Applied Intelligence (2023) 53:26840–26884
https://doi.org/10.1007/s10489-023-04732-z

Improved versions of crow search algorithm for solving global numerical optimization problems

Alaa Sheta[1,5] · Malik Braik[2,5] · Heba Al-Hiary[3] · Seyedali Mirjalili[4,5]

Accepted: 25 May 2023 / Published online: 30 August 2023

Abstract
Over recent decades, research in Artificial Intelligence (AI) has developed a broad range of approaches and methods that can be utilized or adapted to address complex optimization problems. As real-world problems get increasingly complicated, this requires an effective optimization method. Various meta-heuristic algorithms have been developed and applied in the optimization domain. This paper used and ameliorated a promising meta-heuristic approach named Crow Search Algorithm (CSA) to address numerical optimization problems. Although CSA can efficiently optimize many problems, it needs more searchability and early convergence. Its positioning updating process was improved by supporting two adaptive parameters: flight length (fl) and awareness probability (AP) to tackle these curbs. This is to manage the exploration and exploitation conducts of CSA in the search space. This process takes advantage of the randomization of crows in CSA and the adoption of well-known growth functions. These functions were recognized as exponential, power, and S-shaped functions to develop three different improved versions of CSA, referred to as Exponential CSA (ECSA), Power CSA (PCSA), and S-shaped CSA (SCSA). In each of these variants, two different functions were used to amend the values of fl and AP. A new dominant parameter was added to the positioning updating process of these algorithms to enhance exploration and exploitation behaviors further. The reliability of the proposed algorithms was evaluated on 67 benchmark functions, and their performance was quantified using relevant assessment criteria. The functionality of these algorithms was illustrated by tackling four engineering design problems. A comparative study was made to explore the efficacy of the proposed algorithms over the standard one and other methods. Overall results showed that ECSA, PCSA, and SCSA have convincing merits with superior performance compared to the others.

Keywords Crow search algorithm · Optimization · Meta-heuristics · Engineering problems

1 Introduction

Numerous numerical optimization problems have grown more complex over the last few decades, demanding the utilization of quite efficacious optimization algorithms. For instance, accuracy problems in data mining and design cost problems in engineering often require finding the best solutions from many available ones without squandering efforts in looking for sub-optimal areas. Due to the highly non-convex landscapes and the knotty nature of many real-world problems, the search space associated with such issues poses many difficulties in optimization approaches [1]. These extraordinarily knotted modalities of the search space are typically proportionate to the problem size, and the increasing problem dimensionality [2]. Traditional deterministic methods, based on humble calculus rules, conduct a thorough search and cannot present effective solutions as heuristic methods enclosed by limited calculations resources [3]. Typically traditional optimization methods usually, in a large number of cases, presented potent procedures for obtaining the optimal global solutions for problems of humble or even extreme complexity [4]. They intelligibly lead to a fast and efficient optimization process when the optimization problem to be addressed has a simple design with few constraints and decision variables. These methods find it relatively complex to deliver fully efficient solutions for real-world problems of complex designs that have undergone many constraints [4]. They can only identify local optima for some intricate problems, as there is no sureness that the global optimum will constantly be found [5]. This puts a

✉ Seyedali Mirjalili
ali.mirjalili@gmail.com

Extended author information available on the last page of the article

powerful challenge to locate the global optimality of concern. To control such issues, researchers have focused their attention on meta-heuristic techniques. More information about meta-heuristics can be found later in the related works section. A broad range of meta-heuristics has recently been utilized in the literature to cover efficient optimization for various optimization problems. A meta-heuristic algorithm can act effectively for most kinds of problems. It has become evident in line with the 'no-free-lunch' (NFL) theory [6] that there is no general meta-heuristic that can handle all kinds of optimization problems in the best possible manner and can surpass all other meta-heuristics for every problem [7]. A successful meta-heuristic algorithm may be deemed to play well, or at least perform well, on most problems. A crucial problem that impedes researchers from achieving these goals is the sufficient equilibrium between exploration and exploitation conducts in the algorithm of interest. This balance cannot be identified in advance, where the proper balance between these aspects depends on the problem. Practically, some methods have better exploration and exploitation strategies than others. This opens the room for improvement of existing meta-heuristic processes to get a satisfactory balance between exploitation and exploration and then to get to sane solutions in most optimization problems. Simply put, the NFL theory inculcates this area of studies to stay unlocked and motivate researchers to improve the accuracy of present methods or introduce new ones to reinforce optimization with broader performance [8].

In this context, this study was motivated to enhance further a newly well-known meta-heuristic method, called Crow Search Algorithm (CSA) [9], to tackle a good pool of optimization problems. This is developed according to the ideology of ongoing amelioration to develop compelling solutions to real-life problems. CSA is one of the promising swarm intelligence techniques [9], devised about the intelligent behavior of crows while foraging in nature. Although CSA can get the optimum in solving various problems in the field of optimization [10], it usually may be trapped in local optima, particularly when faced with complex problems with several local optima. This may be attributed to its narrow search capability and slow convergence property. To give control over such hurdles, various strategies, and methods have been presented in the literature to promote the accuracy level of CSA further. In this, many variants of CSA were designed from the operators' perspective. In [11], CSA was improved by chaotic-based criteria to embed a chaotic local search to ameliorate the diversity of solutions. Due to the efficacy of chaos, chaos theory was amalgamated into CSA to explore its characteristics [12]. A conscious neighborhood method was employed in CSA to direct the locomotion of crows according to three search mechanisms. The enhanced CSA revealed good performance on benchmark test problems [13]. In [14], a chaos-based strategy was utilized to implement a chaotic local search for CS to hasten the convergence rate and mitigate the incidence of local optima. The chaotic mechanisms are potent to strengthen the search power of CSA under its stochasticity and ergodicity. The strategies of amalgamation of past experience and crafting a non-hideout position help compromise the exploration and exploitation potential of CSA in the search process. A niching method was integrated into CSA to manage the interaction among crows to empower CSA to locate multiple solutions for optimization problems [15]. In [16], each crow uses two update methods to come up with a better solution, where one method achieves intelligence among crows, and the other method assists crows to get out of the local optima. Also, in that work, awareness probability and flight length were adapted inside the iteration loops of CSA to establish a good balance between local and global search strategies to boost searchability. These operators efficiently amend crows' movements. Besides operators, various strategies were also presented in the literature to augment the performance of CSA in addressing various optimization problems. An opposition-based learning method was incorporated into the position updating process of CSA to explore solutions and direct the movements of crows [17, 18]. Kapur's entropy was incorporated with CSA to reinforce its population diversity and assist it in emerging from early convergence [19]. Promising adjusting methods for time-varying flight length based on crows' life convergence times were proposed to deal with the problem of rapid convergence of CSA. In the meantime, an adaptive flight length variable, in reliance on the present and maximum iteration values, improves the hunting ability of crows [20]. A spiral search mechanism was used to strengthen CSA by mitigating early premature convergence while solving numerical optimization problems [21]. Hybrid strategies that integrate several approaches also show outstanding performance in tackling the defects of CSA. CSA was combined with PSO to use both properties sufficiently to strengthen its exploration and exploitation processes [22]. An improved CSA was hybridized with a uniform crossover mechanism to enhance its exploration search capacity and convergence behavior [23]. Moreover, CSA has been fruitfully practiced for many optimization problems. In view of this, variants of CSA have continuous, binary, and mixed types. Continuous type with real-valued variables can address real problems [24], constrained problems [25], and multi-objective problems [18]. For continuous optimization problems, an optimization strategy was embedded into CSA to prop its performance degree in diagnosing Parkinson's disease for twenty benchmark test problems [26]. A binary CSA was presented for binary optimization problems to solve feature selection problems [20]. An integration between CSA and Grey Wolf Optimizer (GWO) was utilized to tackle feature selection problems in addition to unconstrained test functions [27]. For multi-objective optimization problems,

CSA was combined with the fruitfly optimization method to evolve combinatorial interaction test suites in the existence of constraints [28]. A hybrid CSA incorporated with fuzzy c-means and chaos theory has been applied to medical diagnosis problems [29]. Hybrid approaches of CSA with PSO were developed for feature selection, and global optimization problems [22, 30]. In [31], an arithmetic crossover is integrated with CSA to address engineering design problems. More variants and applications of CSA in addressing a broad scope of problems can be found in [10]. The above strategies and approaches have efficiently enhanced the search property of CSA to foster its performance in solving optimization problems. However, each of the revised approaches performed well in the applications deemed, albeit some may fall short of what is required in particular applications. Some of them have limited functionality and modest performance with complex optimization problems. There is still a necessity for more amelioration of CSA, particularly for complex real-world problems.

On the grounds of this, it is worth paying attention and investigating to improve CSA from the point of view of position updating strategy and adaptive strategies for its key parameters. A good positioning updating process with appropriate control parameters can be implemented for CSA to guide further the local and global search processes of crows in the environment. In this paper, an improved CSA with effective flight length and awareness probability is destined to handle the early convergence and modest search capability of CSA. First, the awareness probability of crows is expected to grow as a function of iterations. It is also anticipated that the more the flight length of a crow, the more likely it is to locate food. Thus, we present flight length as a descending function of time and the awareness probability as the rate of change of flight length. This is to foster the exploration and exploitation aptitudes of CSA. We adopted three functions of time for flight length and three functions for awareness probability to achieve this goal. These adaptable functions not only provide adequate guidance for crows in the environment, but they are also beneficial for mitigating the stagnation of CSA. These parametric functions are designed to equipoise the exploration and exploitation features in CSA. In this respect, three versions of CSA were developed, each using different growth functions to update the values of awareness probability and flight length in the course iterations of CSA. Each version adds a sensible enhancement to the original CSA. These versions are named Exponential CSA (ECSA), Power CSA (PCSA), and S-shaped CSA (SCSA). Then, we introduced an enhancement to the positioning updating process of these versions to manage the movement of crows further. The goal of this enhancement is to enable crows to scout and exploit every promising area in the search domain.

Lastly, a new parameter that can provide more exploration and exploitation abilities for these versions was proposed. This parameter can help the crows to explore diverse directions in the search space and exploit each search region to locate other crows' food. To assess the performance of the evolved ECSA, PCSA, and SCSA, many evaluation experiments were conducted to compare them with the basic CSA and many promising meta-heuristic algorithms on three test suites of broadly well-known benchmark functions. Finally, their practicality and practicability were demonstrated in solving four engineering design problems. The comparison findings between the developed algorithms and other rival ones indicate the notable performance of ECSA, PCSA, and SCSA, which implies that these algorithms are competitive and promising. In sum, the critical contributions of this work can be abridged as shown below:

1. Three population-based algorithms, namely ECSA, PCSA, and SCSA, were derived from the basic CSA.
2. A thorough evaluation comparison was conducted between the developed algorithms and other meta-heuristics on three benchmark groups: classical benchmark functions, CEC-2015, and CEC-2017 test suites.
3. The credibility and practicability of the proposed algorithms were investigated on four engineering design optimization problems.
4. The statistical importance and convergence behavior of the proposed algorithms were investigated.

The rest of this work is structured as follows: Section 2 reviews the state-of-the-art optimization problems and methods. The following Section 3 provides a thorough description of the original CSA. In Section 4, we present an elaborated description of the proposed versions of CSA. The evaluation and statistical outcomes of the proposed versions compared to other selected meta-heuristics on extensive test environments are presented in Section 5. The appropriateness of the developed algorithms is verified on four engineering problems in Section 6, with a conclusion and future trends presented in Section 7.

2 Related works

This section presents a description of the random optimization domain. This field has several portions, like single-objective, unconstrained, multi-objective, and dynamic optimization. As the algorithms proposed in this work solve problems of single-objective optimization, the central hub here concerns the difficulties and relevant results in the areas of single-objective optimization problems and algorithms, as described below.

2.1 Single-objective optimization problems

These problems cope with a single objective connoting that, at most, one objective needs to be maximized or minimized. This form of optimization might be concerned with a set of constraints that fall into the groups of inequality and equality and which can be drafted as a minimization problem as shown below:

$$\begin{aligned}&\texttt{Minimize: } F(\vec{x}) = \{f_1(\vec{x})\}\\ &\text{Subject to:}\\ &w_j(\vec{x}) \geq 0,\ j = 1, 2, \ldots, C\\ &z_j(\vec{x}) = 0,\ j = 1, 2, \ldots, L\\ &lb_j \leq x_j \leq ub_j,\ j = 1, 2, 3, \ldots, V\end{aligned}$$

where L and C represent the number of equality and inequality constraints, respectively, V denotes the number of variables, ub_j and lb_j are the upper and lower limits of the jth variable, respectively.

The set and scope of variables, constraints, and objectives create a space in a d-dimensional search domain. For one-, two-and three-dimensional problems, the search space can be readily displayed in the Cartesian coordinate system, allowing its shapes to be viewed. However, problems with dimensions more than three cannot be drawn as these dimensions lie outside of what we encounter in our lives. Thus, real-world optimization problems with many variables present the most significant difficulty when addressing them. The scope of problems' variables limits the scope of the search process, which is diversified. The optimization problems' variables can be either binary or continuous, where these problems deal with either a binary or a continuous search domain. In the first type, there is a finite number of points between any two adjacent points, while there is an unlimited number of points between every two in the second type. Locating the global optimal in a continuous search space differs from that in a binary search space, where each search has its challenges. Even though most optimization problems have a varied scope of decision variables, several problems do not have a precise scope to be deemed during optimization [32].

Anyway, solving such problems demands particular attention. For example, an optimization algorithm might commence with an initial scope and expand it during optimization. The constraints of the optimization problem restrict the search space all the more. Due to the gaps they cause in the search space, the solutions in these areas need to be revised for the optimization problem. The search space can be split into diverse segregated regions with various constraints. Infeasible solutions go beyond the constrained areas that have been set.

Conversely, the feasible solutions are those that can be implemented within constrained areas. The sections of the search domain that are outside and within the confined regions are referred to as feasible areas and infeasible areas, respectively. An optimization technique that performs well in an open search area may need to be more effective when used in a confined search domain. Thus, optimization algorithms must be set up with appropriate operators to handle the constraints [33] properly. Another defiance when dealing with optimization problems is that there are several local optimums. The search space created by the objective function, decision variables, and constraints may be modest or too complex. In reality, the main difficulty of optimization methods in most studies given in the literature is the number of local solutions present in the problems.

In single-objective search space problems, only one optimal solution (i.e., the global best one) is available, which delivers the optimal target value. Many other solutions produce objective values that are not far from the best global ones. This form of solution is referred to as local solutions because they are locally the best ones when the entire search space is considered in its proximity. Still, they are not considered the best solutions globally when the search space is considered as a whole. Having too many local solutions brings about several optimization methods to drop into local optimums. Due to the fact that a real search space frequently includes several local solutions, an optimization algorithm should be able to avoid them in order to consistently obtain the best global solution. Additionally, convergence speed is an intricacy that faces optimization algorithms when addressing optimization problems.

An optimization method capable of averting local solutions is not inevitably eligible to find the global optimal. The estimated position of the global optimal is well defined when the optimization method avoids local solutions. The following stage is to boost the performance level of the got approximate solutions. Rapid convergence rate will certainly cause stagnation in local optima. Conversely, unexpected amendments in the solutions lead to eschewing local optima but slow down the convergence rate to the optimality. The main challenge an optimization technique must overcome when solving real-world problems is these two conflicting goals. The convergence speed is all-important in locating a precise approximation of the global optimal solution. When dealing with single-objective search spaces, there are other kinds of challenges like uncertainty, and isolation of the optimal, dynamic, and noisy objective functions. Each of these difficulties demands particular regard. These concepts are beyond the setting of this study, so concerned readers can refer to the work presented in [34].

2.2 Single-objective optimization algorithms

These algorithms can be broadly distributed into two key classes: deterministic methods, heuristic and meta-heuristic methods, with the latter category being the most common [35].

2.2.1 Deterministic algorithms

Deterministic approaches address optimization problems by making predetermined decisions; for example, if the initial conditions are the same, the ultimate solution will also be the same [36]. Deterministic methods consistently find the exact solution to a specific problem, provided they launch with identical outset points. The essential advantage of these algorithms is their reliability, as they get a solution in each independent run. Their computations could be more effective when acting with large-scale data structures. Yet, the stagnation of local optima is a hurdle to these methods as they usually do not have random conduct when tackling real-world optimization problems [37]. Bharathan et al. [38] evolved an integer linear programming model with penalty coefficients. Global constraint violations are permitted in this model, but they will be penalized appropriately. This strategy is superior to traditional methods when user requirements are complex. Local and Tabu search are examples of deterministic techniques [39]. Deterministic approaches are often ineffective when used to solve high-dimensional problems as the problem's conditions constrain them. Meta-heuristics have been proposed to solve such challenging problems since many real-world problems exhibit one or more of the characteristics indicated above [35].

2.2.2 Stochastic and meta-heuristic algorithms

Stochastic methods use random operators, where different solutions can be found even if the launch point is unaltered, hence making stochastic methods less trustworthy than deterministic algorithms. Anyhow, the random behavior enhances in avoiding the local optima, which is the critical merit of stochastic methods. The accuracy of these methods can be further enhanced by adjusting and using more runs [40]. As per this, heuristic methods can produce high-quality approximate solutions promptly for large-scale data. However, heuristic methods are often created based on the unique knowledge of optimization problems [41]. Hence there will be several limitations to extending the algorithm.

Stochastic algorithms can be split into two classes: collective and individualist. In the first pool, a stochastic algorithm commences and fulfills optimization with a single solution, which is altered, at random, and ameliorated for a predetermined number of iterations or meets a final evaluation method. Simulated Annealing (SA) [42] and hill climbing [43] are the two most popular methods in this group. This family of algorithms demands low computational overhead and only needs a few function evaluations. Comparatively, several random solutions are produced and evolved by collective approaches during the optimization process. Typically, the solutions work together to establish the global optimal in the search space. There are a lot of optimization algorithms in this family; some of the most prominent ones are: Genetic Algorithm (GA) [44], Particle Swarm Optimization (PSO) [45], and Differential Evolution (DE) algorithm [46]. These algorithms might get many solutions by reducing the opportunity for stagnation in local optima, which is a crucial benefit of the collective techniques. Yet each alternative necessitates a single function assessment, and establishing effective collaboration between the solutions is a significant difficulty.

There is no way to be sure that the optimal solution will be found in solving many real-world fusion problems because most problems are NP-hard. Heuristics, which are approximate approaches utilizing iterative trial-and-error procedures, are typically used instead of accurate methods to approach the optimal solution. Many of them take inspiration from nature, and their most recent development is using meta-heuristics [22]. A meta-heuristic is a chief iterative process that directs and alters the actions of subordinate heuristics to provide solutions of a high standard quickly. Each iteration may change either a whole (or partially) single solution or an accumulation of solutions. The subordinate heuristics may be low (or high) level methods, a straightforward local search, or a construction technique. Even though meta-heuristic methods take part in some semblances with each other, they are some fundamental differences between them. In effect, they typically mimic some features of natural behavior, biological or physical phenomena present in nature, or manually tailored search strategies [47]. They can be principally classed into four key pools, according to their source of inspiration, as explained below:

- Evolutionary-based Algorithms (EAs): algorithms in this category simulate concepts and models of biological evolutionary behavior of creatures found in nature based on natural selection and the idea of survival of the fittest. The most widespread techniques in this class are GA [44], DE algorithm [46], and Biogeography-Based Optimizer (BBO) [48]. Some other examples of this type of algorithm include Wildebeests Herd Optimizer (WHO) [49] and Learner Performance-based Behavior (LPB) [50].
- Physics-Based (PB) methods are motivated by the prevailing physical rules that exist. The notable method devised by the inspirations appropriated from physics is SA [42]. Essentially, the physics' laws have already generated significant results when couched into optimization methods, with some eminent examples being:

Equilibrium Optimization (EO) [51] and Archimedes Optimization Algorithm (AOA) [3].

- Swarm Intelligence (SI) techniques: the algorithms in this category simulate the collective social behavior of collections of flocks or colonies, such as swarms of birds, animal herds, insect colonies, schools of fish and flocks of many other species of creatures found in nature. Among the most successful SI-based methods are PSO [45] and ACO Algorithm [52]. Salp Swarm Algorithm (SSA) [32], Capuchin Search Algorithm (CSA) [53], Chameleon Swarm Optimizer (CSO) [54], White Shark Algorithm (WSA) [55] and Trees Social Relations (TSR) [56] are only a small number of the long list of SI-based meta-heuristics.
- Human-based Algorithms (HBA): these methods are generally associated with human activities and behaviors [57]. Some of the newest human-based algorithms in this category include driving training-'based optimization [58], Ali baba and the forty thieves algorithm [47], and stock exchange trading optimization [59].

These and other meta-heuristics have proven to be practical and effective in solving a wide variety of applications in engineering [32] and science [47, 55]. Many applications of similar algorithms can be located in the extended literature concerned with meta-heuristic research. To name a few, business activities [60], industrial designs [61, 62], Feature Selection (FS) [63], motif discovery problem [64], agriculture [65], medical [66] and classification problems [67] are among the beneficiaries [68, 69]. In most cases, these methods in the related applications have led to encouraging and promising results when solving practical, real-world optimization problems.

Optimization with meta-heuristic algorithms begins with a set of random solutions that demand to be amalgamated and altered rapidly and abruptly. This motivates solutions to proceed globally. This procedure, known as exploration or diversification of the search space, draws solutions to different areas of the search space by abrupt shifts [55]. The fundamental objective of this procedure is to identify the most encouraging regions of the search space and to avert local solutions [54]. After satisfactory solutions, solutions commence to change and advance locally progressively. The main goal of this procedure, known as exploitation, is to increase the performance level of the best solutions obtained during the exploration step. Even if the exploitation stage may involve the avoidance of local optima, the coverage of the whole search space is less extensive than it was during the diversification stage. In this situation, local solutions nearby to the global optimal can be avoided. This explanation illustrates how the exploration and exploitation stages tolerated two opposing aims. When is the preferable time to shift from exploration to exploitation? is a critical question [53]. No one can answer this question due to the randomness of meta-heuristics and the unknowable shape and nature of the search space. Because of this, most meta-heuristic optimization methods call for search agents to quickly proceed from exploration to exploitation through flexible mechanisms [70].

The above and other meta-heuristic methods have achieved encouraging levels of performance in a reasonable amount of time in tackling complex real-life problems identified in the intended applications and datasets. Still, they need to ensure that optimum solutions will be located in all experimental runs [53]. Additionally, several complex problems may emerge due to ongoing technological breakthroughs in various technical and scientific domains [32, 55, 71]. These problems must be effectively and continuously solved by getting optimal solutions for all of them. The proficiency of any meta-heuristic to identify the globally optimal solutions for all types of optimization problems cannot be guaranteed. Although only sometimes, an optimization technique is adequate for most problem types. Therefore, it is still thought necessary to facilitate existing algorithms or create new ones to tackle challenging real-world problems. This led to the motivation for this work, which aims to improve the capability of the basic CSA [71]. This is accomplished by creating three enhanced versions of this algorithm and exploring their efficiencies in solving well-known optimization problems with various numbers of dimensions and complexity. The following sections detail the primary CSA and proposed variants of the CSA.

