Expert Systems With Applications 174 (2021) 114685

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems

Malik Shehadeh Braik [1]

Department of Computer Science, Al-Balqa Applied University, Al-salt, Jordan

ARTICLE INFO

Keywords:
Chameleon Swarm Algorithm
Optimization techniques
Meta-heuristics
Nature-inspired algorithms
Evolutionary algorithms
Swarm intelligence algorithms

ABSTRACT

This paper presents a novel meta-heuristic algorithm named Chameleon Swarm Algorithm (CSA) for solving global numerical optimization problems. The base inspiration for CSA is the dynamic behavior of chameleons when navigating and hunting for food sources on trees, deserts and near swamps. This algorithm mathematically models and implements the behavioral steps of chameleons in their search for food, including their behavior in rotating their eyes to a nearly 360°scope of vision to locate prey and grab prey using their sticky tongues that launch at high speed. These foraging mechanisms practiced by chameleons eventually lead to feasible solutions when applied to address optimization problems. The stability of the proposed algorithm was assessed on sixty-seven benchmark test functions and the performance was examined using several evaluation measures. These test functions involve unimodal, multimodal, hybrid and composition functions with different levels of complexity. An extensive comparative study was conducted to demonstrate the efficacy of CSA over other meta-heuristic algorithms in terms of optimization accuracy. The applicability of the proposed algorithm in reliably addressing real-world problems was demonstrated in solving five constrained and computationally expensive engineering design problems. The overall results of CSA show that it offered a favorable global or near global solution and better performance compared to other meta-heuristics.

1. Introduction

Over the past several years, a broad range of mathematical optimization methods based on linear and non-linear programming methods have been applied to solve a wide range of optimization problems in many areas of science (Chumburidze, Basheleishvili, & Khetsuriani, 2019) and engineering (Rodríguez, Gupta, Zabala, & Cabrera-Guerrero, 2018). Linear and non-linear programming methods (Luenberger & Ye, 1984), as some of the most wide used deterministic optimization methods, are distinguished by the use of the gradient information of the problem to search the space and locate a solution. Most of the real-world optimization problems involve simple objective functions because they either represent weight or cost in an optimization problem. The relationships between design variables and objective functions of many real-world problems are mostly linear. However, there are also many problems that are non-linear, non-convex, complex in nature, and entail complex objective functions with designs that are typically subject to many difficult constraints and may concern with a large number of decision variables (Dhiman & Kumar, 2017). Often, the objective functions of non-linear engineering problems comprise of many local optima and might contain sharp or multiple peaks (Sandgren, 1990), while the designer is constantly concerned with locating the global optimum solution. Feasible solutions to engineering design problems are a group of designs distinguished by all practical values of the decision variables. In view of this, if there is more than one local optima in the problem, the obtained outcomes from numerical methods may depend on choosing a starting point at which the gained solution may not necessarily be the global optimal one. Traditional optimization methods typically lead to an effective optimization process when applied to solve an engineering problem of simple design, problems related to linear search spaces, or problems of moderate complexity (Mlinarić, Perić, & Matejaš, 2019; Sheta, Braik, & Al-Hiary, 2019). However, these mathematical methods are subject to local optima entrapment. These methods can only identify local optimum solutions for some complex problems related to non-linear search spaces such as real-world non-convex problems, but there is no surety that they will find the global optimum solution (Qi, Jin, Wang, Xiao, & Zhang, 2019).

To put up with this problem, one might start from an initial design,

E-mail address: mbraik@bau.edu.jo.
[1] ORCID ID: https://orcid.org/0000-0003-4180-3734.

https://doi.org/10.1016/j.eswa.2021.114685
Received 18 April 2020; Received in revised form 3 February 2021; Accepted 4 February 2021
Available online 12 February 2021

tweak this design, or hybridize two algorithms to solve the design problem (Faramarzi & Afshar, 2012). The stochastic methods are an alternative to the traditional methods, which are analogous to meta-heuristic algorithms. These methods create and typically use random variables, and are employed to globally search the space to locate the near global or the global optimum solution. Flexibility, generality and gradient-free nature are some of the advantages of meta-heuristic algorithms (Mirjalili, Mirjalili, & Lewis, 2014). Additionally, they are independent of the nature of the problem to be solved. This implies that they do not demand derivative information about the problem because they employ a stochastic method. This is in contrast to mathematical programming methods which ordinarily require detailed knowledge about the problem (Faramarzi & Afshar, 2012). This independence from the nature of the problem makes them appropriate tools for finding optimal solutions to optimization problems without worrying about the nonlinear kinds of the search space of the problems and their constraints. Moreover, the flexibility of meta-heuristics allows them to address any kind of optimization problems without crucial modifications in the structure of the algorithm. They handle the problem as a black box with only the problem's inputs and outputs. This feature enables them as a potential candidate for an easy-to-use optimization method. In addition, unlike the deterministic nature of mathematical programming methods, meta-heuristics, as a stochastic approach, mostly benefit from the use of random number generators. Thus, the probability of falling into a local optima is lessened compared to the conventional deterministic methods. This property also makes meta-heuristics independent of the initial guess of solutions. Due to their ability of globally exploring the search domain in a fair amount of time and other important features, meta-heuristic methods have become more prevalent and have gained a lot of attention in recent years (Jain, Singh, & Rani, 2019; Sheta et al., 2019; Braik, Al-Zoubi, & Al-Hiary, 2020).

1.1. Meta-heuristic methods

Meta-heuristic algorithms have two substantial features, which are: (1) exploration, and (2) exploitation of a search space (Mirjalili et al., 2014). Exploration is the ability of optimization algorithms to globally explore various areas of the search space. This is related to both the escape from the local optima and the move away from local optima stagnancy. Exploitation is the capacity to locally search for potential solutions in all promising areas in order to improve the quality of solutions. A reasonable performance is realized, by achieving optimal or near optimal solutions, through an adequate tuning balance between these two essential features. Based on the source of inspiration, meta-heuristics generally work by integrating real simulations and nature-inspired rules with randomness to imitate some of the characteristics of biological behavior, natural or physical phenomena found in nature (Jain et al., 2019). Accordingly, meta-heuristics can be bunched up into four groups: (1) the biological evolutionary process, (2) the collective behavior of creatures found in nature, (3) the prevailing physical rules in the universe, and (4) the phenomena associated with humans' behavior and their perceptions.

The first category of Nature-Inspired Algorithms (NIAs) is Evolutionary Algorithms (EAs) that mimic the evolutionary behavior concepts of creatures in nature. Genetic Algorithm (GA) (Goldberg & Holland, 1988) is the most valuable and acclaimed population-based optimization method in EAs that has been developed based on the concept of survival of the fittest. Some of the EAs that have caught the attention of researchers around the world include Differential Evolution (DE) algorithm (Das & Suganthan, 2011) and Evolutionary Strategy (ES) (Rechenberg, 1973).

Swarm Intelligence (SI) is the second category of NIAs that have garnered research interest worldwide (Jain et al., 2019). Optimization algorithms in this class imitate the intelligent concerted and social behavior of groups of swarms or communities such as birds' flocks, insect colonies such as colonies of bees and ants, animal herds and many more flocks of several species of creatures. Particle Swarm Optimization (PSO) (Eberhart & Kennedy, 1995) is the first intelligent model of SI-based methods evolved based on the collective behavior of bird flocking and fish schooling. Other methods in this category include: Grey Wolf Optimizer (GWO) (Mirjalili et al., 2014), Moth-Flame Optimization (MFO) algorithm (Mirjalili, 2015), Salp Swarm Algorithm (SSA) (Mirjalili et al., 2017; Braik, Sheta, Turabieh, & Alhiary, 2020) and capuchin search algorithm (Braik, Sheta, & Al-Hiary, 2020).

Physics-Based (PB) algorithms represent the third category of NIAs. These algorithms are stimulated from the fundamental physical laws existing in the universe, and usually characterize the interconnection of search agents as per the rules predominating in physical processes. The most broadly used algorithm in BP is simulated annealing (Kirkpatrick, Gelatt, & Vecchi, 1983), which is inspired from the physical annealing phenomenon of metallurgy. There are other PB algorithms in the literature such as Gravitational Search Algorithm (GSA) (Rashedi, Nezamabadi-Pour, & Saryazdi, 2009), Multi-Verse Optimizer (MVO) (Mirjalili, Mirjalili, & Hatamlou, 2016) and Equilibrium optimizer (EO) (Faramarzi, Heidarinejad, Stephens, & Mirjalili, 2020).

The last class of NIAs is Human-Based (HB) algorithms that are related to behaviors of humans in nature (Meraihi, Gabis, Ramdane-Cherif, & Acheli, 2020). Harmony Search (HS) (Geem, Kim, & Loganathan, 2001) algorithm can be considered the most popular human-based algorithm.

1.2. Motivations of the work

A popular approach for evaluating new meta-heuristics is to manifest their competitiveness opposed to traditional methods and other meta-heuristics in addressing optimization problems. It is worth noting that it is not possible to present an algorithm that can achieve the global solutions for all kinds of problems. This view can be inferred from the "'no-free-lunch"' (NFL) theory (Koppen, Wolpert, & Macready, 2001). This theory states that there is no general optimization method that can arrive at the optimum solution for all types of problems (Wolpert & Macready, 1997). Although the literature has shown that meta-heuristics are extremely proper for solving challenging problems because they are capable to avert local optima, explore the search domain and exploit the global optimum more reliably compared to mathematical programming methods. Besides, the NFL theory holds this area of research open and urges researchers to propose new optimization algorithms to strengthen optimization and increase accuracy to solve real-life problems that constantly appear in the world due to technological advances (Uymaz, Tezel, & Yel, 2015). Hence, there are still issues that have not yet been addressed or can be better addressed by new algorithms. Also, there is no work in the literature that mimics the behavior of chameleons in nature, while chameleons are widely spread, professionally navigating and foraging in deserts and forests. The above reasons are the major drivers behind this research, in which a modest yet efficient novel optimization algorithm is proposed to optimize real-world problems.

1.3. Proposed algorithm outlines

This work develops a new meta-heuristic algorithm with the aspiration of solving real-world problems that are difficult to solve using current meta-heuristics and with the hope of finding solutions that are better than the current ones. This algorithm is called Chameleon Swarm Algorithm (CSA) which belongs to the second class of NIAs. This algorithm is inspired from the general hunting behavior of chameleons and their effective methods of hunting for food to successfully survive in deserts and forests. Since exploration and exploitation are two key features to the success of any meta-heuristic algorithm, they need to be designed efficiently to strike a good balance between them. In the proposed algorithm, chameleons roam all over the search space to search

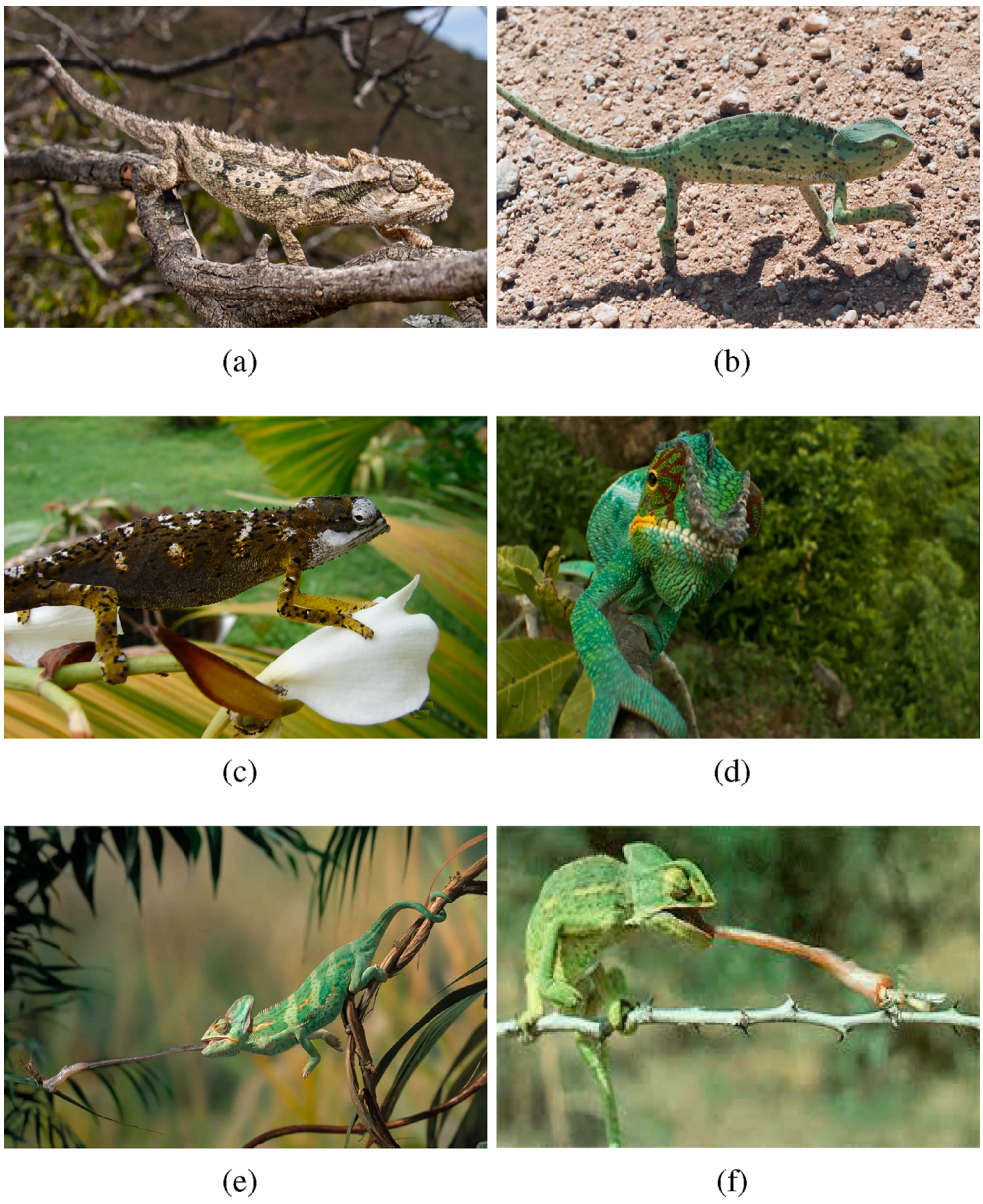

Fig. 1. Hunting behavior of chameleons: (a) searching for prey on a tree, (b) searching for prey in the desert, (c) hunting by rotating its eye backward, (d) visual chasing by rotating its eye to its right's side, (e) attacking prey from a distance with its tongue, (f) catching prey with its tongue.

for prey. In such behavior, chameleons exploit each potential area in the search domain, and use their globular eyes to scan a wide radius of search. When finding prey, they use their terribly long and sticky tongues to rapidly pick up prey with very high performance. To further realize a balance between exploration and exploitation for more reliable performance, an adaptive parameter, over the course of iterations of CSA, was suggested to assist chameleons to better search in the search space.

Based on the author's knowledge, the proposed algorithm is the first meta-heuristic algorithm that imitates the foraging strategy of chameleons in nature with their living aspects described above. It is noteworthy here that there is an algorithm mentioned in the literature called Chameleon algorithm (Karypis, Han, & Kumar, 1999). The algorithm presented here is totally different from that one in terms of inspiration, mathematical models, and formulation in addition to the problems and applications used. Chameleon algorithm is a bottom-up clustering method that has been used in data mining to cluster datasets using dynamic models, while CSA could be used to solve both constrained and global numerical optimization problems.

1.4. Contributions of the work

The main contributions of the proposed CSA can be briefed as follows:

1. A novel nature-inspired optimization algorithm that encompasses all aspects of chameleon's behavior in searching for prey is presented.
2. The performance of CSA is evaluated on 67 publicly available benchmark test functions, referred to as classical test suite, CEC-2015 test suite and CEC-2017 test suite, and compared to other optimization algorithms.
3. A broad comparative study with other competitive optimization algorithms was conducted on five classical engineering design problems.

The remainder of this paper is formulated as follows: Section 2 presents the main inspiration of the proposed algorithm. The following Section 3 presents the concepts of the proposed algorithm along with its mathematical models. The experimental results, convergence analysis

and statistical analysis tests are presented in Section 4. The application of CSA on several engineering problems is described in Section 5. Finally, concluding comments and some further research trends are given in Section 6.

2. Inspiration

Chameleons are a distinctive and highly specialized clade with a wide range of species, and are known for their ability to change color to blend in with their surroundings (Glaw, 2015). They live in a diversity of habitats ranging from lowlands and rainforests to deserts, semi-desert conditions and even mountains. Chameleons are acclimatized for climbing and visual hunting and have excellent eyesight that can see up to 32 feet in front of them. This makes the prey easier to spot. Chameleons generally feed on insects such as snails, mantis, crickets and grasshoppers. There are a few large chameleon species that supplement their diets with lizards and small birds. On the other side, there are many animals that eat chameleons such as birds, snakes and sometimes monkeys. Chameleons' ability to adjust their colors to suit their surroundings is their way to protect themselves when a predator is nearby (Young, 2008). The main stages of a chameleon in hunting prey are as follows:

2.1. Tracking the prey

Like any creature in nature, chameleons roam the desert and trees while searching for prey, and therefore, their position changes accordingly. In Fig. 1(a) and (b), there are two species of chameleons illustrating their behavioral activity during searching for prey.

2.2. Pursuing prey with their eyes

Chameleons' eyes are independently mobile giving them the proficiency to explore the search space to locate prey. Their eyes can look in two different directions simultaneously, affording them a panoramic view of their surroundings. Each eye can move independently of each other to pivot and focus on the location of prey simultaneously. This allows chameleons to observe two different objects at the same time, but in an attempt to find prey. They focus forward in coordination, allowing a stereoscopic view of prey. This allows chameleons a full 360° scope of vision around their bodies! With 180degr on each side to see all around them. When a chameleon sees prey, both eyes can centralize on the same direction for a clearer vision. Then, the chameleon rotates and moves to the location of the prey. In Fig. 1(c) and (d), there are two species of chameleons that use their eyes to scout their hunting area for a possible prey.

2.3. Attacking the prey

Chameleons mainly feed by dropping their clingy tongues to capture prey. They have a very functional mechanism of sticking their tongues to prey once their tongues come into contact with the prey, including surface phenomena, such as wet adhesion and entanglement, where the chameleon rapidly forms a small suction cup (Herrel, Meyers, Aerts, & Nishikawa, 2000). Many species of chameleons can drop their tongues at an acceleration rate of 2,590 meters per second squared (Herrel et al., 2000). This behavior is shown in Fig. 1(e) and (f).

In this work, the hunting behavior of chameleons in nature and their effective methods of tracking, pursuing and capturing prey have led to the mathematical models developed to design CSA and perform optimization. Below is a detailed description of the proposed algorithm.

3. Chameleon Swarm Algorithm

This section shows in detail the mathematical models of the proposed algorithm.

