optimization of engineering problems. *Processes* **9**, 859. https://doi.org/10.3390/pr9050859 (2021).
36. Nouhi, B., Khodadadi, N., Azizi, M., Talatahari, S. & Gandomi, A. H. Multi-objective material generation algorithm (MOMGA) for optimization purposes. *IEEE Access* **10**, 107095–107115 (2022).
37. Azizi, M., BaghalzadehShishehgarkhaneh, M. & Basiri, M. Optimum design of truss structures by Material Generation Algorithm with discrete variables. *Decis. Anal. J.* **3**, 100043. https://doi.org/10.1016/j.dajour.2022.100043 (2022).
38. Hosseini, E., Ghafoor, K. Z., Emrouznejad, A., Sadiq, A. S. & Rawat, D. B. Novel metaheuristic based on multiverse theory for optimization problems in emerging systems. *Appl. Intell.* **51**, 3275–3292. https://doi.org/10.1007/s10489-020-01920-z (2021).
39. Talatahari, S. & Azizi, M. Chaos Game Optimization: A novel metaheuristic algorithm. *Artif. Intell. Rev.* **54**, 917–1004 (2021).
40. Azizi, M., Aickelin, U., Khorshidi, H. A. & BaghalzadehShishehgarkhaneh, M. Shape and size optimization of truss structures by chaos game optimization considering frequency constraints. *J. Adv. Res.* https://doi.org/10.1016/j.jare.2022.01.002 (2022).
41. Hashim, F. A., Hussain, K., Houssein, E. H., Mabrouk, M. S. & Al-Atabany, W. Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems. *Appl. Intell.* **51**, 1531–1551. https://doi.org/10.1007/s10489-020-01893-z (2021).
42. Pereira, J. L. J. *et al.* Lichtenberg algorithm: A novel hybrid physics-based meta-heuristic for global optimization. *Expert Syst. Appl.* **170**, 114522. https://doi.org/10.1016/j.eswa.2020.114522 (2021).
43. Azizi, M., Aickelin, U., Khorshidi, H. & BaghalzadehShishehgarkhaneh, M. Energy valley optimizer: A novel metaheuristic algorEnergy Valley Optimizer (EVO) ithm for global and engineering optimization. *Sci. Rep.* **13**, 226. https://doi.org/10.1038/s41598-022-27344-y (2023).
44. Talatahari, S., Azizi, M., Tolouei, M., Talatahari, B. & Sareh, P. Crystal structure algorithm (CryStAl): A metaheuristic optimization method. *IEEE Access* **9**, 71244–71261. https://doi.org/10.1109/ACCESS.2021.3079161 (2021).
45. Khodadadi, N., Azizi, M., Talatahari, S. & Sareh, P. Multi-objective crystal structure algorithm (MOCryStAl): Introduction and performance evaluation. *IEEE Access* **9**, 117795–117812 (2021).
46. Azizi, M., BaghalzadehShishehgarkhaneh, M. & Basiri, M. Design optimization of truss structures by crystal structure algorithm. *AUT J. Civ. Eng.* **6**, 4 (2022).
47. Kaveh, A. & Dadras, A. A novel meta-heuristic optimization algorithm: Thermal exchange optimization. *Adv. Eng. Softw.* **110**, 69–84. https://doi.org/10.1016/j.advengsoft.2017.03.014 (2017).
48. Faramarzi, A., Heidarinejad, M., Stephens, B. & Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl.-Based Syst.* **191**, 105190. https://doi.org/10.1016/j.knosys.2019.105190 (2020).
49. Dolatabadi, S. Weighted vertices optimizer (WVO): A novel metaheuristic optimization algorithm. *Numer. Algebra Control Optim.* **8**, 461 (2018).
50. Houssein, E. H., Saad, M. R., Hashim, F. A., Shaban, H. & Hassaballah, M. Lévy flight distribution: A new metaheuristic algorithm for solving engineering optimization problems. *Eng. Appl. Artif. Intell.* **94**, 103731. https://doi.org/10.1016/j.engappai.2020.103731 (2020).