3 Basic crow search algorithm

The inspiration for CSA was featured in the crows' behavior in hiding food and the mechanism they follow to find where other crows' food is stashed. Crows frequently conceal extra food in locations where it may be preserved and retrieved when needed. As greedy birds, crows try to find hidden places for other crows' food to find better food by following other crows that act as thieves. Meanwhile, crows prevent their food from being robbed. The anti-tailing ability of crows enables them to notice if another crow follows them by a reasonable probability. If a crow notices that a crow follows it, it will trick that crow into random places in the environment rather than guiding it to where it hides its food. Based on this simulation of crow's intelligence, CSA aims to find the optimum solution to the targeted optimization problem.

The key idea of mimicking the intelligent behavior of crows is the search mechanism and how crows prevent their food from being looted from their lairs by tricking other crows into random positions. The thieves' experience is that the teachers store their food in safe places to protect their caches from being pilfered [72]. Simulation of crow's

behavior in a mathematical model of optimization can be characterized as follows: Several N crows in the flock are the population size, and the d-dimensional environment is the search space that crows need to search in, where d is the number of the decision variables. The position of crow i at iteration t is defined as:

$$x_t^i = [x_1^{i,t}, x_2^{i,t}, \ldots, x_d^{i,t}], \tag{1}$$

where $i = 1, 2, 3, \ldots, N, t = 1, 2, 3, \ldots, T$, where T is the total number of iteration loops, and x_t^i represents the current position of crow i at iteration t.

Each crow has its corresponding memory m to store the hiding position, whereby crow i memorizes the position with the best food so far at iteration t and stores it in m_t^i. During addressing an optimization problem, crow i chases crow j at iteration t to its hiding place m_t^j. There are two situations, and either of them will happen:

- **Situation 1**: Crow i was able to locate the hiding place m_t^j while crow j does not know that it is being pursued by crow i. Next, crow i changes its location as seen below:

$$x_{t+1}^i = x_t^i + r_i \, fl(m_t^j - x_t^i) \tag{2}$$

where r_i is a uniformly random value $\in [0, 1]$, which affects the randomization during the search process and fl is the flight length of the crows.

Figure 1 displays the schematic illustration of the flight length (fl) implied in (2). This parameter constitutes one of the main control parameters of CSA and significantly affects the ability to search. Small values for fl will eventually lead to a local search, where the neighborhood search space will not be far from the present position of crow i.

As Figure 1 illustrates, the next position x_{t+1}^i of the ith crow will be on the dashed line between x_t^i and m_t^j in the state if fl is set to a value less than 1. Alternatively, large values for fl will stimulate crows to global search, where the search space will be far away from the present position x_t^i. When fl is set to larger than 1, the following location of the ith crow, x_{t+1}^i, will be located anywhere on the dashed line. It may override the position m_t^j based on the value of r_i.

- **Situation 2**: If the jth crow observes that the ith crow is following it, it will fly to a random place in the search area to get it away from its hiding place, or denoted as m_t^j. By combining the two states, the mathematical paradigm of CSA can be expressed in (3):

$$x_{t+1}^i = \begin{cases} x_t^i + r_i \, fl(m_t^j - x_t^i) & r_j \geq AP \\ \text{A random position} & r_j > AP \end{cases} \tag{3}$$

where AP is the awareness probability of crows, r_j is a randomly dispensed random value $\in [0, 1]$ that contributes to random distribution during the random search process. The values of fl and AP are fixed at 2.0 and 0.1, respectively.

AP directly controls the counterbalance between diversification (i.e., exploration) and intensification (i.e., exploitation). A small value of AP creates a good solution in the neighborhood area, which will increase intensification. Conversely, large awareness probability values will have a great chance to scout the search space and tend to reach global search, which will increase diversification.

As a synopsis, optimization-based CSA is implemented through an iterative process in which the initially generated memories and positions of crows are valued and amended at each iteration loop up to convergence. The convergence process stops when the termination assessment criterion is attained. The best position reached by crows that have found food is indicated as a solution to the optimization problem. The pseudo-code of the primary CSA is given in Algorithm 1.

Algorithm 1 Pseudo code of the standard CSA.

```
N ← Number of crows
t ← Iteration counter
T ← Total number of iterations
Set the initial values of AP = 0.1, fl = 2.0 and t = 1
x_i ← Crows positions; i = 1, 2, ..., N.
Initialize the positions of crows at random
Evaluate the fitness function of each crow f(·)
Initialize the memory, m, of the crows
while (t < T)
  for i = 1, 2, ..., N
    Choose one of the crows at random, like crow j, and chase it.
    if (r_j ≥ AP)
      x_{t+1}^i = x_t^i + r_i fl(m_t^j − x_t^i)
    else
      x_{t+1}^i = a random position in the search space
    end if
  end for
  Examine the viability of new positions (x_{t+1}^i)
  Assess the crows' new position f(x_{t+1}^i)
  Amend crows' memory m_{t+1}^j
  t = t + 1
end while
```

Each new location of each crow is assessed iteratively inside each loop of CSA in accordance to a pre-defined fitness criterion. Crows' positions are updated if their solutions are

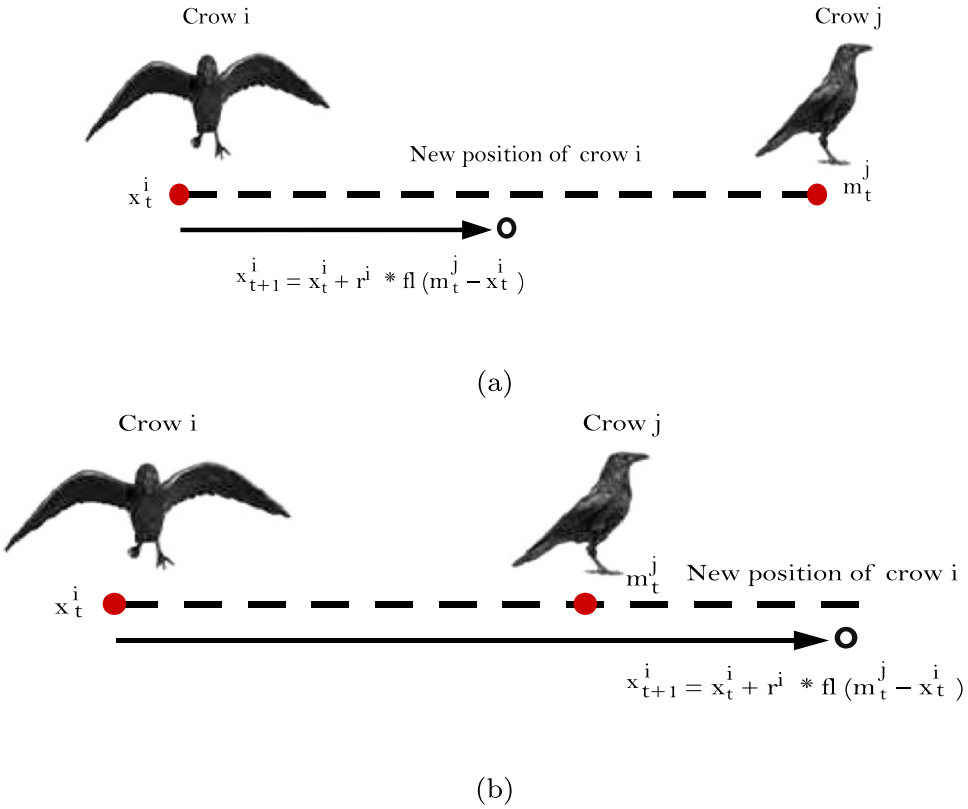

Fig. 1 Flight length of situation 1 (a) $fl < 1$ and (b) $fl > 1$ inspired from [9]

more accurate than the ones of the current positions. Crows remain in their positions if their solutions are of lesser quality than the current ones. Crows update their memories within each iteration loop if the quality level of the new positioning solutions, assessed per the fitness criterion, is better than the saved positions.

4 Proposed variant algorithms of CSA

4.1 Limitations of the basic CSA

Although CSA can search for optimal solutions during solving optimization problems, its search capacity is confined by its native positioning updating mechanism, which may result in local optima iteratively [20, 22, 30]. This phenomenon is illustrated by the fact that CSA is probably to face early convergence when addressing complex or even modest real-world optimization problems [21]. The cause is that crows in CSA rely on constant flight length and awareness probability parameters to navigate and search for other crows' foods repeatedly. However, these fixed selected values for such key parameters cannot ensure that the CSA can escape stagnation or that it is not trapped in local optima. Besides, CSA faces another problem of weak exploration and exploitation capabilities. This phenomenon is clearly encountered due to the process of updating the locations of crows in CSA, which needs to consider the global positions of the crows obtained thus far. Therefore, some strategies must help update the positioning of crows in addition to fine-adjusting the key parameters, namely flight length and awareness probability. A desired exploration process can find a favorable region in the search space with an optimal solution. Then, a desired exploitation process can eventually locate this optimal solution. However, the exploration power of CSA needs to be improved in the initial search phase so that its low exploitation causes it tough to locate the global optimum solution in the overdue search stage. As a result of this fact, local optima is customarily received. Thus, a sensible trade-off should be formed between exploration and exploitation to reinforce its search aptitude. Accordingly, the movements of crows in the enhanced CSA in this work are carried out by a new positioning updating process with the global best position got so far that the best crows offer, as explained in more detail in the next section. Further, more is needed to support the best crows to find other crows' food sources using only fixed flight length and awareness probability parameters. Because this implies that the best crow's capacities to guide all the crows are weakened, the exploration and exploitation competencies of CSA are less and less in the late foraging process. As per this, the flight length value should increase gradually with time, while the value of awareness probability should gradually decrease over time. This is to assist the best crow in directing all the crows towards the food so that the explo-

ration and exploitation abilities of CSA will be higher and higher in the overdue search process. Thereby, it is essential to strengthening the exploitation feature of CSA in the final search iterations to speed up the speed of convergence and shun suffering from quick convergence. To cope with the above issues, a new positioning updating model that uses adaptive flight length and awareness probability parameters in the implementation phase of CSA is presented to enhance the performance degree of CSA greatly. The following sections offer detailed descriptions of the proposed positioning updating model, in addition to the proposed enhanced versions of CSA, referred to as Exponential CSA (ECSA), Power CSA (PCSA), and S-shaped CSA (SCSA).

4.2 Proposed positioning updating process

In ECSA, PCSA, and SCSA, the position of the robber and owner crows needs to be updated at each iteration. A crow possesses a food source, and a robber crow endeavors to steal that food. In such a scenario, the position of both the owner and robber crows has changed accordingly. The owner crow's memory is also updated based on its observation of the robber crow. The position of the crows in ECSA, PCSA, and SCSA are updated as per the mechanism proposed in (4).

$$x_{t+1}^{i} = \begin{cases} x_t^i + fl_t(m_t^j - x_t^i)r_1 & r_j \leq AP_t, r < 0.5 \\ x_t^i - (1 - fl_t)(m_t^j - x_t^i)r_2\alpha & Otherwise \\ \tau\left(l_j - (l_j - u_j)r_3\right) & r_j \geq AP_t \end{cases} \quad (4)$$

where x_{t+1}^{i} designates the next position of the ith crow at iteration $t+1$, x_t^i identifies the present position of the ith crow at the current iteration, m_t^j stands for the memory of the best crow throughput the iterative process of the entire swarms at iteration t, AP_t is the awareness probability of crows at iteration t which is updated iteratively during the iteration loops, fl_t represents the unit step of crows' movement upon iteration t, r_j, r, r_1, r_2 and r_3 are random values yielded in the extent from 0 to 1, α represents the component sgn$(rand - 0.5)$ that is either 1 or -1 which effects the search direction, l_j and u_j stand for the lower and upper boundaries of the search domain at dimension j and τ is defined as a function of iterations as drafted in (5).

$$\tau = a_0 e^{-(a_1 t/T)^{a_2}} \quad (5)$$

where the coefficients a_0, a_1 and a_1 are constant values used to automatically update the parameter τ at each iteration. These coefficients are basically beneficial for strengthening exploration and exploitation conducts. The coefficients a_0, a_1 and a_2 are all set to 2.0, 4.0, and 2.0, respectively, for all of the problems that are later handled in this work. These values were captured by pilot testing for a bunch of test functions.

The parameter τ is presented as a function of time to dominate the random movement of crows iteratively and thus decreases with the number of iterative generations. Specifically, this parameter was applied to strengthen the dynamic system of convergence by diminishing the search speed as well as enhancing the exploration and exploitation features of the evolved algorithms. This parameter can enable crows to explore more foraging space and exploit each area while foraging for food or other crows' food. This is to arrive at an efficient convergence process, which can further strengthen the performance degree of ECSA, PCSA, and SCSA in tackling optimization problems.

The first two cases of (4), (i.e., when $r_j < AP_t$), were suggested to allow crows to take advantage of random numbers, and the component sgn$(rand - 0.5)$ was proposed to enable crows to be very effective in exploiting and scouting the search space in different directions and locations. The third case of (4), (i.e., when $r_j \geq AP_t$) was suggested to empower crows to scout several random positions in the search domain to improve local and global search capabilities and to get a sufficient balance between exploration and exploitation. This gives crows in ECSA, PCSA, and SCSA great power to explore every potential position in the search area. The parameters fl_t and AP_t were used as interactive operators to manage these algorithms' exploration and exploitation capabilities. With different values for fl_t and AP_t, the proposed algorithms can alternate between global and local searches.

In the position updating mechanism in (4), the algorithms tend to evolve a new solution x_{t+1}^{i} for a particular problem that is better than the present solution x_t^i. In the mechanism of updating crows' positions in CSA [9] as shown in (3), the values of the parameters AP_t and fl_t are constants during the execution phase of CSA. This indicates that exploration and exploitation features of the standard CSA rely on fixed figures of these parameters. This affects the search behavior of CSA, which can turn into good exploration and exploitation without a rigid structure. Accordingly, to improve exploration and exploitation, the values of fl_t and AP_t parameters must be updated iteratively at each iteration of CSA.

To recapitulate, we have proposed three new variant algorithms of the primary CSA. The developed variants of CSA aim to boost the convergence rate of CSA to reach optimality by carefully enhancing two folds: exploration and exploitation of the search space. Each evolved algorithm uses a different mathematical growth model for each of the fl_t and AP_t coefficients, which are described below.

4.3 Exponential model-based CSA (ECSA)

The exponential model was first introduced in [73]. The exponential growth functions shown in (6) and (7) were proposed to represent the flight length and awareness probability of

crows in ECSA, respectively.

$$fl_t(k;\beta_0,\beta_1) = \beta_0(1-e^{-\beta_1 k}) \tag{6}$$

$$AP_t(k;\beta_0,\beta_1) = \beta_1\beta_0 e^{-\beta_1 k} \tag{7}$$

where $k = \frac{T}{t}$, β_0 stands for the initial estimation of the flight length, β_1 stands for the final estimation of the flight length of crows that could be fulfilled approximately at the end of the ECSA's iterative process. These parameters represent the coefficients of the exponential growth function.

It is important to be aware of the following:

$$AP_t(k;\beta_0,\beta_1) = \frac{\partial fl_t(k;\beta_0,\beta_1)}{\partial k} \tag{8}$$

Several conventional and intelligent conventional are mentioned in the literature to estimate the parameters β_0, and β_1 for the functions of flight length and awareness probability [74]. One famous traditional parameter estimation method is the least square estimation method [75]. This method struggles with estimating accuracy and requires a lot of measurements to be able to offer accurate parameter estimates. Other approaches include using meta-heuristics, which may require great computational efforts to estimate parameters [74]. In this work, the parameters β_0 and β_1 used for the adaptive functions of fl_t and AP_t were picked by applying practical design by examining ECSA on a significant subset of test problems. The coefficients β_0 and β_1 are equal to 2.0 and 1.0, respectively, for all of the problems solved in this paper using the proposed ECSA. However, only optimal values are often experimentally obtained, perhaps, not the 'best' ones.

4.4 Power model-based CSA (PCSA)

The non-homogeneous Poisson process serves as the model's foundation [76]. The functions shown in (9) and (10) were used to implement fl_t and AP_t of the crows in PCSA, respectively.

$$fl_t(k;\beta_0,\beta_1) = \beta_0 k^{\beta_1} \tag{9}$$

$$AP_t(k;\beta_0,\beta_1) = \beta_0\beta_1 k^{\beta_1-1} \tag{10}$$

where $k = \frac{T}{t}$.

Equations 9 and 10 were utilized to amend fl_t and AP_t throughout the iterative process of PCSA. It is significant to know that (8) was used to find AP_t in this model. A range of values from 0 to 5 was applied to β_0 and β_1. For all of the benchmark problems optimized in this paper using the proposed PCSA, β_1 and β_0 are equal to 0.05 and 2.0, respectively. These values were found by empirical investigation of a large subset of test functions of varied complexity, where these values reported the best performance of the proposed PCSA.

It is evident from (9) and (10) that when t is small, the value of fl_t is as its maximum value and rapidly drops to its lowest value. On the contrary, AP_t is small at small t and gradually increases towards its maximum value. In such a context, crows in PCSA can find a food source at the end of their foraging. Using power functions for fl_t and AP_t can improve exploration and exploitation, as shown in the evaluation results.

4.5 Delayed s-shaped model-based CSA (SCSA)

The S-shaped model used in the proposed SCSA was introduced in [77, 78]. The proposed growth S-shaped model for fl_t is given in (11).

$$fl_t(k;\beta_0,\beta_1) = \beta_0\left(1-(1+\beta_1 k)\,e^{-\beta_1 k}\right) \tag{11}$$

where $k = \frac{T}{t}$.

Equation 8 was used to derive AP_t of the SCSA model from fl_t presented in (11), where the formula produced for AP_t can be expressed as follows:

$$AP_t(k;\beta_0,\beta_1) = \beta_0\beta_1^2 k e^{-\beta_1 k} \tag{12}$$

This work solves all testing problems using the proposed SCSA, where β_1 and β_0 are equal to 7.0 and 2.0, respectively. These values were determined by experimental testing of a large number of benchmark functions, in which these coefficients were altered many times until the proposed SCSA obtained a sensible solution.

To sum up, the parameters β_0 and β_1 in (6), (7), (9), (10), (11) and (12) can be fine-tuned to other problems as demanded. The three new versions of the basic CSA were proposed to provide an adequate setting for fl_t and AP_t to foster exploration and exploitation features of CSA. The above models are expected to achieve effective convergence and improve the performance of the proposed ECSA, PCSA, and SCSA in optimizing optimization problems. Moreover, they could provide great potential for ECSA, PCSA, and SCSA to evade stagnation in local optima areas and assist them in determining the global optimum solution.

4.6 Exploration ability

Many parameters improve exploration in the proposed algorithms, ECSA, PCSA, and SCSA, which are described as shown below:

- τ: It manages the exploration quantity of the proposed algorithms and identifies how far the new location would be from the food source. It further intensifies exploitation aptness, eludes premature convergence, and prevents the descent of solutions into local optima.
- sgn$(rand-0.5)$: It directs the direction of the exploration process. Since r implements a random number in the interval from 0 to 1, the likelihood of both positive and negative signs is comparable.
- fl_t: This adaptive function was elected based on several empirical tests. In the initial iterations, the owner crow and the robber crow are far away from each other, and the crows are all onward away from food or other crows' food. Updating fl_t improves the proposed algorithms' ability to search the space globally.
- AP_t: This adaptive function was selected based on an empirical test. Updating the values of AP_t for each proposed algorithm helps the crows in each algorithm to find unknown search areas at initial iterations when crows are very far from food or from each other.

4.7 Exploitation ability

The following describes the main parameters used in ECSA, PCSA, and SCSA to perform exploitation and local search in the search space:

- τ: With the passage of iteration time, exploration wanes, and exploitation expands. In the last iterations, where a crow approaches a food source, updating the crow's position with this control variable will aid the ability to locally search for the food source, which leads to exploitation.
- sgn$(rand-0.5)$: It also manages the exploitation feature and identifies the direction of the local search.
- a_0: This quantity manages the exploitation property of the proposed algorithms by drilling around the optimum solution.

4.8 Computational complexity analysis

Typically, the complexity issue of optimization algorithms can be represented by a function that links the problem's input size to the algorithm's run-time. In doing so, the complexity issue of ECSA can be described as presented in (14).

$$\begin{aligned}\mathcal{O}(ECSA) = {} & \mathcal{O}(initialization) + \mathcal{O}(problem\ Def.) \\ & + \mathcal{O}(t(cost\ function)) + \mathcal{O}(t(Sol.\ update)) \\ & + \mathcal{O}(t(memory\ update))\end{aligned} \tag{13}$$

where N, d, and t denote the number of crows, problem dimension, and iteration counter, respectively, and c identifies the cost of the objective function.

The parameters in (14) form the basic components of the complexity issue of the optimization method. In consequence, the general computational complexity issue can be identified as shown below:

$$\mathcal{O}(ECSA) = \mathcal{O}(1 + nd + tcn + tn + tnd) \tag{14}$$

As $nd \ll tnd$ and $tn \ll tcn$, (14) is reduced to (15):

$$\mathcal{O}(ECSA) \cong O(tcn + tnd) \tag{15}$$

Notably, the complexity issue for PCSA and SCSA is the same as the complex issue for ECSA, which is given in (15). The complexity issue of ECSA, PCSA, and SCSA is of the polynomial order. In conclusion, these proposed algorithms can be considered efficient optimization algorithms. The fundamental steps of these algorithms can be abridged by the steps given in Algorithm 2, and the flowchart describing the

Algorithm 2 A pseudo-code summarizing the main steps of the proposed ECSA, PCSA, and SCSA algorithms.