3.1. Mathematical models of CSA

This subsection describes the mathematical models that characterize the behavior of chameleons while foraging. This includes tracking prey, searching for prey and hunting prey. Then the proposed algorithm is outlined.

3.1.1. Initialization and function evaluation

As CSA is a population-based algorithm, it employs the initial population to initiate the optimization process. A population of n chameleons, in a d-dimensional search space, with each chameleon denotes a candidate solution to a problem, can be represented in a two-dimensional matrix y of size $n \times d$. The position of chameleon i at iteration t in the search domain can be represented by a vector, as shown below:

$$y_t^i = \left[y_{t,1}^i, y_{t,2}^i, \ldots, y_{t,d}^i\right] \tag{1}$$

where $i = 1, 2, \ldots, n$, t is the current iteration number, d stands for the dimension of the problem, $y_{t,d}^i$ represents the position of the ith chameleon at dth dimension.

The initial population is created based on the number of chameleons and dimension of the problem with uniform random initialization in the search space as shown below:

$$y^i = l_j + r \times (u_j - l_j) \tag{2}$$

where y^i is the initial vector of the ith chameleon, l_j and u_j refer to the lower and upper bounds of the search area in the jth dimension, respectively, and r is a uniformly created random number in the range of $[0, 1]$.

The quality of the solution is assessed for each new position of a chameleon based on a fitness function. Then, the current position is updated if the solution quality of the new position is better than the solution quality of the current one. However, the chameleons in CSA model remains at its current position if its solution quality is more qualified than the new one.

3.1.2. Search for prey

The behavior of chameleons' movement during foraging can be mathematically modeled using the position updating strategy as proposed below:

$$y_{t+1}^{i,j} = \begin{cases} y_t^{i,j} + p_1(P_t^{i,j} - G_t^j)r_2 + p_2(G_t^j - y_t^{i,j})r_1 & r_i \geqslant Pp \\ y_t^{i,j} + \mu\left((u^j - l^j)r_3 + l_b^j\right)\mathrm{sgn}(rand - 0.5) & r_i < Pp \end{cases} \tag{3}$$

where the attributes of Eq. (3) are defined as follows:

- $y_{t+1}^{i,j}$ is the new position of the ith chameleon in the jth dimension in the $(t+1)$th iteration step,
- $y_t^{i,j}$ is the current position of the ith chameleon in the jth dimension in the tth iteration step,
- $P_t^{i,j}$ represents the best position that has scored so far by chameleon i in the jth dimension at iteration loop t,
- G_t^j represents the global best position in the jth dimension obtained so far by any chameleon in the tth iteration,
- p_1 and p_2 are two positive numbers that control exploration ability,
- r_1, r_2 and r_3 are random numbers generated uniformly in the range of $[0, 1]$,
- r_i is a random number generated uniformly at index i in the interval $[0, 1]$,
- Pp represents the probability of the chameleon perceiving prey, which is equal to 0.1, where this value was found after extensive experimental testing,

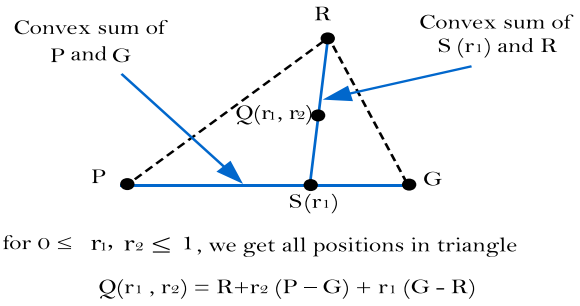

Fig. 2. Flowchart for simulating position updating strategy in CSA when $Pp \geqslant 0.1$.

- $\mathrm{sgn}(rand - 0.5)$ has an effect on the direction of exploration and exploitation, and it can be either 1 or −1,
- μ is a parameter defined as a function of iterations that decreases with the number of iterations as given in Eq. (6).

The first state of Eq. (3), (i.e., when $Pp \geqslant 0.1$), was proposed by taking advantage of formation of a plane in an affine space as shown in Fig. 2.

From simple geometry, it is known that three positions not on the same line identify a unique plane. Suppose that P, G, and R are three such positions in an affine space. The line segment that connects the positions P and G is the set of positions of the form defined in Eq. (4).

$$S(r_1) = r_1 P + (1 - r_1) G \tag{4}$$

where r_1 is a uniformly distributed random number created within $[0,1]$.

Suppose we pick a random position on this line and form a path from this arbitrary position to R, as illustrated in Fig. 2. Using a random parameter r_2, we can characterize positions along this line path as given in Eq. (5).

$$Q(r_1, r_2) = r_2 S + (1 - r_2) R 0 \leqslant r_2 \leqslant 1 \tag{5}$$

Eqs. (4) and (5) indicate that there is a high potentiality to explore every position on this streak path. We simulated this feature from affine space, by contrasting the main notations of Eq. (3) to the notations in Fig. 2, where $y_{t+1}^{i,j} = Q(r_1, r_2)$, $y_t^{i,j} = R, P_t^{i,j} = P$ and $G_t^j = G$. This gives chameleons the ability to explore every possible position in the search space. Appropriately, p_1 and p_2 in Eq. (3) were suggested as scaling factors to manage exploration ability in CSA. With a range of small and large values for p_1 and p_2, CSA can alternate between local search and global search. The second state of Eq. (3), (i.e., when $Pp < 0.1$), was proposed to allow chameleons to explore random locations in the search

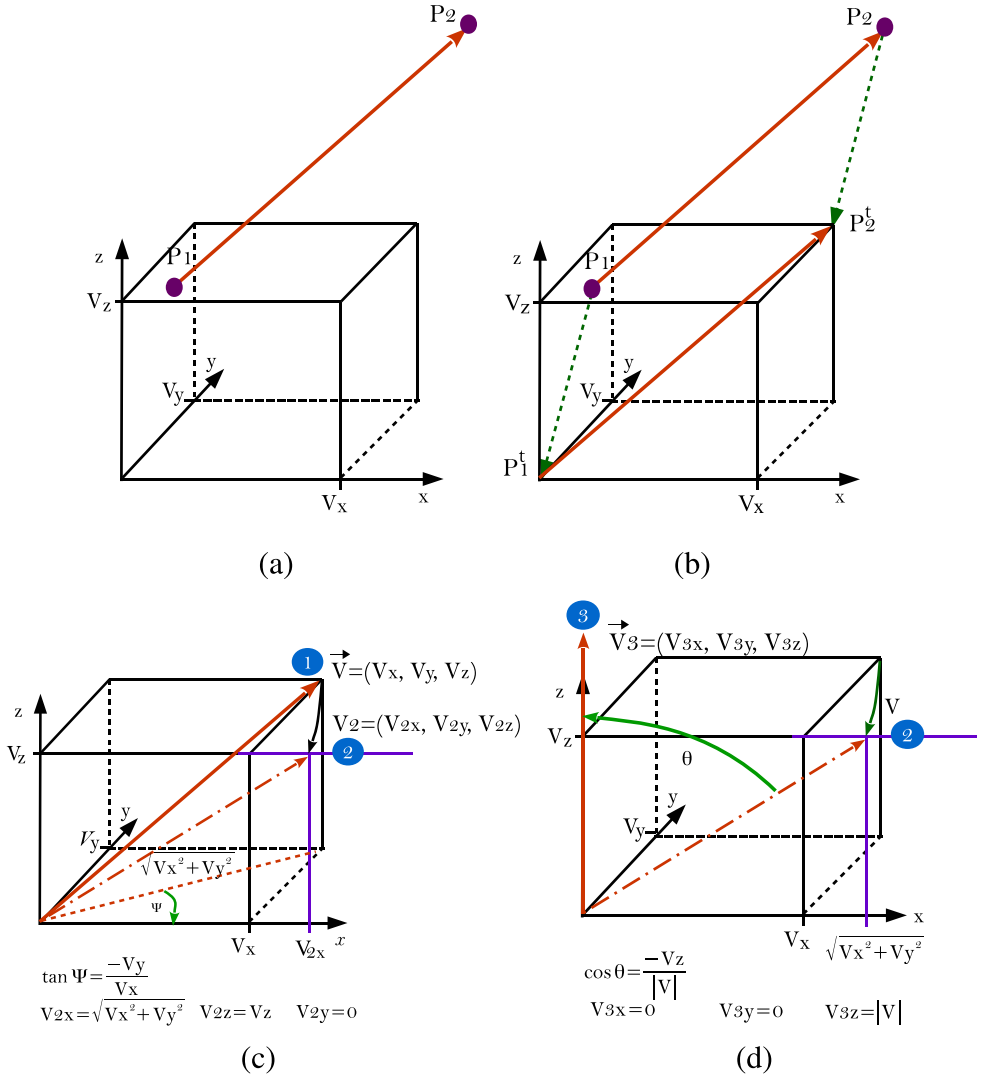

Fig. 3. Rotating and translating a vector in 3-dimensional space: (a) a vector in the space, (b) translating the vector to the origin, (c) rotating the vector in the z space, (d) rotating and aligning the vector by an angle in the z-axis.

space to enhance the search capability of CSA.

$$\mu = \gamma e^{(-\alpha t/T)^{\beta}} \tag{6}$$

where t and T represent the current and maximum number of iterations, respectively, γ, α and β are three constant values used to control exploration and exploitation capabilities.

Eq. (6) was proposed to ensure convergence by slowing down the search speed as well as enhancing exploration and exploitation capacities of the proposed algorithm. The values of the parameters γ, α and β for all of the benchmark test functions thereafter solved in this work are equal to 1.0, 3.5 and 3.0, respectively. These parameters were chosen by empirical testing for a subset of problems. However, they can be adjusted for other optimization problems as necessary. Also, the values of p_1 and p_2 are equal to 0.25 and 1.50, respectively. These parameters are also identified by experimental testing, where they can be further adjusted for other problems as demanded.

In Eq. (3), G_t^j represents the current candidate solution that is close to the optimum solution because the global optimal solution is not known a priori in the search space. $Pp \geqslant 0.1$ spells out that chameleons can change their position according to the observation of prey in the search space. On the other hand, when $Pp < 0.1$, chameleons randomly explore the search space in many different directions and areas while hunting for prey. This gives them a great possibility to perceive any nearby prey that represents their optimum targets.

3.1.3. Chameleon eyes' rotation

As mentioned earlier, chameleons have the ability to recognize the position of prey using the rotation feature of their eyes, which provide them with the ability to spot prey over 360degr. In order to mathematically simulate the hunting behavior of chameleons using their eyes, we realize that chameleons will update their position according to the position of their prey. In such a context, chameleons will rotate and move to the prey. The following steps are proposed in this regard.

- Translate the original position of the chameleon to the center of gravity (i.e., the origin),
- Find the rotation matrix that identifies the position of prey,
- Update the position of the chameleons using the rotation matrix at the center of gravity, and finally,
- Translate the chameleons back to the original position.

The above four steps, in updating the position of chameleons, can be performed using rotation and translation of vectors in space. The positions of chameleons and prey can be represented as vectors. In this connection, the following strategies used to rotate the vector $\overrightarrow{P_2P_1}$ shown in Fig. 3(a) with a preset angle in 3-dimensional search space, were used to update the position of chameleons with an angle:

1. Translate the vector $\overrightarrow{P_2P_1}$ to the origin. This can be performed by subtracting the tail of the vector from its head, $\overrightarrow{V} = P_2 - P_1$, as illustrated by vector $\overrightarrow{V} = P_2^t P_1^t$ in Fig. 3(b).
2. Rotate $\overrightarrow{V} = (V_x, V_y, V_z)$ in Fig. 3(c), with its tail at the origin and head at position (1), in the z-axis through the angle ψ. This step leads to the vector $\overrightarrow{V2} = (V2_x, V2_y, V2_z)$, which is positioned in the $x-z$ plane as shown at position (2) in Fig. 3(c).

   In Fig. 3(c), the angle of rotation ψ is computed from the components V_x and V_y of the vector $\overrightarrow{V}$ using the following formula:

$$\tan(\psi) = -V_y/V_x \tag{7}$$

   The angle ψ in Eq. (7) is negative due to the use of a positive value for V_y and the right-hand rule for the rotation of $\overrightarrow{V}$. The x component of the rotated vector $\overrightarrow{V2}$ at position (2) can be defined as follows:

$$V2x = \sqrt{V_x^2 + V_y^2} \tag{8}$$

   where the components y and z of the vector $\overrightarrow{V2}$, named V_y and V_z, are 0 and $|V|$, respectively.
3. Rotate the vector $\overrightarrow{V2}$ at position (2) in the $x-z$ plane by an angle θ to align it with the z-axis. This step leads to $\overrightarrow{V3}$ as shown at position (3) in Fig. 3(d).

   In Fig. 3(d), the angle θ can be calculated according to the mathematical formula shown in Eq. (15).

$$\cos(\theta) = -V_z/|V| \tag{9}$$

   where θ is the angle of rotation $\overrightarrow{V2}$ at position 2 to create $\overrightarrow{V3}$ at position 3, V_z is the z component of the vector $\overrightarrow{V}$ and $|V|$ is the magnitude of the vector $\overrightarrow{V}$.

   With the right-hand rule for vector rotation, the angle θ in Fig. 3(d) is negative as shown in Eq. (15). The z component of the rotated vector $\overrightarrow{V3}$ at position (3), as shown in Fig. 3(d), can be defined as follows:

$$V3_z = |V| \tag{10}$$

   where $|V|$ is the magnitude of the vector $\overrightarrow{V}$, and the x and y components of $\overrightarrow{V3}$, named $V3_x$ and $V3_y$, respectively, are both 0.

The above steps, used to rotate a vector from one position to another one, were used to model chameleons' position update when they locate prey using their eyes' rotation. The new position of a chameleon can be updated using the following mathematical formula:

$$y_{t+1}^i = yr_t^i + \overline{y}_t^i \tag{11}$$

where y_{t+1}^i represents the new position of a chameleon after rotation, $\overline{y}_t^i$ represents the center of the current position of the chameleon before rotation and yr_t^i represents the rotating centered coordinates of the chameleon in the search space that can be defined as shown in Eq. (12).

$$yr_t^i = m \times yc_t^i \tag{12}$$

where m is a rotation matrix that represents the rotation of a chameleon and yc_t^i represents the centering coordinates at iteration t. These two components are defined as shown in Eqs. (13) and (14), respectively.

$$yc_t^i = y_t^i - \overline{y}_t^i; \tag{13}$$

y_t^i is the current position of the chameleons at iteration t.

$$m = R\left(\theta, \overrightarrow{V}z_1, \overrightarrow{z_2}\right) \tag{14}$$

where $\overrightarrow{z_1}$ and $\overrightarrow{z_2}$ are two orthonormal vectors in the n-dimensional search space where the size of each vector is $d \times 1$, R refers to the rotation matrices in the respective axes as defined below and θ represents the rotation angle of a chameleon defined as shown below:

$$\theta = r\mathrm{sgn}(rand - 0.5) \times 180^{\circ} \tag{15}$$

where r is a random number created in the range $[0, 1]$ to implement a rotation angle from 0 to 180degr and $\mathrm{sgn}(rand - 0.5)$ is the direction of rotation that is either 1 or -1.

The following are the rotation matrices along x and y axes in three dimensions:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{16}$$

where ϕ represents the angle of rotation about the x-axis.

$$R_y = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \tag{17}$$

where θ represents the angle of rotation about the y-axis.

3.1.4. Hunting prey

Chameleons finish hunting by attacking prey when it is too close to a chameleon. The chameleon that gets closes to prey is supposed to be the best chameleon, and it is the optimal. This chameleon uses its tongue to attack prey. Hence, its position is updated slightly as it can drop its tongue as double as its length. This mechanism assists chameleons to exploit the search space by effectively grabbing prey.

The velocity of a chameleon's tongue when it falls toward prey can be mathematically modeled according to Eq. (18).

$$v_{t+1}^{i,j} = \omega v_t^{i,j} + c_1\left(G_t^j - y_t^{i,j}\right)r_1 + c_2\left(P_t^{i,j} - y_t^{i,j}\right)r_2 \tag{18}$$

where $v_{t+1}^{i,j}$ represents the new velocity of the ith chameleon in the jth dimension at iteration $t+1$, $v_t^{i,j}$ represents the current velocity of the ith chameleon in the jth dimension, $y_t^{i,j}$ represents the current position of the ith chameleon, $P_t^{i,j}$ is the ith chameleon's best known position and G_t^j is the best global position known so far to the chameleons, c_1 and c_2 are two positive constants that control the influence of $P_t^{i,j}$ and G_t^j on dropping the chameleon's tongue, r_1 and r_2 are two random numbers distributed in the range from 0 to 1 and ω is the inertia weight that is linearly decreased with the iterative generations as given in Eq. (19).

$$\omega = (1 - t/T)^{(\rho\sqrt{(t/T)})} \tag{19}$$

where ρ is a positive number used to manage exploitation capacity. For all of the benchmark problems solved in this work, ρ is equal to 1.

The inertia weight parameter ω has a good effect in improving the convergence behavior in CSA. The position of the chameleons' tongue when projected towards prey represents, implicitly, the position of the chameleon, which can be computed in accordance to the third equation of motion, as shown below:

$$y_{t+1}^{i,j} = y_t^{i,j} + \left((v_t^{i,j})^2 - (v_{t-1}^{i,j})^2\right)\Big/(2a) \tag{20}$$

where $v_{t-1}^{i,j}$ represents the previous speed of the ith chameleon in the jth dimension and a represents the acceleration rate of the chameleon's tongue projection, which increases gradually until it reaches its maximum value of 2,590 meters per second squared. This rate can be defined as given in Eq. (21).

$$a = 2590 \times \left(1 - e^{-log(t)}\right) \tag{21}$$

Eqs. (3), (11) and (20) were proposed to simulate the foraging behavior of chameleons. These formulas are useful to allow chameleons, over the course of iterations, to explore different random locations in the search space and identify the most promising areas where the optimal prey is found.

3.2. Exploration ability of CSA

To summarize the exploration capacity of CSA, there are many parameters in CSA that lead to exploration as given below:

- μ: controls the amount of exploration of the algorithm. It identifies to what extent the new position would be to the prey. It heightens exploitation capacity, avoids early convergence and prevents solutions from falling into local optima. The higher the α and β values, the higher the exploration power. Numbers greater than four would degenerate exploration performance. It was found based on empirical testing that values greater than four induce search agents to search on boundaries.
- $\mathrm{sgn}(rand - 0.5)$ in Eq. (3): controls the direction of exploration. Due to that *rand* falls into $[0, 1]$ with a uniform distribution, there is an equal probability of positive and negative signs.
- p_1 and p_2: these parameters influence on the position update of chameleons. The p_1 parameter controls the influence of the best position of chameleons on the best position achieved, while p_2 controls the effect of the global best position achieved so far. These parameters range from 0 to 2, where $p_1 = 0$ or $p_2 = 0$ means that one of the parts of Eq. (3) will not participate in the optimization process, which expands the recession probability in local optima. The state of $p_1 = 2$ and $p_2 = 0$ and conversely, emphasizes reasonable exploration ability, which might lead to non-accurate solutions. Based on empirical testing, $p_1 = 0.25$, or $p_1 = 0.5$, and $p_2 = 1.5$ or $p_2 = 1.0$, provides a sensible balance between exploration and exploitation phases.
- *Pp*: the value of this parameter was chosen on the basis of experimental tests, but it should not be greater than 0.15, as this would degrade the exploration performance. In the first few iterations, chameleons are all far away from prey. Updating the position of chameleons based on Eq. (3) improves the algorithm's capability to globally search the space.