51. Rao, R. V., Savsani, V. J. & Vakharia, D. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput. Aided Des.* **43**, 303–315 (2011).
52. Reynolds, R. G. In *Proceedings of the 3rd annual conference on evolutionary programming, World Scientific Publishing* 131–139 (World Scientific).
53. Geem, Z. W., Kim, J. H. & Loganathan, G. V. A new heuristic optimization algorithm: harmony search. *Simulation* **76**, 60–68 (2001).
54. Yue, C. T. *et al. Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization* (Nanyang Technological University, 2020).
55. Kumar, A. *et al.* A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm Evol. Comput.* **56**, 100693. https://doi.org/10.1016/j.swevo.2020.100693 (2020).
56. Canayaz, M. MH-COVIDNet: Diagnosis of COVID-19 using deep neural networks and meta-heuristic-based feature selection on X-ray images. *Biomed. Signal Process. Control* **64**, 102257. https://doi.org/10.1016/j.bspc.2020.102257 (2021).
57. Basu, A., Sheikh, K. H., Cuevas, E. & Sarkar, R. COVID-19 detection from CT scans using a two-stage framework. *Expert Syst. Appl.* **193**, 116377. https://doi.org/10.1016/j.eswa.2021.116377 (2022).
58. Bandyopadhyay, R., Basu, A., Cuevas, E. & Sarkar, R. Harris Hawks optimisation with Simulated Annealing as a deep feature selection method for screening of COVID-19 CT-scans. *Appl. Soft Comput.* **111**, 107698. https://doi.org/10.1016/j.asoc.2021.107698 (2021).
59. Hosseini, E., Ghafoor, K. Z., Sadiq, A. S., Guizani, M. & Emrouznejad, A. COVID-19 optimizer algorithm, modeling and controlling of coronavirus distribution process. *IEEE J. Biomed. Health Inform.* **24**, 2765–2775. https://doi.org/10.1109/JBHI.2020.3012487 (2020).

60. Tsai, C.-W., Chiang, M.-C., Ksentini, A. & Chen, M. Metaheuristic algorithms for healthcare: open issues and challenges. *Comput. Electr. Eng.* **53**, 421–434. https://doi.org/10.1016/j.compeleceng.2016.03.005 (2016).
61. Nilanjan, D. & Amira, S. A. In *Advancements in Applied Metaheuristic Computing* (ed, Dey, N.) 185–203 (IGI Global, 2018).
62. Bejinariu, S. I., Costin, H., Rotaru, F., Luca, R. & Niţă, C. D. In *2015 International Symposium on Signals, Circuits and Systems (ISSCS)* 1–4.
63. Nakib, A. *Metaheuristics for Medicine and Biology* (eds Amir, N. & El-Ghazali, T.) 1–22 (Springer, 2017).
64. Rai, D. & Thakkar, H. K. *Cognitive Big Data Intelligence with a Metaheuristic Approach* (eds, Sushruta, M. *et al.*) 239–258 (Academic Press, 2022).
65. Braik, M., Sheta, A. & Aljahdali, S. 603–614 (Springer International Publishing).
66. Kumar, S., Vig, G., Varshney, S. & Bansal, P. Brain tumor detection based on multilevel 2D histogram image segmentation using DEWO optimization algorithm. *Int. J. E-Health Med. Commun.* **11**, 71–85 (2020).
67. Altay, E. V. & Alatas, B. Association analysis of Parkinson disease with vocal change characteristics using multi-objective metaheuristic optimization. *Med. Hypotheses* **141**, 109722. https://doi.org/10.1016/j.mehy.2020.109722 (2020).
68. Nakib, A. *Metaheuristics for Medicine and Biology* (eds Amir, N. & El-Ghazali, T.) 121–135 (Springer, 2017).
69. Chen, X., Du, W. & Qian, F. Solving chemical dynamic optimization problems with ranking-based differential evolution algorithms. *Chin. J. Chem. Eng.* **24**, 1600–1608. https://doi.org/10.1016/j.cjche.2016.04.044 (2016).