$r, rand, r_1, r_2, r_3, r_j$ are random values from 0 and 1
l_j and u_j are the lower and upper limits at dimension j
$d \leftarrow$ dimension of the problem
Initialize and evaluate the position, x, of all N crows
Initialize the crows' memory, m
Set $t \leftarrow 1$
while $(t < T)$ **do**
  Define AP_t using Equations 7, 10 or 12 depending on the model used.
  Define fl_t using Equations 6, 9 or 11 depending on the model used.
  Define τ using Equation 5
  for i = 1 to N **do**
    Choose one of the crows at random, such as crow j, to follow it.
    if$(r_j < AP_t)$ **then**
      if$(r < 0.5)$ **then**
        $x_t^i + 2fl_t(m_t^j - x_t^i)r_1$
      else
        $x_t^i - 2(1 - fl_t)(m_t^j - x_t^i)r_2\,\text{sgn}(rand - 0.5)$
      end if
    else
      for j=1 to d **do**
        $x_{t+1}^{i,j} = \tau\left(l_j - (l_j - u_j)r_3\right)$
      end for
    end if
  end for
  Assess the new crows' positions $f(x_{t+1}^i)$
  Amend crows' memory m_{t+1}^j
  t = t+1
end while

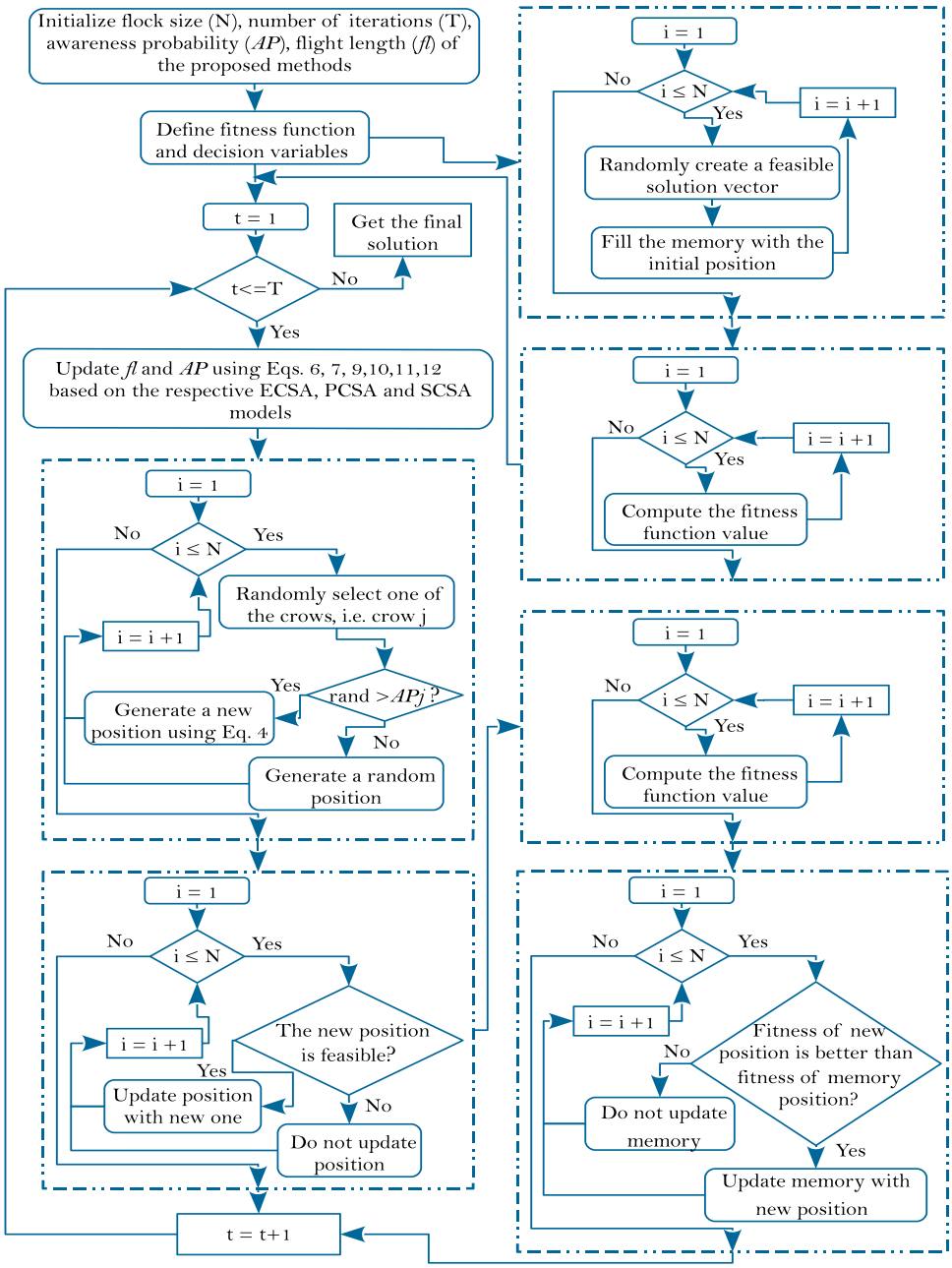

Fig. 2 Schematic diagram of the proposed algorithms of CSA for global optimization

general steps of these algorithms (i.e., ECSA, PCSA, and SCSA) are presented in Fig. 2.

5 Experimental results and discussion

This section shows and explains the experimental results of the developed ECSA, PCSA, and SCSA on sixty-seven broadly well-known benchmark functions. A characterization of these test functions is also provided in this section. The outcomes are explained and compared with promising meta-heuristic optimization methods.

5.1 Description and purpose of the functions used

In this study, 67 optimization functions were used to demonstrate the effectiveness of the developed ECSA, PCSA, and SCSA. These functions can be clustered into the following classes: unimodal with 7 test functions [79], multimodal with 6 test functions [80], fixed-dimension multimodal with 10 test functions [79, 80], CEC-2015 with 15 benchmark functions [81] and CEC-2017 with 29 stable functions and one unstable test function [47, 82]. Details of these test functions, involving the test environment, functions' dimensions, search spaces' limits, and the optimum obtained value, are

presented in Appendix A in Tables 23, 24 and 25, respectively. Each set of these test functions was used to assess specific views of the developed algorithms.

The first class (i.e., unimodal functions) that includes F_1-F_7 has only one optimum solution. These test functions were chosen to judge the proposed algorithms' exploitation feature and convergence. Multimodal test functions in the second class, which involve F_8-F_{13}, have more than one optimum solution. These functions were chosen in the current work to assess the exploration behavior of the proposed algorithms, where they have many local optimum solutions and more than one global optima. However, a good optimization algorithm demands the ability to search the space globally to identify the global optimal and bypass the local entrapment. The third class (i.e., fixed-dimension multimodal functions), which includes F_{14}-F_{23}, are homologous to multimodal functions, but they have fixed and low dimensions. These functions were employed here to assess the proposed algorithms' exploration feature further. The desired algorithm must avoid local optimal solutions and quickly approach the optimal global solution. In a nutshell, the test functions F_1-F_{23} were chosen in this study because they are adequate in verifying the local optimum avoidance, diversification, and intensification behaviors of the proposed algorithms as well as their suitability for testing the convergence rate of the proposed algorithms.

The last two classes, CEC-2015 and CEC-2017 test groups include composite and hybrid benchmark test functions. These tests' functions mimic the complexity of a real search domain by having several local optima and various function shapes in diverse test areas. As detailed in Appendix A, these functions are formed by shifting, rotation, extension, and hybridization of unimodal and multimodal test functions. These test problems implement more challenging optimization problems and are chosen in this work to present more challenges in evaluating the accuracy of the proposed algorithms. Besides, these cases were prepared to assess optimal local avoidance and the proposed algorithms' exploration and exploitation behaviors. As per the above discussions, an adept optimization algorithm should be capable of bypassing local optimal solutions and speedily converging to the global optimum. Subsequently, the above test groups were selected to assess the efficacy of the proposed algorithms from the perspective of evading local optimal solutions and finding the optimum global ones, especially CEC-2015 and CEC-2017 benchmarks with extremely challenging test functions. With this, estimating and judging the exploration and exploitation aptitudes of the developed algorithms in this work is simple.

5.2 Experimental setup

To manifest the general efficacy of the developed ECSA, PCSA, and SCSA, their outcomes are compared with those of the standard CSA and other optimization methods on unimodal, multimodal, fixed-dimensional, CEC-2015, and CEC-2017 benchmark test groups presented in Appendix A. The competing methods presented here include four categories of meta-heuristic optimization algorithms: (i) GA [83] as an evolutionary algorithm, (ii) PSO [84], Spotted Hyena Optimizer (SHO) [85], GWO [86], Emperor Penguin Optimizer (EPO) [87], and CSA [9] as swarm intelligence algorithms, (iii) Gravitational Search Algorithm (GSA) [88] and Multi-Verse Optimizer (MVO) [89] as physics-based algorithms and (iv) Sine Cosine Algorithm (SCA) [90] as a mathematics-based algorithm. PSO and GA are the most popular and well-studied swarm intelligence and evolutionary algorithms in these comparative algorithms. Besides, SHO, GWO, EPO, CSA, and SCA are practical meta-heuristic algorithms, and finally, MVO and GSA are reliable and well-known physics-based algorithms. Parameter settings of the proposed ECSA, PCSA, and SCSA algorithms, the primary CSA, and the other comparative algorithms are provided in Table 1. The comparative meta-heuristics mentioned above were selected in this comparison because they have been broadly applied in the literature to address the aforementioned benchmark functions, where they provided promising performance. Moreover, these algorithms share many similarities with the proposed algorithms, including flexibility, generality, and simplicity. In addition, they are independent of the nature of the benchmark functions to be addressed.

The settings of the parameters displayed in Table 1 were determined to fit the settings broadly presented in the literature. The initialization process of the proposed algorithms is similar to that used in the other ones for a fair-minded comparison between the proposed algorithms and those competitive ones. The proposed algorithms used 30 search agents associated with 1000 iterations (30,000 maximum number of function evaluations (FEs)) for all test functions in all test classes. Likewise, to realize a fair comparison, the other algorithms also used a maximum number of 30,000 FEs. The comparisons between the algorithms were made with similar floating-point precision. In this, the margins of differences between the findings are attributable to the degree of performance of the comparative methods. As presented in Table 1, the algorithms were assessed in 30 separate runs for each test problem in each experiment.

5.3 Performance Evaluation

This section presents the accuracy of the proposed algorithms and compares them with other optimization algorithms on classical unimodal (F_1-F_7), multimodal (F_8-F_{13}) and fixed-dimension multimodal (F_{14}-F_{23}) benchmark test functions. The average (`AVG`) and standard deviation (`STD`) values were employed as the best statistical measures on the benchmark test functions. These statistical values were calculated

Table 1 Parameter settings of ECSA, PCSA, and SCSA and other optimization methods

Algorithm	Parameter	Value
∀ algorithms	Search agents	30
	Number of generations	1000
CSA	fl	2
	AP Constant	0.1
ECSA	β_0, β_1	2.0, 1.0
	a_0, a_1, a_2	2, 4, 2
PCSA	β_0, β_1	2.0, 0.05
	a_0, a_1, a_2	2, 4, 2
SCSA	β_0, β_1	2.0, 7.0
	a_0, a_1, a_2	2, 4, 2
SHO	$\vec{M}$ Constant	[0.5, 1]
	Control parameter ($\vec{h}$)	[5, 0]
GWO	Control parameter ($\vec{a}$)	[2, 0]
PSO	Inertia coefficient	0.75
	Cognitive coefficient	1.8
	Social coefficient	2
MVO	Traveling 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
GA	Selection	Roulette wheel
	Crossover	0.9
	Mutation	0.05
EPO	Temperature profile ($\grave{T}$)	[1, 1000]
	$\vec{A}$ constant	[-1.5, 1.5]
	Parameters M, f, l	2, [2, 3], [1.5, 2]
	Function S()	[0, 1.5]

at the final iteration to evaluate the algorithms' accuracy convincingly. The standard deviation results were computed to test the stability of the proposed algorithms during the independent runs. The algorithms' stopping conditions were assigned to the total number of iterations. The best findings are emboldened in all tables.

5.3.1 Evaluation of functions F_1-F_7

These functions are convenient for judging the exploitation ability of the proposed algorithms since they only have one global optimum and no local ones. Table 2 shows the average (AVG) and standard deviation (STD) obtained, over 30 separate runs, by the proposed algorithms and the meta-heuristics mentioned above on unimodal functions.

From the findings on unimodal functions in Table 2, it is apparent that the proposed algorithms, ECSA, PCSA, and SCSA, reveal their strength in delivering very reasonable outcomes in comparison to the parent CSA and other promising methods. In particular, these proposed algorithms achieved the global optimum solution in many test functions compared to others. Notably, as it is seen, the proposed SCSA was the most efficient algorithm for the functions F_1, F_3, and F_6, where it achieved the global optimum solutions compared to the others. For the function F_6, the average and standard deviation values of SCSA were the best, with values of 0.00E+00 in both measures. Besides, it delivered competitive outcomes in the other test functions and was much better than many competing algorithms. This reliable performance is also seen for the proposed ECSA on average and standard deviations in F_4, F_6, F_3, F_2, and F_1, respectively in which it achieved very promising results in these functions, had the second and third ranks in these test functions after EPO and SCSA. More particularly, ECSA was the most proficient algorithm among the other algorithms for function F_5, as it identified the best global perfect solution in this function.

The proposed PSCA achieved the second-best result in F_1, F_3, and F_6 with only slight differences in the outcomes in comparison to SCSA. In F_5, ECSA ranked first with an optimal result of 7.97E-01, whereas PSCA ranked second with only a little amount of difference in the mean result, and SCSA collected the third rank after ECSA and PSCA in respect of the average score with only a very slight difference. PSCA and SCSA are the second and third-best optimization methods regarding the average results in function F_6. Specifically, SCSA ranked first with an optimal result of 0.00, where PSCA received the second rank after SCSA with a mean score of 7.08E-33, and ECSA received the third rank after SCSA and PSCA with a mean score of 5.81E-28. The results of ECSA, PCSA, and SCSA on the unimodal functions (F_1 - F_3) confirm that these algorithms outperformed the basic CSA, PSO, MVO, SCA, GSA, and others in these test cases. However, in F_2, F_4, and F_7, these algorithms provided relatively comparable results to EPO and GWO algorithms. Therefore, it can be deduced that the evolved algorithms in this work were capable of finding optimal results on many unimodal functions. Further, the small standard deviation results of the proposed ECSA, PCSA, and SCSA in all unimodal functions expose that these algorithms are stable and that this excellence is entrenched. On the basis of the characteristics of the unimodal functions under study, it can be certainly stated that ECSA, PCSA, and SCSA benefited from the high exploitation capability.

5.3.2 Evaluation of functions F_8-F_{23}

The functions F_8 to F_{23} are well-suited for examining the exploration conduct of the developed algorithms. Tables 3 and 4 exhibit the performance scores of different algorithms, over 30 separate runs, for high dimensional functions (F_8-F_{13}) and fixed-dimensional functions (F_{14}-F_{23}), respectively.

Table 2 Results of ECSA, PCSA, SCSA, and other optimization algorithms in unimodal benchmark functions

F	ECSA		PCSA		SCSA		CSA	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F_1	7.62E-28	1.02E-27	1.41E-32	1.87E-32	8.36E-35	1.33E-34	1.33E-10	1.35E-10
F_2	1.42E-11	2.13E-11	1.02E-11	1.78E-11	3.14E-12	7.20E-12	1.27E-05	1.15E-05
F_3	1.35E-22	5.36E-22	2.27E-24	5.43E-24	1.0E-24	4.37E-24	1.41E-06	4.54E-06
F_4	2.87E-13	4.35E-13	7.64E-13	1.28E-12	4.20E-13	6.27E-12	3.36E-05	2.18E-05
F_5	**7.97E-01**	1.62	1.25	1.83	1.32	1.91	3.37	1.77
F_6	5.81E-28	1.08E-27	7.08E-33	9.91E-33	**0.00E+00**	**0.00E+00**	9.80E-11	1.11E-10
F_7	4.72E-04	2.72E-04	4.50E-04	2.98E-04	4.88E-05	3.31E-05	5.41E-01	2.62E-01
F	PSO		MVO		SCA		GSA	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F_1	4.98E-09	1.40E-08	28.10E-02	1.11E-01	3.55E-02	1.06E-01	1.16E-16	6.10E-17
F_2	7.29E-04	1.84E-03	39.60E-02	14.10E-02	3.23E-05	8.57E-05	17.00E-02	92.90E-02
F_3	1.40E+01	7.13	43.10	8.97	4.91E+03	3.89E+03	41.60E+01	15.60E+01
F_4	6.00E-01	17.20E-02	88.00E-02	25.00E-02	18.70E+00	8.21	1.12	98.90E-02
F_5	4.93E+01	38.90	11.80E+01	14.3E+01	73.7E+01	19.80E+02	38.50E+00	3 4.70E+00
F_6	9.23E-09	1.78E-08	31.50E-02	9.98E-02	4.88	97 .50E-02	1.08E-16	4.00E-17
F_7	6.92E-02	28.70E-03	2.02E-02	7.43E-03	3.88E-02	5.79E-02	76.80E-02	2.77
F	SHO		GWO		GA		EPO	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F_1	**0.00E+00**	**0.00E+00**	46.10E-24	73.70E-24	19.50E-13	20.10E-12	57.10E-29	83.10E-30
F_2	**0.00E+00**	**0.00E+00**	12.00E-35	13.00E-35	65.30E-19	51.00E-18	62.00E-41	33.20E-41
F_3	**0.00E+00**	**0.00E+00**	10.00E-13	41.00E-15	77.00E-11	73.60E-10	20.50E-20	91.70E-21
F_4	77.80E-13	89.60E-13	20.20E-15	24.30E-15	91.70	56.70	**4.32E-18**	**3.98E-19**
F_5	8.59	55.30E-02	27.90	1.84	55.70E+01	41.60	5.07	**49.00E-02**
F_6	24.6E-02	17.80E-02	65.80E-02	33.80E-02	31.50E-02	99.80E-03	70.10E-20	43.90E-21
F_7	32.90E-06	24.30E-06	78.00E-05	38.50E-05	67.90E-05	32.90E-04	**27.10E-06**	**92.60E-07**

Table 3 substantiates that the proposed ECSA, PCSA, and SCSA algorithms were able to get promising results in the majority of multimodal test functions (i.e., F_8 - F_{13}) while some of the other methods did not. The results obtained by ECSA, PCSA, and SCSA in these functions were in the vicinity of the global optimum solutions, with only a slight margin difference to these solutions. For a thorough discussion, it is apparent from the findings in Table 3 that SCSA outperformed other methods in F_{12}. For F_8, which is the most challenging function in this group, ECSA, PCSA, and SCSA got better scores than those reported by the parent CSA, which roughly reached results reasonably approaching the global optimum reported by SHO. For F_9, HS presented better accuracy than other optimization methods, while ECSA, PCSA, and SCSA still presented very sensible results in this function. Also, the proposed ECSA, PCSA, and SCSA delivered highly effective results in F_{10} and F_{11}. For F_{12}, SCSA is the first best optimizer. When reading Table 3 once more, one can notice that almost all optimization algorithms acted sensibly well, with the proposed ECSA, PCSA, and SCSA achieving high-performance levels better than those performed by CSA. These findings confirm that ECSA, PCSA, and SCSA have good scores concerning exploration capacity. The standard deviation figures of ECSA, PCSA, and SCSA are tiny in these benchmark test functions, which asserts that their performance is stable.

The experimental tests presented in Table 4 are purposed to corroborate the exploration tact of the proposed ECSA, PCSA, and SCSA algorithms on more sophisticated test functions with complex search spaces than the tests presented in Tables 2 and 3. It is evident from the outcomes shown in Table 4 that ECSA, PCSA, and SCSA are superior to many other promising algorithms in the majority of fixed-dimensional functions concerning the accuracy values. Their performance is also very comparable to other rivals in the other test functions. Eventually, the AVG outcomes of the best solutions got during the 30 independent runs prove that ECSA, PCSA, and SCSA show excellent and consistent performance on average. In more detail, it is apparent from the results in Table 4 that ECSA, PCSA, and SCSA are the best optimizers in F_{14} - F_{19}, where they got convincing statistical results in these functions markedly better than those obtained by other algo-

Table 3 Results of ECSA, PCSA, SCSA, and other optimization algorithms in multimodal test functions

F	ECSA		PCSA		SCSA		CSA	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F_8	-2.45E+03	3.53E+02	-2.66E+03	3.09E+02	-2.81E+03	3.76E+02	-2.91E+03	3.21E+02
F_9	2.60	7.06	4.44	6.00	3.70	7.08	6.40	3.60
F_{10}	1.11E-01	2.20E-01	1.09E-01	3.02E-01	0.91E-01	1.36E-01	4.34E-01	6.38E-01
F_{11}	2.68E-02	2.86E-02	1.23E-02	3.48E-02	1.00E-02	3.19E-02	9.08E-02	6.84E-02
F_{12}	2.23E-08	1.47E-07	9.04E-08	1.76E-07	**8.11E-11**	**1.49E-10**	2.69E-09	7.12E-09
F_{13}	1.11E-04	1.93E-04	1.05E-02	2.12E-02	1.25E-02	2.00E-02	1.79E-03	4.93E-03
F	PSO		MVO		SCA		GSA	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F_8	-6.01E+02	13.00E+01	-69.20E+01	91.90E+01	-38.10E+01	28.30	-27.50E+01	57.20
F_9	4.72E+01	10.30	10.10E+01	18.90	22.30	32.50	33.50	11.90
F_{10}	3.86E-02	21.10E-02	1.15	78.70E-02	15.50	8.11	82.5E-10	19.00E-10
F_{11}	5.50E-03	73.90E-04	57.40E-02	11.20E-02	30.10E-02	28.90E-02	8.19	3.70
F_{12}	1.05E-10	20.60E-11	1.27	1.02	52.10	24.70E+01	26.50E-02	31.40E-02
F_{13}	4.03E-03	53.90E-04	66.00E-03	43.30E-03	28.10E+01	86.3E+01	57.30E-33	89.50E-33
F	SHO		GWO		GA		EPO	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F_8	**-1.16E+02**	**27.20**	-61.40E+01	93.20	-51.10E+01	43.70	-87.60E+01	59.20
F_9	**0.00**	**0.00**	43.40E-02	1.66	12.30E-02	41.10	69.00E-02	48.10E-02
F_{10}	2.48	1.41	16.30E-15	**31.40E-16**	53.10E-12	11.10E-11	**80.30E-17**	27.40E-15
F_{11}	**0.00**	**0.00**	22.90E-04	52.40E-04	33.10E-07	42.30E-06	42.00E-06	47.30E-05
F_{12}	36.80E-03	11.50E-03	39.30E-03	24.20E-03	91.60E-09	48.80E-028	50.90E-04	37.50E-04
F_{13}	92.90E-021	95.20E-03	47.50E-02	23.80E-02	63.90E-03	44.90E-03	**0.00**	**0.00**

rithms. There is no noteworthy difference between the results of ECSA, PCSA, and SCSA algorithms. Still, there is a large of difference between the findings obtained by these algorithms and those obtained by the other algorithms in F_{14}, F_{15}, F_{16}, and F_{17}. The average accuracy values of ECSA, PCSA, and SCSA are better than the average accuracy values of CSA in F_{22}, F_{21}, F_{20} and F_{19}. In comparison, the STD values of CSA are better than ECSA, PCSA, and SCSA in these test functions. Specifically, there is a minimal statistical difference between ECSA, PCSA, SCSA, GA, EPO, and GWO in F_{17}, and there is a slight difference between ECSA, PCSA, SCSA, and SHO in F_{19} and F_{20}. Some of the optimum results are marked in favor of the proposed algorithm, while another meta-heuristic algorithm finds a better solution. For example, the optimum result of F_{21} is -10.153; the proposed ECSA, PCSA, and SCSA obtained mean values of -2.05, -6.72, and -6.57, where these values are close to the optimal results. The same applies to F_{22} and F_{23}, where the performance of ECSA, PCSA, and SCSA is comparable to SHO in F_{22} and F_{23}. Moreover, the proposed ECSA, PCSA, and SCSA algorithms are characterized as having the slightest STD results in most of the fixed-dimensional functions in comparison to all other competing algorithms. These small STD results divulge that the dominance of ECSA, PCSA, and SCSA is well-established. Eventually, these results prove that the ECSA, PCSA, and SCSA have excellent and stable performance on average. In short, as can be seen, statistically, the proposed ECSA, PCSA, and SCSA are superior to CSA, MVO, PSO, GSA, SCA, GA, SHO, GWO, and EPS, as the mean outcomes of ECSA, PCSA, and SCSA in the majority of fixed-dimensional functions over 30 separate runs is, in each function, a much lower.