3.3. Exploitation ability of CSA

The main parameters to perform exploitation feature in CSA are described as follows:

- p_1 and p_2: these parameters are important to the success of CSA. This may be due to their behavior in balancing global and local search processes. These parameters can further present an adequate balance between exploration and exploitation of CSA.
- μ: by the lapse of iteration, exploration fades out and exploitation fades in. In the last iterations, where a chameleon is in close proximity to prey, the position updating process with this parameter will help in the local search around prey, leading to exploitation. So, this parameter can address more balance between exploration and exploitation.
- γ: this parameter controls the amount of exploitation by prospecting for the best solution.
- ρ: this parameter is used to manage exploitation capacity of the algorithm.
- $\mathrm{sgn}(rand - 0.5)$ in Eq. (3): enhances the exploitation quality, as it changes the direction of search.

3.4. Implementation and analysis of CSA

The basic steps of the proposed CSA can be outlined by the iterative steps shown in Algorithm 1.

Algorithm 1. A pseudo-code describing the basic steps of the proposed algorithm.

1: $Pp \leftarrow 0.1$ (the position update probability)
2: $r_1,\ r_2,\ r_3,\ r^i$ are random numbers between 0 and 1
3: u and l are the upper and lower bounds of the search area
4: $d \leftarrow$ dimension of the problem
5: $\overline{y}_t^i$ is the center of the current position of chameleon i at iteration t
6: yr_t^i is the rotating centered coordinates of chameleon i at iteration t which can be defined using Eq. (12)
7: Randomly initialize the position of a swarm of n chameleons in the search space using Eq. (2)
8: Initialize the velocity of dropping chameleons' tongues
9: Evaluate the position of the chameleons
10: **while** $(t < T)$ **do**
11: Define the parameter μ using Eq. (6)
12: Define the inertia weight ω using Eq. (19)
13: Define the acceleration rate a using Eq. (21)
14: **for** i = 1 to n **do**
15: **for** j = 1 to d **do**
16: **if** $r^i \geqslant P_p$ **then**
17: $y_{t+1}^{i,j} = y_t^{i,j} + p_1(P_t^{i,j} - G_t^j)r_1 + p_2(G_t^j - y_t^{i,j})r_2$
18: else
19: $y_{t+1}^{i,j} = y_t^{i,j} + \mu\left(\left(u^j - l^j\right)r_3 + l_b^j\right)sgn(rand - 0.5)$
20: **end if**
21: **end for**
22: **end for**
23: **for** i = 1 to n **do**
24: $y_{t+1}^i = yr_t^i + \overline{y}_t^i$
25: **end for**
26: **for** i = 1 to n **do**
27: **for** j = 1 to d **do**
28: $v_{t+1}^{i,j} = \omega v_t^{i,j} + c_1\left(G_t^j - y_t^{i,j}\right)r_1 + c_2\left(P_t^{i,j} - y_t^{i,j}\right)r_2$
29: $y_{t+1}^{i,j} = y_t^{i,j} + \left((v_t^{i,j})^2 - (v_{t-1}^{i,j})^2\right)/(2a)$
30: **end for**
31: **end for**
32: Adjust the chameleons' positions according to u and l
33: Evaluate the new positions of the chameleons
34: Update the position of the chameleons
35: t = t + 1
36: **end while**

Algorithm 1 shows that CSA initiates the optimization process by randomly generating the position of the chameleons in the search space. At each iteration loop, the position of the chameleons is updated consecutively using Eqs. (3), (11) and (20). If any of the chameleons moves out of the search area, it will be pushed back to the boundary based on the simulated steps proposed for CSA. After that, the solutions are assessed using a fitness function that is recalculated inside each iteration loop to identify the chameleon with the best fitness. The fittest solution is referred to as the best position of the chameleon that finds prey. All the steps in Algorithm 1 except the initialization step are repeated at each iteration until the maximum number of iterations is realized.

In short, in the swarm behavior model of CSA, chameleons are constantly exploring and exploiting the surrounding area for both stationary and mobile prey and heading towards prey to capture it. It is anticipated that the mathematical models proposed for CSA advocate its potentiality to solve optimization problems.

4. Experimental results and discussion

This section describes the results of the proposed CSA in sixty-seven benchmark test functions. A detailed description of these functions is given below. The results were discussed, analyzed and compared to other meta-heuristic algorithms that have reported promising performance in the literature.

Table 1
Parameter values of CSA and other meta-heuristic algorithms.

Algorithm	Parameter	Value
All algorithms	Population size	30
	Iterations	1000
CSA	p_1, p_2, ρ	0.25, 1.50, 1.0
	c_1, c_2	1.75, 1.75
SSA	Position update probability	0.5
MFO	Logarithmic spiral	0.75
MVO	Travelling distance rate	[0.6, 1]
	Wormhole existence prob.	[0.2, 1]
SCA	Number of elites	2
GSA	Alpha coefficient	20
	Gravitational constant (G_0)	100
	Rnorm, Rpower	2, 1
GA	Selection	Roulette wheel
	Crossover	0.9
	Mutation	0.05
HS	Harmony memory and rate	30, 0.95
	Discrete set and fret width	17700, 1
GWO	Convergence parameter ($\vec{a}$)	[2.0 to 0.0]
PSO	Cognitive constant	1.8
	Social constant	2.0
	$w_{min},\ w_{max}$	0.1, 0.9
EO	$a_1,\ a_2$	2, 1
	GP	0.5

4.1. Description of benchmark test functions

The sixty-seven test functions were used to substantiate the efficacy and applicability of the proposed CSA. These functions can be grouped into five main categories: unimodal functions (Digalakis & Margaritis, 2001), multimodal functions (Yang, 2010a), fixed-dimension multimodal functions (Digalakis & Margaritis, 2001; Yang, 2010a), CEC-2015 test suite (Chen et al., 2014) and CEC-2017 test suite (Awad, Ali, & Suganthan, 2017). Details of these functions, including dimensions of the functions (*Dim*), limits of the search space (Range) and the optimal reported value (f_{min}), are presented in Appendix A in Tables A.1–A.3, respectively.

Each group of these functions was utilized to benchmark specific perspectives of the proposed algorithm. The first category that includes unimodal functions (F_1–F_7) have one optimum solution, which purposely challenges the exploitation performance and convergence property of the algorithm. Multimodal functions (F_8–F_{13}), in the second category, have more than one optimal solution. Local optimal solutions in these test functions assess the exploration behavior of the algorithm, while an algorithm demands to be capable of globally searching the space to locate the global optima and avoid being trapped in local optima. The third category involves fixed-dimension multimodal functions (F_{14}–F_{23}), which are analogous to multimodal functions, but have low and fixed dimensions. A good algorithm must evade local optimal solutions and rapidly converge to the global optimal solution. The fourth and fifth categories named CEC-2015 (C-f1 to C-f15) and CEC-2017 (C-f1 to C-f30) test suites involve hybrid and composite test functions. These functions represent more challenging optimization functions and are utilized to further challenge the performance of the proposed CSA. These challenge functions simulate the complexity of a real search space by having a big number of local optimums and functions of various shapes. These functions were created by rotating, shifting, expanding and hybridizing unimodal and multimodal test functions. More details about

Table 2
Average and standard deviation of best optimal solution for thirty independent runs in unimodal test functions (F_1–F_7).

Functions	CSA		SSA		MFO		MVO	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F_1	**4.01E−24**	**3.16E−23**	1.24E−08	2.67E−09	3.15E−04	5.99E−04	2.81E−01	1.11E−01
F_2	2.12E−17	2.62E−16	2.44E−02	1.06E−01	3.71E+01	2.16E+01	3.96E−01	1.41E−01
F_3	**3.02E−15**	**3.48E−14**	1.93E−09	1.10E−09	4.42E+03	3.71E+03	4.31E+01	8.97E+00
F_4	**1.67E−13**	**1.32E−12**	1.54E−05	3.20e-06	6.70E+01	1.06E+01	8.80E−01	2.50E−01
F_5	**4.57E+00**	**3.33E−01**	2.20E+02	4.87E+02	3.50E+03	3.98E+03	1.18E+02	1.43E+02
F_6	**1.65E−21**	**9.31E−19**	7.15E−10	1.84E−10	1.66E−04	2.01E−04	3.15E−01	9.98E−02
F_7	**1.44E−04**	**1.49E−03**	7.06E−03	4.99E−03	3.22E−01	2.93E−01	2.02E−02	7.43E−03
Functions	SCA		GSA		GA		HS	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F_1	3.55E−02	1.06E−01	1.16E−16	6.10E−17	1.95E−12	2.01E−11	7.86E−10	8.11E−09
F_2	3.23E−05	8.57E−05	1.70E−01	9.29E−01	**6.53E−18**	**5.10E−17**	5.99E−20	1.11E−01
F_3	4.91E+03	3.89E+03	4.16E+02	1.56E+02	7.70E−10	7.36E−09	9.19E−05	6.16E−04
F_4	1.87E+01	8.21E+00	1.12E+00	9.89E−01	9.17E+01	5.67E+01	8.73E−01	1.19E−01
F_5	7.37E+02	1.98E+03	3.85E+01	3.47E+01	5.57E+02	4.16E+01	8.91E+02	2.97E+02
F_6	4.88E+00	9.75E−01	1.08E−16	4.00E−17	3.15E−01	9.98E−02	8.18E−17	1.70E−18
F_7	3.88E−02	5.79E−02	7.68E−01	2.77E+00	6.79E−04	3.29E−03	5.37E−01	1.89E−01

the composition of these functions can be found in Appendix A.

4.2. Experimental setup

To demonstrate the overall efficacy of the proposed CSA, its results are compared with those of the highly respected meta-heuristic algorithms on the aforementioned benchmark test functions. The comparative algorithms in this section include some classes of optimization methods: (i) GA (Bonabeau, Dorigo, Marco, Theraulaz, & Théraulaz, 1999) and PSO (Kennedy & Eberhart, 1995) as the most well-studied evolutionary and swarm intelligence algorithms, (ii) GWO (Mirjalili et al., 2014), GSA (Rashedi et al., 2009), SSA (Mirjalili et al., 2017) and MFO (Mirjalili, 2015) as recent and reliable meta-heuristic algorithms, (iii) HS (Geem et al., 2001) as a popular human-based algorithm that simulates the process of generating harmonies in music, (iv) MVO (Mirjalili et al., 2016) and EO (Faramarzi et al., 2020) as recent and efficacious physics-based optimizers, and (v) SCA as an algorithm developed based on sine and cosine mathematical functions. The parameter settings of CSA and those comparative algorithms are given in Table 1.

The parameter settings in Table 1 for optimization algorithms excluding CSA were specified according to the settings mentioned in their original references. The initialization process for CSA is similar to that commonly adopted in other optimization algorithms. This is to provide an equitable comparison between CSA and those comparative algorithms. The parameter settings of CSA were remained the same in the evaluation of all benchmark test functions. For all test categories, CSA uses 30 particles along with 1000 iterations, with a maximum number of Function Evaluations (FEs) of 30,000. Similarly, the other comparative algorithms also used 30,000 as a maximum number of FEs. It could be mentioned that the comparison for all algorithms is done with equal floating point precision, so the differences between the outcomes are due to the performance of the algorithms. Each algorithm in Table 1 was evaluated thirty independent runs for each test function. The stop criterion for each algorithm is set to the maximum number of iterations. Best results are shown in bold throughout the paper.

4.3. Performance evaluation

This subsection presents and discusses the performance of the pro-

Table 3
Average and standard deviation of best optimal solution for thirty independent runs in multimodal functions (F_8–F_{13}).

Functions	CSA		SSA		MFO		MVO	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F_8	-2.10E+03	2.95E+02	−2.81E+03	2.86E+02	-8.04E+03	8.80E+02	-6.92E+03	9.19E+02
F_9	5.00E+00	**3.20E+00**	1.58E+01	6.73E+00	1.63E+02	3.74E+01	1.01E+02	1.89E+01
F_{10}	**1.30E−15**	**5.42E−14**	7.18E−01	8.21E−01	1.60E+01	6.18E+00	1.15E+00	7.87E−01
F_{11}	9.42E−06	**2.23E−05**	7.74E−04	1.29E−01	5.03E−02	1.74E−01	5.74E−01	1.12E−01
F_{12}	**2.83E−25**	**1.46E−25**	2.92E−01	5.95E−01	1.26E+00	1.83E+00	1.27E+00	1.02E+00
F_{13}	1.28E−25	5.40E−25	2.19E−03	4.47E−03	7.24E−01	1.48E+00	6.60E−02	4.33E−02
Functions	SCA		GSA		GA		HS	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F_8	-3.81E+03	**2.83E+02**	-2.75E+03	5.72E+02	-5.11E+03	4.37E+02	**−4.69E+02**	3.94E+02
F_9	2.23E+01	3.25E+01	3.35E+01	1.19E+01	1.23E−01	4.11E+01	**4.85E−02**	3.91E+01
F_{10}	1.55E+01	8.11E+00	8.25E−09	1.90E−09	5.31E−11	1.11E−10	2.83E−08	4.34E−07
F_{11}	3.01E−01	2.89E−01	8.19E+00	3.70E+00	**3.31E−06**	4.23E−05	2.49E−05	1.34E−04
F_{12}	5.21E+01	2.47E+02	2.65E−01	3.14E−01	9.16E−08	4.88E−07	1.34E−05	6.23E−04
F_{13}	2.81E+02	8.63E+02	**5.73E−32**	**8.95E−32**	6.39E−02	4.49E−02	9.94E−08	2.61E−07

Table 4
Average and standard deviation of best optimal solution for thirty independent runs in fixed-dimension multimodal test functions (F_{14}–F_{23}).

Functions	CSA		SSA		MFO		MVO	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F_{14}	**9.98E−01**	**2.73E−16**	1.031E+00	1.81E−01	2.21E+00	1.80E+00	**9.98E−01**	9.14E−01
F_{15}	**1.07E−04**	5.03E−03	2.13E−03	4.96E−03	1.58E−03	3.50E−03	7.15E−03	1.26E−02
F_{16}	**−103.16E−02**	5.53E−16	**−103.16E−02**	1.44E−14	**−103.16E−02**	**0.00E+00**	**−103.16E−02**	4.74E−08
F_{17}	**3.97E−01**	**0.00E+00**	7.30E−02	3.98E−01	1.13E−16	3.98E−01	3.98E−01	1.15E−07
F_{18}	**2.99E+00**	**2.74E−15**	3.00E+00	7.31E−14	3.00E+00	4.25E−15	5.70E+00	1.48E+01
F_{19}	-3.86E+00	**2.43E−15**	-3.86E+00	3.77E−14	−3.86E+00	3.16E−15	−3.86E+00	3.53E−07
F_{20}	-1.53E+00	**4.10E−02**	-3.24E+00	5.79E−02	-3.23E+00	6.65E−02	-3.23E+00	5.37E−02
F_{21}	-2.86E+00	**2.31E−02**	−7.55E+00	3.32E+00	-6.20E+00	3.52E+00	−7.38E+00	2.91E+00
F_{22}	-2.91E+00	1.05E−02	−8.80E+00	2.98E+00	-7.95E+00	3.20E+00	−8.50E+00	3.02E+00
F_{23}	**−2.87E+00**	**2.03E−02**	−9.48E+00	2.43E+00	−7.50E+00	3.68E+00	-8.41E+00	3.13E+00
Functions	SCA		GSA		GA		HS	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F_{14}	1.26E+00	6.86E−01	3.61E+00	2.96E+00	4.39E+00	4.41E−02	6.79E+00	1.12E+00
F_{15}	1.01E−03	3.75E−04	6.84E−03	7.37E−03	7.36E−03	**2.39E−04**	5.15E−03	3.45E−04
F_{16}	**−103.16E−02**	3.23E−05	**−103.16E−02**	**0.00E+00**	−1.04E+00	4.19E−07	**−103.16E−02**	3.64E−08
F_{17}	3.99E−01	7.61E−04	3.98E−01	1.13E−16	3.98E−01	3.71E−17	3.99E−01	9.45E−15
F_{18}	3.00E+00	2.25E−05	3.01E+00	3.24E−02	3.01E+00	6.33E−07	3.00E+00	1.94E−10
F_{19}	-3.86E+00	2.55E−03	**−3.22E+00**	4.15E−01	-3.30E+00	4.37E−10	-3.29E+00	9.69E−04
F_{20}	-2.84E+00	3.71E−01	**−1.47E+00**	5.32E−01	-2.39E+00	4.37E−01	-2.17E+00	1.64E−01
F_{21}	**−2.28E+00**	1.80E+00	-4.57E+00	1.30E+00	−5.19E+00	2.34E+00	-7.33E+00	1.29E+00
F_{22}	-3.99E+00	1.99E+00	-6.58E+00	2.64E+00	-2.97E+00	1.37E−02	**−1.00E+00**	**2.89E−04**
F_{23}	-4.49E+00	1.96E+00	-9.37E+00	2.75E+00	-3.10E+00	2.37E+00	-2.46E+00	1.19E+00

posed CSA compared to other well-known optimization algorithms in unimodal (F_1–F_7), multimodal (F_8–F_{13}) and fixed-dimension multimodal (F_{14}–F_{23}) benchmark functions. The average (*Ave*) and standard deviation (*Std*) criteria were typically considered as the best solutions in these benchmark functions. The values of these criteria were reported at the last iteration loop to give a persuasive assessment of the performance of the algorithms. The results of *Ave* and *Std* for the other algorithms were obtained from the literature (Dhiman & Kumar, 2017, 2019). Standard deviation analysis aims to ensure that the algorithms have reached stable performance throughout the thirty independent runs.

4.3.1. *Evaluation of test functions F_1–F_7*

The functions F_1 to F_7 are suitable for the experimental tests here to assess the exploitation potentiality of the proposed algorithm. Table 2 displays the mean and standard deviation of the optimum solution obtained, over thirty independent runs, by the proposed and the aforementioned algorithms in unimodal benchmark functions. From the results in unimodal test functions in Table 2, it is apparent that CSA outperformed almost all comparative algorithms in the majority of test functions. As it is seen, CSA was able to deliver convincing results in unimodal functions much better than other algorithms. This achievement is also shown on both the mean and standard deviation in the functions F_1, F_3, F_4, F_6 and F_7. In F_2, CSA was ranked third after GA and HS with only a slight difference in the mean outcome, and CSA received the second rank after GA in terms of the standard deviation score with only a very small difference. Further, the small (*Std*) results of CSA in all unimodal functions reveal that CSA is stable and this superiority is rooted. Based on the properties of unimodal test functions, it can be clearly expressed that CSA benefited from high exploitation capability.