70. Cheema, J. J. S., Sankpal, N. V., Tambe, S. S. & Kulkarni, B. D. Genetic programming assisted stochastic optimization strategies for optimization of glucose to gluconic acid fermentation. *Biotechnol. Prog.* **18**, 1356–1365. https://doi.org/10.1021/bp015509s (2002).
71. Mohd Zain, M. Z. B., Kanesan, J., Kendall, G. & Chuah, J. H. Optimization of fed-batch fermentation processes using the Backtracking Search Algorithm. *Expert Syst. Appl.* **91**, 286–297. https://doi.org/10.1016/j.eswa.2017.07.034 (2018).
72. Geem, Z. W. & Kim, J.-H. Wastewater treatment optimization for fish migration using harmony search. *Math. Probl. Eng.* **2014**, 313157. https://doi.org/10.1155/2014/313157 (2014).
73. Piotrowski, R., Lewandowski, M. & Paul, A. Mixed Integer nonlinear optimization of biological processes in wastewater sequencing batch reactor. *J. Process Control* **84**, 89–100. https://doi.org/10.1016/j.jprocont.2019.10.002 (2019).
74. Çerçevik, A. E., Avşar, Ö. & Hasançebi, O. Optimum design of seismic isolation systems using metaheuristic search methods. *Soil Dyn. Earthq. Eng.* **131**, 106012. https://doi.org/10.1016/j.soildyn.2019.106012 (2020).
75. Kaveh, A. & Mahdavi, V. R. Shape optimization of arch dams under earthquake loading using meta-heuristic algorithms. *KSCE J. Civ. Eng.* **17**, 1690–1699. https://doi.org/10.1007/s12205-013-0463-1 (2013).
76. Azizi, M., Talatahari, S. & Giaralis, A. Optimization of engineering design problems using atomic orbital search algorithm. *IEEE Access* **9**, 102497–102519. https://doi.org/10.1109/ACCESS.2021.3096726 (2021).
77. Kaveh, A. & Khosravian, M. Size/layout optimization of truss structures using vibrating particles system meta-heuristic algorithm and its improved version. *Period. Polytech. Civ. Eng.* **66**, 1–17. https://doi.org/10.3311/PPci.18670 (2022).
78. Gandomi, A. H., Talatahari, S., Yang, X.-S. & Deb, S. Design optimization of truss structures using cuckoo search algorithm. *Struct. Design Tall Spec. Build.* **22**, 1330–1349. https://doi.org/10.1002/tal.1033 (2013).
79. Zhang, J., Huang, Y., Wang, Y. & Ma, G. Multi-objective optimization of concrete mixture proportions using machine learning and metaheuristic algorithms. *Constr. Build. Mater.* **253**, 119208. https://doi.org/10.1016/j.conbuildmat.2020.119208 (2020).
80. Sun, L., Koopialipoor, M., JahedArmaghani, D., Tarinejad, R. & Tahir, M. M. Applying a meta-heuristic algorithm to predict and optimize compressive strength of concrete samples. *Eng. Comput.* **37**, 1133–1145. https://doi.org/10.1007/s00366-019-00875-1 (2021).
81. Chou, J.-S., Ngo, N.-T. & Pham, A.-D. Shear strength prediction in reinforced concrete deep beams using nature-inspired metaheuristic support vector regression. *J. Comput. Civ. Eng.* **30**, 04015002. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000466 (2016).
82. Muthu, P., Dhanalakshmi, V. & Sankaranarayanasamy, K. Optimal tolerance design of assembly for minimum quality loss and manufacturing cost using metaheuristic algorithms. *Int. J. Adv. Manuf. Technol.* **44**, 1154–1164. https://doi.org/10.1007/s00170-009-1930-1 (2009).