5.3.3 Implementation time

In addition to the average (AVG) and standard deviation (STD) values used in comparisons of competing algorithms above, the computational time taken by the algorithms to accomplish computations is also a root criterion that is frequently used to exemplify the efficacy of algorithms. Alternatively stated, the computational time is crucial to reveal whether the computational burden of new amended optimization algorithms is satisfactory and within the bounds of other competing methods. The proposed algorithms of CSA (i.e., ECSA, PCSA, SCSA) and the other competing algorithms were carried out in MATLAB 2021 A platform. All of these algorithms were carried out under identical conditions on a Windows 10 with an Intel Core i7-5200U^{TM}

Table 4 Results of ECSA, PCSA, SCSA, and other optimization algorithms in fixed-dimensional test functions

F	ECSA		PCSA		SCSA		CSA	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F_{14}	**9.98E-01**	3.43E-02	**9.98E-01**	9.95E-02	**9.98E-01**	5.64E-02	**9.98E-01**	**0.00**
F_{15}	3.27E-03	5.61E-03	4.41E-03	8.11E-03	**1.23E-03**	3.64E-03	3.19E-03	6.85E-03
F_{16}	-1.0316	5.77E-16	-1.0316	6.19E-16	-1.0316	6.42E-16	-1.0316	6.51E-16
F_{17}	**3.97E-01**	**0.00**	**3.97E-01**	**0.00**	3.97E-01	**0.00**	**3.97E-01**	**0.00**
F_{18}	**2.99**	2.16E-15	**2.99**	2.13E-15	**2.99**	2.04E-15	**2.99**	**1.87E-15**
F_{19}	-3.862	2.61E-15	-3.862	2.42E-15	-3.862	**2.37E-15**	-3.862	2.69E-15
F_{20}	-3.27	4.79E-02	-3.27	5.29E-02	-3.27	5.92E-02	-3.31	**2.17E-02**
F_{21}	**-2.05**	3.31	-6.72	3.38	-6.57	3.69	-1.01E+01	5.62E-15
F_{22}	-5.43	3.52	-3.96	3.80	-4.61	3.53	-1.04E+01	4.66E-16
F_{23}	-6.00	3.67	-5.50	3.70	-4.55	3.84	-1.05E+01	1.58E-15
F	PSO		MVO		SCA		GSA	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F_{14}	2.77	2.32	**9.98E-01**	9.14E-12	1.26	6.86E-01	3.61	2.96
F_{15}	9.09E-03	**2.38E-03**	7.15E-02	1.26E-01	1.01E-02	3.75E-03	6.84E-02	73.70E-03
F_{16}	**-1.02**	**0.00**	**-1.02**	4.74E-08	**-1.02**	3.23E-05	**-1.02**	**0.00**
F_{17}	**3.97E-01**	9.03E-16	3.98E-01	1.15E-07	3.98E-01	7.61E-04	3.98E-01	11.30E-17
F_{18}	3.00	6.59E-05	3.00	1.48E+01	3.00	2.25E-05	3.00	32.40E-03
F_{19}	-3.80	3.37E-15	-3.77	3.53E-07	-3.75	2.55E-03	-3.86	41.50E-02
F_{20}	-3.32	2.66E-01	-3.23	5.37E-02	-2.84	3.71E-01	-1.47	53.20E-02
F_{21}	-7.54	2.77	-7.38	2.91	-2.28	1.80	-4.57	1.30
F_{22}	-8.55	3.08	-8.50	3.02	-3.99	1.99	-6.58	2.64
F_{23}	-9.19	2.52	-8.41	3.13	-4.49	1.96	-9.37	2.75
F	SHO		GWO		GA		EPO	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F_{14}	9.68	3.29	3.71	3.86	4.39	4.41E-02	1.08	41.10E-03
F_{15}	9.01E-03	1.06E-03	3.66E-02	7.60E-02	7.36E-02	2.39E-03	8.21E-03	40.90E-04
F_{16}	-1.03	2.86E-11	**-1.02**	7.02E-09	**-1.02**	4.19E-07	**-1.02**	98.00E-08
F_{17}	**3.97E-01**	2.46E-01	3.98E-01	7.00E-07	3.98E-01	3.71E-17	3.98E-01	53.90E-06
F_{18}	3.00	9.05	3.00	7.16E-06	3.00	6.33E-07	3.00	11.50E-09
F_{19}	**-3.71**	4.39E-01	-3.84	1.57E-03	-3.81	4.37E-10	-3.86	65.00E-08
F_{20}	**-1.44**	5.47E-01	-3.27	7.27E-02	-2.39	4.37E-01	-2.81	71.10E-02
F_{21}	-2.08	**3.80E-01**	-9.65	1.54	-5.19	2.34	-8.07	2.29
F_{22}	**-1.61**	**2.04E-04**	-1.04	2.73E-04	-2.97	1.37E-02	-10.01	39.70E-03
F_{23}	**-1.68**	2.64E-01	-1.05E+01	**1.81E-04**	-3.10	2.37	-3.41	11.10E-03

CPU at 2.2 GHz and 8.0 GB of RAM. This is to ascertain that the comparison is fair. Table 5 shows the average execution times and their associated standard deviation values taken by ECSA, PCSA, SCSA, and all other competitors, over 30 separate runs, in optimizing the functions (i.e., F_1 to F_{23}). The best results in this table are emboldened to give them more significance than the other results.

The comparison presented in Table 5 between the competitor algorithms was made in respect of the execution time that the algorithm takes to accomplish the computation. Although the execution times of ECSA, PCSA, and SCSA, as displayed in this table, are sometimes slightly more significant than those of CSA and some of the other algorithms, the increase in execution times of these proposed algorithms is not significant in many cases. Appropriately, the execution times of the proposed algorithms were reasonable since they are fallen within the scope of execution times of other algorithms. In short, the proposed ECSA, PCSA, and SCSA algorithms' computational times are relatively short and associated with small Std values.

Table 5 Results of computational times (in seconds) of the proposed algorithms and other competing algorithms in solving benchmark test functions F_1 - F_{23}

F	Metric	ECSA	PCSA	SCSA	CSA	SHO	GWO	PSO	MVO	SCA	GSA	GA	EPO
F_1	AVG	1.2011	1.0684	1.1083	0.7801	1.2760	1.6144	**0.7749**	2.1986	2.1943	1.4449	1.2760	0.8589
	STD	0.3960	0.4086	0.4118	0.3956	0.7027	0.7964	**0.3917**	1.1359	1.1626	0.7449	0.7027	0.4436
F_2	AVG	1.3402	0.9749	1.0877	**0.6990**	1.2473	1.5244	0.8148	2.1086	2.0496	1.3625	1.2473	0.7913
	STD	**0.4447**	0.5335	0.5018	0.4772	0.7058	0.8866	0.3994	1.2700	1.1209	0.7132	0.7058	0.4667
F_3	AVG	2.7190	2.1961	2.1581	**1.9749**	2.8200	2.8183	2.0430	3.6284	2.0681	3.1176	2.8200	2.5448
	STD	1.5607	1.2599	1.2346	1.1437	1.8580	1.6018	**1.1233**	2.0350	1.2558	1.7982	1.8580	1.4691
F_4	AVG	1.3207	0.8290	1.0575	**0.6241**	2.3015	1.2904	0.6508	2.3000	2.4572	1.3718	2.3015	0.8354
	STD	0.7228	0.4745	0.6009	**0.3489**	1.2658	0.7623	0.3602	1.2820	1.2610	0.7597	1.2658	0.4631
F_5	AVG	1.7359	1.2469	1.3613	1.1030	2.6449	2.0170	**1.0283**	3.0772	1.8931	1.9922	2.6449	1.2506
	STD	1.0243	0.6827	0.7700	0.6067	1.4526	1.1449	**0.5820**	1.9437	1.0801	1.0624	1.4526	0.6552
F_6	AVG	1.2253	0.8466	0.9441	**0.6023**	2.0106	1.3746	0.6479	1.9824	1.8794	1.4041	2.0106	0.6395
	STD	0.6669	0.4901	0.5400	**0.3323**	1.1230	0.7694	0.3512	1.1759	1.1138	0.7876	1.1230	0.3788
F_7	AVG	2.2518	1.6044	1.8633	1.5796	3.3438	2.6673	**1.5228**	3.4718	2.0617	2.4616	3.3438	1.6765
	STD	1.3033	0.9457	1.0405	0.8674	1.8774	1.4989	**0.8653**	1.9379	1.1871	1.4017	1.8774	0.9595
F_8	AVG	1.7182	0.9455	1.2104	0.9476	2.4399	1.7041	**0.8457**	1.9647	1.9769	1.6315	2.4399	0.9767
	STD	0.9395	0.5384	0.6832	**0.5016**	1.3337	0.9763	0.5088	1.1432	1.0926	0.8992	1.3337	0.5721
F_9	AVG	1.3268	0.8853	0.9957	0.7853	2.3267	1.3382	**0.6876**	2.3873	2.2004	1.3956	2.3267	1.0274
	STD	0.8077	0.5005	0.5842	0.4351	1.2554	0.8022	**0.3804**	1.3088	1.2612	0.8020	1.2554	0.5407
F_{10}	AVG	1.4868	0.9445	1.0437	0.8069	2.3557	1.5850	**0.7337**	2.3691	2.0803	1.3764	2.3557	0.9644
	STD	0.8400	0.5281	0.5781	0.4505	1.3338	0.9046	**0.4381**	1.2702	1.2388	0.8041	1.3338	0.5401
F_{11}	AVG	2.0739	1.1835	1.3210	1.0448	2.7112	1.9830	**1.0061**	2.6545	2.0441	1.9159	2.7112	1.2478
	STD	1.1902	0.6845	0.7592	**0.5826**	1.5242	1.1193	0.5939	1.6066	1.1507	1.0716	1.5242	0.6573

Table 5 continued

F	Metric	ECSA	PCSA	SCSA	CSA	SHO	GWO	PSO	MVO	SCA	GSA	GA	EPO
F_{12}	AVG	4.7037	3.5033	3.7520	3.4145	6.0310	4.6815	3.4495	5.9345	**2.1522**	4.6242	6.0310	4.4346
	STD	2.6898	1.9583	2.0828	2.0068	3.4170	2.6718	1.9689	3.3202	**1.2638**	2.6475	3.4170	2.4816
F_{13}	AVG	4.6213	3.5375	3.6659	3.5688	5.4460	4.8408	3.3723	5.8627	**2.2055**	4.8930	5.4460	4.0912
	STD	2.6820	1.9962	2.0435	2.0398	3.0645	2.7130	1.9662	3.2641	**1.2784**	2.8695	3.0645	2.3118
F_{14}	AVG	13.9432	11.0024	10.4939	10.7455	14.9488	14.5606	11.2409	15.3779	**2.4527**	13.8411	14.9488	13.2094
	STD	7.8223	6.2784	5.9342	6.0572	8.4406	8.3444	6.3942	8.6558	**1.3531**	8.0631	8.4406	7.5177
F_{15}	AVG	**0.6359**	0.7341	0.7105	0.6779	1.9591	1.1527	0.7467	1.4527	1.9744	1.0355	1.9591	0.6880
	STD	0.4954	0.5588	0.5231	**0.3452**	1.1045	0.6973	0.4236	0.8203	1.1488	0.6088	1.1045	0.4318
F_{16}	AVG	0.7076	0.7338	0.7278	**0.6524**	1.8093	1.1478	0.7445	1.4118	2.0560	1.0691	1.8093	0.7819
	STD	0.4311	**0.3731**	0.4291	0.3746	1.0528	0.6888	0.4344	0.7941	1.1422	0.6246	1.0528	0.4273
F_{17}	AVG	0.8435	1.0896	0.6461	0.5668	2.9676	1.1251	**0.5274**	1.5005	2.1370	0.8590	2.9676	0.5567
	STD	0.4308	0.5700	0.3365	**0.2739**	1.5921	0.5886	0.2842	0.7990	1.1311	0.4766	1.5921	0.3156
F_{18}	AVG	0.6438	0.8296	0.6606	0.3801	1.6481	0.9199	0.5390	1.1560	1.7806	0.7410	1.6481	**0.3789**
	STD	0.3645	0.4201	0.3653	**0.2136**	0.8734	0.5388	0.3086	0.6215	0.9879	0.3991	0.8734	0.2287
F_{19}	AVG	1.1276	0.8990	0.9072	0.7243	2.0884	1.2718	0.9169	1.6770	1.9994	1.2834	2.0884	**0.7013**
	STD	0.6257	0.5320	0.5089	0.4250	1.2172	0.6981	0.5395	0.9225	1.1416	0.6864	1.2172	**0.3928**
F_{20}	AVG	**0.8819**	0.8894	0.8830	0.8835	2.1534	1.4573	1.0976	1.8967	1.9177	1.4813	2.1534	0.9056
	STD	0.6855	0.5642	0.5203	0.5146	1.1968	0.8689	0.5690	1.0317	1.1012	0.8111	1.1968	**0.4829**
F_{21}	AVG	0.92272	0.8974	**0.8401**	0.8448	2.3263	1.5256	0.9706	1.7110	1.8800	1.3206	2.3263	0.9645
	STD	0.7039	0.6044	0.5535	**0.4635**	1.2605	0.8698	0.5639	1.0378	1.0677	0.7505	1.2605	0.5527
F_{22}	AVG	1.3008	**1.1768**	1.1288	1.2361	2.3664	1.7165	1.0830	1.7734	1.8844	1.6431	2.3664	1.1463
	STD	0.7629	0.6905	0.6135	**0.5685**	1.3365	0.9811	0.6266	1.0458	1.1046	0.9143	1.3365	0.6308
F_{23}	AVG	1.6735	1.4363	**1.2097**	1.2127	2.7880	2.0737	1.2957	2.2381	2.0608	1.7298	2.7880	1.2908
	STD	0.7083	0.7003	**0.6514**	0.6820	1.5397	1.2242	0.7199	1.3274	1.1630	1.0137	1.5397	0.7383

5.4 Convergence curves of the developed algorithms

The most common qualitative results of optimization techniques utilized in the literature are convergence curves. In this situation, an algorithm's best results to date are recorded after each iteration loop. To demonstrate how closely an algorithm approximates the global optimal solutions over a certain number of iterations, the convergence curves are represented as lines. The convergence curves of the proposed ECSA, PCSA and SCSA, and the parent CSA, in respect of the best fitness values of all test functions of the standard benchmark functions, were obtained in a two-dimensional environment, as shown in Figs. 3 and 4, where the preceding Fig. 3 exhibits the convergence curves for F_1 to F_{12}, while the latter Fig. 4 presents the curves for F_{13} to F_{23}.

The convergence curves of the three algorithms derived from CSA are displayed over 1000 iterations specified on the x-axis in Figs. 3 and 4 against the best fitness values obtained so far on the y-axis. In these curve trends, the best optimization method is the one that illustrates rapid convergence and reaches a minor error. This implies that we support the algorithm that settles at a low fitness value after a few number of iterations. An attractive outcome is that all the proposed algorithms of CSA have credibly found the minimum of $F_1 - F_{12}$. Hence, these problems may be relatively easy to solve

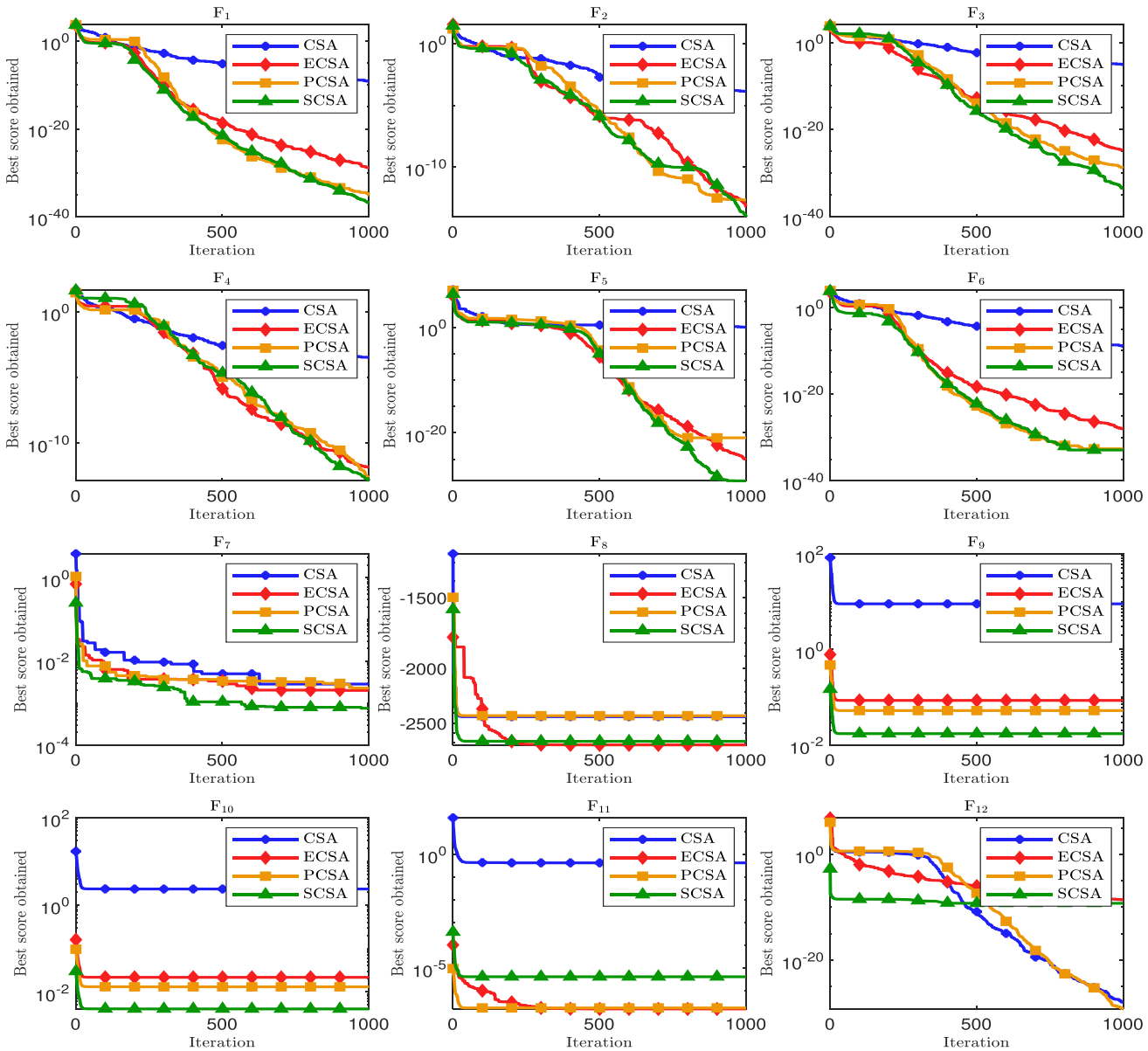

Fig. 3 Convergence curves of the proposed algorithms, ECSA, PCSA, SCSA, and the basic CSA for F_1 - F_{12}

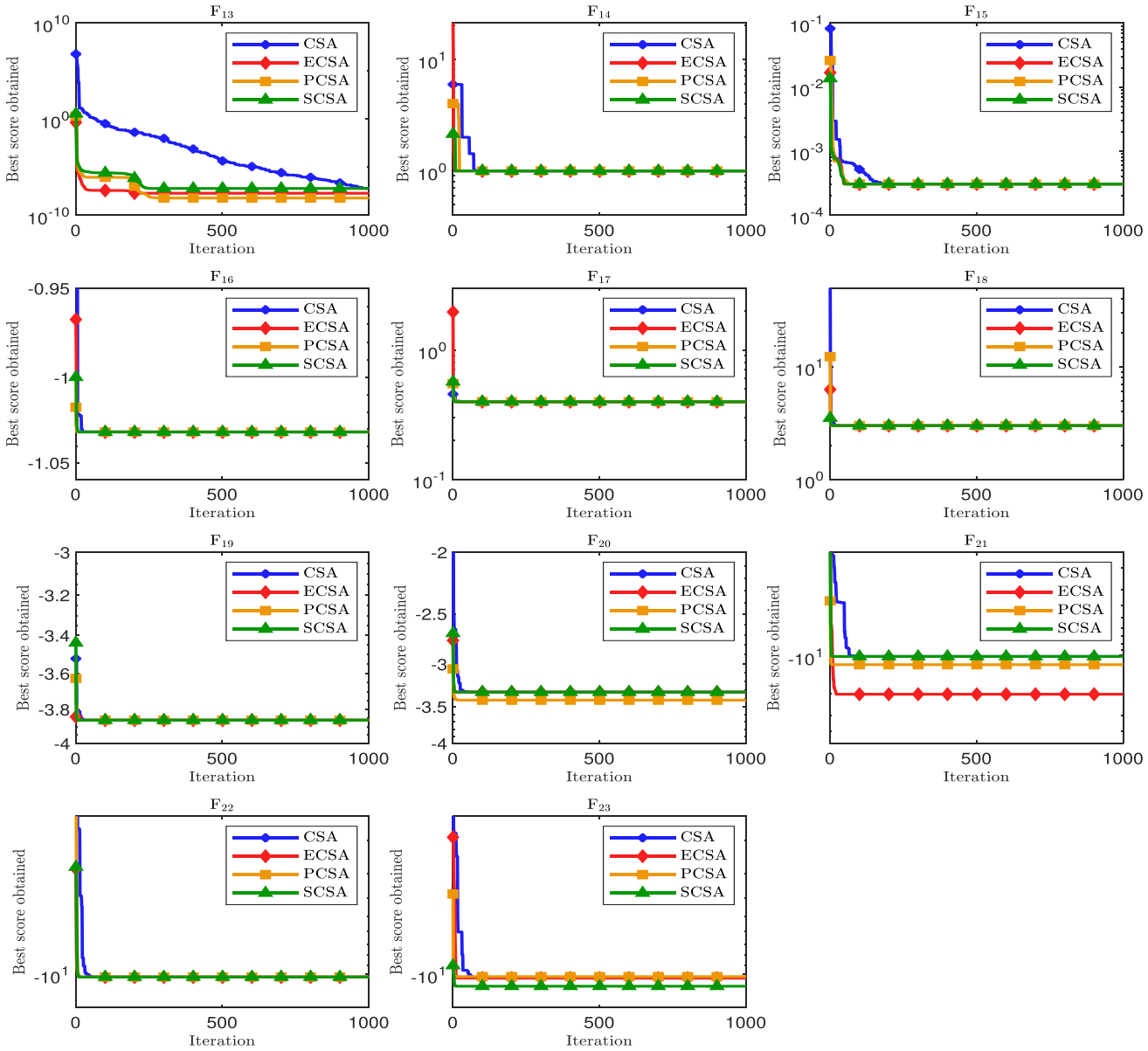

Fig. 4 Convergence curves of the proposed algorithms, ECSA, PCSA, SCSA, and the basic CSA for $F_{13}-F_{23}$

with a 100% success rate. Looking deeply at Figs. 3 and 4, one can notice reasonable variations in the behaviors of the three developed algorithms of CSA. This variation is ascribed to how the optimization method acts regarding exploitation and exploration features.

This also varies from one test function to another test function for the same optimization algorithm based on the nature of the function. In general, the convergence curves stabilize at or after approximately 200 iterations. The proposed versions of CSA show good convergence behaviors in all test functions and outperform the native CSA. Looking at each plot separately, one can say that SCSA has surpassed its companion versions in benchmark function F_{21}, where it realized a meager fitness value in less than ten iterations. After then, the curve continued to fall, although only very little. Again, one can perceive that SCSA excelled in getting the best visual results, followed by PCSA, which behaved similarly to PCSA, and both are superior to ECSA, and CSA is the worst among them. Overall, the convergence curves in Figs. 3 and 4 demonstrate that the proposed ECSA, PCSA, and SCSA maintain an appropriate balance between exploration and exploitation for locating the optimal global solution reliably. Thus, the success rate of these algorithms is high for solving optimization functions.