4.3.2. *Evaluation of test functions F_8–F_{23}*

The benchmark test functions F_8 to F_{23} are suitable for testing the exploration behavior of the proposed CSA. Tables 3 and 4 display the average (*Ave*) and standard deviation (*Std*) results obtained by CSA and other optimization algorithms in multimodal (F_8–F_{13}) and fixed-dimensional multimodal (F_{14}–F_{23}) test functions, respectively. Tables 3 and 4 substantiate that CSA is able to reach the global optimum in the majority of multimodal and fixed-dimension multimodal functions (i.e., F_8, F_{10}–F_{23}) while some of the other methods did not. For the remaining test functions (i.e., F_9 and F_{11}), the outcomes of CSA were very close to the global optimum. For in-depth discussion and clarification, it is obvious from these outcomes that CSA outperformed other optimization algorithms in F_{10}, F_{12}, F_{14}, F_{15}, F_{16}, F_{17}, F_{18} and F_{23} benchmark test functions. For F_8 test function, which is the hardest test function in this category, CSA won the second rank after HS optimizer, which almost reached the global optimum. For F_9 benchmark test function, HS and GA both showed better performance than other optimization algorithms, while CSA still showed competitive results in this function and was the third best optimizer. GSA achieved the best optimal average results for F_{19} and F_{20} benchmark test functions, where CSA also delivered highly effective results in these test functions. For F_{21} test function, SCA and CSA are the first and second best optimizers, respectively. In short, by reading Table 3, it is important to note that CSA ranked first in multimodal test functions. In fixed-dimension problems, when reading Table 4, almost all optimization methods performed reasonably well, with the proposed CSA achieved the first-best performance optimizer. These findings confirm that CSA has a good score in terms of exploration ability. The standard deviation values of CSA are very small in these functions in the targeted categories, which assert that the performance of CSA is stable in these test functions.

4.3.3. *A comparison between CSA and adaptive PSO*

A selected set of eleven benchmark test functions mentioned in Table A.1 was solved using a proposed variant of PSO, referred to as Adaptive PSO (APSO) (Zhan, Zhang, Li, & Chung, 2009). Due to the high performance level of APSO in terms of solution accuracy and global optimality, it was chosen here for comparison with the proposed algorithm. In Zhan et al. (2009), APSO only solved unimodal and multimodal functions in Table A.1 with parameter settings somewhat different from those widely used in the literature. Therefore, the performance of CSA was compared with APSO in a separate table. To get a fair comparison between CSA and APSO, both were tested using the same population size

Table 5
A comparison between CSA and APSO in terms of *Ave* and *Std* solutions for 30 independent runs in unimodal and multimodal test functions.

F	CSA		APSO	
	Ave	*Std*	*Ave*	*Std*
F_1	2.59×10^{-23}	4.45×10^{-22}	$\mathbf{1.45}\times 10^{-150}$	$\mathbf{5.37}\times 10^{-150}$
F_2	3.90×10^{-13}	4.17×10^{-12}	$\mathbf{5.15}\times 10^{-84}$	$\mathbf{1.44}\times 10^{-84}$
F_3	$\mathbf{1.27}\times 10^{-21}$	$\mathbf{7.56}\times 10^{-20}$	1.00×10^{-10}	2.13×10^{-10}
F_5	$\mathbf{3.16}\times 10^{-01}$	$\mathbf{4.51}\times 10^{-01}$	$2.84\times 10^{+00}$	$3.27\times 10^{+00}$
F_6	1.73×10^{-23}	4.67×10^{-22}	$\mathbf{0.00}\times 10^{+00}$	$\mathbf{0.00}\times 10^{+00}$
F_7	$\mathbf{1.39}\times 10^{-04}$	$\mathbf{0.11}\times 10^{-04}$	$4.66\times 10^{-\mathbf{03}}$	1.70×10^{-03}
F_8	$-\mathbf{271.98}\times 10^{+01}$	$297.98\times 10^{+00}$	$-1256.95\times 10^{+01}$	$\mathbf{5.22}\times 10^{-11}$
F_9	$\mathbf{0.00}\times 10^{+00}$	$\mathbf{0.00}\times 10^{+\mathbf{00}}$	5.80×10^{-15}	1.01×10^{-14}
F_{10}	$\mathbf{1.20}\times 10^{-15}$	8.76×10^{-14}	1.11×10^{-14}	$\mathbf{3.55}\times 10^{-15}$
F_{11}	$\mathbf{0.08}\times 10^{-02}$	$\mathbf{0.29}\times 10^{-02}$	1.67×10^{-02}	2.41×10^{-02}
F_{12}	1.22×10^{-25}	9.09×10^{-25}	$\mathbf{3.76}\times 10^{-31}$	$\mathbf{1.20}\times 10^{-30}$

of 20, the same number of FEs of 2.0×10^5 for each test function associated with 1000 iterations. To reduce statistical errors and verify the stability of the algorithms, each function was experimented over thirty independent runs, and their mean and standard deviation results were used in the comparison. In the following test, the algorithm settings for the APSO are as follows: the inertia weight is initialized to 0.9, and both of c_1 and c_2 are initialized to 2.0. These parameters are then adaptively dominated during running the experiments. The performance of CSA versus APSO in terms of average and standard deviation solutions, obtained in the thirty independent runs, is shown in Table 5.

The comparison in Table 5 showed that CSA offered a high level of performance in many of the unimodal test functions. In particular, it presented higher accuracy in functions F_3, F_5 and F_7 than those achieved by APSO, while APSO delivered higher accuracy in functions F_1, F_2 and F_6 than those achieved by CSA. In regards to multimodal functions, both of CSA and APSO have achieved the global optimum in the optimization of functions F_8, F_9, F_{10} and F_{12}. Although CSA outperformed APSO in benchmark functions F_8, F_9, F_{10} and F_{11}, it did not perform as well as APSO in function F_{12} in both mean and standard deviation solutions. Notably, APSO reported very small mean and standard deviation solutions in several unimodal and multimodal test functions. However, the optimum values of the objective function in the functions given in Table A.1 are zero whether they are unimodal or multimodal functions, except for F_8. From an engineering point of view, it does not matter whether an algorithm finds the optimal result as 10^{-5} or 10^{-25}. Both are considered zero. Therefore, finding a much smaller value does not necessarily indicate better performance from a practical point of view.

4.4. Evaluation of IEEE CEC-2015 test functions

A challenging and recent benchmark test suite, IEEE CEC-2015, was used to further evaluate the performance of CSA to illustrate its effectiveness in these challenging functions. This test set involves unimodal, multimodal, hybrid and composition test functions (Kaur, Awasthi, Sangal, & Dhiman, 2020). The search space for all test functions in this group is $[-100, 100]$ with 30 dimensions for each test function. The parameter settings for CSA and all other comparative algorithms applied to CEC-2015 test suite are presented in Table 1. The number of FEs for each test function in this group was set to 30,000 bearing in mind that the number of search agents and iterations for each algorithm were set to 30 and 1000, respectively. Table 6 reveals the performance of CSA and other algorithms in CEC-2015 benchmark test functions. The results of other algorithms were possessed from the literature (Kaur et al., 2020).

It is apparent from Table 6 that CSA surpassed other algorithms in most of the test functions and scored better average scores for C-f1, C-f2, C-f4-f6, C-f8, C-f12 and C-f14 benchmark test functions. Moreover, it produced results close to optimality for C-f3 and C-f9 test functions that are similar to other comparative algorithms. Also, CSA generated near optimal results for C-f7 test function similar to those reported by GWO, PSO, MFO and MVO. Furthermore, CSA reported an average fitness value for C-f11 test function that is comparable to that value reported by PSO. However, CSA did not perform as well as MVO for C-f10 test function. The second best optimization algorithm in all CEC-2015 test

Table 6
Optimization results of CSA and other algorithms in the CEC-2015 test functions.

F		CSA	GWO	PSO	MFO	MVO	SCA	GSA	GA
C-f1	*Ave*	**3.36E+05**	2.02E+06	4.37E+05	1.47E+06	6.06E+05	7.65E+06	3.20E+07	8.89E+06
	Std	**1.08E+05**	2.01E+06	1.81E+05	1.00E+06	5.02E+05	3.07E+06	2.98E+06	6.95E+06
C-f2	*Ave*	**2.16E+03**	5.65E+06	9.41E+03	1.97E+04	1.43E+04	7.33E+08	4.58E+03	2.97E+05
	Std	**4.41E+02**	2.19E+06	4.82E+03	1.46E+04	1.03E+04	2.33E+08	1.09E+03	2.85E+03
C-f3	*Ave*	**3.20E+02**	**3.20E+02**	**3.20E+02**	**3.20E+02**	**3.20E+02**	**3.20E+02**	**3.20E+02**	**3.20E+02**
	Std	**1.02E−03**	7.08E−02	8.61E−02	9.14E−02	3.19E−02	7.53E−02	**1.02E−03**	2.78E−02
C-f4	*Ave*	**4.07E+02**	4.16E+02	4.09E+02	4.26E+02	4.18E+02	4.42E+02	4.39E+02	6.99E+02
	Std	**2.37E+00**	1.03E+01	3.96E+00	1.17E+01	1.03E+01	7.72E+00	7.25E+00	6.43E+00
C-f5	*Ave*	**8.32E+02**	9.20E+02	8.65E+02	1.33E+03	1.09E+03	1.76E+03	1.75E+03	1.26E+03
	Std	6.24E+02	1.78E+02	2.16E+02	3.45E+02	2.81E+02	2.30E+02	2.79E+02	**1.86E+02**
C-f6	*Ave*	**1.77E+03**	2.26E+04	1.86E+03	7.35E+03	3.82E+03	2.30E+04	3.91E+06	2.91E+05
	Std	**2.05E+02**	2.07E+04	1.28E+03	3.82E+03	2.44E+03	1.91E+04	2.70E+06	1.67E+05
C-f7	*Ave*	**7.02E+02**	**7.02E+02**	**7.02E+02**	**7.02E+02**	**7.02E+02**	7.06E+02	7.08E+02	7.08E+02
	Std	1.25E+00	**7.07E−01**	7.75E−01	1.10E+00	9.40E−01	9.07E−01	1.32E+00	2.97E+00
C-f8	*Ave*	**1.02E+03**	3.49E+03	3.43E+03	9.93E+03	2.58E+03	6.73E+03	6.07E+05	5.79E+04
	Std	2.08E+03	**2.04E+03**	2.77E+03	8.74E+03	1.61E+03	3.36E+03	4.81E+05	2.76E+04
C-f9	*Ave*	**1.00E+03**	**1.00E+03**	**1.00E+03**	**1.00E+03**	**1.00E+03**	**1.00E+03**	**1.00E+03**	**1.00E+03**
	Std	**4.19E−02**	1.28E−01	7.23E−02	2.20E−01	5.29E−02	9.79E−01	5.33E+00	3.97E+00
C-f10	*Ave*	3.18E+03	4.00E+03	3.27E+03	8.39E+03	**2.62E+03**	9.91E+03	3.42E+05	4.13E+04
	Std	**1.14E+03**	2.82E+03	1.84E+03	3.82E+03	1.18E+03	8.83E+03	1.74E+05	2.39E+04
C-f11	*Ave*	1.36E+03	1.40E+03	**1.35E+03**	1.37E+03	1.39E+03	**1.35E+03**	1.41E+03	1.36E+03
	Std	**4.18E+01**	5.81E+01	1.12E+02	8.97E+01	5.42E+01	1.11E+02	7.73E+01	5.39E+01
C-f12	*Ave*	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	1.31E+03	1.31E+03	1.31E+03
	Std	**4.69E−01**	6.69E−01	6.94E−01	9.14E−01	8.07E−01	1.54E+00	2.05E+00	1.65E+00
C-f13	*Ave*	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	1.35E+03	1.35E+03
	Std	**1.21E−07**	1.92E−04	5.44E−03	1.04E−03	2.43E−04	3.78E−03	4.70E+01	3.97E+01
C-f14	*Ave*	**3.51E+03**	7.29E+03	7.10E+03	7.60E+03	7.34E+03	7.51E+03	9.30E+03	8.96E+03
	Std	1.32E+03	2.45E+03	3.12E+03	**1.29E+03**	2.47E+03	1.52E+03	1.94E+03	6.32E+03
C-f15	*Ave*	**1.60E+03**	1.61E+03	**1.60E+03**	1.61E+03	**1.60E+03**	1.62E+03	1.64E+03	1.63E+03
	Std	**7.57E−11**	4.94E+00	1.06E−02	1.13E+01	1.80E−02	3.64E+00	1.12E+01	3.67E+01

Table 7
Optimization results of CSA and other meta-heuristic algorithms in the CEC-BC-2017 test functions.

F	CSA		EO		PSO		GWO		GSA		SSA	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
C-f1	**276.56**	**2.77E−01**	2465.30	2206.20	3959.60	4456.60	325132.00	107351.00	296.00	275.10	3396.25	3673.08
C-f3	**300.00**	5.78E−14	**300.00**	2.40E−08	**300.00**	1.90E−10	1538.00	1886.02	10829.20	1620.74	**300.00**	**0.00**
C-f4	**400.22**	**1.12E−04**	404.48	0.79	405.94	3.28	409.50	7.55	406.60	2.92	406.27	10.07
C-f5	**505.96**	5.26	510.73	**3.67**	513.06	6.54	513.50	6.10	556.70	8.40	521.82	10.50
C-f6	**600.00**	**7.88E−07**	**600.00**	1.50E−04	600.24	0.98	600.60	0.88	621.60	9.01	609.77	8.26
C-f7	716.26	7.17	720.93	5.74	718.98	5.10	729.80	8.60	**714.60**	**1.55**	740.88	16.62
C-f8	**803.96**	5.12	809.51	2.91	811.39	5.47	814.30	8.26	820.50	**4.69**	823.45	9.95
C-f9	**900.00**	**5.31E−14**	**900.00**	2.27E−02	**900.00**	5.90E−14	911.30	19.53	**900.00**	6.90E−14	944.07	104.66
C-f10	**1023.60**	216.17	1418.70	261.63	1473.30	**214.97**	1530.50	286.67	2694.60	297.62	1858.85	294.50
C-f11	**1101.31**	**3.63**	1105.20	5.02	1110.50	6.28	1140.20	54.13	1134.70	10.45	1180.50	59.80
C-f12	**1781.00**	**1699.00**	1.03E+04	9790.00	1.45E+04	1.12E+04	6.25E+05	1.12E+06	7.02E+05	42.07E+03	1.98E+06	1.90E+06
C-f13	**1328.46**	**211.41**	8023.00	6720.80	8601.10	5123.60	9842.30	5633.43	11053.00	2110.55	16098.60	10537.20
C-f14	**1402.29**	74.44	1463.30	**32.49**	1482.10	42.46	3403.53	1953.33	7147.50	1489.52	1508.94	51.05
C-f15	**1503.02**	122.13	1585.60	**48.01**	1714.30	282.89	3806.60	3860.66	18001.00	5498.67	2236.69	571.19
C-f16	**1600.38**	51.18	1649.00	**50.91**	1860.00	127.65	1725.78	123.85	2149.70	105.80	1726.26	126.97
C-f17	**1702.15**	33.31	1731.60	18.07	1761.60	47.50	1759.61	**31.29**	1857.70	108.32	1774.57	34.23
C-f18	**1850.01**	**750.28**	12450.00	11405.00	14599.00	11852.20	25806.10	15766.90	8720.50	5060.10	23429.10	14045.70
C-f19	**1903.79**	**24.23**	1951.50	47.11	2602.80	2185.02	9866.10	6371.09	13670.00	19168.00	2916.10	1871.20
C-f20	**2011.99**	23.44	2020.60	**22.28**	2085.10	62.25	2075.60	52.04	2272.30	81.72	2089.30	49.28
C-f21	**2200.00**	41.45	2307.50	20.96	2281.70	54.02	2317.10	**7.00**	2357.70	28.20	2249.80	60.44
C-f22	**2223.36**	17.93	2297.40	18.40	2314.80	66.10	2310.10	16.75	2300.00	**7.2E−02**	2301.50	11.80
C-f23	2616.15	7.05	**2615.80**	**5.52**	2620.80	9.23	2616.40	8.47	2736.50	39.14	2621.70	8.69
C-f24	**2630.15**	15.07	2743.80	6.90	2692.20	108.20	2741.70	8.73	2742.20	**5.52**	2733.20	64.43
C-f25	**2897.94**	19.52	2934.30	19.76	2924.00	25.02	2938.00	23.61	2937.50	**15.36**	2923.50	23.86
C-f26	**2600.00**	**44.51**	2967.80	164.98	2952.10	249.66	3222.50	427.02	34407.50	628.73	2900.90	36.56
C-f27	3092.64	10.19	**3091.30**	**2.24**	3116.20	24.99	3104.10	21.81	3259.50	41.66	3092.60	2.78
C-f28	**3160.75**	113.51	3302.70	133.92	3315.90	121.83	3391.20	101.50	3459.40	**33.84**	3210.50	113.17
C-f29	**3140.19**	**13.87**	3169.90	24.65	3203.80	52.26	3190.50	42.90	3449.50	171.33	3214.10	51.69
C-f30	**4.15E+03**	**1.20E+05**	2.97E+05	4.58E+05	3.50E+05	5.04E+05	2.97E+05	5.27E+05	1.30E+06	3.63E+05	4.21E+05	5.68E+05

functions is PSO that scored near optimal average results in seven out of fifteen test functions. Also, other optimization algorithms have sensible results in some CEC-2015 benchmark test functions. The difference between the average result reported by PSO and SCA and that reported by CSA in C-f11 function is very small, but the standard deviation of CSA is very small compared to those reported by these optimizers in this test function. Regarding the standard deviation results, the proposed CSA has the smallest *Std* values in most of the CEC-2015 test functions compared to other competitors. These results emphasize that the preeminence of CSA is stable.