83. Hassan, S., Kumar, K., Raj, C. D. & Sridhar, K. Design and optimisation of pressure vessel using metaheuristic approach. *Appl. Mech. Mater.* **465–466**, 401–406. https://doi.org/10.4028/www.scientific.net/AMM.465-466.401 (2014).
84. Acharya, B. B., Dhakal, S., Bhattarai, A. & Bhattarai, N. PID speed control of DC motor using meta-heuristic algorithms. *Int. J. Power Electron. Drive Syst.* **12**, 822 (2021).
85. Pham, H.-A., Huong, T. Q. & Dang, H. X. In *Modern Mechanics and Applications.* (eds Nguyen T. K. *et al.*) 229–239 (Springer Singapore).
86. Khodadadi, N., Snasel, V. & Mirjalili, S. Dynamic arithmetic optimization algorithm for truss optimization under natural frequency constraints. *IEEE Access* **10**, 16188–16208 (2022).
87. Khodadadi, N. & Mirjalili, S. Truss optimization with natural frequency constraints using generalized normal distribution optimization. *Appl. Intell.* **52**, 10384–10397 (2022).
88. Khodadadi, N., Talatahari, S. & DadrasEslamlou, A. MOTEO: A novel multi-objective thermal exchange optimization algorithm for engineering problems. *Soft Comput.* **26**, 6659–6684 (2022).
89. Khodadadi, N., SoleimanianGharehchopogh, F. & Mirjalili, S. MOAVOA: A new multi-objective artificial vultures optimization algorithm. *Neural Comput. Appl.* **34**, 20791–20829 (2022).
90. Khodadadi, N., Abualigah, L., El-Kenawy, E.-S.M., Snasel, V. & Mirjalili, S. An archive-based multi-objective arithmetic optimization algorithm for solving industrial engineering problems. *IEEE Access* **10**, 106673–106698 (2022).
91. Eid, M. M. *et al.* Meta-heuristic optimization of LSTM-based deep network for boosting the prediction of monkeypox cases. *Mathematics* **10**, 3845 (2022).
92. El-Kenawy, E.-S.M. *et al.* Metaheuristic optimization for improving weed detection in wheat images captured by drones. *Mathematics* **10**, 4421 (2022).
93. Sharma, S., Khodadadi, N., Saha, A. K., Gharehchopogh, F. S. & Mirjalili, S. Non-dominated sorting advanced butterfly optimization algorithm for multi-objective problems. *J. Bionic Eng.* **20**, 1–25 (2022).
94. Khazalah, A. *et al. Classification Applications with Deep Learning and Machine Learning Technologies* 107–127 (Springer, 2022).
95. Khodadadi, N. *et al. Advances in Swarm Intelligence: Variations and Adaptations for Optimization Problems* 407–419 (Springer, 2022).
96. Abdelhamid, A. A. *et al.* Deep learning with dipper throated optimization algorithm for energy consumption forecasting in smart households. *Energies* **15**, 9125 (2022).
97. El-Kenawy, E.-S.M. *et al.* Advanced dipper-throated meta-heuristic optimization algorithm for digital image watermarking. *Appl. Sci.* **12**, 10642 (2022).
98. Gharehchopogh, F. S., Abdollahzadeh, B., Khodadadi, N. & Mirjalili, S. *Advances in Swarm Intelligence: Variations and Adaptations for Optimization Problems* 241–254 (Springer, 2022).
99. Mirjalili, S. Z. *et al.* In *Proceedings of 7th International Conference on Harmony Search, Soft Computing and Applications: ICHSA 2022* 185–194 (Springer).
100. Khafaga, D. S. *et al.* An Al-Biruni earth radius optimization-based deep convolutional neural network for classifying monkeypox disease. *Diagnostics* **12**, 2892 (2022).

101. Zhao, W. *et al.* An effective multi-objective artificial hummingbird algorithm with dynamic elimination-based crowding distance for solving engineering design problems. *Comput. Methods Appl. Mech. Eng.* **398**, 115223 (2022).