5.5 Qualitative analysis of ECSA

To show the qualitative outcomes of ECSA, four metric measures were used in a 2-D environment given by solving functions F_1, F_4, F_6, F_{10}, F_{11} and F_{16}.

The metric measures employed in the qualitative results can be characterized as follows:

- The first metric displays the convergence of the best global crow through a path of iterations. Convergence analysis of optimization algorithms is vital to grasp methods' exploration and exploitation features better. Remarkably, the results of the convergence curves reveal that ECSA exhibits sensible convergence rates and has encouraging conduct in all of the considered functions. Within the first 500 iterations, ECSA quickly converges to the most favorable regions in the search space. In the following 500 iterations, ECSA gradually converges to the global or near-global optimal in F_1, F_6, and F_{10}.

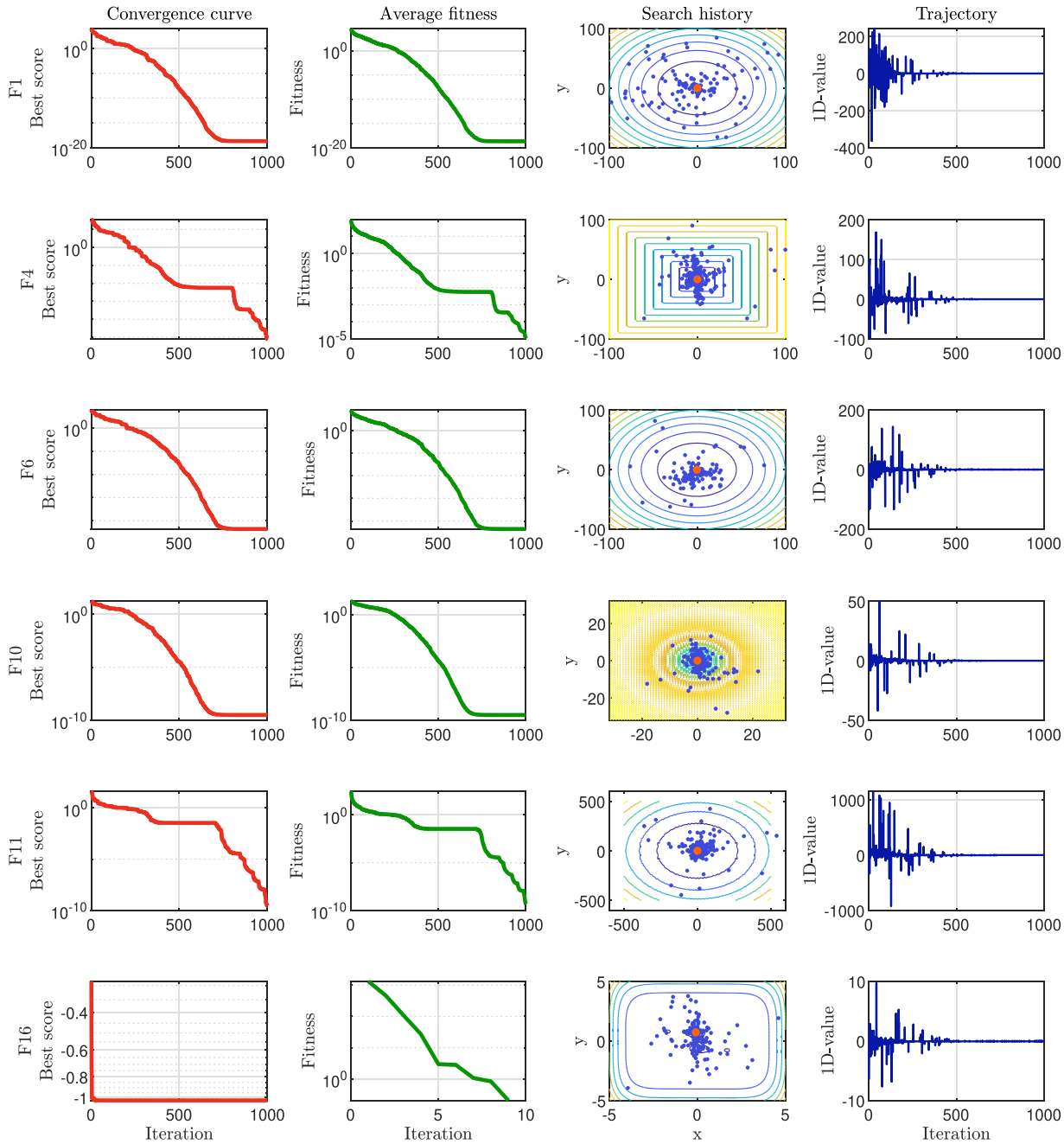

Fig. 5 Qualitative results for F_1, F_4, F_6, F_{10}, F_{11} and F_{16}: convergence curve, the average fitness of all crows, search history and path in the first dimension of the first crow

For F_4 and F_{11}, ECSA continued to approach the global solutions. The stability of ECSA with various kinds of functions supports the ability to obtain a steady convergence. The soft convergence curves and the regularity with which these curves converge to the slightest error throughout an iterative process serve as evidence. The crows are prompted to travel to the global optima and are prompted to move locally rather than globally, demonstrating how ECSA utilizes the search space.

- Average fitness curves read the average objective values of all crows at each iteration of ECSA. These fitness curves decrease significantly throughout iterations in all tested test functions. In more detail, ECSA gave a fast convergence response for F_1, F_6, F_{10} and F_{16}, has rational convergence for F_4, and found optimal solutions with a sensible convergence response for F_{11}. This ascertains that ECSA promotes the global best crow and enhances the fitness values of all crows.
- The search history records the crows' positions during optimization. ECSA is exploring the most favorable regions in the search space for the benchmark functions. The proposed ECSA does not become stuck in local optima and is exploring the whole search space. For F_1, F_4, and F_6, the sample points are slightly split into the unpromising regions. Most of the sample points in the test functions, F_{10} - F_{11}, and F_{16}, are distributed around unpromising regions. This is owing to the complexity of these test functions. This insinuates that ECSA explored the whole search space and averted falling into local optima. The sample points are dispersed around the optimal solution, which ensures that ECSA can diversify and intensify the search process in the search space efficiently and effectively.
- At each loop of the iterative ECSA process, the first crow's path shows the value of the first variable. The route curves demonstrate that crows exhibit significant favorable regions during the initial stages of optimization.

The average fitness and convergence curves in Fig. 5 demonstrate that ECSA has an appropriate balance between exploration and exploitation to effectively locate the global optimal solution. Generally, the above qualitative outcomes substantiate the coveted features of ECSA and affirm that its success level in addressing benchmark test functions is mainly reliable. This is attributed to the robust global and local search mechanisms of the developed ECSA in the search space.

5.6 Evaluation of CEC-2015 benchmark

The performance level of the proposed variants of CSA was evaluated using a more complicated benchmark set, CEC-2015. These functions include hybrid and composition test functions [91]. Therefore, they are valuable for assessing the exploration and exploitation features of the proposed algorithms. These test functions have a search space equal to $[-100, 100]$ with dimensions equal to 30 for each. Appendix A in Table 24 details these functions. The settings of the parameters for ECSA, PCSA, and SCSA and all other competing algorithms used in this test set are exhibited in Table 1. The number of FEs for each function was assigned to 30,000, considering that each method's number of iterations and crows are 1000 and 30, respectively. The performance degree of ECSA, PCSA and SCSA, and other competing methods on the CEC-2015 benchmark group is provided in Table 6.

Table 6 compares the performance of the proposed ECSA, PCSA, and SCSA algorithms with the primary CSA and eight other meta-heuristics for CEC-2015 benchmark functions. It is evident from this table that ECSA, PCSA, and SCSA excelled other algorithms in the majority of the studied functions. In this context, ECSA achieved better mean values for test functions C15-f1, C15-f4, C15-f8, and C15-f11. PCSA arrived at better mean scores for C15-f2 and C15-f5, and SCSA realized better mean scores for C15-f6 and C15-f10. Additionally, the proposed ECSA, PCSA, and SCSA yielded findings near optimality for C15-f3, C15-f7, C15-f9, C15-f12, C15-f13, and C15-f15, respectively, and these results are analogous to other algorithms. For the test functions C15-f3 and C15-f9, all the competing methods achieved the same results of 3.20E+02 and 1.00E+03, respectively. Also, CSA produced near-optimal outcomes for the C15-f4 test function comparable to the results revealed by ECSA, PCSA, and SCSA. On the C15-f7 test function, CSA reported an average fitness value of 7.11E+02, close to the result of 7.02 E+02 reported by ECSA, PCSA, and SCSA. For the C15-f8 test function, ECSA reported an optimal result of 1.43E+03, whereas PCSA, SCSA, and CSA reported good results close to optimality for this function 1.65E+03, 1.56E+03, and 7.75E+03, respectively. For C15-f10, SCSA reported an optimal result of 1.17E+03, whereas ECSA and PCSA reported 1.24E+03 and 2.87E+03, respectively, comparable to the result obtained by SCSA. However, CSA did not perform as well as ECSA, PCSA, and SCSA for the C15-f10 test function. Even though ECSA, PCSA, and SCSA have very sensible results in CEC-2015 benchmark functions, the edges of differences between the mean scores obtained by CSA and those obtained by ECSA, PCSA, and SCSA in the C15-f14 test function are minimal. Still, the standard deviations of ECSA, PCSA, and SCSA are minimal compared to that reported by CSA for this function. For the remaining functions, C15-f11, C15-f12, C15-f13, and C15-f15, ECSA, PCSA, and SCSA scored optimal or near-optimal average outcomes. In sum, the results obtained by ECSA, PCSA, and SCSA are outstanding and almost better

Table 6 Results of ECSA, PCSA, and SCSA and other methods on the CEC-2015 test group

F	Metric	ECSA	PCSA	SCSA	CSA	SHO	GWO	PSO	MVO	SCA	GSA	GA	EPO
C15-f1	AVG	**1.03E+05**	1.11E+05	1.08E+05	2.75E+06	2.28E+06	2.02E+06	4.37E+05	1.47E+06	6.06E+05	7.65E+06	3.20E+07	1.50E+05
	STD	3.60E+05	**1.39E+05**	7.50E+05	4.01E+06	2.18E+06	2.08E+06	4.73E+05	2.63E+06	5.02E+05	3.07E+06	8.37E+06	1.21E+06
C15-f2	AVG	2.02E+03	**2.00E+03**	2.11E+03	5.34E+05	3.13E+05	5.65E+06	9.41E+03	1.97E+04	1.43E+04	7.33E+08	4.58E+03	6.70E+06
	STD	3.68E+04	9.90E+04	4.39E+04	1.77E+06	4.19E+05	6.03E+06	1.08E+04	1.46E+04	1.03E+04	2.33E+08	**1.09E+03**	1.34E+08
C15-f3	AVG	**3.20E+02**	**3.20E+02**	**3.20E+02**	**3.20E+02**	**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	1.30E-05	1.29E-05	1.38E-05	2.12E-01	3.76E-02	7.08E-02	8.61E-02	9.14E-02	3.19E-02	7.53E-02	**1.11E-05**	1.16E-03
C15-f4	AVG	**4.09E+02**	4.15E+02	4.13E+02	4.87E+02	4.11E+02	4.16E+02	**4.09E+02**	4.26E+02	4.18E+02	4.42E+02	4.39E+02	4.10E+02
	STD	4.44E+01	3.82E+01	3.56E+01	2.91E+01	1.71E+01	1.03E+01	**3.96E+00**	1.17E+01	1.03E+01	7.72E+00	7.25E+00	5.61E+01
C15-f5	AVG	3.38E+02	**3.12E+02**	3.35E+02	2.23E+03	9.13E+02	9.20E+02	8.65E+02	1.33E+03	1.09E+03	1.76E+03	1.75E+03	9.81E+02
	STD	6.11E+02	6.95E+02	6.85E+02	5.92E+02	1.85E+02	**1.78E+02**	2.16E+02	3.45E+02	2.81E+02	2.30E+02	2.79E+02	2.06E+02
C15-f6	AVG	1.27E+03	2.24E+03	**1.13E+03**	1.94E+04	1.29E+04	2.26E+04	1.86E+03	7.35E+03	3.82E+03	2.30E+04	3.91E+06	2.05E+03
	STD	1.97E+04	6.12E+04	6.01E+04	7.87E+04	1.15E+04	2.45E+04	**1.93E+03**	3.82E+03	2.44E+03	2.41E+04	2.70E+06	1.05E+04
C15-f7	AVG	**7.02E+02**	**7.02E+02**	**7.02E+02**	7.11E+02	**7.02E+02**	**7.02E+02**	**7.02E+02**	**7.02E+02**	**7.02E+02**	7.06E+02	7.08E+02	**7.02E+02**
	STD	1.72E+00	2.74E+00	2.58E+00	9.19E+00	6.76E-01	7.07E-01	7.75E-01	1.10E+00	9.40E-01	9.07E-01	1.32E+00	**5.50E-01**
C15-f8	AVG	**1.43E+03**	1.65E+03	1.56E+03	7.75E+03	1.86E+03	3.49E+03	3.43E+03	9.93E+03	2.58E+03	6.73E+03	6.07E+05	1.47E+03
	STD	3.37E+03	**1.47E+03**	2.49E+03	3.47E+03	1.98E+03	2.04E+03	2.77E+03	8.74E+03	1.61E+03	3.36E+03	4.81E+05	2.34E+03
C15-f9	AVG	**1.00E+03**	**1.00E+03**	**1.00E+03**	**1.00E+03**	**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	1.08E+00	1.08E+00	1.13E+00	1.63E+00	1.43E-01	1.28E-01	7.23E-02	2.20E-01	**5.29E-02**	9.79E-01	5.33E+00	1.51E+01
C15-f10	AVG	1.24E+03	2.87E+03	**1.17E+03**	1.56E+04	2.00E+03	4.00E+03	3.27E+03	8.39E+03	2.62E+03	9.91E+03	3.42E+05	1.23E+03
	STD	1.58E+04	1.82E+04	4.22E+04	3.86E+04	2.73E+03	2.82E+03	1.84E+03	1.12E+04	**1.78E+03**	8.83E+03	1.74E+05	2.51E+04
C15-f11	AVG	**1.30E+03**	1.31E+03	1.31E+03	1.40E+03	1.38E+03	1.40E+03	1.35E+03	1.37E+03	1.39E+03	1.35E+03	1.41E+03	1.35E+03
	STD	2.29E+02	2.81E+02	2.79E+02	3.69E+02	2.42E+01	5.81E+01	1.12E+02	8.97E+01	5.42E+01	1.11E+02	7.73E+01	**1.41E+01**
C15-f12	AVG	**1.30E+03**	**1.30E+03**	**1.30E+03**	1.31E+03	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	1.31E+03	1.31E+03	**1.30E+03**
	STD	4.70E+01	4.56E+01	4.88E+01	1.33E+01	7.89E-01	**6.69E-01**	6.94E-01	9.14E-01	8.07E-01	1.54E+00	2.05E+00	7.50E+00
C15-f13	AVG	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	**1.30E+03**	1.35E+03	**1.30E+03**
	STD	6.03E-01	8.40E-01	8.37E-01	9.39E-01	2.76E-04	1.92E-04	5.44E-03	1.04E-03	2.43E-04	3.78E-03	4.70E+01	**6.43E-05**
C15-f14	AVG	3.20E+03	3.72E+03	3.13E+03	**1.92E+03**	4.25E+03	7.29E+03	7.10E+03	7.60E+03	7.34E+03	7.51E+03	9.30E+03	3.22E+03
	STD	2.70E+04	2.33E+04	2.19E+04	2.38E+04	1.73E+03	2.45E+03	3.12E+03	**1.29E+03**	2.47E+03	1.52E+03	4.04E+02	2.12E+03
C15-f15	AVG	**1.60E+03**	**1.60E+03**	**1.60E+03**	**1.60E+03**	**1.60E+03**	1.61E+03	**1.60E+03**	1.61E+03	**1.60E+03**	1.62E+03	1.64E+03	**1.60E+03**
	STD	1.27E+00	2.65E+00	1.85E+00	6.89E-01	3.76E+00	4.94E+00	**2.66E-07**	1.13E+01	1.80E-02	3.64E+00	1.12E+01	5.69E+01

than the competitors, namely CSA, GWO, SHO, MVO, PSO, GSA, EPO, SCA, and GA, in the majority of the CEC-2015 test functions. Regarding the standard deviation outcomes, ECSA, PCSA, and SCSA have tiny figures in most of these functions compared to others. This confirms that the excellence of these algorithms is stable, and this stability is solid.

5.7 Evaluation of CEC-2017 benchmark

For further testing, the degree of reliability of the proposed ECSA, PCSA, and SCSA against more challenging benchmark problems, the CEC-2017 as a recent and challenging test group was employed. This group comprises hybrid and composition functions, as well as unimodal, multimodal, and multimodal functions that have been rotated and shifted [82]. As a result, it is sufficiently adequate for rating the exploration and exploitation conducts of the evolved algorithms. It is worth mentioning that given the unsteady behavior of the C17-f2 function, it was taken away from this suite. The search area for all of these functions in this test suite is $[-100, 100]$ with ten dimensions for each test problem. Due to the high complexity of these problems and because they contain a lot of local optima, they were chosen to judge the strength level of the evolved algorithms in avoiding local optimums and getting the optimal global solutions. Appendix A in Table 25 gives more details about these functions.

The performance of ECSA, PCSA, and SCSA was tested via this test set, and the outcomes were compared with previous meta-heuristics. The majority of the functions in this group rank among the most difficult hybrid and compositional functions. All competing methods' findings were gathered using 50 search agents, 1000 iterations, and 30 separate runs. Taking into account the predetermined number of iterations and search agents, 50,000 FEs were allocated to each test function. The parameter settings of the competing methods can be found in Table 1. The outcomes of ECSA, PCSA and SCSA, and other comparative algorithms are displayed in Table 7.

The outcomes in Table 7 underscore the notability of the proposed ECSA, PCSA, and SCSA over other meta-heuristics in optimizing challenge functions. These algorithms presented the best exclusive average fitness scores in 6 out of 29 functions (C17-f1, C17-f3, C17-f4, C17-f19, C17-f22, and C17-f23). In further detail, the proposed algorithms have remarkable performance degrees in uni-modal test problems (C17-f1, C17-f3), where they could constantly find the global optimum solution over 30 separate runs. Also, they could find the optimum solutions in three cases in the hybrid functions, namely C17-f15, C17-f18, and C17-f19. It is obviously seen that ECSA is the best algorithm among all other competitors, which scored the best average results in C17-f7, C17-f10, C17-f17, C17-f21, C17-f18, C17-f11, and C17-f14. At the same time, PCSA is the second-best optimizer that scored the best results in 8 out of 29 test functions, namely C17-f4, C17-f1, C17-f3, C17-f23, C17-f19, C17-f25, C17-f9 and C17-f16. Besides, SCSA is the third best optimizer, which reported the best average outcomes in C17-f29, C17-f26, C17-f30, C17-f24, C17-f5, and C17-f20. Moreover, ECSA and PCSA reported the best average scores in the C17-f25 test function. This confirms that we have formerly settled that SCSA has high accuracy when solving optimization functions in different search domains. However, the proposed ECSA, PCSA, and SCSA flopped to acquire the optimal solutions in a few test functions, such as C17-f6. As a result, these proposed algorithms are sometimes trapped in local optimums, but they are not far from the global optimum.

For the C17-f8 test function, CSA reported the best average score of 811.04, whereas SCSA, ECSA, and PCSA reported the second, third, and fourth best scores with slight differences from that reported by CSA. For the C17-f9 function, PCSA achieved the best mean value, whereas ECSA and SCSA reported the third and fourth best outcomes with modicum differences from that reported by PCSA. For the C17-f12 test function, the performance of GWO is better than other methods in respect of the average fitness value. SHO is the best optimizer for C17-f13, which reported the best average score and is better than ECSA, PCSA, and SCSA, with slight difference margins. In connection with the composition functions that make up the most complicated functions in CEC-2017, the optimal solutions were revealed by the proposed methods in C17-23, C17-25, and C17-26 test cases. ECSA and SCSA won the best average fitness value for C17-f27. Hence, the performance of ECSA, PCSA, and SCSA is better than other algorithms regarding average fitness values, and they won the top three optimizers for these test functions. Regarding the STD figures in Table 7, the presented ECSA, PCSA, and SCSA performed remarkably better than other optimization methods in most test functions. This sustains the conviction that the proposed algorithms have significant stability when applied to complex test functions in different search areas. These results divulge that ECSA, PCSA, and SCSA are ranked first, following their strength in exploration and exploitation capacities.

Lastly, CSA, SHO, and GWO algorithms provided plausible solutions in different test functions of CEC-2017. The SCA, GSA, and GA algorithms behaved almost poorly in these test functions, while PSO and MVO behaved almost modestly. Overall, SCSA, ECSA, PCSA, SHO, and GWO functioned much better than the others in most CEC-2017 functions. The top three optimizers were the overall ranking of the proposed ECSA, PCSA, and SCSA algorithms in this test bed.