4.5. Evaluation of IEEE CEC-2017 test functions

In order to further challenge the proposed CSA, a more recent and challenging test suite, CEC-BC-2017, was used. This test suite involves shifted and rotated unimodal and multimodal, hybrid, and composition functions (Awad et al., 2017). It should also be noted that due to the unstable behavior of C-f2 function, it was removed from this test suite (Awad et al., 2017). The search domain for all functions, in this test suite, is $[-100, 100]$ with 10 dimensions for each test function. The performance of CSA was tested against this test group and the obtained results were compared with other well-known meta-heuristic algorithms used previously. The majority of the test functions offered in this test suite are among the more challenging hybrid and composition functions. The results of all algorithms were obtained using 50 search agents with 1000 iterations and 30 independent runs. The maximum number of FEs for each test function in this test suite was set to 50,000 taking into account the number of search agents and number of iterations. The parameter settings for each algorithm are the same as those listed in Table 1. Five optimization algorithms, as shown in Table 7, were compared with CSA in the optimization of the IEEE CEC-2017 benchmark test functions. The results of the other algorithms given in Table 7 have been reported in the literature (Faramarzi et al., 2020).

The results in Table 7 bear out the superiority of the proposed algorithm over other optimization algorithms in optimizing extremely challenging functions. The proposed algorithm reported best average fitness values in twenty-six out of twenty-nine benchmark test functions (that is, C-f1 - C-f6, C-f8-C-f22, C-f24- C-f26, C-f28, C-f29 and C-f30). For C-f7 benchmark test function, GSA reported the best average score, where the proposed algorithm reported the second best score with slight difference from that reported by GSA. For C-f23 and C-f27 benchmark test functions, the performance of EO is better than other algorithms in terms of average fitness value. CSA is the second best optimizer for these test functions that reported comparable results to EO with small difference margins. When reading the *Std* values in Table 7, the proposed CSA performed substantially better than other algorithms in thirteen out of twenty-nine benchmark test functions. This supports the fact what we concluded earlier that the proposed algorithm has a high level of stability when applied to different search areas and test beds. These results reveal that CSA ranks first for robustness in exploration and exploitation capabilities. In a nutshell, the overall ranking that the proposed algorithm came to in this test was the first. This accomplishment is further evidence of the ability of CSA to excel in well-examined algorithms while achieving very encouraging results in very challenging functions.

4.6. Qualitative results of CSA

Qualitative results are most often derived from different visualization tools. The best qualitative results of optimization algorithms in the literature are convergence behavior curves. The convergence curve is plotted as a line to characterize how well an algorithm augments the approximation of the global optimum over a number of iterations. In this work, qualitative solution results including convergence curves, average fitness of chameleons, trajectory of the first chameleon and search history of all chameleons were used to clearly analyze the performance of CSA when addressing some of the benchmark functions described above. These qualitative results aim to further emphasize that CSA effectively performs exploration and exploitation mechanisms while solving basic benchmark test functions. Fig. 4 shows the qualitative results of CSA while solving some unimodal, multimodal and fixed-dimensional

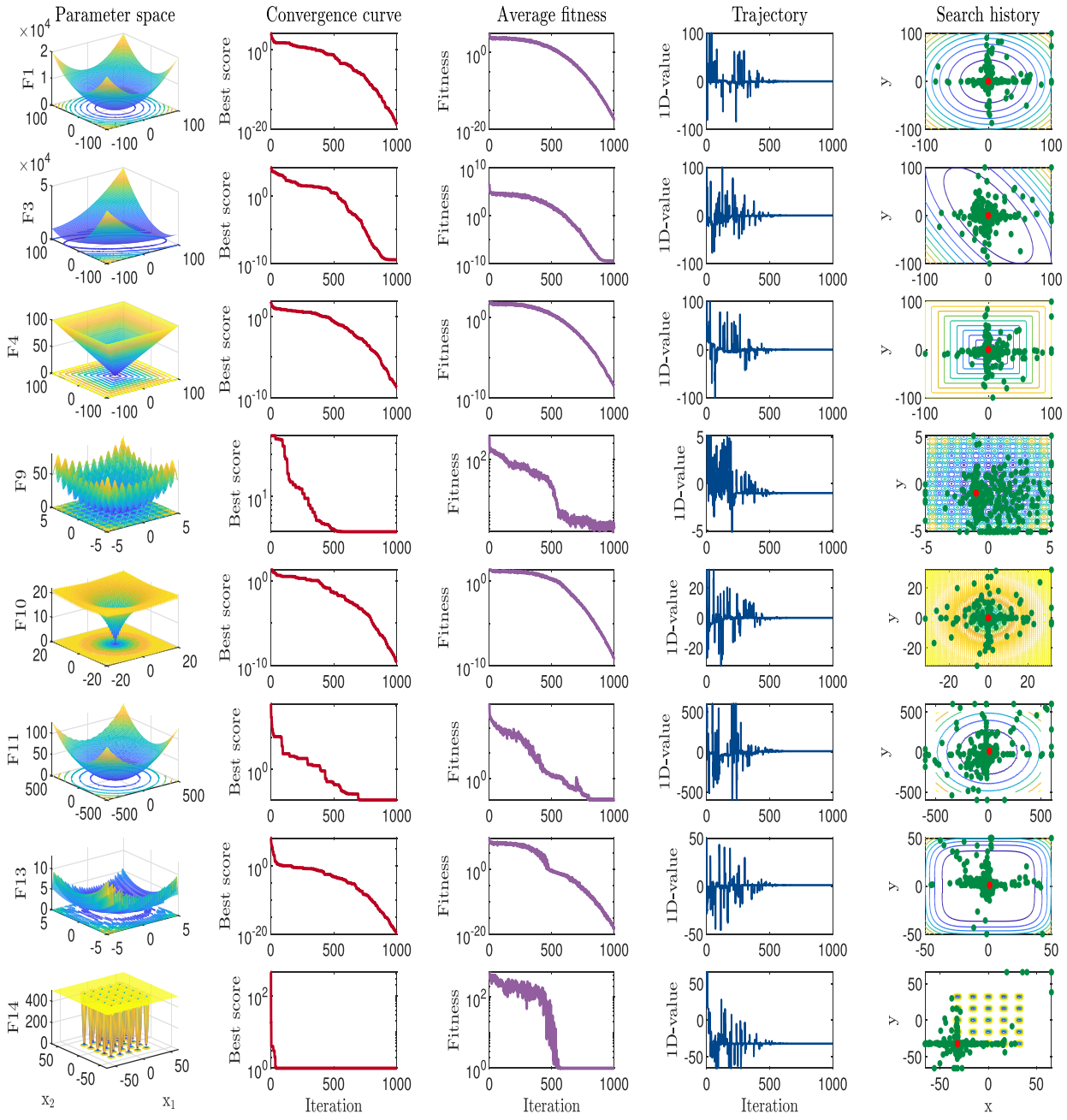

Fig. 4. Qualitative results of CSA for F_1, F_3, F_4, F_9, F_{10}, F_{11}, F_{13} and F_{14}: Parameter space of the benchmark functions, convergence curves, average fitness curves of all chameleons, trajectory in the first dimension and search history for all chameleons.

multimodal functions, where convergence and average fitness curves are presented in logarithmic scales.

There are several metrics used in Fig. 4 to visualize the qualitative performance of CSA, which can be interpreted as follows:

- The first qualitative measure represents graphically the convergence curve of CSA in terms of obtaining the global optimal solution during the path of iterations. Convergence measures are an essential feature to better understand exploration and exploitation mechanisms in optimization algorithms. The interesting finding is that all of these curves exhibit that CSA delivered promising convergence behaviors in optimizing all of the studied test functions. In the initial iteration steps, CSA swiftly converges to the most promising areas in the search space. In the next and final iteration steps, CSA progressively converges to the near global or the global optimum in the majority of test functions such as F_1, F_3, F_4, F_{10}, F_{13} and F_{14} with relatively small errors in some functions such as F_9 and F_{11}. It is the stability of the proposed algorithm with variant function types that supports it to obtain a stable convergence. This is evidenced by the smooth convergence curves and the consistent way in which these curves approach the minimum error over a pre-defined number of iterations. As per these observations, chameleons in CSA initially roam the search space to explore various promising regions. Then, CSA exploits the search space by inspiring the chameleons to search for the global optimum solution and motivating them to move locally, not globally.
- The average fitness curves of CSA gradually converges towards the optimal solutions after the first 400 iterations to exploit the global optimum solutions to achieve plausible behavior in the final iteration steps. Moreover, the proposed algorithm prevents solutions from

rapidly converging towards a local optimal solution. This verifies that CSA not only enhances the global best chameleon but also the fitness values of all chameleons. In more detail, CSA provides fast convergence response for F_1, F_3, F_4 and F_{10}, has a sensible convergence response for F_9 and F_{11}, and finds optimal solutions with convincing convergence behaviors for F_{13} and F_{14}.

- The path in the first dimension shows the value of the first variable of the function at each iteration loop. The trajectory curves illustrate that the chameleons render large and sudden changes in the first 500 iteration steps of optimization.
- Search history displays the location history of chameleons while optimizing a test function. It is observed from Fig. 4 that the chameleons explore the most promising areas in the given search space for the considered test functions. For unimodal functions F_1, F_3 and F_4, the sample points are slightly divvied up in the non-promising regions. On the other hand, most of the sample points in the multimodal functions, F_9–F_{11}, F_{13} and F_{14}, are widely distributed around non-promising areas. This is due to the level of difficulty of these test functions. This implies that the proposed CSA explores the entire search space and avoids being caught in local optima. The sample points are scattered around the optimum solution that ensures the ability of CSA in reliably exploiting the search space. Therefore, CSA has the ability to explore and exploit the search space well.

In short, the convergence and average fitness curves in Fig. 4 reveal that CSA can provide a stable optimization for a range of test functions that vary in dimensions and complexity of the search space. The search history and trajectory plots show that CSA can conserve an adequate balance between exploration and exploitation. This feature basically helps CSA to find the global optimum solution.

4.7. Sensitivity analysis of CSA

The process of finding the best parameter settings for a meta-heuristic algorithm in solving a given problem is a major issue (Yang, Deb, Loomes, & Karamanoglu, 2013). Different values of the control parameters of an algorithm might yield different results as reported in the literature (Braik et al., 2020). This is because most of the meta-heuristic algorithms act on a general concept and do not have specific domain knowledge for each problem (Sree Ranjini & Murugan, 2017). Hence, there is a need to tune the control parameters of each meta-heuristic that could provide more adaptability and robustness in solving a broad range of problems in various fields of science (Jain et al., 2019). In the literature, there are various parameter setting methods that have been developed for various meta-heuristics as a role in enhancing optimization algorithms (Jain et al., 2019; Braik et al., 2020).

In this work, the optimal parameter set of the proposed algorithm was selected based on the use of Design of Experiment (DoE) framework through empirical testing of the proposed algorithm on a selected set of test functions. This was carried out by assessing the sensitivity of the proposed algorithm to its control parameters. This strategy for setting the optimal set of parameters for CSA to solve a problem was tested here using different values for each control parameter. In this context, a reasonable range for each control parameter of CSA was initially defined and experiments were systematically performed. The trends of the values of the control parameters were defined to identify whether the best values are within the range or further experimentation is needed. The parameter values were varied many times until a reasonable solution was obtained. This parameter setting method is useful to identify which parameters of CSA are powerful, sensitive to various inputs and which of these parameters have a significant influence on the accuracy of the proposed algorithm.

Here, the proposed parameter setting method examines the sensitivity of the number of iterations (T), number of search agents (n) and three control parameters of CSA, which are respectively, ρ, α and β. While designing the CSA, this study conducted a complete design with these parameters on a subset of functions selected from each category of unimodal, multimodal, fixed-dimension multimodal, CEC-2015 and CEC-2017 test suites. These test functions are respectively F_1, F_{10}, F_{14}, CEC-2015-f1 and CEC-2017-f8. The values of each parameter for the sensitivity analysis design were defined as follows: $T = \{100, 500, 800, 1000\}$, $n = \{30, 50, 80, 100\}$, $\rho = \{1.0, 2.0, 3.0, 4.0\}$, $\alpha = \{2.5, 3.0, 3.5, 4.0\}$, $\beta = \{2.5, 3.0, 3.5, 4.0\}$. Each function was solved with a maximum of 30,000 function evaluations and 30 independent runs. The results of the sensitivity analysis, in terms of the mean fitness and convergence curves, for the above five functions with the five control parameters and their associated values as described above are illustrated below:

Table 8
Average fitness values of CSA using different simulation runs.

Iterations	Functions				
	F_1	F_{10}	F_{14}	C-2015-f1	C-2017-f8
100	0.25E−01	0.32E−03	1.99E+00	3.40E+07	809.96
500	1.10E−09	1.78E−12	9.98E−01	5.31E+06	808.97
800	2.70E−20	1.50E−14	9.98E−01	2.74E+06	805.97
1000	2.05E−26	7.99E−15	9.98E−01	3.33E+05	804.07

1. Maximum number of iterations (T): CSA was simulated for various numbers of iterations. Table 8 displays the average fitness of CSA when it was applied to simulate five different benchmark functions with different number of iterations. Fig. 5(a) displays the convergence curves of CSA for F_1 using different numbers of iterations.

   The computational and visual results in Table 8 and Fig. 5(a), respectively, demonstrate that CSA converges to the optimum solution when the number of iterations increases. This supports the importance of the number of iterations on the robustness and convergence behavior of CSA.
2. Number of chameleons (n): CSA was simulated for different numbers of chameleons, with number of iterations is fixed to 1000. Table 9 shows the average fitness of CSA when it was applied to solve F_1, F_{10}, F_{14}, C-2015-f1 and C-2017-f8 with different number of chameleons. Fig. 5(b) show the convergence curves of CSA on the F_1 benchmark function associated with different number of chameleons.

   Table 9 shows that the best performance was obtained with a large number of search agents. Moreover, what could be deduced from Fig. 5(b) is that the fitness value decreases when the number of chameleons increases. As shown in Fig. 5(b), the sensitivity of CSA increases with the number of search agents.
3. Control parameter (ρ): to check the effect of the parameter ρ on the proposed algorithm, CSA was run for different values of ρ, with 1000 iterations and 80 search agents, keeping other parameters unchanged. The values of ρ used in the experiments are 1.0, 2.0, 3.0 and 4. The sensitivity of CSA with different values of ρ on F_1 is shown by the convergence curves in Fig. 5(c).

   Table 10 shows that CSA has relatively slight sensitivity to this control parameter, where it presented small differences between the results. Fig. 5(c) displays relatively stable behavior of CSA when ρ ranges from 2.0 to 3.0. Based on this analysis, the best value for ρ, with CSA having the best results, can be suggested as its smallest value (i.e., $\rho = 1.0$).
4. Control parameter (α): to investigate the sensitivity of CSA to this parameter, CSA was simulated for various values of α including 2.50, 3.0, 3.50, and 4.0. The number of search agents is 80 and the other parameters remain the same. Table 11 shows the effect of α on the average fitness of CSA in various benchmark functions. The effect of different values of α on the convergence curves of CSA on function F_1 is shown in Fig. 5(d).

   It is evident from Table 11 and Fig. 5(d) that CSA is highly sensitive to the parameter α, as there is a large difference between the

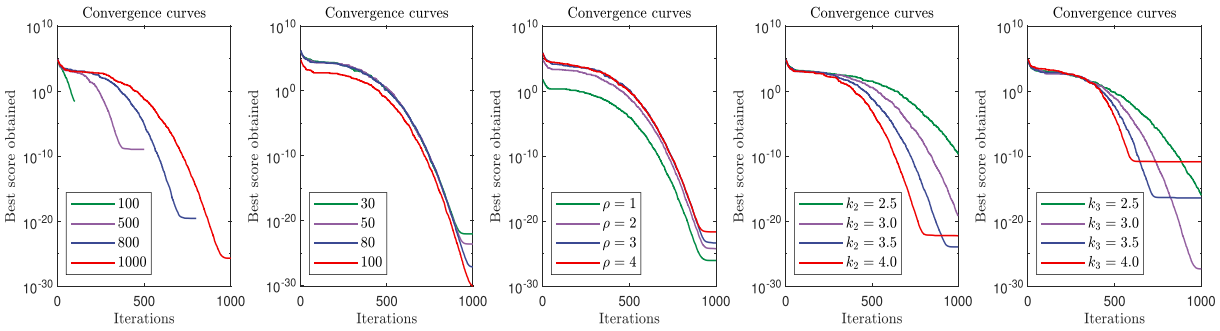

Fig. 5. Sensitivity analysis of the proposed CSA for: (a) Number of iterations, (b) Number of search agents, (c) Control parameter ρ, (d) Control parameter α, (e) Control parameter β.

Table 9
Average fitness values of CSA using different number of chameleons with 1000 iterations.

Chameleons	Functions				
	F_1	F_{10}	F_{14}	C-2015-f1	C-2017-f8
30	2.05E−26	7.99E−15	9.98E−01	3.33E+05	804.07
50	9.79E−27	2.76E−15	9.98E−01	2.77E+05	803.38
80	2.70E−29	2.34E−16	9.98E−01	1.55E+05	803.07
100	7.44E−31	4.44E−17	9.98E−01	1.43E+05	802.92

Table 10
Average fitness values of CSA using different values for the parameter ρ.

ρ	Functions				
	F_1	F_{10}	F_{14}	C-2015-f1	C-2017-f8
1.0	2.70E−29	2.34E−16	9.98E−01	1.55E+05	803.07
2.0	6.03E−25	7.99E−14	9.98E−01	6.20E+05	806.97
3.0	4.54E−25	4.44E−12	9.98E−01	2.72E+06	808.77
4.0	2.29E−23	5.12E−09	9.98E−01	1.91E+06	811.96

results based on the values of this parameter. CSA clearly achieved the best results when the value of α is 3.5.

5. Control parameter (β): to explore the effect of β on the sensitivity of CSA, it was simulated for different values of this parameter while keeping other parameters unchanged and set n to 80. Table 12 shows the average fitness values of CSA when it was used to solve various functions with different values of β, with Fig. 5(e) shows the effect of different values of β on F_1.

It is obvious from Table 12 and Fig. 5(e) that CSA is largely sensitive to β, as there is a big difference between the results obtained using different values for β. It is also noticed that CSA yielded the best results when the value of β was set to 3.0, indicating that the balance between exploration and exploitation is effective with this value.

In short, we can observe from the results reported in Tables 8–12 that different solutions were obtained for the same benchmark test functions using different parameter settings. Based on the above analysis, the general behavior of CSA indicates that it is sensitive to the number of iterations and search agents in which the performance level of CSA increases with the increase in the values of these parameters. Moreover, we can identify that the most appropriate set of parameters for the proposed algorithm are the ones that offered the best performance. Therefore, we can state that the best values of the control parameters of CSA are as follows: $\rho = 1.0$, $\alpha = 3.5$ and $\beta = 3.0$, where these values were used for all of the problems solved in this paper. However, these parameters can be adjusted for other problems as needed, as is often the case, only proper settings are obtained, perhaps, not the "'best"' settings.

Table 11
Average fitness values of CSA using different values for the parameter α.

α	Functions				
	F_1	F_{10}	F_{14}	C-2015-f1	C-2017-f8
2.50	2.13E−10	1.24E−07	9.98E−01	2.15E+06	805.99
3.0	6.55E−20	3.67E−12	9.98E−01	1.91E+06	804.97
3.5	2.70E−29	2.34E−16	9.98E−01	1.55E+05	803.07
4.0	5.69E−23	1.73E−12	9.98E−01	1.02E+06	806.96

Table 12
Average fitness values of CSA using different values for the parameter β.