102. Khodadadi, N., Mirjalili, S. M. & Mirjalili, S. *Handbook of Moth-Flame Optimization Algorithm* 79–96 (CRC Press, 2022).
103. Abualigah, L. *et al. Integrating Meta-Heuristics and Machine Learning for Real-World Optimization Problems* 481–497 (Springer, 2022).
104. Kaveh, A., Khodadadi, N., Azar, B. F. & Talatahari, S. Optimal design of large-scale frames with an advanced charged system search algorithm using box-shaped sections. *Eng. Comput.* **37**, 2521–2541 (2021).
105. Kaveh, A., Talatahari, S. & Khodadadi, N. The hybrid invasive weed optimization-shuffled frog-leaping algorithm applied to optimal design of frame structures. *Period. Polytech. Civ. Eng.* **63**, 882–897 (2019).
106. Kaveh, A., Talatahari, S. & Khodadadi, N. Hybrid invasive weed optimization-shuffled frog-leaping algorithm for optimal design of truss structures. *Iran. J. Sci. Technol. Trans. Civ. Eng.* **44**, 405–420 (2020).
107. Kaveh, A., Eslamlou, A. D. & Khodadadi, N. Dynamic water strider algorithm for optimal design of skeletal structures. *Period. Polytech. Civ. Eng.* **64**, 904–916 (2020).
108. Kaveh, A., Khodadadi, N. & Talatahari, S. A comparative study for the optimal design of steel structures using CSS and ACSS algorithms. *Iran Univ. Sci. Technol.* **11**, 31–54 (2021).
109. Khodadadi, N., Gharehchopogh, F. S., Abdollahzadeh, B. & Mirjalili, S. In *Proceedings of 7th International Conference on Harmony Search, Soft Computing and Applications: ICHSA 2022* 259–269 (Springer).
110. Mirjalili, S. M., Mirjalili, S. Z., Khodadadi, N., Snasel, V. & Mirjalili, S. In *Advances in Swarm Intelligence: Variations and Adaptations for Optimization Problems* 169–179 (Springer, 2022).
111. Azizi, M., Talatahari, S., Basiri, M. & Shishehgarkhaneh, M. B. Optimal design of low-and high-rise building structures by Tribe-Harmony Search algorithm. *Decis. Anal. J.* **3**, 100067 (2022).
112. Shishehgarkhaneh, M. B., Azizi, M., Basiri, M. & Moehler, R. C. BIM-based resource tradeoff in project scheduling using fire hawk optimizer (FHO). *Buildings* **12**, 1472 (2022).
113. BaghalzadehShishehgarkhaneh, M., Keivani, A., Moehler, R. C., Jelodari, N. & RoshdiLaleh, S. Internet of Things (IoT), Building Information Modeling (BIM), and Digital Twin (DT) in construction industry: A review, bibliometric, and network analysis. *Buildings* **12**, 1503 (2022).
114. BaghalzadehShishehgarkhaneh, M., Moradinia, S. F., Keivani, A. & Azizi, M. Application of classic and novel metaheuristic algorithms in a BIM-based resource Tradeoff in dam projects. *Smart Cities* **5**, 1441–1464 (2022).
115. Abd-Alsabour, N. & Ramakrishnan, S. Hybrid metaheuristics for classification problems. *Pattern Recogn.* **10**, 65253 (2016).
116. Lin, L. & Gen, M. Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation. *Soft. Comput.* **13**, 157–168 (2009).
117. Sorensen, K., Sevaux, M. & Glover, F. A history of metaheuristics. *arXiv preprint* arXiv:1704.00853 *(2017).*
118. Wolpert, D. H. & Macready, W. G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1**, 67–82 (1997).
119. Khodadadi, N., Mirjalili, S. M., Mirjalili, S. Z. & Mirjalili, S. In *Proceedings of 7th International Conference on Harmony Search, Soft Computing and Applications: ICHSA 2022* 195–205 (Springer).