Table 7 Results of ECSA, PCSA, and SCSA and other algorithms in the CEC-2017 test group

F	Metric	ECSA	PCSA	SCSA	CSA	SHO	GWO	PSO	MVO	SCA	GSA	GA
C17-f1	AVG	**100E+00**	**100E+00**	**100E+00**	853.43E+00	2.38E+05	2.12E+05	4.47E+04	1.57E+05	6.16E+04	7.75E+05	3.30E+06
	STD	**8.20E-15**	**8.20E-15**	**6.69E-15**	1.02E+03	2.28E+07	2.18E+07	4.83E+06	2.73E+07	5.12E+06	3.17E+07	8.47E+07
C17-f3	AVG	**300E+00**	**300E+00**	**300E+00**	**300E+00**	3.30E+02	3.30E+02	3.30E+02	3.30E+02	3.30E+02	3.30E+02	3.30E+02
	STD	**3.78E-14**	**3.28E-14**	**4.23E-14**	8.47E-06	3.86E-03	7.18E-03	8.71E-03	9.24E-03	3.29E-03	7.63E-03	1.21E-06
C17-f4	AVG	**400E+00**	**400E+00**	**400E+00**	**400E+00**	4.21E+02	4.26E+02	4.19E+02	4.36E+02	4.28E+02	4.52E+02	4.49E+02
	STD	5.91E-02	**1.84E-01**	7.16E-05	7.42E-01	1.81E+02	1.13E+02	3.06E+01	1.27E+02	1.13E+02	7.82E+02	7.35E+02
C17-f5	AVG	518.35E+00	516.57E+00	**514.22E+00**	519.10E+00	9.23E+02	9.30E+02	8.75E+02	1.43E+04	1.19E+04	1.86E+03	1.85E+03
	STD	2.18E+00	1.21E+00	**1.11E+00**	4.29E+00	1.95E+03	1.88E+03	2.26E+03	3.55E+04	2.91E+03	2.40E+03	2.89E+03
C17-f6	AVG	605.74E+00	604.63E+00	605.59E+00	**601.60E+00**	1.39E+04	2.36E+03	1.96E+03	7.45E+03	3.92E+04	2.40E+04	3.01E+05
	STD	1.28E+00	7.20E+00	**1.17E+00**	1.98E+00	1.25E+05	2.55E+05	1.03E+04	3.92E+04	2.54E+04	2.51E+05	2.80E+07
C17-f7	AVG	**7.11E+02**	7.12E+02	7.16E+02	7.22E+02	7.12E+02	7.12E+02	7.12E+03	7.12E+02	7.12E+03	7.16E+02	7.18E+02
	STD	2.01E+00	1.12E+00	1.70E+00	5.51E+00	**6.86E-02**	7.17E-02	7.85E-02	1.20E+01	9.50E-02	9.17E-02	1.42E+01
C17-f8	AVG	814.07E+00	818.58E+00	811.90E+00	**811.04**	1.96E+03	3.59E+04	3.53E+03	9.03E+03	2.68E+04	6.83E+03	6.17E+04
	STD	9.28E+00	**4.77E+00**	7.79E+00	4.88E+00	1.08E+04	2.14E+04	2.87E+04	8.84E+05	1.71E+04	3.46E+04	4.91E+08
C17-f9	AVG	908.54E+00	**902.44E+00**	912.81E+00	902.52E+00	1.10E+04	1.10E+04	1.10E+03	1.10E+04	1.10E+03	1.10E+03	1.10E+04
	STD	8.25E+01	1.13E+01	1.46E+01	3.31E+00	1.53E-02	1.38E-02	7.33E-02	2.30E-01	**5.39E-03**	9.89E-02	5.43E+01
C17-f10	AVG	**1.45E+03**	1.65E+03	1.60E+03	1.64E+03	2.10E+03	4.10E+04	3.37E+03	8.49E+04	2.72E+03	9.01E+04	3.52E+04
	STD	2.46E+02	3.98E+02	**1.92E+02**	2.62E+02	2.83E+04	2.92E+04	1.94E+04	1.22E+05	1.88E+04	8.93E+04	1.84E+06
C17-f11	AVG	**1.11E+03**	1.13E+03	1.13E+03	1.12E+03	1.48E+03	1.50E+04	1.45E+03	1.47E+04	1.49E+04	1.45E+04	1.51E+03
	STD	3.02E+01	3.79E+01	**2.19E+01**	**2.19E+01**	2.52E+02	5.91E+02	1.22E+03	8.07E+02	5.52E+02	1.21E+03	7.83E+02
C17-f12	AVG	2.72E+03	2.78E+03	2.61E+03	4.95E+03	1.40E+04	**1.40E+03**	1.40E+04	**1.40E+03**	1.40E+05	1.41E+03	1.41E+06
	STD	1.12E+02	1.49E+02	1.06E+02	3.79E+03	7.99E-02	6.79E-02	**6.04E-02**	9.24E-02	8.17E-02	1.64E+01	2.15E+01
C17-f13	AVG	1.50E+03	1.58E+03	1.58E+03	1.58E+03	**1.40E+03**	1.40E+06	**1.40E+03**	**1.40E+03**	**1.40E+03**	1.40E+04	1.45E+03
	STD	1.45E+01	2.95E+01	2.59E+01	1.88E+02	2.86E-05	**1.02E-05**	5.54E-04	1.14E-04	2.53E-05	3.88E-04	4.80E+02
C17-f14	AVG	**1.40E+03**	1.42E+03	1.42E+03	1.42E+03	4.35E+04	7.39E+03	7.20E+03	7.70E+04	7.44E+04	7.61E+03	9.40E+03
	STD	4.60E+00	3.72E+00	**2.21E+00**	5.10E+00	1.83E+04	2.55E+04	3.22E+04	1.39E+04	2.57E+04	1.62E+04	4.14E+03
C17-f15	AVG	1.54E+03	1.53E+03	**1.52E+03**	1.54E+03	1.70E+03	1.71E+04	1.70E+03	1.71E+06	1.70E+03	1.72E+06	1.74E+06
	STD	4.11E+01	9.03E+01	4.97E+01	2.06E+01	3.86E+01	4.04E+01	**2.76E-05**	1.23E+02	1.90E-03	3.74E+01	1.22E+01
C17-f16	AVG	1.73E+03	**1.71E+03**	1.79E+03	1.73E+03	3.28E+05	3.02E+06	5.37E+05	2.47E+05	7.06E+05	8.65E+05	4.20E+06
	STD	**1.12E+01**	1.52E+01	1.44E+01	9.19E+01	3.18E+09	3.08E+09	5.73E+08	3.63E+09	6.02E+09	4.07E+09	9.37E+09
C17-f17	AVG	**1.71E+03**	1.75E+03	1.79E+03	1.74E+03	4.13E+04	6.65E+06	8.41E+04	2.97E+04	2.43E+05	8.33E+06	5.58E+03

Table 7 continued

F	Metric	ECSA	PCSA	SCSA	CSA	SHO	GWO	PSO	MVO	SCA	GSA	GA
	STD	5.53E+00	**4.47E+00**	6.91E+00	7.64E+00	5.19E+04	7.03E+05	2.08E+04	2.46E+04	2.03E+03	3.33E+07	2.09E+03
C17-f18	AVG	**1.79E+03**	1.87E+03	1.81E+03	1.86E+03	4.20E+03	4.20E+03	4.20E+03	4.20E+03	4.20E+03	4.20E+03	4.20E+03
	STD	1.12E+01	1.25E+01	1.38E+01	5.51E+01	4.76E-04	8.08E-04	9.61E-04	8.14E-04	4.19E-04	8.53E-04	**2.11E-07**
C17-f19	AVG	**1.91E+03**	**1.91E+03**	**1.91E+03**	**1.91E+03**	5.11E+03	5.16E+03	1.09E+04	5.26E+03	5.18E+03	5.42E+03	5.39E+03
	STD	0.19E+01	1.59E+01	8.71E+00	8.78E+00	2.71E+02	2.03E+02	**4.90E-01**	2.17E+02	2.03E+02	8.72E+02	8.25E+02
C17-20	AVG	2.02E+03	2.07E+03	**2.02E+03**	2.06E+03	8.13E+03	8.20E+03	7.65E+03	2.33E+03	2.09E+03	2.76E+04	2.75E+03
	STD	1.59E+01	2.58E+01	1.85E+01	4.88E+01	2.85E+01	**2.78E+00**	3.16E+01	4.45E+02	3.81E+01	3.30E+02	3.79E+01
C17-21	AVG	**2.10E+03**	2.12E+03	2.17E+03	2.20E+03	2.29E+03	3.26E+04	2.86E+03	8.35E+04	4.82E+04	3.30E+04	4.91E+06
	STD	5.48E+01	**4.44E+01**	6.81E+01	2.93E+01	2.15E+04	3.45E+05	2.93E+03	4.82E+03	3.44E+04	3.41E+04	3.70E+07
C17-22	AVG	**2.30E+03**	**2.30E+03**	**2.30E+03**	**2.30E+03**	8.02E+03	8.02E+03	8.02E+03	8.02E+03	8.02E+03	8.06E+03	8.08E+03
	STD	5.83E+01	2.89E+01	1.78E+01	1.64E+00	7.76E-01	8.07E-01	8.75E-01	2.10E+01	8.40E-01	**8.07E-02**	2.32E+00
C17-23	AVG	**2.60E+03**	**2.60E+03**	**2.60E+03**	2.61E+03	2.86E+03	4.49E+04	4.43E+04	8.93E+04	3.58E+03	7.73E+03	7.07E+04
	STD	2.45E+01	2.31E+01	2.90E+01	**1.03E+01**	2.98E+04	3.04E+04	3.77E+04	9.74E+04	2.61E+04	4.36E+04	5.81E+06
C17-24	AVG	2.49E+03	2.45E+03	**2.42E+03**	2.54E+03	2.00E+04	2.00E+04	2.00E+04	2.00E+04	2.00E+04	2.00E+04	2.00E+04
	STD	1.31E+01	9.57E+01	1.21E+01	1.03E+02	2.43E-02	2.28E-02	7.23E-03	3.20E-02	**6.29E-03**	8.79E-02	6.33E+01
C17-25	AVG	**2.72E+03**	**2.72E+03**	2.73E+03	2.89E+03	3.00E+04	5.00E+04	4.27E+04	9.39E+04	3.62E+04	8.91E+04	4.42E+06
	STD	2.66E+01	2.50E+01	2.34E+01	**7.98E-01**	3.73E+04	3.82E+04	2.84E+04	2.12E+05	2.78E+04	6.83E+04	2.74E+06
C17-26	AVG	2.91E+03	2.91E+03	**2.90E+03**	2.91E+03	2.38E+04	2.40E+04	2.35E+05	2.37E+04	2.39E+04	2.35E+04	2.41E+04
	STD	**1.35E+01**	2.23E+01	2.32E+01	6.90E+01	3.42E+02	6.81E+02	2.12E+03	9.97E+02	6.42E+02	2.11E+03	8.73E+02
C17-27	AVG	**3.08E+03**	3.09E+03	**3.08E+03**	3.10E+03	2.30E+04	2.30E+04	2.30E+04	2.30E+04	2.30E+04	2.31E+04	2.31E+04
	STD	2.03E+01	1.32E+01	1.60E+01	4.27E+00	8.89E-02	**7.69E-02**	7.94E-02	8.14E-02	9.07E-02	3.54E+01	3.05E+00
C17-28	AVG	**3.12E+03**	3.14E+03	3.13E+03	**3.12E+03**	5.30E+04	5.30E+04	5.30E+04	5.30E+04	5.30E+04	5.30E+04	5.35E+04
	STD	1.98E+01	1.68E+01	1.19E+01	8.98E+01	**3.76E-05**	3.92E-05	6.44E-04	2.04E-04	3.43E-05	4.78E-04	4.70E+02
C17-29	AVG	3.14E+03	3.18E+03	**3.13E+03**	3.17E+03	5.25E+04	8.29E+03	8.10E+04	8.60E+03	8.34E+04	8.51E+03	8.30E+04
	STD	3.70E+01	3.39E+01	4.28E+01	**2.53E+01**	2.73E+04	3.45E+04	4.12E+04	2.29E+05	3.47E+05	2.52E+04	5.04E+03
C17-f30	AVG	4.74E+03	4.49E+03	**4.22E+03**	4.75E+03	2.60E+04	2.61E+04	2.60E+04	2.61E+04	2.60E+04	2.62E+04	2.64E+04
	STD	8.46E+03	1.86E+03	1.44E+03	1.60E+03	4.76E+01	5.94E+01	**3.66E-04**	2.13E+02	2.80E-03	4.64E+01	2.12E+02

Table 8 Average rank of all methods obtained using Friedman's test in the first test group of unimodal, multimodal, and fixed-dimensional test functions

Algorithm	Rank
ECSA	4.500000
PCSA	4.478260
SCSA	4.130434
CSA	4.695652
SHO	6.934782
GWO	6.217391
PSO	6.782608
MVO	9.282608
SCA	9.782608
GSA	8.326086
GA	8.021739
EPO	4.847826

5.8 Statistical test analysis

To determine if the performance differences of all methods in the benchmark test functions examined in this study are statistically significant using the non-parametric Friedman's test, a statistical analysis test was first done in this section. More than five algorithms must be tested and compared on more than 10 test functions for a reliable comparison. This study compares the performance of ECSA, PCSA, and SCSA concerning the primary CSA and eight other meta-heuristics. In consideration of the benchmark functions, this study thoroughly tested three test groups:

- Group 1, which consists of unimodal, multimodal, and fixed-dimensional functions with 23 test functions,
- Group 2, which is the CEC-2015 test functions, including 15 set functions, and
- Group 3, this test set is the CEC-2017 test function which includes 29 set functions.

Friedman's test requires computing the mean ranked value, for which a comparison is needed to examine the $p-$ values gained for a level of significance ($\alpha = 0.05$) with Friedman's test to recognize if the null hypothesis is rejected or not. The mathematical formulation and commentaries regarding Friedman's test can be found in [92, 93]. The null hypothesis was rejected for all three test sets of benchmark functions, indicating a statistically large difference between the performances of the competing methods in each test set. The lowest-ranked algorithm by Friedman's test is commonly utilized as a control one for post-hoc analysis. Further steps are needed to determine the performance of the optimization methods that differ significantly from ECSA or SCSA, which are reported as the best algorithms in the first, second, and third test groups, as shown below, and find out which algorithms have similar performance as ECSA or SCSA. To this effect, we performed a post-hoc statistical test using Holm's test procedure [92] to conduct a pairwise comparison between the control algorithm and the other algorithms. This test method shows that the performance of the two algorithms is considerably dissimilar if the difference in the mean ranking of the compared algorithms is greater than the $p-$ values. This statistical test is carried out here to discern which algorithms are better, similar, or worse than ECSA, PCSA, and SCSA at a significance level of 0.05.

Holm's method is a widely applicable multiple-test method based on successive rejection manner. It ranks all algorithms based on their p-values and compares them with $\alpha/k - i$, where k is the degree of freedom and i represents the algorithm number. This procedure commences with the most significant p-value and consecutively rejects the null hypothesis as long as that $pi < \alpha/k - i$. Once the algorithm cannot deny the hypothesis, it stops and considers all the rest hypotheses agreeable.

Holm's technique is a multiple-test approach based on successive rejection that is generally applicable. It compares

Table 9 Results of Holm's method based on the statistical results of Group 1 for $\alpha = 0.05$

i	Algorithm	z	p-value	$\alpha \div i$	Hypothesis
11	SCA	5.316095	1.060176E-7	0.004545	Rejected
10	MVO	4.845825	1.260863E-6	0.005	Rejected
9	GSA	3.946178	7.940843E-5	0.005555	Rejected
8	GA	3.659927	2.522869E-4	0.00625	Rejected
7	SHO	2.637601	0.008349	0.007142	Rejected
6	PSO	2.494475	0.012614	0.008333	Rejected
5	GWO	1.962865	0.049661	0.01	Rejected
4	EPO	0.674735	0.499844 3	0.0125	Not rejected
3	CSA	0.531609	0.594996	0.016666	Not rejected
2	ECSA	0.347590	0.728147	0.025	Not rejected
1	PCSA	0.327144 4	0.743558 8	0.05	Not rejected

Table 10 Average ranking of all algorithms obtained using Friedman's test on the test functions of the second group (i.e., CEC-2015)

Algorithm	Rank
ECSA	3.666666
PCSA	4.733333
SCSA	3.933333
CSA	9.000000
SHO	5.900000
GWO	7.466666
PSO	5.066666
MVO	7.566666
SCA	6.299999
GSA	9.266666
GA	10.200000
EPO	4.900000

all algorithms with $alpha/k-i$ and ranks them according to their p-values, where k is the degree of freedom and i is the algorithm number. As long as $pialpha/k-i$, this technique sequentially rejects the null hypothesis starting with the most significant p-value. Once the algorithm is unable to refute the hypothesis, it pauses and accepts all remaining hypothese.

Table 8 shows the mean rank of all algorithms obtained by utilizing Friedman's test in light of algorithms' results in unimodal, multimodal, and fixed-dimensional multimodal benchmark functions.

The p-value retrieved by Friedman's test based on the mean outcomes of Group 1 is 4.747802E-11. In this test group, SCSA surpassed all evaluated methods, with the lowest mean rank of 4.130434, and the performance score was substantially better than PCSA, ECSA, CSA, EPO, GWO, PSO, SHO, GA, GSA, MVO, and SCA. The statistical outcomes of Holm's method obtained on the test functions of Group 1 are given in Table 9.

Holm's test in Table 9 rejects those hypotheses with p-value ≤ 0.007142. It is seen from these results that ECSA, PCSA, and SCSA are proficient in yielding promising results such as those of other efficacious optimization methods presented in the literature.

Table 10 displays the average ranking of all algorithms determined by Friedman's test after the results of all algorithms in the CEC-2015 test suite (i.e., Group 2).

The p-value evaluated by Friedman's test based on the average accuracy outcomes for Group 2 is 7.596059E-09. The results for Group 2, presented in Table 10, exhibit that ECSA ranked first, SCSA ranked second, and PCSA ranked fourth. In sum, ECSA significantly outperformed SCSA, EPO, PCSA, PSO, SHO, SCA, GWO, MVO, CSA, GSA, and GA at the examined level of significance.

The results of Holm's method got after the application of Friedman's test on the test functions of Group 2 is shown in Table 11.

Holm's test in Table 11 rejects those hypotheses with p-value ≤ 0.008333. It is evident from these findings that the proposed algorithms are promising optimization algorithms as those evaluated in this study.

Table 12 displays the average ranking of all algorithms as determined by Friedman's test on the outcomes of all CEC-2017 test functions (i.e., Group 3).

Finally, the p-value retrieved by Friedman's test for Group 3 is 6.930422E-11. In the results of this group, shown in Table 12, which compares the performance of each optimization algorithm in all test functions of CEC-2017, ECSA placed first, SCSA placed second, and PCSA placed fourth. In short, the lowest average rank belonged to ECSA, with an average rank of 3.762068, and vastly outperformed SCSA, CSA, PCSA, PSO, SHO, MVO, GWO, GSA, SCA, and GA.

The results of Holm's test obtained after applying Friedman's test on Group 3 functions are shown in Table 13.

In Table 13, Holm's procedure rejects those hypotheses that have p-value ≤ 0.016666. From the outcomes shown in

Table 11 Results of Holm's method based on the average ranking results of Group 2 for $\alpha = 0.05$)

i	Method	z	p-value	$\alpha \div i$	Hypothesis
11	GA	4.962422	6.961920E-7	0.004545	Rejected
10	GSA	4.253505	2.1044991E-5	0.005	Rejected
9	CSA	4.050957	5.100847E-5	0.0055555	Rejected
8	MVO	2.962262	0.003053	0.00625	Rejected
7	GWO	2.886307	0.003897	0.007142 5	Rejected
6	SCA	2.000160	0.045482	0.008333	Rejected
5	SHO	1.696338	0.089821	0.01	Rejected
4	PSO	1.063376	0.287611	0.0125	Not rejected
3	EPO	0.936783	0.348869	0.016666	Not rejected
2	PCSA	0.810191	0.417830	0.025	Not rejected
1	SCSA	0.202547	0.839488	0.05	Not rejected

Table 12 Average ranking of all algorithms obtained using Friedman's test on the test functions of the third test group (i.e., CEC-2017)

Algorithm	Rank
ECSA	3.762068
PCSA	4.102586
SCSA	3.810310
CSA	4.017241
SHO	6.465517
GWO	7.258620
PSO	6.017241
MVO	7.137931
SCA	8.172413
GSA	8.103448
GA	9.258620

Table 13, it is clear that the proposed versions of CSA are as practical optimization algorithms as those evaluated in this study.

As a principal inference drawn from all CEC-2015 and CEC-2017 functions' statistical analysis in this paper, ECSA performs considerably better than PCSA, SCSA, and CSA. In concise terms, the results reported in Tables 8, 9, 10, 11, 12 and 13 show that the proposed algorithms, ECSA, PCSA, and SCSA, are statistically highly similar to each other in these test groups. These proposed algorithms have successfully avoided local optimum solutions and are highly efficient in both exploration and exploitation features.

6 Engineering design problems

This section explores the competence of the proposed ECSA, PCSA, and SCSA algorithms in solving four well-known classical engineering design problems: (1) the speed reducer problem, (2) the tension/compression spring problem, (3) the pressure vessel problem, and (4) the welded beam problem.

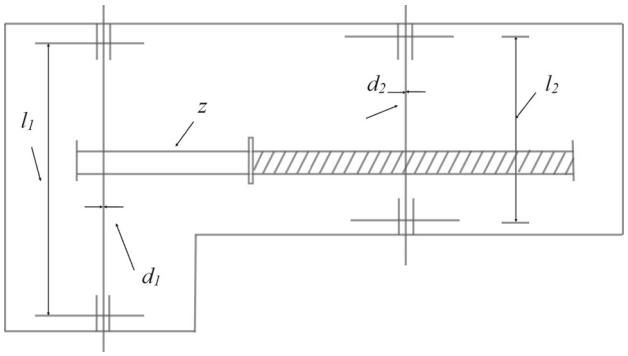

Fig. 6 A structural design of a speed reducer problem

These design problems reflect challenging benchmark test problems with different characteristics, dimensions, and varied levels of complexity and constraints. Because of this, their search spaces are highly comparable to those that the proposed ECSA, PCSA, and SCSA may encounter while addressing constrained optimization problems.

As presented below, the effectiveness of the proposed algorithms in solving these design problems was compared with other meta-heuristic algorithms mentioned above and in Table 1. These comparative algorithms were selected in this comparison because they were extensively applied in the literature to address these engineering design problems, where they provided promising performance. Further, these algorithms share many similarities with the proposed algorithms, including flexibility, generality, and simplicity. In addition, these algorithms are independent of the nature of the engineering design problems to be addressed. To achieve a fair comparison between the proposed ECSA, PCSA, and SCSA algorithms, (EPO [87], SHO [85], GWO [86], PSO [84], MVO [89], SCA [90], GSA [88] and GA [83]) the primary CSA, and the other competing meta-heuristics, the settings of standard parameters such as the maximum number of iterations and the number of search agents used to solve these problems were the same and set to 1000 and 30, respectively. Moreover, several constraints should not be infringed by

Table 13 Results of Holm's test method based on the average statistical results of Group 3 for $\alpha = 0.05$

i	Method	z	p-value	$\alpha \div i$	Hypothesis
10	GA	7.344015	2.072793E-13	0.005	Rejected
9	SCA	6.096918	1.081326E-9	0.005555	Rejected
8	GSA	6.017737	1.768715E-9	0.00625	Rejected
7	GWO	5.047773	4.4698895E-7	0.007142 5	Rejected
6	MVO	4.909207	9.144540E-7	0.008333	Rejected
5	SHO	4.137194	3.515778E-5	0.01	Rejected
4	PSO	3.622519	2.917472E-4	0.0125	Rejected
3	CSA	1.3262777	0.184747	0.016666	Not rejected
2	SCSA	0.593855	0.552608	0.025	Not rejected
1	PCSA	0.534470	0.593016	0.05	Not rejected

Table 14 A comparison of the results achieved by ECSA, PCSA, SCSA, and other algorithms for the speed reducer design problem

Algorithm	Optimum variables							Optimum cost
	b	m	z	l_1	l_2	d_1	d_2	
ECSA	3.5	0.7	17	7.3	7.71533	3.35021	5.28665	2994.47144045
PCSA	3.5	0.7	17	7.30034	7.71558	3.35026	5.28666	2994.51947720
SCSA	3.5	0.7	17	7.30196	7.71684	3.35039	5.28683	2994.82887327
CSA	3.5	0.7	17	7.90415	7.71738	3.35142	5.28669	3000.21223231
EPO	3.50123	0.7	17	7.3	7.8	3.33421	5.26536	2994.2472
SHO	3.50159	0.7	17	7.3	7.8	3.35127	5.28874	2998.5507
GWO	3.506690	0.7	17	7.380933	7.815726	3.357847	5.286768	3001.288
PSO	3.500019	0.7	17	8.3	7.8	3.352412	5.286715	3005.763
MVO	.508502	0.7	17	7.392843	7.816034	3.358073	5.286777	3002.928
SCA	.508755	0.7	17	7.3	7.8	3.461020	5.289213	3030.563
GSA	3.6	0.7	17	8.3	7.8	3.369658	5.289224	3051.120
GA	3.510253	0.7	17	8.35	7.8	3.362201	5.287723	3067.561

the optimum solution(s) arrived at while solving these engineering design problems. Therefore, when addressing these design problems, the proposed algorithms are equipped with a static penalty function in handling the constraints while solving these problems as described below:

$$\zeta(z) = f(z) \pm \left[\sum_{i=1}^{m} l_i \cdot max(0, t_i(z))^{\alpha} + \sum_{j=1}^{n} o_j \left| U_j(z) \right|^{\beta} \right] \tag{16}$$

where $\zeta(z)$ is the objective function, o_j and l_i stand for positive penalty constants, $U_j(z)$ and $t_i(z)$ are the constraints of the objective function. The values of α and β were assigned to 1 and 2, respectively.