β	Functions				
	F_1	F_8	F_{14}	C-2015-f1	C-2017-f8
2.50	9.98E−17	7.01E−11	9.98E−01	1.23E+06	806.96
3.0	2.70E−29	2.34E−16	9.98E−01	1.55E+05	803.07
3.50	3.78E−17	4.44E−15	9.98E−01	4.19E+06	806.97
4.0	1.38E−11	5.11E−10	9.98E−01	4.28E+06	807.96

4.8. Statistical test results of CSA

This subsection conducted multiple statistical tests to first comprehend whether the differences in the accuracy of all optimization methods in the test functions are statistically meaningful using Friedman's test (Pereira, Afonso, & Medeiros, 2015). Friedman's test is a non-parametric statistical test method commonly used to assess the performance of algorithms. In order to have a trustworthy comparison, there is a need to perform a comparison between more than ten test functions with more than five different algorithms (Demšar, 2006). In terms of the number of algorithms, this study actually looked at more than five algorithms. With regard to the test functions, this work has been fully tested three test suites. The first group comprises of unimodal, multimodal and fixed-dimension multimodal with 23 test functions. The second group is the CEC-2015 test suite which includes 15 test functions. Finally, the third group is the CEC-2017 test suite which includes 29 test functions.

Friedman's test demands the computation of the average ranked value. Next, a comparison is required to evaluate the critical values got for the examined significance level ($\alpha = 0.05$) with Friedman's test to determine whether or not the null hypothesis is rejected. In this context, if the significance level (i.e., p-value) revealed by Friedman's test is equal or less than 0.05, the null hypothesis is rejected, indicating that there is a large difference between the performance of the algorithms (Liu, Mernik, Hrnčič, & Črepinšek, 2013). Next, more steps are required to find out which algorithms' performance is significantly different from CSA and which algorithms are analogous. This is to verify whether the results achieved by CSA diverge from the results of other optimization methods in a statistically significant way. For this purpose, this study

Table 13
Average rankings of algorithms using Friedman's test based on their results in benchmark functions (F_1–F_{23}).

Algorithm	Rank
CSA	**1.847826**
SSA	4.673913
MFO	6.021739
MVO	5.913043
SCA	5.434782
GSA	4.434782
GA	3.934782
HS	3.739130

Table 14
Holm's test for benchmark functions F_1–F_{23} (CSA is the control algorithm).

i	CSA vs.	z-value	p-value	α/i (0.05)	Hypothesis
7	MFO	5.778520	7.536016E−09	0.007142	Rejected
6	MVO	5.628038	1.822702E−08	0.008333	Rejected
5	SCA	4.965916	6.837739E−07	0.010000	Rejected
4	SSA	3.912540	9.133031E−05	0.012500	Rejected
3	GSA	3.581479	3.416544E−04	0.016666	Rejected
2	GA	2.889260	0.003861	0.025000	Rejected
1	HS	2.618392	0.008834	0.050000	Rejected

Table 15
Average rankings of algorithms using Friedman's test based on their results in the CEC-2015 benchmark functions.

Algorithm	Rank
CSA	**2.200000**
GWO	4.266666
PSO	2.733333
MFO	4.600000
MVO	3.433333
SCA	5.600000
GSA	6.800000
GA	6.366666

considered a post hoc statistical analysis of Holm's test method (Holm, 1979) to know which methods are better or worse than CSA and at which significance level. Holm's test is a broadly applicable test procedure based upon the sequent rejective manner. It sorts all optimization methods on the basis of their p-values and compares them to $\alpha/k-i$, where α is the level of significance, k stands for the degree of freedom and i refers to the algorithm number. The method begins with the most important p-value and successively rejects null hypothesis provided that $p_i < \alpha/k-i$. Once the method is unable to reject the hypothesis, it ceases and deems all the remaining hypotheses acceptable. In the application of Friedman's test, the best performing algorithm fetches the lowest rank while the worst acting algorithm gets the highest rank. The lowest ranked algorithm is used as a control method for post hoc analysis. A summary of the ranking results for different algorithms using Friedman's test in the functions (F_1–F_{23}) with $\alpha = 0.05$ is shown in Table 13 based on the results given in Tables 2 and 3.

In Table 13, Friedman's statistic distributed according to chi-square with 7 degrees of freedom is 50.420289. The p-value computed by Friedman's test based on the results of the algorithms in the functions F_1 to F_{23} is 1.198945E−08, confirming that there are statistically noteworthy differences between the accuracy of these algorithms. As per the ranking results in Table 13, CSA is statistically significant and the control algorithm that performed best among all other algorithms. CSA outperformed all other algorithms, with the lowest rank of 1.847826 at the considered significance level of 5%. CSA is clearly followed successively by HS, GA, GSA, SSA, SCA, MVO and finally MFO. Holm's test

Table 16
Holm's test for CEC-2015 benchmark functions (CSA is the control algorithm).

i	CSA vs.	z-value	p-value	α/i (0.05)	Hypothesis
7	GSA	5.142956	2.704484E−07	0.007142	Rejected
6	GA	4.658474	3.185605E−06	0.008333	Rejected
5	SCA	3.801315	1.439298E−04	0.010000	Rejected
4	MFO	2.683281	0.007290	0.012500	Rejected
3	GWO	2.310603	0.020854	0.016666	Rejected
2	MVO	1.378908	0.167922	0.025000	Not rejected
1	PSO	0.596284	0.550984	0.050000	Not rejected

Table 17
Average rankings of algorithms using Friedman's test based upon their results in the CEC-BC-2017 benchmark functions.

Algorithm	Rank
CSA	**1.258620**
EO	2.465517
PSO	3.482758
GWO	4.448275
GSA	5.051724
SSA	4.293103

Table 18
Holm's test for CEC-2017 benchmark functions (CSA is the control algorithm).

i	CSA vs.	z-value	p-value	α/i (0.05)	Hypothesis
5	GSA	7.720486	1.158865E−14	0.010000	Rejected
4	GWO	6.492227	8.457654E−11	0.012500	Rejected
3	SSA	6.176389	6.558420E−10	0.016666	Rejected
2	PSO	4.527012	5.982339E−06	0.025000	Rejected
1	EO	2.456518	0.014029	0.050000	Rejected

method was then applied to ascertain if there are statistically large differences between the control algorithm and the other algorithms. The statistical results obtained using Holm's method are given in Table 14.

Holm's procedure in Table 14 rejects those hypotheses that have a p-value $\leqslant$0.016666. As it is evidenced from the results of Holm's method for the functions F_1–F_{23}, CSA is statistically significant and effective in providing substantially competitive results such as those obtained by other meta-heuristics reported in the literature. Table 15 shows the average rank of all algorithms using Friedman's test in the CEC-2015 benchmark test functions.

In Table 15, Friedman's statistic distributed according to chi-square with 7 degrees of freedom is 48.994444. The p-value obtained by Friedman's test in the CSC-2015 functions is 2.278411E−08, substantiating that there are statistically momentous differences between the performance of CSA and other algorithms. In Table 15, the proposed CSA represents the control algorithm since it is able to surpass other algorithms. It reported a performance level that outperformed other algorithms with the lowest average rank of 2.2, and the performance was significantly better than PSO, MVO, GWO, MFO, SCA, GA and GSA.

The results of Holm's method in the CEC-2015 functions are shown in Table 16.

In Table 16, Holm's test rejects the hypotheses with a p-value $\leqslant$0.016666. The results in Table 16 show that CSA is statistically significant and is largely better than all other competitive algorithms that have reported a high degree of performance in the literature.

The average ranks of algorithms using Friedman's test in the CEC-2017 benchmark test suite are shown in Table 17.

In Table 17, Friedman's test distributed according to chi-square with 7 degrees of freedom is 83.108374. The p-value obtained by Friedman's test in the CSC-2017 functions is 7.740075E−11, which points out that there are statistically large differences between the performance of CSA and other rivals. In Table 17, CSA placed first with an average rank of

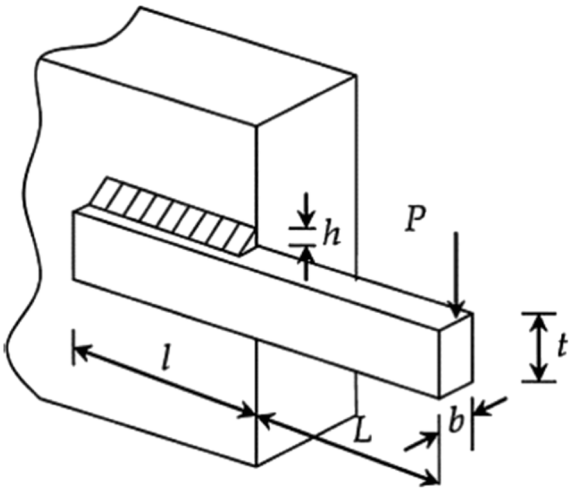

Fig. 6. A schematic diagram of a welded beam structure.

1.258620. This means that CSA outperformed other algorithms, which can be placed in order after CSA as follows: EO, PSO, SSA, GWO and GSA in the last rank.

The results of the Holm's test method, which was applied after applying Friedman's test in the CEC-2017 test functions, are given in Table 18.

Holm's procedure in Table 18 rejects those hypotheses that have a *p*-value $\leqslant 0.016666$. As can be inferred from Table 18, CSA's performance is statistically significant and its performance outperformed other promising algorithms mentioned in the literature.

As a main conclusion drawn from the statistical analysis of all benchmark functions in this study, overall, CSA performed markedly better than other well-studied and effective optimization algorithms in the literature such as SSA, MFO, MVO, SCA, GSA, GA, HS, GWO, PSO and EO. This bears out the satisfactory performance and reliability of the proposed algorithm, and affirms that this algorithm can effectively explore the search space whether it contains one optima or many optimums. These conclusions are splendid motivations to test this algorithm on more challenging problems that we usually encounter in real-world problems.

5. Applications of CSA on engineering problems

The reliability of CSA in solving real-world problems, particularly constrained optimization problems, is revealed by its evaluation on classical engineering design problems. In this section, CSA was applied to solve five well-studied engineering design problems: the welded beam design problem, the pressure vessel design problem, the tension/compression spring design problem, the speed reducer design problem and the rolling element bearing design problem. These design problems have a variety of constraints that require a constraint handling method to be used to optimize them.

Table 19
Optimization results of the welded beam design problem obtained by CSA and other optimization algorithms.

Algorithm	Optimal values for variables				Optimum cost
	h	l	t	b	
CSA	0.205730	3.470489	9.036624	0.205730	**1.724852**
SHO	0.205563	3.474846	9.035799	0.205811	1.725661
GWO	0.205678	3.475403	9.036964	0.206229	1.726995
PSO	0.197411	3.315061	10.000000	0.201395	1.820395
MFO	0.203567	3.443025	9.230278	0.212359	1.732541
MVO	0.205611	3.472103	9.040931	0.205709	1.725472
SCA	0.204695	3.536291	9.004290	0.210025	1.759173
GSA	0.147098	5.490744	10.000000	0.217725	2.172858
GA	0.164171	4.032541	10.000000	0.223647	1.873971
HS	0.206487	3.635872	10.000000	0.203249	1.836250

5.1. Constraint handling

To deal with the constraints of the aforementioned design problems, CSA was adapted with a static penalty handling method (Yang, 2010b). This is to arrive at a fair comparison between CSA and the comparative methods used in this work. The penalty function of this constraint handling method can be defined as follows:

$$\zeta(z) = f(z) \pm \left[\sum_{i=1}^{m} l_i \cdot max(0, t_i(z))^{\gamma} + \sum_{j=1}^{n} o_j \left| U_j(z) \right|^{\psi} \right] \tag{22}$$

where $\zeta(z)$ is the objective function, o_j and l_i are two positive penalty constants, $U_j(z)$ and $t_i(z)$ are constraint functions. The values of the parameters ψ and γ were set to 2 and 1, respectively.

This constraint method is characterized by its simplicity and low computational burden cost. It is essential to mention that all the engineering problems below are solved using the parameter settings used by CSA as shown in Table 1, but the parameters p_1 and p_2 are each set to 0.85.

5.2. Welded beam design problem

The design of this problem is a cantilever beam welded at one end and subjected to a spot load at the other end. The aim of this problem is to design a welded beam for the structure shown in Fig. 6 (Wang, Guo, Gandomi, Hao, & Wang, 2014) to achieve the lowest manufacturing cost.

The welded beam structure consists of a beam, *A*, and a welding required to attach it to the member, *B*. This problem is subject to some constraints, including: shear stress (τ), bending stress in the beam (θ), buckling load on the bar (P_c) and an end deflection of the beam (δ). To solve this problem, there is a need to locate the possible combination of the following structural parameters of the welded beam design: thickness of the weld (h), length of the clamped bar (l), height of the bar (t) and thickness of the bar (b). These parameters were represented by a vector as follows: $\vec{x} = [x_1, x_2, x_3, x_4]$, where x_1, x_2, x_3 and x_4 stand for h, l, t and b, respectively. The mathematical formula of the cost function for this problem is given as follows:

$$f(\vec{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2)$$

Subject to the following restrictions,

$$g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \leqslant 0$$

$$g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \leqslant 0$$

$$g_3(\vec{x}) = x_1 - x_4 \leqslant 0$$

$$g_4(\vec{x}) = 1.10471 x_1^2 + 0.04811 x_3 x_4 (14.0 + x_2) - 5.0 \leqslant 0$$

$$g_5(\vec{x}) = 0.125 - x_1 \leqslant 0$$

$$g_6(\vec{x}) = \delta(\vec{x}) - \delta_{max} \leqslant 0$$

$$g_7(\vec{x}) = P - P_c(\vec{x}) \leqslant 0$$

Other parameters of this design problem are identified as follows:

$$\tau(\vec{x}) = \sqrt{((\tau')^2 + (\tau'')^2) + \frac{2\tau'\tau'' x_2}{2R}}, \quad \tau' = \frac{p}{\sqrt{2} x_1 x_2}$$

$$\tau'' = \frac{MR}{J}, \quad M = P(L + \frac{x_2}{2}), \quad R = \sqrt{(\frac{x_1 + x_3}{2})^2 + \frac{x_2^2}{4}}$$

Table 20
Statistical results of CSA and other optimization algorithms for the welded beam design problem.

Algorithm	Best	*Ave*	Worst	*Std*
CSA	1.724852	1.724853	1.724854	2.235211E−05
SHO	1.725661	1.725828	1.726064	0.000287
GWO	1.726995	1.727128	1.727564	0.001157
PSO	1.820395	2.230310	3.048231	0.324525
MFO	1.732541	1.775231	1.802364	0.012397
MVO	1.725472	1.729680	1.741651	0.004866
SCA	1.759173	1.817657	1.873408	0.027543
GSA	2.172858	2.544239	3.003657	0.255859
GA	1.873971	2.119240	2.320125	0.034820
HS	1.836250	1.363527	2.035247	0.139485

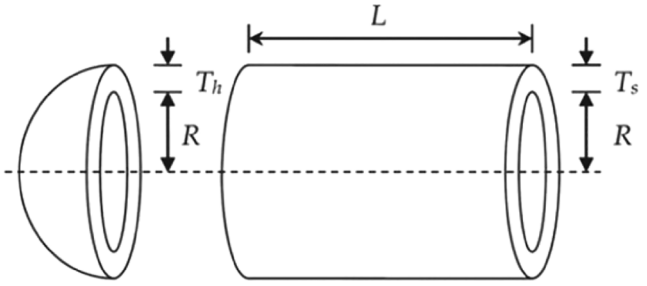

Fig. 7. A schematic structure of the cross-section of pressure vessel design problem.

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}, \sigma(\vec{x}) = \frac{6PL}{x_4x_3^2}$$

$$\delta(\vec{x}) = \frac{4PL^3}{Ex_4x_3^3}, P_c(\vec{x}) = \frac{4.013\sqrt{EGx_3^2x_4^6/36}}{L^2}\left(1 - \frac{x^3}{2L}\sqrt{\frac{E}{4G}}\right)$$

where $P = 6000lb$, $L = 14$in, $\delta_{max} = 0.25$ inch, $E = 30 * 10^6$ psi, $G = 12 * 10^6$ psi, $\delta_{max} = 13600psi$, $\sigma_{max} = 30000$ psi.

The range of the parameters h, l, t and b were taken as $0.1 \leqslant x_1 \leqslant 2$, $0.1 \leqslant x_2 \leqslant 10$, $0.1 \leqslant x_3 \leqslant 10$, and $0.1 \leqslant x_4 \leqslant 2$, respectively.

Some of the optimization methods formerly applied to solve this optimization problem include SHO (Dhiman & Kumar, 2017), GWO (Mirjalili et al., 2014), PSO (Kennedy & Eberhart, 1995), MFO (Mirjalili, 2015), MVO (Mirjalili et al., 2016), SCA (Mirjalili, 2016), GSA (Rashedi et al., 2009), GA (Bonabeau et al., 1999) and HS (Geem et al., 2001). A comparison of the optimal solutions reached by CSA and those aforementioned algorithms is presented in Table 19.

The results in Table 19 point out that the proposed CSA located the best design for the welded beam structure by finding the optimum cost of approximately 1.724852, which is the lowest cost among all other algorithms. Table 20 shows the statistical comparison of CSA and other algorithms after 30 independent runs in terms of the best result, worst result, average result and standard deviation result.

The results of Table 20 indicate that CSA outperformed other algorithms with lowest average and standard deviation scores as compared to others.

Table 22
Statistical results of CSA and other optimization algorithms for the pressure vessel design problem.

Algorithm	Best	*Ave*	Worst	*Std*
CSA	5885.3327	5886.7022	5887.0112	2.2501
SHO	5885.5773	5887.4441	5892.3205	002.893‘
GWO	5889.3689	5891.5247	5894.6238	013.9103
PSO	5891.3879	6531.5032	7394.5879	534.1195
MFO	6055.6378	6360.6854	7023.8521	365.5971
MVO	6011.5148	6477.3050	7250.9170	327.0074
SCA	6137.3724	6326.7606	6512.3541	126.6094
GSA	11550.2976	23342.2909	33226.2526	5790.6251
GA	5890.3279	6264.0053	7005.7500	496.1288
HS	6550.0230	6643.9870	8005.4397	657.5237

5.3. Pressure vessel design problem

This engineering problem is one of the widely used design benchmark tests (Kannan & Kramer, 1994). The goal of this problem is to reduce the overall cost of materials, formation and welding of the cylindrical pressure vessel reinforced at both ends by hemispherical heads as shown in Fig. 7.