120. Jamil, M. & Yang, X.-S. A literature survey of benchmark functions for global optimisation problems. *Int. J. Math. Model. Numer. Optim.* **4**, 150–194 (2013).
121. Jamil, M., Yang, X.-S. & Zepernick, H.-J. Test functions for global optimization: A comprehensive survey. *Swarm Intelligence and Bio-inspired Computation* 193–222 (2013).
122. Yang, X.-S. Test problems in optimization. *arXiv preprint* arXiv:1008.0549 *(2010).*
123. Liang, J.-J., Suganthan, P. N. & Deb, K. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.* 68–75 (IEEE).
124. Karaboga, D. & Basturk, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **39**, 459–471 (2007).
125. Eberhart, R. & Kennedy, J. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science* 39–43 (1995).
126. Mirjalili, S. & Lewis, A. The Whale Optimization algorithm. *Adv. Eng. Softw.* **95**, 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008 (2016).
127. Mezura-Montes, E., Coello, C. & Landa-Becerra, R. *Engineering Optimization Using Simple Evolutionary Algorithm* (2003).
128. Akhtar, S., Tai, K. & Ray, T. A socio-behavioural simulation model for engineering design optimization. *Eng. Optim.* **34**, 341–354. https://doi.org/10.1080/03052150212723 (2002).
129. Gandomi, A. H., Yang, X.-S. & Alavi, A. H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **29**, 17–35. https://doi.org/10.1007/s00366-011-0241-y (2013).
130. Zhang, M., Luo, W. & Wang, X. Differential evolution with dynamic stochastic selection for constrained optimization. *Inf. Sci.* **178**, 3043–3074. https://doi.org/10.1016/j.ins.2008.02.014 (2008).
131. Talatahari, S. & Azizi, M. Optimization of constrained mathematical and engineering design problems using chaos game optimization. *Comput. Ind. Eng.* **145**, 106560. https://doi.org/10.1016/j.cie.2020.106560 (2020).
132. Siddall, J. N. *Optimal Engineering Design: Principles and Applications* (CRC Press, 1982).
133. Deb, K. & Goyal, M. In *ICGA*.
134. Hernandez-Aguirre, A., Botello, S., Coello, C. & Lizárraga, G. *Use of Multiobjective Optimization Concepts to Handle Constraints in Single-Objective Optimization*, Vol. 2723 (2003).
135. Şahin, I., Dörterler, M. & Gokce, H. Optimization of hydrostatic thrust bearing using enhanced grey wolf optimizer. *Mechanika* **25**, 480–486. https://doi.org/10.5755/j01.mech.25.6.22512 (2019).
136. Rao, R. V. & Waghmare, G. G. A new optimization algorithm for solving complex constrained design optimization problems. *Eng. Optim.* **49**, 60–83. https://doi.org/10.1080/0305215X.2016.1164855 (2017).
137. Baghlani, A. & Makiabadi, M. H. Teaching-learning-based optimization algorithm for shape and size optimization of truss structures with dynamic frequency constraints. *Iran. J. Sci. Technol.* **37**, 409–421 (2013).
138. Yildiz, A. R., Abderazek, H. & Mirjalili, S. A comparative study of recent non-traditional methods for mechanical design optimization. *Arch. Comput. Methods Eng.* **27**, 1031–1048. https://doi.org/10.1007/s11831-019-09343-x (2020).

## Author contributions

Conceptualization, M.A. and M.B.S.; methodology, M.A. and M.B.S.; software, M.A.; validation, M.A. and M.B.; formal analysis, M.B.S., and M.A.; investigation, M.B.S.; resources, M.B.S.; data curation, M.A.; writing—original draft preparation, M.B.S., and M.A.; writing—review and editing, M.A. and R.C.M.; visualization, M.B.S., and M.B.; supervision, M.A.; project administration, R.C.M.All authors reviewed the manuscript.

## Competing interests

The authors declare no competing interests.

## Additional information

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1038/s41598-023-32465-z.

**Correspondence** and requests for materials should be addressed to M.A.