This approach sets the penalty value for each solution in the static penalty function, which can help the proposed algorithms' search agents move into the problem's search space.

The results of solving the aforementioned engineering design problems by the proposed algorithms compared to other competitors are presented below.

6.1 Speed reducer design problem

The structural design of the speed reducer design problem is shown in Fig. 6. This design is complex because it consists of seven design variables [94].

The weight that should be reduced in this design problem is subject to four constraints [85], which are explained as follows:

- Transverse deflections of the shafts
- Surface stress
- Bending stress of the gear teeth
- Stresses in the shafts

Table 15 Statistical results obtained from ECSA, PCSA and SCSA and other optimization algorithms for speed reducer design problem

Algorithm	Best	AVG	Worst	STD
ECSA	2994.47106669	2994.51536693	2994.90762336	1.37829792E-01
PCSA	2994.47107258	2994.47243595	2994.47723809	2.41749483E-03
SCSA	2994.47107658	2994.48965105	2994.55203370	3.05108129E-02
CSA	2994.60512139	2997.50385785	3002.59911565	2.78670160
EPO	2994.2472	2997.482	2999.092	1.78091
SHO	2998.5507	2999.640	3003.889	1.93193
GWO	3001.288	3005.845	3008.752	5.83794
PSO	3005.763	3105.252	3211.174	79.6381
MVO	3002.928	3028.841	3060.958	13.0186
SCA	3030.563	3065.917	3104.779	18.0742
GSA	3051.120	3170.334	3363.873	92.5726
GA	3067.561	3186.523	3313.199	17.1186

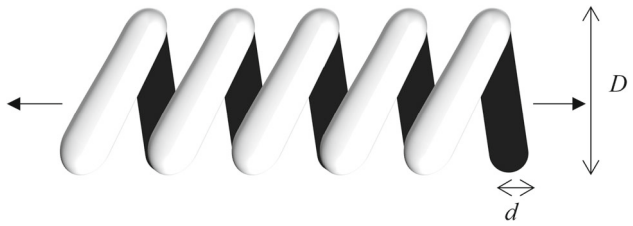

Fig. 7 A schematic diagram of a tension/compression spring design

The variables of this design problem were set as follows: l_1, l_2, d_1, d_2, b, m, and z. These parameters are the first shaft's distance between bearings, the second shaft's distance between bearings, the first and second shafts' diameters, the faces of the shafts, the module of teeth, and the number of teeth in the pinion, respectively. These variables were successively implemented while solving this problem by a vector as $\vec{x} = [x_1 x_2 x_3 x_4 x_5 x_6 x_7]$. The mathematical formula for the speed reducer problem can be formulated as follows:

$$\begin{aligned}\texttt{Minimize}: f(\vec{x}) = {} & 0.7854 x_1 x_2^2 (3.3333 x_3^2 + 14.9334 x_3 \\ & - 43.0934) - 1.508 x_1 (x_6^2 + x_7^2) \\ & + 7.4777 (x_6^3 + x_7^3) \\ & + 0.7854 (x_4 x_6^2 + x_5 x_7^2)\end{aligned}$$

This function is subject to the following eleven constraints:

$$g_1(\vec{x}) = \frac{27}{x_1 x_2^2 x_3} - 1 \leq 0$$

$$g_2(\vec{x}) = \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \leq 0$$

$$g_3(\vec{x}) = \frac{1.9 x_4^3}{x_2 x_6^4 x_3} - 1 \leq 0$$

$$g_4(\vec{x}) = \frac{1.93 x_5^3}{x_2 x_7^4 x_3} - 1 \leq 0$$

$$g_5(\vec{x}) = \frac{[(745(x_4/x_2 x_3))^2 + 16.9 \times 10^6]^{1/2}}{110 x_6^3} - 1 \leq 0$$

$$g_6(\vec{x}) = \frac{[(745(x_5/x_2 x_3))^2 + 157.5 \times 10^6]^{1/2}}{85 x_7^3} - 1 \leq 0$$

$$g_7(\vec{x}) = \frac{x_2 x_3}{40} - 1 \leq 0$$

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

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

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

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

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

The optimum cost and best designs obtained by ECSA, PCSA, SCSA, and other optimization methods for the speed reducer design problem are in Table 14.

Per the optimum costs shown in Table 14, ECSA, PCSA, and SCSA are adept at finding optimum designs for the speed reducer problem with the lowest costs. In terms of best, worst, average, and standard deviation results, a summary of the statistical outcomes for ECSA, PCSA, SCSA, and other

Table 16 A comparison of the outcomes reached by ECSA, PCSA, SCSA, and other algorithms for the tension/compression spring design problem

Algorithm	Optimum variables			Optimum weight
	d	D	N	
ECSA	0.0516891	0.356718	11.289	0.01266523
PCSA	0.0516891	0.356718	11.289	0.01266523
SCSA	0.0516891	0.356718	11.289	0.01266523
CSA	0.0516895	0.356729	11.2883	0.01266523
EPO	0.051087	0.342908	12.0898	0.012656987
SHO	0.051144	0.343751	12.0955	0.012674000
GWO	0.050178	0.341541	12.07349	0.012678321
PSO	0.05000	0.310414	15.0000	0.013192580
MVO	0.05000	0.315956	14.22623	0.012816930
SCA	0.050780	0.334779	12.72269	0.012709667
GSA	0.05000	0.317312	14.22867	0.012873881
GA	0.05010	0.310111	14.0000	0.013036251

Table 17 Statistical results obtained by ECSA, PCSA, and SCSA and other algorithms for tension/compression spring design problem

Algorithm	Best	AVG	Worst	STD
ECSA	0.01266523	0.01266523	0.01266523	8.3790149E-16
PCSA	0.01266523	0.01266523	0.01266523	2.58597307E-18
SCSA	0.01266523	0.01266523	0.01266523	2.55932959E-17
CSA	0.01266523 -02	0.01266523	0.01266524	4.14047743E-09
EPO	0.012656987	0.012678903	0.012667902	0.001021
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
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

meta-heuristics for this design problem, over 30 independent runs, is exhibited in Table 15.

As per the results in Table 15, the proposed ECSA, PCSA, and SCSA identified the best statistical solutions with respect to best, average, worst, and standard deviation values among all other competitors. This is for more certainty that the proposed algorithms are superior to other existing algorithms in terms of these statistical results.

6.2 tension/compression spring design

The design of the tension/compression spring problem is shown in Figure 7 [95].

This problem aims to lessen the design's weight of the tension/compression spring. This design problem has several constraints: minimum deflection, shear stress, and surge frequency. The parameters of this problem are mean coil diameter (D), wire diameter (d), and the number of active coils (N). These variables can be drafted as a vector as: $\vec{x} = [x_1, x_2, x_3]$, where the parameters of $\vec{x}$ are d, D and N, respectively. The mathematical formulation of this problem can be described as given below:

$\texttt{Minimize: } f(\vec{x}) = (x_3 + 2)x_2x_1^2$

The following restrictions apply to this design problem:

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

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

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

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

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

A comparison of the optimal designs gained by ECSA, PCSA, SCSA, and those algorithms mentioned above for the tension/compression spring design problem is shown in Table 16.

As per the results in Table 16, the proposed ECSA, PCSA, SCSA, and CSA have reached the optimum design for this problem with an optimal cost of 0.01266523. This cost is a little lower than the costs obtained by other comparative algorithms. A summary of the statistical results of the tension/compression spring design problem collected by ECSA, PCSA, SCSA, and different algorithms over 30 independent runs is given in Table 17.

It is perceived from the outcomes reported in Table 17 that ECSA, PCSA, and SCSA performed better again by providing many improved statistical results in terms of best, average, worst, and standard deviation results compared to the others.

6.3 Pressure vessel design

The pressure vessel design problem has been widely used in optimization [90]. This problem aims to reduce the total cost of material formation and welding of the cylindrical vessel, which is covered on both ends with hemispherical heads, as shown in Figure 8.

The variables of this design problem are defined as follows:

- Inner radius (R)
- Thickness of the shell (T_s)

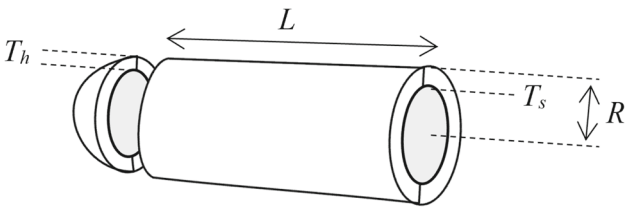

Fig. 8 A representative structure of the cross-section of a pressure vessel design problem

Table 18 A comparison of the results achieved by ECSA, PCSA, SCSA, and other algorithms for the pressure vessel design problem

Algorithm	Optimal values for variables T_s	T_h	R	L	Optimum cost
ECSA	12.4507	6.154387	40.31962	200	5885.332773
PCSA	12.4507	6.154387	40.31962	200	5885.332773
SCSA	12.4507	6.154387	40.31962	200	5885.332773
CSA	12.45072	6.154399	40.31967	199.9994	5885.340485
EPO	0.778099	0.383241	40.315121	200.00000	5880.0700
SHO	0.778210	0.384889	40.315040	200.00000	5885.5773
GWO	0.779035	0.384660	40.327793	199.65029	5889.3689
PSO	0.778961	0.384683	40.320913	200.00000	5891.3879
MVO	0.845719	0.418564	43.816270	156.38164	6011.5148
SCA	0.817577	0.417932	41.74939	183.57270	6137.3724
GSA	1.085800	0.949614	49.345231	169.48741	11550.2976
GA	0.752362	0.399540	40.452514	198.00268	5890.3279

- Length of the cylindrical section of the vessel without looking at the head (L)
- Thickness of the head (T_h)

The variable vector of this design problem can be formulated as follows: $\vec{x} = [x_1, x_2, x_3, x_4]$, where the parameters of this vector represent T_s, T_h, R and L, respectively. The mathematical formulation of this design problem can be defined as shown below:

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

This problem is subject to the following constraints:

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

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

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

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

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

This design problem has been addressed in the literature using various optimization methods, such as those mentioned above. Table 18 illustrates the cost results achieved by ECSA, PCSA, SCSA, and other algorithms for the pressure vessel design problem.

As shown in Table 18, the proposed ECSA, PCSA, and SCSA can find the optimal design for the pressure vessel problem with the lowest cost of 5885.332773. Table 19 displays the statistical results for the pressure vessel design problem obtained by ECSA, PCSA, SCSA, and other algorithms, over 30 independent runs, regarding the best, worst, mean, and standard deviation costs.

Table 19 Statistical results obtained by ECSA, PCSA, SCSA, and other algorithms for pressure vessel design problem

Algorithm	Best	AVG	Worst	STD
ECSA	5885.332773	5885.332773	5885.332773	2.541289E-09
PCSA	5885.332773	5885.332773	5885.332773	4.605113E-11
SCSA	5885.332773	5885.332773	5885.332773	1.174152E-11
CSA	5885.336392	5885.339615	5885.344878	2.950500E-03
EPO	5880.0700	5884.1401	5891.3099	024.341
SHO	5885.5773	5887.4441	5892.3207	002.893
GWO	5889.3689	5891.5247	5894.6238	013.910
PSO	5891.3879	6531.5032	7394.5879	534.119
MVO	6011.5148	6477.3050	7250.9170	327.007
SCA	6137.3724	6326.7606	6512.3541	126.609
GSA	1550.2976	23342.2909	33226.2526	5790.625
GA	5890.3279	6264.0053	7005.7500	496.128

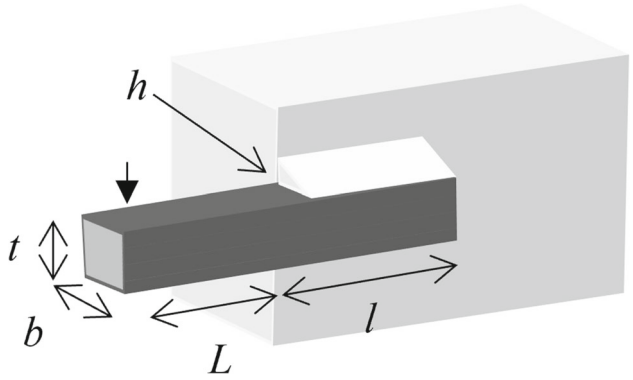

Fig. 9 A schematic structure of a welded beam design

It may be ascertained from Table 19 that ECSA, PCSA, and SCSA outperformed other algorithms and provided highly competitive statistical results regarding standard deviation and average cost values compared to others.

6.4 Welded beam design

The welded beam problem always strives to reduce the manufacturing cost of the welded beam structure shown in Fig. 9 [96].

The welded beam structure in Fig. 9 comprises a beam, A, and the welding coveted to be linked to the section, B. This design problem undergoes a set of constraints identified as follows:

- End deflection of the beam (δ)
- Buckling load on the bar (P_c)
- Shear stress (τ)
- Bending stress in the beam (θ)

There is a need to find the parameters of the welded beam structure to optimize this design problem, which is: the clamped bar's length (l), the thickness of the weld (h), the thickness of the bar (b) and the height of the bar (t). The design variable vector can be written as: $\vec{x} = [x_1, x_2, x_3, x_4]$, where the parameters of $\vec{x}$ stand for h, l, t and b, respectively. The mathematical formulation of the cost function of this design problem to be minimized is given as follows:

$\texttt{Minimize}: f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$

Subject to the following constraints,

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

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

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

$g_4(\vec{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0$

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

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

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

where the other parameters are defined as presented below:

$$\tau(\vec{x}) = \sqrt{((\tau')^2 + (\tau'')^2) + \frac{2\tau'\tau''x_2}{2R}},\ \tau' = \frac{p}{\sqrt{2}x_1x_2}$$

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

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

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

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 ranges of the variables were used as $0.1 \leq x_i \leq 2.0$ when $i = 1$ and 4 and $0.1 \leq x_i \leq 10.0$ when $i = 2$ and 3.

This problem was addressed by several algorithms such as EPO [87], SHO [85], GWO [86], PSO [84], MVO [89], SCA [90], GSA [88] and GA [83]. The results obtained by

Table 20 A comparison of the cost results achieved by ECSA, PCSA, and SCSA, and other algorithms for the welded beam problem

Algorithm	Optimal values for variables				Optimum cost
	h	l	t	b	
ECSA	0.20573	3.4705	9.0366	0.20573	1.72485230
PCSA	0.20573	3.4705	9.0366	0.20573	1.72485230
SCSA	0.20573	3.4705	9.0366	0.20573	1.72485230
CSA	0.20573	3.4705	9.0366	0.20573	1.72485233
EPO	0.205411	3.472341	9.035215	0.201153	1.723589
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	10.00000	0.201395	1.820395
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	10.00000	0.217725	2.172858
GA	0.164171	4.032541	10.00000	0.223647	1.873971

Table 21 Statistical results obtained from ECSA, PCSA, and SCSA and other algorithms for welded beam design problem

Algorithm	Best	AVG	Worst	STD
ECSA	1.724852	1.724852	1.724852 0	2.792645E-13
PCSA	1.724852	1.724852	1.724852	3.427136E-15
SCSA	1.724852	1.724852	1.724852	8.757557E-16
CSA	1.724852	1.724852	1.724852	3.215889E-08
EPO	1.723589	1.725124	1.727211	0.004325
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
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

Table 22 Average running time of the proposed algorithms of CSA and other algorithms in solving various engineering design problems

Algorithm	Average time (in seconds)
ECSA	2.392594
PCSA	2.382520
SCSA	2.469050
CSA	3.789843
PSO	2.033712
MVO	3.502070
SCA	3.961583
GSA	4.371697
SHO	3.612047
GWO	3.810658
GA	6.552614
EPO	3.996441

ECSA, PCSA, SCSA, and other algorithms for the welded beam design problem are presented in Table 20.

It is evident from the outcomes in Table 20 that ECSA, PCSA, and SCSA delivered optimal designs for the welded beam structure by finding the optimum cost of approximately 1.72485230, thus exceeding all other algorithms in terms of accuracy in optimization. Table 21 displays the statistical results of ECSA, PCSA, SCSA, and different algorithms over 30 independent runs concerning best, worst, mean, and standard deviation results.

The results in Table 21 show that ECSA, PCSA, and SCSA outperformed all other algorithms with minimal standard deviation and average cost values compared to different algorithms.

The efficiency of an optimization algorithm can also be judged based on how long it takes to solve a problem. The processing time was calculated to evaluate the proposed algorithms' performance accurately. No matter how effective the optimization is, it is useless if the algorithm takes too long to solve a problem. Therefore, the running time must be within the acceptable range. The implementation time is influenced by the size and complexity of the problem as well as the capability of the machine being used. The average computational times taken by ECSA, PCSA, SCSA, and other algorithms to solve the above four classical engineering design problems are presented in Table 22, where the machine and software specifications are as follows: previously specified.

It can be noticed from Table 22 that the average computational times required by the proposed ECSA, PCSA, and SCSA in optimizing different engineering design problems are within the range of other algorithms. The computational times for ECSA, PCSA, and SCSA are also better than some of the other rivals. This demonstrates that the proposed algorithms are computationally efficient on a machine with moderate specifications like the one mentioned above.

In short, as can be seen from the results of the proposed ECSA, PCSA, and SCSA algorithms in solving the above engineering design problems, several advantages could be noted by the performance of these algorithms in solving these problems.

- First, these algorithms are independent of the nature of these design problems, which means there is no bias for these algorithms in solving these problems. This can provide a good judgment of the efficiency and practicality of the proposed algorithms in solving real-world optimization problems.
- Second, as per the design costs and statistical results are shown in Tables 20 to 15, one can say that the proposed algorithms are suitable for solving these design problems as the designs costs of these algorithms are the lowest among all other competing algorithms. In other words, the proposed ECSA, PCSA, and SCSA algorithms could identify the optimal designs for these problems by finding the optimal costs in each, which implies that these proposed could escape out of local optimums.
- Another benefit can be captured from the performance levels of the proposed algorithms in solving these problems, where one can suggest that these algorithms can be used for further studies in judging the efficiency of newly developed meta-heuristic algorithms. This includes conducting comparative studies and verification of new meta-heuristic optimization algorithms. The above advantages of the proposed algorithms are positive points to recommend the proposed ECSA, PCSA, and SCSA algorithms

Table 23 Characteristics of the unimodal, multimodal, and fixed-dimensional test functions. U: Unimodal, M: Multimodal, and F: fixed-dimensional

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	$\max_i \{\lvert x_i \rvert, 1 \le i \le d\}$	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	$\sum_{i=1}^{d} ([x_i + 0.5])^2$	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}	$-20\exp(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}) - \exp\left(\frac{1}{d}\sum_{i=1}^{d} cos(2\pi x_i)\right) + 20 + e$	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	$x_i \in [-600,600]$
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}	$0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{d}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_d - 1)^2\right.$ $\left.\left[1 + \sin^2(2\pi x_d)\right]\right\} + \sum_{i=1}^{d} u(x_i, 5, 100, 4)$	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}	$\sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	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	$x_i \in [-2,2]$

Table 23 continued

Key	Function formulation	$f(x^*)$	Category	Dim	Range
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	$x_i \in [0.10]$

as promising candidates for solving other engineering design problems.

As per the NFL theorem [6] mentioned in the introduction section, it is beyond the bounds of the possibility of finding a single algorithm that can solve all kinds of problems with a high level of efficiency. Because of this, an algorithm may tackle a specific problem very efficiently while bringing poor results to another problem. Knowing the properties of the fitness function is necessary to comment on which algorithm should be selected to solve a problem. Hence, to choose an algorithm when encountering a similar problem, it is best to study it first and then carefully select the most suitable algorithm. In this, it is better to choose and test many algorithms to solve the problem of interest under the same conditions and settings with a particular fitness function. Then, the algorithm with superior performance and the most efficient among all other algorithms in solving the problem can be selected and recommended for solving different problems similar to the solved problem.