The optimization variables of this problem are four which can be described as follows: thickness of the shell (T_s), thickness of the head (T_h), inner radius (R) and length of the cylindrical section of the vessel without looking at the head (L). These variables can be represented by a vector as: $\vec{x} = [x_1, x_2, x_3, x_4]$, where x_1, x_2, x_3 and x_4 represent T_s, T_h, R and L, respectively. The mathematical formula for this design problem can be specified as follows:

Minimize the function: $f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$.

This problem is subject to four constraints as shown below:

$$g_1(\vec{x}) = -x_1 + 0.0193x_3 \leqslant 0$$

$$g_2(\vec{x}) = -x_2 + 0.00954x_3 \leqslant 0$$

$$g_3(\vec{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leqslant 0$$

$$g4(\vec{x}) = x_4 - 240 \leqslant 0$$

where $0 \leqslant x1 \leqslant 99$, $0 \leqslant x_2 \leqslant 99$, $10 \leqslant x_3 \leqslant 200$ and $10 \leqslant x_4 \leqslant 200$.

The pressure vessel design problem is one of the most common optimization problems that researchers have used in numerous studies to confirm the efficacy of their new optimization algorithms. Table 21

Table 21
Optimization results of the pressure vessel design optimization problem obtained by CSA and other optimization algorithms.

Algorithm	Optimal values for variables				Optimum cost
	T_s	T_h	R	L	
CSA	12.450698	6.154387	40.319619	200.000000	**5885.3327**
SHO	0.778210	0.384889	40.315040	200.000000	5885.5773
GWO	0.779035	0.384660	40.327793	199.650290	5889.3689
PSO	0.778961	0.384683	40.320913	200.000000	5891.3879
MFO	0.835241	0.409854	43.578621	152.215200	6055.6378
MVO	0.845719	0.418564	43.816270	156.381640	6011.5148
SCA	0.817577	0.417932	41.749390	183.572700	6137.3724
GSA	1.085800	0.949614	49.345231	169.487410	11550.2976
GA	0.752362	0.399540	40.452514	198.002680	5890.3279
HS	1.099523	0.906579	44.456397	179.658870	6550.0230

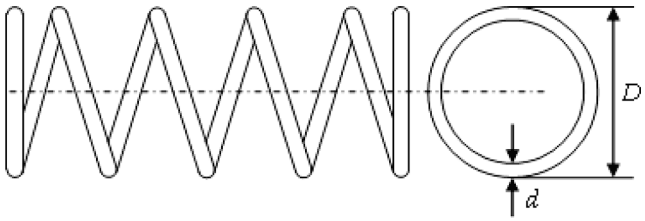

Fig. 8. A schematic structural diagram of a tension/compression spring.

Table 23
Optimization results of the tension/compression spring design problem obtained by CSA and other algorithms.

Algorithm	Optimum variables			Optimum cost
	d	D	N	
CSA	0.051778	0.358851	11.164981	**0.012665370**
SHO	0.051144	0.343751	12.095500	0.012674000
GWO	0.050178	0.341541	12.073490	0.012678321
PSO	0.050000	0.310414	15.000000	0.013192580
MFO	0.050000	0.313501	14.032790	0.012753902
MVO	0.050000	0.315956	14.226230	0.012816930
SCA	0.050780	0.334779	12.722690	0.012709667
GSA	0.050000	0.317312	14.228670	0.012873881
GA	0.050100	0.310111	14.000000	0.013036251
HS	0.050250	0.316351	15.239600	0.012776352

presents a comparison of the optimal results obtained for the pressure vessel design problem by CSA and the other algorithms mentioned above.

According to the cost results obtained for the pressure vessel design problem in Table 21, CSA reported the lowest cost of 5885.3327. A comparison of the statistical results for the pressure vessel design problem over 30 independent runs is shown in Table 22.

It may be observed from Table 22 that CSA exceeded other algorithms and presented very competitive results in terms of *Ave* and *Std* values compared to other algorithms.

5.4. Tension/compression spring design problem

Another popular engineering problem is the design of a tension/compression spring with a structure shown in Fig. 8.

The goal of this optimization problem is to lessen the weight of a tension/compression spring design. This problem is subject to few constraints, including shear stress, surge frequency and minimum deflection. The parameters in this design problem are: diameter of the wire (d), diameter of the mean coil (D) and number of active coils (N). The parameters of this problem were represented by a vector as: $\vec{x} = [x_1, x_2, x_3]$, where x_1, x_2 and x_3 identify the parameters d, D and N, respectively. The mathematical formula for this design problem is defined as follows:

Minimize the objective function: $f(\vec{x}) = (x_3 + 2)x_2x_1^2$.

This problem is subject to the following constraints:

$$g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leqslant 0$$

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leqslant 0$$

$$g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leqslant 0$$

$$g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \leqslant 0$$

where $0.05 \leqslant x_1 \leqslant 2.0$, $0.25 \leqslant x_2 \leqslant 1.3$ and $2 \leqslant x_3 \leqslant 15.0$.

Table 24
Statistical results of different optimization algorithms for the tension/compression spring design problem.

Algorithm	Best	*Ave*	Worst	*Std*
CSA	0.012667428	0.012669345	0.012672344	1.232107E−03
SHO	0.012674000	0.012684106	0.012715185	0.000027
GWO	0.012678321	0.012697116	0.012720757	0.000041
PSO	0.013192580	0.014817181	0.017862507	0.002272
MFO	0.012753902	0.014023657	0.017236590	0.001390
MVO	0.012816930	0.014464372	0.017839737	0.001622
SCA	0.012709667	0.012839637	0.012998448	0.000078
GSA	0.012873881	0.013438871	0.014211731	0.000287
GA	0.013036251	0.014036254	0.016251423	0.002073
HS	0.012776352	0.013069872	0.015214230	0.000375

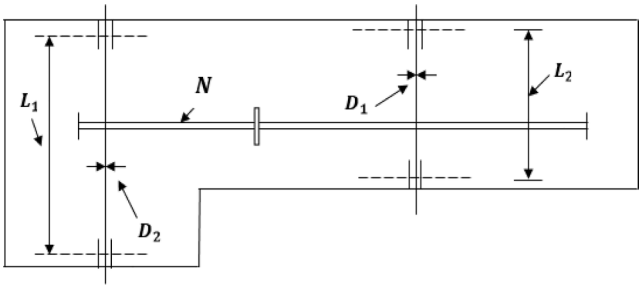

Fig. 9. A schematic diagram of a speed reducer design.

The tension/compression spring design problem has been extensively addressed by various optimization algorithms such as SHO, GWO, PSO, MFO, MVO, SCA, GSA, GA and HS. Table 23 shows a comparison between the proposed CSA and other competitors' algorithms in terms of design variables' values and cost values for the tension/compression spring design problem.

The results in Table 23 manifest that CSA is able to find the optimal design for this problem with a cost of 0.012665370. This cost is slightly less than the costs obtained by other algorithms. A summary of the statistical results of this design problem obtained by CSA and other algorithms is given in Table 24.

The results in Table 24 reveal that CSA again performs much better in statistical results by providing better results in terms of best, average, worst and standard deviation compared to other optimization methods.

5.5. Speed reducer design problem

The speed reducer design, with the structural diagram shown in Fig. 9, is another practical example often used as a benchmark case for testing optimization algorithms. This is a difficult design problem since it is correlated with seven variables (Gandomi & Yang, 2011).

The weight to be reduced in this design problem is subject to some constraints (Mezura-Montes & Coello, 2005), explained as follows: bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shafts. The design variables of this optimization problem are: b, m, z, l_1, l_2, d_1 and d_2. These variables are defined as the face width, module of teeth, number of teeth in the pinion, length of the first shaft between bearings, length of the second shaft between bearings, the diameter of the first shafts and the diameter of second shafts, respectively. They were represented by a vector while solving this problem, such as $\vec{x} = [x_1x_2x_3x_4x_5x_6x_7]$. The mathematical formula for this problem is:

$$f(\vec{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

Table 25
Optimization results of the speed reducer design problem obtained by CSA and other optimization algorithms.

Algorithm	Optimum variables							Optimum cost
	b	m	z	l_1	l_2	d_1	d_2	
CSA	3.500	0.7	17	7.3	7.715320	3.350215	5.286654	**2994.4710**
SHO	3.50159	0.7	17	7.3	7.815454	3.351272	5.288745	2998.5507
GWO	3.506690	0.7	17	7.3	7.815726	3.357847	5.286768	3001.2883
PSO	3.500019	0.7	17	8.3	7.803454	3.352412	5.286715	3005.7631
MFO	3.507524	0.7	17	7.3	7.802364	3.323541	5.287524	3009.5712
MVO	3.508502	0.7	17	7.3	7.816034	3.358073	5.286777	3002.9281
SCA	3.508755	0.7	17	7.3	7.814353	3.461020	5.289213	3030.5636
GSA	3.600	0.7	17	8.3	7.802442	3.369658	5.289224	3051.1209
GA	3.510253	0.7	17	8.3	7.818452	3.362201	5.287723	3067.5611
HS	3.520124	0.7	17	8.3	7.802354	3.366970	5.288719	3029.0020

Subject to the following constraints:

$$g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3} - 1 \leqslant 0$$

$$g_2(\vec{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leqslant 0$$

$$g_3(\vec{x}) = \frac{1.9x_4^3}{x_2x_6^4x_3} - 1 \leqslant 0$$

$$g_4(\vec{x}) = \frac{1.93x_5^3}{x_2x_7^4x_3} - 1 \leqslant 0$$

$$g_5(\vec{x}) = \frac{[(745(x_4/x_2x_3))^2 + 16.9 \times 10^6]^{1/2}}{110x_6^3} - 1 \leqslant 0$$

$$g_6(\vec{x}) = \frac{[(745(x_5/x_2x_3))^2 + 157.5 \times 10^6]^{1/2}}{85x_7^3} - 1 \leqslant 0$$

$$g_7(\vec{x}) = \frac{x_2x_3}{40} - 1 \leqslant 0$$

$$g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \leqslant 0$$

$$g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \leqslant 0$$

$$g_{10}(\vec{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leqslant 0$$

$$g_{11}(\vec{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leqslant 0$$

where the range of the design variables b, m, z, l_1, l_2, d_1 and d_2 were given as $2.6 \leqslant x_1 \leqslant 3.6$, $0.7 \leqslant x_2 \leqslant 0.8$, $17 \leqslant x_3 \leqslant 28$, $7.3 \leqslant x_4 \leqslant 8.3$, $7.3 \leqslant x_5 \leqslant 8.3$, $2.9 \leqslant x_6 \leqslant 3.9$ and $5.0 \leqslant x_4 \leqslant 5.5$, respectively.

Table 25 reveals a comparison of the best solutions that CSA has come up with and other optimization algorithms for the speed reducer design problem.

As shown in Table 25, CSA outperformed other algorithms by having the minimum cost of around 2994.4710, which indicates that CSA is able to find the optimum design for this design problem. A summary of the statistical results of CSA and other nine optimization methods for this optimization problem is given in Table 26.

Based on the results in Table 26, CSA located the best optimal solutions among other algorithms. This clarifies that CSA is superior to other algorithms in terms of these statistical results.

Table 26
Statistical results of CSA and other optimization algorithms for the speed reducer design problem.

Algorithm	Best	*Ave*	Worst	*Std*
CSA	2994.4710	2995.1210	2998.0923	2.9012E−02
SHO	2998.5507	2999.6400	3003.8892	1.9319
GWO	3001.2882	3005.8451	3008.7524	5.8379
PSO	3005.7632	3105.2525	3211.1749	79.6381
MFO	3009.5717	3021.2565	3054.5248	11.0235
MVO	3002.9281	3028.8411	3060.9582	13.0186
SCA	3030.5633	3065.9172	3104.7791	18.0742
GSA	3051.1205	3170.3347	3363.8735	92.5726
GA	3067.5617	3186.5236	3313.1992	17.1186
HS	3029.0025	3295.3296	3619.4651	57.0235

5.6. *Rolling element bearing design problem*

The main goal of this design problem is to maximize the dynamic load carrying power of a rolling element bearing with a schematic view shown in Fig. 10 (Dhiman & Kumar, 2017).

This problem consists of ten decision variables defined as ball

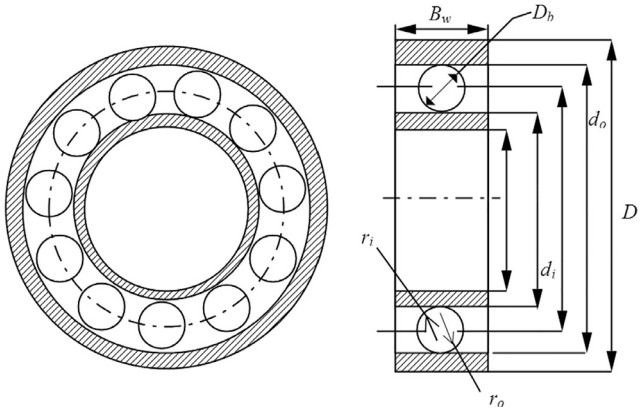

Fig. 10. A schematic diagram of a rolling element bearing.

Table 27
Optimization results of the rolling element bearing design problem obtained by CSA and other algorithms.

Algorithm	Optimum variables										Optimum cost
	D_m	D_b	X	f_i	f_o	K_{Dmin}	K_{Dmax}	$\in$	e	ζ	
CSA	125	21.418	11.356	0.515	0.515	0.4	0.700	0.3	0.02	0.612	**85201.641**
SHO	125	21.407	10.932	0.515	0.515	0.4	0.700	0.3	0.02	0.600	85054.532
GWO	125	21.351	10.987	0.515	0.515	0.5	0.688	0.3	0.03	0.627	84807.111
PSO	125	20.753	11.173	0.515	0.515	0.5	0.615	0.3	0.05	0.600	81691.202
MFO	125	21.032	10.965	0.515	0.515	0.5	0.675	0.3	0.02	0.610	84002.524
MVO	125	21.322	10.973	0.515	0.515	0.5	0.687	0.3	0.03	0.610	84491.266
SCA	125	21.148	10.969	0.515	0.515	0.5	0.700	0.3	0.02	0.629	83431.117
GSA	125	20.854	11.149	0.515	0.517	0.5	0.618	0.3	0.02	0.624	82276.941
GA	125	20.775	11.012	0.515	0.515	0.5	0.613	0.3	0.05	0.610	82773.982
HS	125	20.871	11.166	0.515	0.516	0.5	0.619	0.3	0.05	0.614	81569.527

diameter, D_b, pitch diameter, D_m, inner, f_i, and outer, f_o, raceway curvature factors, number of balls, $X, K_{Dmin}, K_{Dmax}, e, \in$ and ζ. The mathematical formulation for this problem is:

Maximize $C_d = f_c X^{2/3} D_b^{1.8}$ *if* $D \leqslant 25.4\ mm$.

$$C_d = 3.647 f_c X^{2/3} D_b^{1.4} \text{ if } D > 25.4\ mm$$

The constraints and f_c for this design problem are as follows:

$$g_1\left(\vec{x}\right) = \frac{\phi_0}{2sin^{-1}\left(D_b/D_m\right)} - X + 1 \geq 0$$

$$g_2(\vec{x}) = 2D_b - K_{Dmin}(D-d) \geqslant 0$$

$$g_3(\vec{x}) = K_{Dmax}(D-d) - 2D_b \geqslant 0$$

$$g_4(\vec{x}) = \zeta B_w - D_b \leqslant 0$$

$$g_5(\vec{x}) = D_m - 0.5(D+d) \geqslant 0$$

$$g_6(\vec{x}) = (0.5+e)(D+d) - D_m \geqslant 0$$

$$g_7(\vec{x}) = 0.5(D - D_m - D_b) - \in D_b \geqslant 0$$

$$g_8(\vec{x}) = f_i \geqslant 0.515$$

$$g_9(\vec{x}) = f_o \geqslant 0.515$$

Table 28
Statistical results of different optimization algorithms for the rolling element bearing design problem.

Algorithm	Best	*Ave*	Worst	*Std*
CSA	85201.641	85157.112	850945.807	117.123
SHO	85054.532	85024.858	85853.876	186.682
GWO	84807.111	84791.613	84517.923	137.186
PSO	81691.202	50435.017	32761.546	13962.150
MFO	84002.524	82357.493	83979.254	1401.524
MVO	84491.266	84353.685	84100.834	392.431
SCA	83431.117	81005.232	77992.482	1710.777
GSA	82276.941	78002.107	71043.110	3119.904
GA	82773.982	81198.753	80687.239	1679.367
HS	81569.527	80397.998	79412.779	1756.902

$$\phi_0 = 2\pi - 2cos^{-1}\left(\frac{x}{y}\right)$$

$$\gamma = \frac{D_b}{D_m},\ f_i = \frac{r_i}{D_b},\ f_o = \frac{r_o}{D_b}, T = D - d - 2D_b$$

$$D = 160,\ d = 90,\ B_w = 30,\ r_i = r_o = 11.033$$

$$0.5(D+d) \leqslant D_m \leqslant 0.6(D+d),\ 0.15(D-d) \leqslant D_b \leqslant 0.45(D-d)$$

$$4 \leqslant X \leqslant 50,\ 0.515 \leqslant f_i \text{ and } f_o \leqslant 0.6.$$

$$0.4 \leqslant K_{Dmin} \leqslant 0.5,\ 0.6 \leqslant K_{Dmax} \leqslant 0.7$$

$$0.3 \leqslant e \leqslant 0.4,\ 0.02 \leqslant e \leqslant 0.1,\ 0.6 \leqslant \zeta \leqslant 0.85$$

A comparison of the best optimization solutions for the rolling

$$\begin{aligned} f_c &= 37.91\left[1 + \left\{1.04\left(\frac{1-\gamma}{1+\gamma}\right)^{1.72}\left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)}\right)^{0.41}\right\}^{10/3}\right]^{-0.3} \times \left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{1/3}}\right]\left[\frac{2f_i}{2f_i-1}\right]^{0.41} \\ x &= \left\{\frac{(D-d)}{2} - 3\frac{T}{4}\right\}^2 + \left\{\frac{D}{2} - \frac{T}{4} - D_b\right\}^2 \\ &\quad - \left\{\frac{d}{2} + \frac{T}{4}\right\}^2 \\ y &= 2\left\{\frac{(D-d)}{2} - 3\frac{T}{4}\right\}\left\{\frac{D}{2} - \frac{T}{4} - D_b\right\} \end{aligned} \tag{23}$$

element bearing design problem arrived at by CSA and other algorithms is presented in Table 27.

As per the optimum costs reported in Table 27, CSA came up with the best design for this design problem with a cost of about 85201.641. The statistical results of CSA and other optimization methods such as SHO, GWO, PSO, MFO, MVO, SCA, GSA, GA and HS for the rolling element bearing design problem are shown in Table 28.

When reading the results in Table 28, the proposed optimization method obtained the best statistical solutions for the rolling element bearing design problem over other optimization methods.