Table 24 A description of the CEC-2015 benchmark test functions

Function	Function name	Related basic functions	Dim	f_{min}
C15-f1	Rotated Bent Cigar function	Bent Cigar function	30	100
C15-f2	Rotated Discus function	Discus function	30	200
C15-f3	Rotated and shifted Weierstrass function	Weierstrass function	30	300
C15-f4	Rotated and shifted Schwefel's function	Schwefel's function	30	400
C15-f5	Rotated and shifted Katsuura function	Katsuura function	30	500
C15-f6	Rotated and shifted HappyCat function	HappyCat function	30	600
C15-f7	Rotated and shifted HGBat function	HGBat function	30	700
C15-f8	Rotated and shifted expanded Griewank's with Rosenbrock's function	Griewank's function Rosenbrock's Function	30	800
C15-f9	Rotated and shifted expanded Scaffer's F6 function	Expanded Scaffer's F6 function	30	900
C15-f10	Hybrid function 1 (three functions)	Rastrigin's function High Conditioned Elliptic function Schwefel's function	30	1000
C15-f11	Hybrid function 2 (four functions)	Rosenbrock's function Scaffer's F6 function Griewank's function Weierstrass function	30	1100
C15-f12	Hybrid function 3 (five functions)	Ackley's function Schwefel's function Expanded Griewank's with Rosenbrock's function HappyCat function Katsuura function	30	1200
C15-f13	Composition function 1 (five functions)	Rosenbrock's function Bent Cigar function Discus function High Conditioned Elliptic function High Conditioned Elliptic function	30	1300
C15-f14	Composition function 2 (three functions)	Rastrigin's function High Conditioned Elliptic function Schwefel's function	30	1400
C15-f15	Composition function 3 (five functions)	Schwefel's function Weierstrass function HGBat function Rastrigin's function High Conditioned Elliptic function	30	1500

7 Conclusion and future works

In this study, three improved variants of the Crow Search Algorithm (CSA), named Exponential CSA (ECSA), Power CSA (PCSA), and S-shaped CSA (SCSA), were proposed. Exponential, power, and s-shaped growth functions were used to implement these versions' flight length and awareness probability to improve their exploration and exploitation capabilities. Another key adaptive control parameter was suggested in the positioning updating mechanism of ECSA, PCSA, and SCSA to improve their exploration and exploitation features further. Extensive experiments were conducted to reveal the performance degree of the proposed algorithms by solving three test groups of benchmark functions consisting of 67 familiar optimization problems with different levels of complexity. Besides, the effectiveness of ECSA, PCSA, and SCSA was also proven by their applications on four engineering design problems. In line with the computational and statistical results, it can be realized and confirmed that the proposed algorithms provided very competitive outcomes with several advantages over a group of widely well-known meta-heuristics regarding the stability and quality of the solu-

Table 25 Characteristics of the CEC-2017 benchmark test functions: U: Unimodal, M: Multimodal, H: Hybrid, and C: Composition

Function	Function name	Dim	f_{min}	Type
C17-f1	Rotated and shifted Bent Cigar function	10	100	U
C17-f3	Rotated and shifted Zakharov function	10	300	U
C17-f4	Rotated and shifted Rosenbrock's function	10	400	M
C17-f5	Rotated and shifted Rastrigin's function	10	500	M
C17-f6	Rotated and shifted Expanded Scaffer's function	10	600	M
C17-f7	Rotated and shifted Lunacek Bi-Rastrigin function	10	700	M
C17-f8	Rotated and shifted Non-Continuous Rastrigin's function	10	800	M
C17-f9	Rotated and shifted Levy function	10	900	M
C17-f10	Rotated and shifted Schwefel's function	10	1000	M
C17-f11	Hybrid function of Rosenbrock, Zakharov and Rastrigin's	10	1100	H
C17-f12	Hybrid function of Modified Schwefel, High Conditioned Elliptic, and Bent Cigar	10	1200	H
C17-f13	Hybrid function of Rosenbrock, Bent Ciagr and Lunache Bi-Rastrigin	10	1300	H
C17-f14	Hybrid function of Ackley, Eliptic, Schaffer, and Rastrigin	10	1400	H
C17-f15	Hybrid function of HGBat, Bent Cigar, Rosenbrock, and Rastrigin	10	1500	H
C17-f16	Hybrid function of Expanded Schaffer, Modified Schwefel, Rosenbrock and HGBat	10	1600	H
C17-f17	Hybrid function of Ackley, Katsuura, Expanded Griewank plus Rosenbrock, Rastrigin and Modified Schwefel	10	1700	H
C17-f18	Hybrid function of Ackley, High Conditioned Elliptic, Discus, Rastrigin and HGBat	10	1800	H
C17-f19	Hybrid function of Rastrigin, Bent Cigar, Expanded Grienwank plus Rosenbrock, expanded Schaffer and Weierstrass	10	1900	H
C17-f20	Hybrid function of Happycat, Schaffer, Katsuura, Modified Schwefel, Rastrigin and Ackley	10	2000	H
C17-f21	Composition of High Conditioned Elliptic, Rosenbrock and Rastrigin	10	2100	C
C17-f22	Composition of Griewank's, Rastrigin's and Modified Schwefel's	10	2200	C
C17-f23	Composition of Ackley, Rosenbrock, Rastrigin and Modified Schwefel	10	2300	C
C17-f24	Composition of Girewank, High Conditioned Elliptic, Ackley and Rastrigin	10	2400	C
C17-f25	Composition of Rastrigin, Discus, Happycat, Rosenbrock and Ackley	10	2500	C
C17-f26	Composition of Rastrigin, Rosenbrock, Griewank, Modified Schwefel and Expanded Scaffer	10	2600	C
C17-f27	Composition of Rastrigin, HGBat, Expanded Scaffer, Bent-Cigar, Modified Schwefel, and High Conditioned Elliptic	10	2700	C
C17-f28	Composition function of Discus, Griewank, Rosenbrock, Ackley, Expanded and HappyCat Scaffer	10	2800	C
C17-f29	Composition function of Rotated and shifted Rastrigin, Lunacek Bi-Rastrigin and Expanded Scaffer	10	2900	C
C17-f30	Composition function of Rotated and shifted Rastrigin, Levy function and Non-Continuous Rastrigin	10	3000	C

tions obtained. Upon reading the outcomes, the proposed algorithms provide a fundamental framework for relatively low-dimensional optimization problems, which may expand their reliability to solve large-scale problems. In this, there is a need for further study to solve other real-world problems with several search fields and diverse kinds of constraints. It also has the potential to expand the applications of ECSA, PCSA, and SCSA to address multi-objective problems in diverse fields.

Appendix A. Unimodal, multimodal, CEC-2015 and CEC-BC-2017 test functions

The classical unimodal, multimodal, fixed-dimensional, CEC-2015 and CEC-2017 benchmark functions are tabulated in Tables 23, 24 and 25, respectively.

Funding No funding was received for conducting this study.

Declarations

Conflict of Interest The authors declare that there is no conflict of interest regarding the publication of this paper.

References

1. Elaziz MA, Heidari AA, Fujita H, Moayedi H (2020) A competitive chain-based harris hawks optimizer for global optimization and multilevel image thresholding problems. Appl Soft Comput 95:106347
2. Taradeh M, Mafarja M, Heidari AA, Faris H, Aljarah I, Mirjalili S, Fujita H (2019) An evolutionary gravitational searchbased feature selection. Inf Sci 497:219–239
3. Hashim FA, Hussain K, Houssein EH, Mabrouk MS, Al-Atabany W (2021) Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. Appl Intell 51(3):1531–1551
4. Mlinarić D, Perić T, Matejaš J (2019) Multiobjective programming methodology for solving economic diplomacy resource allocation problem. Croat Oper Res Rev, pp 165-174
5. 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 Trans Neural Netw Learn Syst
6. Koppen M, Wolpert DH, Macready WG (2001) Remarks on a recent paper on the" no free lunch" theorems. IEEE Trans Evol Comput 5(3):295–296
7. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
8. Yong W, Tao W, Cheng-Zhi Z, Hua-Juan H (2016) A new stochastic optimization approachdolphin swarm optimization algorithm. Int J Comput Intell Appl 15(02):1650011
9. Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems:crow search algorithm. Comput & Struct 169:1–12
10. Meraihi Y, Gabis AB, Ramdane-Cherif A, Acheli D (2021) A comprehensive survey of crow search algorithm and its applications. Artif Intell Rev 54(4):2669–2716
11. Hinojosa S, Oliva D, Cuevas E, Pajares G, Avalos O, Gálvez J (2018) Improving multi-criterion optimization with chaos: a novel multiobjective chaotic crow search algorithm. Neural Comput Applic 29(8):319–335
12. Rizk-Allah RM, Hassanien AE, Bhattacharyya S (2018) Chaotic crow search algorithm for fractional optimization problems. Appl Soft Comput 71:1161–1175
13. Zamani H, Nadimi-Shahraki MH, Gandomi AH (2019) Ccsa: conscious neighborhood-based crow search algorithm for solving global optimization problems. Appl Soft Comput 85:105583
14. Sayed GI, Hassanien AE, Azar AT (2019) Feature selection via a novel chaotic crow search algorithm. Neural Comput Applic 31(1):171–188
15. Islam J, Vasant PM, Negash BM, Watada J (2019) A modified crow search algorithm with niching technique for numerical optimization. In 2019 IEEE Student Conference on Research and Development (SCOReD), pages 170-175. IEEE
16. Bhullar AK, Kaur R, Sondhi S (2020) Enhanced crow search algorithm for avr optimization. Soft Comput 24(16):11957–11987
17. Shekhawat S, Saxena A (2020) Development and applications of an intelligent crow search algorithm based on opposition based learning. ISA Trans 99:210–230
18. Rizk-Allah RM, Hassanien AE, Slowik A (2020) Multi-objective orthogonal opposition-based crow search algorithm for large-scale multi-objective optimization. Neural Comput & Applic 32(17):13715–13746
19. Upadhyay P, Chhabra JK (2020) Kapurs'entropy based optimal multilevel image segmentation using crow search algorithm. Appl Soft Comput 97:105522
20. Chaudhuri A, Sahu TP (2021) Feature selection using binary crow search algorithm with time varying flight length. Expert Syst Appl 168:114288
21. Han X, Xu Q, Yue L, Dong Y, Xie G, Xu X (2020) An improved crow search algorithm based on spiral search mechanism for solving numerical and engineering optimization problems. IEEE Access 8:92363–92382
22. Braik M, Al-Zoubi H, Ryalat M, Sheta A, Alzubi O (2022) Memory based hybrid crow search algorithm for solving numerical and constrained global optimization problems. Artificial Intelligence Review, pages 1-73
23. El-Ashmawi WH, Ali AF, Slowik A (2021) Hybrid crow search and uniform crossover algorithmbased clustering for top-n recommendation system. Neural Comput Appl 33(12):7145–7164
24. Guha D, Roy PK, Banerjee S (2022) Performance evolution of different controllers for frequency regulation of a hybrid energy power system employing chaotic crow search algorithm. ISA Trans 120:128–146
25. Das S, Sahu TP, Janghel RR, Sahu BK (2022) Effective forecasting of stock market price by using extreme learning machine optimized

by pso-based group oriented crow search algorithm. Neural Comput Appl 34(1):555–591
26. Gupta D, Sundaram S, Khanna A, Hassanien AE, De Albu-querque VHC (2018) Improved diagnosis of parkinson's disease using optimized crow search algorithm. Comput & Electr Eng 68:412–424
27. Arora S, Singh H, Sharma M, Sharma S, Anand P (2019) A new hybrid algorithm based on grey wolf optimization and crow search algorithm for unconstrained function optimization and feature selection. Ieee Access 7:26343–26361
28. Ramgouda P, Chandraprakash V (2019) Constraints handling in combinatorial interaction testing using multi-objective crow search and fruitfly optimization. Soft Comput 23(8):2713–2726
29. Anter AM, Ali M (2020) Feature selection strategy based on hybrid crow search optimization algorithm integrated with chaos theory and fuzzy c-means algorithm for medical diagnosis problems. Soft Comput 24(3):1565–1584
30. Adamu A, Abdullahi M, Junaidu SB, Hassan IH (2021) An hybrid particle swarm optimization with crow search algorithm for feature selection. Mach Learn Appl 6:100108
31. Kumar S, Fred AL, Miriam LJ, Padmanabhan P, Gulyás B, Kumar A, Dayana N (2022) Improved crow search algorithm based on arithmetic crossover-a novel metaheuristic technique for solving engineering optimization problems. Elsevier, In Multi-Objective Combinatorial Optimization Problems and Solution Methods, pp 71–91
32. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191
33. Wang J, Wan D (2020) Application progress of computational fluid dynamic techniques for complex viscous flows in ship and ocean engineering. J Mar Sci Appl 19(1):1–16
34. Liu Q, Li X, Liu H, Guo Z (2020) Multi-objective metaheuristics for discrete optimization problems: A review of the state-of-the-art. Appl Soft Comput 93:106382
35. Hernández-Pérez LG, Ponce-Ortega JM (2022) Use of statistic functions to consider uncertainty in multi-objective optimization methods based on metaheuristic algorithms. Process Integr Optim Sustain 6(1):161–174
36. Kvasov DE, Mukhametzhanov MS (2018) Metaheuristic vs. deterministic global optimization algorithms: The univariate case. Appl Math Comput 318:245–259
37. Alshamrani AM, Alrasheedi AF, Alnowibet KA, Mahdi S, Mohamed AW (2022) A hybrid stochastic deterministic algorithm for solving unconstrained optimization problems. Mathematics 10(17):3032
38. Bharathan S, Rajendran C, Sundarraj R (2017) Penalty based mathematical models for web service composition in a geo-distributed cloud environment. In 2017 IEEE Int Conf Web Serv (ICWS), IEEE pp 886–889
39. Venkateswarlu C (2021) A metaheuristic tabu search optimization algorithm: Applications to chemical and environmental processes. IntechOpen, In Engineering Problems-Uncertainties, Constraints and Optimization Techniques
40. Sangaiah AK, Hosseinabadi AAR, Shareh MB, Rad SYB, Zolfagharian A, Chilamkurti N (2020) Iot resource allocation and optimization based on heuristic algorithm. Sensors 20(2):539
41. Hafeez G, Alimgeer KS, Khan I (2020) Electric load forecasting based on deep learning and optimized by heuristic algorithm in smart grid. Appl Energy 269:114915
42. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
43. Chinnasamy S, Ramachandran M, Amudha M, Ramu K (2022) A review on hill climbing optimization methodology
44. Holland JH (1992) Genetic algorithms. Scientific american 267(1):66–73
45. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on, pages 39–43 IEEE
46. Das S, Suganthan PN (2011) Differential evolution: A survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31
47. Braik M, Ryalat MH, Al-Zoubi H (2022) A novel meta-heuristic algorithm for solving numerical optimization problems: Ali baba and the forty thieves. Neural Comput Appl 34(1):409–455
48. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12(6):702–713
49. Motevali MM, Shanghooshabad AM, Aram RZ, Keshavarz H (2019) Who: A new evolutionary algorithm bio-inspired by wildebeests with a case study on bank customer segmentation. Int J Pattern Recog Artif Intell 33(05):1959017
50. Rahman CM, Rashid TA (2021) A new evolutionary algorithm: Learner performance based behavior algorithm. Egypt Inform J 22(2):213–223
51. Faramarzi A, Heidarinejad M, Stephens B, Mirjalili S (2020) Equilibrium optimizer: A novel optimization algorithm. Knowl-Based Syst 191:105190
52. Dorigo M, Stützle T (2019) Ant colony optimization: overview and recent advances. Handb Metaheuristics, pp 311–351
53. Braik M, Sheta A, Al-Hiary H (2020) A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm. Neural Comput Appl, pp 1–33
54. Braik MS (2021) Chameleon swarm algorithm: A bio-inspired optimizer for solving engineering design problems. Expert Syst Appl, pp 114685
55. Braik M, Hammouri A, Atwan J, Al-Betar MA, Awadallah MA (2022) White shark optimizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems. Knowl-Based Syst 243:108457
56. Alimoradi M, Azgomi H, Asghari A (2022) Trees social relations optimization algorithm: A new swarm-based metaheuristic technique to solve continuous and discrete optimization problems. Math Comput Simul 194:629–664
57. Meraihi Y, Gabis AB, Ramdane-Cherif A, Acheli D (2020) A comprehensive survey of crow search algorithm and its applications. Artif Intell Rev, pp 1–48
58. Dehghani M, Trojovská E, Trojovskỳ P (2022) Driving training-based optimization: A new human-based metaheuristic algorithm for solving optimization problems
59. Emamin H (2022) Stock exchange trading optimization algorithm: a human-inspired method for global optimization. J Supercomput 78(2):2125–2174
60. Olga A, Olga Z, Julia O (2019) The optimization of business processes at the enterprises of agro-industrial complex. Int Multidiscip Sci GeoConference: SGEM, 19(5.3):863–868
61. Braik M, Al-Zoubi H, Al-Hiary H (2021) Artificial neural networks training via bio-inspired optimisation algorithms: modelling industrial winding process, case study. Soft Comput 25(6):4545–4569
62. Braik M, Sheta A, Al-Hiary H, Aljahdali S (2022) Enhanced cuckoo search algorithm for industrial winding process modeling. J Intell Manuf, pp 1–30
63. Braik M (2022) Enhanced ali baba and the forty thieves algorithm for feature selection. Neural Comput Appl, pp 1–32
64. Hashim FA, Houssein EH, Hussain K, Mabrouk MS, Al-Atabany W (2020) A modified henry gas solubility optimization for solving motif discovery problem. Neural Comput Appl 32(14):10759–10771
65. Ortega-Sánchez N, Rodríguez-Esparza E, Oliva D, Pérez-Cisneros M, Mohamed AW, Dhiman G, Hernández-Montelongo R (2022) Identification of apple diseases in digital images by using the gaining-sharing knowledge-based algorithm for multilevel thresholding. Soft Comput 26(5):2587–2623

66. Aranguren I, Valdivia A, Morales-Castañeda B, Oliva D, Elaziz MA, Perez-Cisneros M (2021) Improving the segmentation of magnetic resonance brain images using the lshade optimization algorithm. Biomed Sig Process Control 64:102259
67. Devi SG, Sabrigiriraj M (2019) A hybrid multi-objective firefly and simulated annealing based algorithm for big data classification. Concurr Comput: Pract Experience 31(14):e4985
68. Hussain K, Salleh MNM, Cheng S, Shi Y (2019) Metaheuristic research: a comprehensive survey. Artif Intell Rev 52(4):2191–2233
69. Braik M (2021) A hybrid multi-gene genetic programming with capuchin search algorithm for modeling a nonlinear challenge problem: Modeling industrial winding process, case study. Neural Process Lett 53(4):2873–2916
70. Morales-Castañeda B, Zaldivar D, Cuevas E, Fausto F, Rodríguez A (2020) A better balance in metaheuristic algorithms: Does it exist? Swarm and Evolutionary Computation 54:100671
71. Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput & Struct 169:1–12
72. Clayton N, Emery N (2005) Corvid cognition. Curr Biol 15(3):R80–R81
73. Musa JD (1975) A theory of software reliability and its application. IEEE Trans Softw Eng 3:312–327
74. Braik M, Sheta A, Turabieh H, Alhiary H (2020) A novel lifetime scheme for enhancing the convergence performance of salp swarm algorithm. Soft Comput, pp 1–26
75. Sheta A (2006) Reliability growth modeling for software fault detection using particle swarm optimization. In 2006 IEEE International Conference on Evolutionary Computation, IEEE, pp 3071–3078
76. Crow L (1974) Reliability analysis for complex repairable systems, soc. industrial and applied mathematics, reliability and biometry. Proc Stat Anal Life Length 25:248–253
77. Yamadab S, Ohba M, Osaki S (1984) S-shaped software reliability growth models and their applications. IEEE Trans Reliab 33(4):289–292
78. Yamada S, Ohba M, Osaki S (1983) S-shaped reliability growth modeling for software error detection. IEEE Trans Reliab 32(5):475–484
79. Digalakis JG, Margaritis KG (2001) On benchmarking functions for genetic algorithms. Int J Computer Math 77(4):481–506
80. Yang X-S (2010) Firefly algorithm, stochastic test functions and design optimisation. arXiv:arXiv:1003.1409
81. 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
82. Awad NH, Ali MZ, Suganthan PN (2017) Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving cec2017 benchmark problems. In 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE pp 372–379
83. Bonabeau E, Du Fnrs Marco DR, Dorigo M, Theraulaz G, et al (1999) Swarm intelligence: from natural to artificial systems. Number 1. Oxford university press
84. Kennedy J, Eberhart R (1995) Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks, volume 4, IEEE, pp 1942–1948
85. Dhiman G, Kumar V (2017) Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications. Adv Eng Softw 114:48–70
86. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
87. Dhiman G, Kumar V (2018) Emperor penguin optimizer: A bio-inspired algorithm for engineering problems. Knowl-Based Syst 159:20–50
88. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) Gsa: a gravitational search algorithm. Inf Sci 179(13):2232–2248
89. Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 27(2):495–513
90. Mirjalili S (2016) Sca: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120–133
91. Kaur S, Awasthi LK, Sangal A, Dhiman G (2020) Tunicate swarm algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. Eng Appl Artif Intell 90:103541
92. Holm S (1979) A simple sequentially rejective multiple test procedure. Scand J Stat, pages 65–70
93. Liu S-H, Mernik M, HrnčIč D, Matej Črepinšek (2013) A parameter control method of evolutionary algorithms using exploration and exploitation measures with a practical application for fitting sovova's mass transfer model. Appl Soft Comput 13(9):3792–3805
94. Gandomi AH, Yang X-S (2011) Benchmark problems in structural optimization. In Computational optimization, methods and algorithms, Springer pp 259–281
95. Arora J (2004) Optimum design concepts: optimality conditions. Introduction to Optimum Design
96. Wang G-G, Guo L, Gandomi AH, Hao G-S, Wang H (2014) Chaotic krill herd algorithm. Inf Sci 274:17–34

Alaa Sheta holds a Ph.D. in computer science from the Computer Science Department at the School of Information Technology and Engineering, George Mason University, Fairfax, VA, USA, in 1997. He earned his B.E. and M.Sc. degrees in Electronics and Communication Engineering from Cairo University, Faculty of Engineering, in 1988 and 1994, respectively. Dr. Sheta is a tenured Professor at the Computer Science Department, Southern Connecticut State University, New Haven, CT, USA. The National Science Foundation, USA, has funded his research. He has authored or co-authored more than 170 refereed journal and conference papers, six book chapters, and three books. He supervised more than 30 master's and Ph.D. students. He is a senior member of the IEEE Society. Alaa is an Associate Editor for the International Journal of Advanced Computer Science and Applications (IJACSA) and the International Journal of Computational Complexity and Intelligent Algorithms (IJCCIA). His research includes meta-heuristics search algorithms, machine learning, data mining, digital image processing, deep learning, and robotics.

Malik Sh. Braik received a B.Sc. degree in Electrical engineering from the Faculty of Engineering, Jordan University of Science and Technology, Jordan in 2000. Five years later, he received his M.Sc. degree in computer science from the Department of Information Technology, Al-Balqa Applied University, Jordan. Nine years later, he received his Ph.D. in Computer Engineering from the University of Birmingham, the U.K. He has more than 70 publications in various fields of optimization, signal processing, image processing, computer vision and artificial intelligence. His research interests include evolutionary computation, computer vision, pattern recognition, control, and system identification, computational intelligence, image processing, and meta-heuristic algorithms. He is currently working in the Department of Computer Science, Prince Abdullah bin Ghazi Faculty of Communications and Information Technology, Al-Balqa Applied University, Al-Salt, Jordan.

Heba H. Al-Hiary completed her B.Sc. degree in Computer Engineering from the Computer Engineering Department, Faculty of Engineering Technology, Al-Balqa' Applied University in 2001. Later she received her M.Sc. degree in Computer Science from the Information Technology Department, Al-Balqa Applied University, Jordan in 2005. She has been working on a variety of image processing topics related to image restoration, enhancement, edge detection, segmentation, and compression. Her current work focuses on image processing, parallel processing, software reliability and modelling, system identification and pattern recognition, evolutionary computation algorithms, and meta-heuristics. She is currently working with the Department of Computer Information Systems, Faculty of Information Technology, Al-Balqa Applied University, Al-Salt, Jordan.

Seyedali Mirjalili is a Professor at the Center for Artificial Intelligence Research and Optimization at Torrens University. He has gained international recognition for his contributions to nature-inspired artificial intelligence techniques, with over 500 published works that have received more than 80,000 citations and an H-index of 90. He has been on the list of the top 1% of highly-cited researchers since 2019, and Web of Science named him one of the most influential researchers in the world. In 2022 and 2023, The Australian newspaper recognized him as a global leader in Artificial Intelligence and a national leader in the Evolutionary Computation and Fuzzy Systems fields. He serves as a senior member of IEEE and holds editorial positions at several top AI journals, including Engineering Applications of Artificial Intelligence, Applied Soft Computing, Neurocomputing, Advances in Engineering Software, Computers in Biology and Medicine, Healthcare Analytics, Applied Intelligence, and Decision Analytics.

Authors and Affiliations

Alaa Sheta[1,5] · Malik Braik[2,5] · Heba Al-Hiary[3] · Seyedali Mirjalili[4,5]

Alaa Sheta
shetaa1@southernct.edu

Malik Braik
mbraik@bau.edu.jo

Heba Al-Hiary
hhiary@bau.edu.jo

[1] Computer Science Department, Southern Connecticut State University, 501 Crescent St, New Haven, CT 06515, USA

[2] Department of Computer Science, Al-Balqa Applied University, Al-Salt, Jordan

[3] Department of Computer Information Systems, Al-Balqa Applied University, Al-Salt, Jordan

[4] Centre for Artificial Intelligence Research and Optimisation, Torrens University Australia, Adelaide, Australia

[5] University Research and Innovation Center, Obuda University, Budapest 1034, Hungary