In short, CSA, as a new meta-heuristic-based optimization algorithm, has confirmed its reliability and efficiency in addressing several engineering design problems over other well-known optimization algorithms. Therefore, we can conclude that CSA is a suitable optimization algorithm and it is definitely a promising candidate for solving real-world contemporary problems.

6. Conclusion and future works

This paper presented a novel swarm intelligence optimization algorithm, referred to as Chameleon Swarm Algorithm (CSA), to solve optimization problems. The key inspiration for this newly developed algorithm is inspired from the hunting behavior of chameleons in nature. This behavior is studied and modeled mathematically, including each feature of chameleon's search for prey to achieve the desired optimization. The best solution of CSA, come into possessed so far, is considered the best global position of prey entrapped by a chameleon. A series of experiments was conducted to prove the performance of the proposed algorithm, analyze the exploration power, exploitation strength and local optima avoidance through solving a set of 67 well-known optimization tasks. These tasks consist of a set of 23 basic unimodal and multimodal functions, CEC-2015 test suite with 15 test functions and CEC-2017 test suite with 29 functions. In addition, the efficacy of CSA was also illustrated by its application to five engineering design problems. From the experimental results, the proposed algorithm offers a root framework for relatively low-dimension optimization issues, which may be extended to corroborate its credibility to solve very large-scale optimization problems. Further study is needed to develop a binary version of CSA which could be a valuable contribution to solving real-life problems with different search areas and various types of constraints. Moreover, it may be possible to broaden the applications of CSA to solve multi-objective problems in various fields.

Credit Author Statement

Malik Shehadeh Braik is the corresponding and only author for this work. He is responsible for ensuring that the descriptions are all accurate. He proposed and developed the mathematical models of the proposed approach. The author also designed the experiments, and executed the programs and experimental scenarios. Malik wrote the entire paper: the literature, the mathematical models, the diagrams, the Tables, the pseudo-code of the proposed algorithm, the statistical tests and discussed all the experimental and statistical test results. Finally, the corresponding and only author (Malik) checked the validation of the results and checked the references. He also examined the technical concepts in the paper, verified the readability of the paper, checked the grammar and supported it in proof of English.

Declaration of Competing Interest

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

Acknowledgment

We would like to acknowledge the anonymous reviewers for the valuable comments that improved the quality of this paper. Moreover, we would also like to thank the Editors for their generous comments and support during the review process.

Appendix A. Unimodal, multimodal and fixed-dimension multimodal test functions

A detailed description of the unimodal benchmark functions (F_1–F_7), multimodal benchmark functions (F_8–F_{13}) and fixed-dimension multimodal benchmark functions (F_{14}–F_{23}) is tabulated in Table A.1.

A description of the CEC-2015 benchmark test functions is described in Table A.2.

A description of the CEC-BC-2017 benchmark test functions is described in Table A.3.

Table A.1
Characteristics of the unimodal, multimodal and fixed-dimension multimodal test functions. U: Unimodal, M: Multimodal, and F: fixed-dimension multimodal.

Key	Function formulation	$f(x^*)$	Category	*Dim*	Range
f_1	$\sum_{i=1}^{d} x_i^2$	0	U	10	$x_i \in$ [-100,100]
f_2	$\sum_{i=1}^{d} \lvert x_i \rvert + \prod_{i=1}^{d} \lvert x_i \rvert$	0	U	30	$x_i \in$ [-10,10]
f_3	$\sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j \right)^2$	0	U	10	$x_i \in$ [-100,100]
f_4	$\max_i \{ \lvert x_i \rvert, 1 \leqslant i \leqslant d \}$	0	U	10	$x_i \in$ [-100,100]
f_5	$\sum_{i=1}^{d-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	0	U	10	$x_i \in$ [-30,30]
f_6	$\sum_{i=1}^{d} (\lvert x_i + 0.5 \rvert)^2$	0	U	10	$x_i \in$ [-100,100]
f_7	$\sum_{i=1}^{d} i x_i^4 + random[0,1)$	0	U	10	$x_i \in$ [-128,128]
f_8	$\sum_{i=1}^{d} -x_i \sin \left(\sqrt{\lvert x_i \rvert} \right)$	−418.9829E+5	M	10	$x_i \in$ [-500,500]
f_9	$\sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$	0	M	10	$x_i \in$ [-5.12,5.12]
f_{10}	$-20\exp\left(-0.2 \sqrt{\frac{1}{d} \sum_{i=1}^{d} x_i^2} \right) - \exp\left(\frac{1}{d} \sum_{i=1}^{d} cos(2\pi x_i) \right) + 20 + e$	0	M	10	$x_i \in$ [-32,32]
f_{11}	$\frac{1}{4000} \sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos \left(\frac{x_i}{\sqrt{i}} \right) + 1$	0	M	10	$x_i \in$ [-600,600]

(*continued on next page*)

Table A.1 (*continued*)

Key	Function formulation	$f(x^*)$	Category	*Dim*	Range
f_{12}	$\frac{\pi}{d}\left\{10\sin(\pi y_1)+\sum_{i=1}^{d-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_d-1)^2\right\}+\sum_{i=1}^{d}u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0-a & <x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	0	M	10	$x_i \in$ [-50,50]
f_{13}	$0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{d}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right]+(x_d-1)^2\left[1+\sin^2(2\pi x_d)\right]\right\}+\sum_{i=1}^{d}u(x_i,5,100,4)$	0	M	10	$x_i \in$ [-50,50]
f_{14}	$\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	1	F	2	$x_i \in [-65,65]$
f_{15}	$\sum_{i=1}^{11}\left[a_i-\frac{x_1\left(b_i^2+b_ix_2\right)}{b_i^2+b_ix_3+x_4}\right]^2$	0.0003	F	4	$x_i \in [-5,5]$
f_{16}	$4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	−1.0316	F	2	$x_i \in [-5,5]$
f_{17}	$\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	0.398	F	2	$x_i \in [-5,5]$
f_{18}	$\left[1+(x_1+x_2+1)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]$ $\times\left[30+(2x_1-3x_2)^2\times\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$	3	F	2	$x_i \in [-2,2]$
f_{19}	$-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}\left(x_j-p_{ij}\right)^2\right)$	−3.86	F	3	$x_i \in [1,3]$
f_{20}	$-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}\left(x_j-p_{ij}\right)^2\right)$	−3.32	F	6	$x_i \in [0,1]$
f_{21}	$-\sum_{i=1}^{5}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	−10.1532	F	4	$x_i \in [0,10]$
f_{22}	$-\sum_{i=1}^{7}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	−10.4028	F	4	$x_i \in [0,10]$
f_{23}	$-\sum_{i=1}^{10}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	−10.5363	F	4	$x_i \in [0.10]$

Table A.2
A description of the IEEE CEC-2015 benchmark test functions.

Function	Function name	Related basic functions	*Dim*	f_{min}
C-f1	Rotated Bent Cigar Function	Bent Cigar Function	30	100
C-f2	Rotated Discus Function	Discus Function	30	200
C-f3	Shifted and Rotated Weierstrass Function	Weierstrass Function	30	300
C-f4	Shifted and Rotated Schwefel's Function	Schwefel's Function	30	400
C-f5	Shifted and Rotated Katsuura Function	Katsuura Function	30	500
C-f6	Shifted and Rotated HappyCat Function	HappyCat Function	30	600
C-f7	Shifted and Rotated HGBat Function	HGBat Function	30	700
C-f8	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	Griewank's Function Rosenbrock's Function	30	800
C-f9	Shifted and Rotated Expanded Scaffer's F6 Function	Expanded Scaffer's F6 Function	30	900
C-f10	Hybrid Function 1 (Three functions)	Schwefel's Function Rastrigin's Function High Conditioned Elliptic Function	30	1000
C-f11	Hybrid Function 2 (Four functions)	Griewank's Function Weierstrass Function Rosenbrock's Function Scaffer's F6 Function	30	1100
C-f12	Hybrid Function 3 (Five functions)	Katsuura Function HappyCat Function Expanded Griewank's with Rosenbrock's Function Schwefel's Function Ackley's Function	30	1200
C-f13	Composition Function 1 (Five functions)	Rosenbrock's Function High Conditioned Elliptic Function Bent Cigar Function Discus Function High Conditioned Elliptic Function	30	1300
C-f14	Composition Function 2 (Three functions)	Schwefel's Function Rastrigin's Function High Conditioned Elliptic Function	30	1400
C-f15	Composition Function 3 (Five functions)	HGBat Function Rastrigin's Function Schwefel's Function Weierstrass Function High Conditioned Elliptic Function	30	1500

Table A.3
Characteristics of the CEC-BC-2017 benchmark test functions: U: Unimodal, M: Multimodal, H: Hybrid, and C: Composition.

Function	Function name	*Dim*	f_{min}	Type
C-f1	Shifted and Rotated Bent Cigar function	10	100	U
C-f3	Shifted and Rotated Zakharov function	10	300	U
C-f4	Shifted and Rotated Rosenbrock's function	10	400	M
C-f5	Shifted and Rotated Rastrigin's function	10	500	M
C-f6	Shifted and Rotated Expanded Scaffer's function	10	600	M
C-f7	Shifted and Rotated Lunacek Bi-Rastrigin function	10	700	M
C-f8	Shifted and Rotated Non-Continuous Rastrigin's function	10	800	M
C-f9	Shifted and Rotated Levy Function	10	900	M
C-f10	Shifted and Rotated Schwefel's Function	10	1000	M
C-f11	Hybrid function of Zakharov, Rosenbrock and Rastrigin's	10	1100	H
C-f12	Hybrid function of High Conditioned Elliptic, Modified Schwefel and Bent Cigar	10	1200	H
C-f13	Hybrid function of Bent Ciagr, Rosenbrock and Lunache Bi-Rastrigin	10	1300	H
C-f14	Hybrid function of Eliptic, Ackley, Schaffer and Rastrigin	10	1400	H
C-f15	Hybrid function of Bent Cigar, HGBat, Rastrigin and Rosenbrock	10	1500	H
C-f16	Hybrid function of Expanded Schaffer, HGBat, Rosenbrock and Modified Schwefel	10	1600	H
C-f17	Hybrid function of Katsuura, Ackley, Expanded Griewank plus Rosenbrock, Modified Schwefel and Rastrigin	10	1700	H
C-f18	Hybrid function of high conditioned Elliptic, Ackley, Rastrigin, HGBat and Discus	10	1800	H
C-f19	Hybrid function of Bent Cigar, Rastrigin, Expanded Grienwank plus Rosenbrock, Weierstrass and expanded Schaffer	10	1900	H
C-f20	Hybrid function of Happycat, Katsuura, Ackley, Rastrigin, Modified Schwefel and Schaffer	10	2000	H
C-f21	Composition of Rosenbrock, High Conditioned Elliptic and Rastrigin	10	2100	C
C-f22	Composition of Rastrigin's, Griewank's and Modifed Schwefel's	10	2200	C
C-f23	Composition of Rosenbrock, Ackley, Modified Schwefel and Rastrigin	10	2300	C
C-f24	Composition of Ackley, High Conditioned Elliptic, Girewank and Rastrigin	10	2400	C
C-f25	Composition of Rastrigin, Happycat, Ackley, Discus and Rosenbrock	10	2500	C
C-f26	Composition of Expanded Scaffer, Modified Schwefel, Griewank, Rosenbrock and Rastrigin	10	2600	C
C-f27	Composition of HGBat, Rastrigin, Modified Schwefel, Bent-Cigar, High Conditioned Elliptic and Expanded Scaffer	10	2700	C
C-f28	Composition function of Ackley, Griewank, Discus, Rosenbrock, HappyCat, Expanded Scaffer	10	2800	C
C-f29	Composition function of shifted and rotated Rastrigin, Expanded Scaffer and Lunacek Bi-Rastrigin	10	2900	C
C-f30	Composition function of shifted and rotated Rastrigin, Non-Continuous Rastrigin and Levy function	10	3000	C

References

Awad, N. H., Ali, M. Z. & Suganthan, P. N. (2017). Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving cec2017 benchmark problems. In *2017 IEEE congress on evolutionary computation (cec)* (pp. 372–379).

Bonabeau, E., Dorigo, M., Marco, D. d. R.D.F., Theraulaz, G., & Théraulaz, G. (1999). Swarm intelligence: From natural to artificial systems (No 1). Oxford university press.

Braik, M., Al-Zoubi, H., & Al-Hiary, H. (2020). Artificial neural networks training via bio-inspired optimisation algorithms: Modelling industrial winding process, case study. *Soft Computing*, 1–25.

Braik, M., Sheta, A., & Al-Hiary, H. (2020). A novel meta-heuristic search algorithm for solving optimization problems: Capuchin search algorithm. *Neural Computing and Applications*, 1–33.

Braik, M., Sheta, A., Turabieh, H., & Alhiary, H. (2020). A novel lifetime scheme for enhancing the convergence performance of salp swarm algorithm. *Soft Computing*, 1–26.

Chen, Q., Liu, B., Zhang, Q., Liang, J., Suganthan, P., & Qu, B. (2014). Problem definitions and evaluation criteria for cec 2015 special session on bound constrained single-objective computationally expensive numerical optimization. Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Technical Report, Nanyang Technological University.

Chumburidze, M., Basheleishvili, I., & Khetsuriani, A. (2019). Dynamic programming and greedy algorithm strategy for solving several classes of graph optimization problems. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience, 10*(1), 101–107.

Das, S., & Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation, 15*(1), 4–31.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

Dhiman, G., & Kumar, V. (2017). Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Advances in Engineering Software, 114*, 48–70.

Dhiman, G., & Kumar, V. (2019). Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. *Knowledge-Based Systems, 165*, 169–196.

Digalakis, J. G., & Margaritis, K. G. (2001). On benchmarking functions for genetic algorithms. *International Journal of Computer Mathematics, 77*(4), 481–506.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science, 1995. mhs'95* (pp. 39–43).

Faramarzi, A., & Afshar, M. (2012). Application of cellular automata to size and topology optimization of truss structures. *Scientia Iranica, 19*(3), 373–380.

Faramarzi, A., Heidarinejad, M., Stephens, B., & Mirjalili, S. (2020). Equilibrium optimizer: A novel optimization algorithm. *Knowledge-Based Systems, 191*, Article 105190.

Gandomi, A. H., & Yang, X. S. (2011). *Computational optimization, methods and algorithms (259–281)*. Springer.

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

Glaw, F. (2015). Taxonomic checklist of chameleons (Squamata: Chamaeleonidae) Taxonomic checklist of chameleons (squamata: Chamaeleonidae). *Vertebrate Zoology, 65*(2), 167–246.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning, 3*(2), 95–99.

Herrel, A., Meyers, J., Aerts, P., & Nishikawa, K. C. (2000). The mechanics of prey prehension in chameleons. *Journal of Experimental Biology, 203*(21), 3255–3263.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 65–70.

Jain, M., Singh, V., & Rani, A. (2019). A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm and Evolutionary Computation, 44*, 148–175.

Kannan, B., & Kramer, S. N. (1994). An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design, 116*(2), 405–411.

Karypis, G., Han, E. H., & Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. *Computer, 32*(8), 68–75.

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

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of icnn'95-international conference on neural networks, 4* (pp. 1942–1948).

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

Koppen, M., Wolpert, D. H., & Macready, W. G. (2001). Remarks on a recent paper on the no free lunch theorems. *IEEE Transactions on Evolutionary Computation, 53*, 295–296.

Sree Ranjini, K. S., & Murugan, S. (2017). Memory based hybrid dragonfly algorithm for numerical optimization problems. *Expert Systems with Applications, 83*, 63–78.

Liu, S. H., Mernik, M., Hrnčič, D., & Črepinšek, M. (2013). A parameter control method of evolutionary algorithms using exploration and exploitation measures with a practical application for fitting sovova's mass transfer model. *Applied Soft Computing, 13*(9), 3792–3805.

Luenberger, D. G., & Ye, Y. (1984). *Linear and nonlinear programming (2)*. Springer.

Meraihi, Y., Gabis, A. B., Ramdane-Cherif, A., & Acheli, D. (2020). A comprehensive survey of Crow Search Algorithm and its applications. *Artificial Intelligence Review*, 1–48.

Mezura-Montes, E., & Coello, C. A. C. (2005). Useful infeasible solutions in engineering optimization with evolutionary algorithms. In *Mexican international conference on artificial intelligence* (pp. 652–662).

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

Mirjalili, S. (2016). Sca: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems, 96*, 120–133.

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

Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Computing and Applications, 27*(2), 495–513.
Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.
Mlinarić, D., Perić, T., & Matejaš, J. (2019). Multi-objective programming methodology for solving economic diplomacy resource allocation problem. *Croatian Operational Research Review*, 165–174.
Pereira, D. G., Afonso, A., & Medeiros, F. M. (2015). Overview of friedman's test and post-hoc analysis. *Communications in Statistics-Simulation and Computation, 44*(10), 2636–2653.
Qi, Y., Jin, L., Wang, Y., Xiao, L., & Zhang, J. (2019). Complex-valued discrete-time neural dynamics for perturbed time-dependent complex quadratic programming with applications. *IEEE Transactions on Neural Networks and Learning Systems*.
Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). Gsa: A gravitational search algorithm. *Information Sciences, 179*(13), 2232–2248.
Rechenberg, I. (1973). Evolution strategy: Optimization of technical systems by means of biological evolution. *Fromman-Holzboog, Stuttgart, 104*, 15–16.
Rodríguez, N., Gupta, A., Zabala, P. L., & Cabrera-Guerrero, G. (2018). Optimization algorithms combining (meta) heuristics and mathematical programming and its application in engineering. *Mathematical Problems in Engineering, 2018*.
Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design optimization.
Sheta, A., Braik, M., & Al-Hiary, H. (2019). Modeling the tennessee eastman chemical process reactor using bio-inspired feedforward neural network (bi-ff-nn). *The International Journal of Advanced Manufacturing Technology, 103*(1–4), 1359–1380.
Uymaz, S. A., Tezel, G., & Yel, E. (2015). Artificial algae algorithm (aaa) for nonlinear global optimization. *Applied Soft Computing, 31*, 153–171.
Wang, G. G., Guo, L., Gandomi, A. H., Hao, G. S., & Wang, H. (2014). Chaotic krill herd algorithm. *Information Sciences, 274*, 17–34.
Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67–82.
Yang, X. S. (2010). *Firefly algorithm, stochastic test functions and design optimisation*. arXiv preprint arXiv:1003.1409.
Yang, X. S. (2010b). *Nature-inspired metaheuristic algorithms*. Luniver press.
Yang, X. S., Deb, S., Loomes, M., & Karamanoglu, M. (2013). A framework for self-tuning optimization algorithm. *Neural Computing and Applications, 23*(7–8), 2051–2057.
Young, E. (2008). Chameleons fine-tune camouflage to predator's vision. *New Scientist, 21*.
Zhan, Z. H., Zhang, J., Li, Y., & Chung, H. S. H. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39*(6), 1362–1381.