Evolutionary Intelligence
https://doi.org/10.1007/s12065-022-00762-7

RESEARCH PAPER

# Single candidate optimizer: a novel optimization algorithm

Tareq M. Shami[1] · David Grace[1] · Alister Burr[1] · Paul D. Mitchell[1]

Received: 8 June 2022 / Revised: 15 July 2022 / Accepted: 22 July 2022

**Abstract**
Single-solution-based optimization algorithms have gained little to no attention by the research community, unlike population-based approaches. This paper proposes a novel optimization algorithm, called Single Candidate Optimizer (SCO), that relies only on a single candidate solution throughout the whole optimization process. The proposed algorithm implements a unique set of equations to effectively update the position of the candidate solution. To balance exploration and exploitation, SCO is integrated with the two-phase strategy where the candidate solution updates its position differently in each phase. The effectiveness of the proposed approach is validated by testing it on thirty three classical benchmarking functions and four real-world engineering problems. SCO is compared with three well-known optimization algorithms, i.e., Particle Swarm Optimization, Grey Wolf Optimizer, and Gravitational Search Algorithm and with four recent high-performance algorithms: Equilibrium Optimizer, Archimedes Optimization Algorithm, Mayfly Algorithm, and Salp Swarm Algorithm. According to Friedman and Wilcoxon rank-sum tests, SCO can significantly outperform all other algorithms for the majority of the investigated problems. The results achieved by SCO motivates the design and development of new single-solution-based optimization algorithms to further improve the performance. The source code of SCO is publicly available at: https://uk.mathworks.com/matlabcentral/fileexchange/116100-single-candidate-optimizer.

**Keywords** Single candidate optimizer · Particle swarm optimization · Novel optimization algorithm · Metaheuristic · SCO

## 1 Introduction

The rapid advances in science and technology in the last decade has increased the difficulty level of real-world optimization problems and this motivates the development of fast and efficient optimization algorithms. The first step in optimization is to formulate an objective function that can be maximized or minimized. Once the optimization problem is formulated, an optimization algorithm is needed to search for the best variables that can achieve the best solution. Real-world optimization problems are mathematically formulated as follows:

$$\begin{aligned} \min \quad & f(\mathbf{x}), \mathbf{x} = x_1, x_2, ..., x_D \\ \text{s. t.} \quad & h_k(\mathbf{x}) = 0, k = 1, 2, ..., K \\ & g_l(\mathbf{x}) > 0, l = 1, 2, ..., L \\ & lb_j < x_j < ub_j, j = 1, 2, ..., D \end{aligned} \tag{1}$$

where $f(\mathbf{x})$ is the objective function that needs to be optimized, $D$, $L$ and $K$ are the numbers of dimensions (variables), inequality constraints, equality constraints, respectively, $ub_j$ and $lb_j$ represents the upper and lower bounds of variable $x$ at dimension $j$.

Generally, optimization problems can be solved by deterministic or stochastic methods. Utilizing gradient information, linear and non-linear programming are two prominent deterministic methods that can be used to find the optimal solution of a given problem. However, these conventional deterministic methods can converge to local optima [1] [2]. To overcome the limitations of the conventional approaches, meta-heuristic algorithms, as a stochastic approach, can be used to solve complicated real-world optimization problems. Meta-heuristic algorithms have shown robust performance when applied to different optimization problems in various fields such as wireless communications [3–5] and artificial intelligence [6–8].

The main merits of meta-heuristic algorithms are their simplicity, flexibility, ability to avoid a local optimum, and derivative-free mechanisms [9]. The searching process of

✉ Tareq M. Shami
tareq.al-shami@york.ac.uk

[1] Department of Electronic Engineering, University of York, Heslington, York YO10 5DD, UK

meta-heuristic algorithms is split into two phases: exploration and exploitation. The exploration stage broadly spans the search space with the aim of finding promising regions that can lead towards the optimal solution [10]. Poor exploration can lead to local optima entrapment. The exploitation phase focuses on searching around the promising regions discovered in the exploration phase. The inability to perform successful exploitation can significantly reduce the solution accuracy. Balancing between exploration and exploitation is one of the major challenges faced by meta-heuristic approaches . An efficient meta-heuristic algorithm is a one that:

1 Balances well between exploration and exploitation
2 Provides high level of accuracy
3 Converges towards the optimal solution and escapes from local optima
4 Has a stable performance where results are not significantly different from one independent run to another

The purpose of this paper is to develop a robust optimization algorithm that can be used to solve diverse real-world optimization problems. The rest of the paper is organized as follows. In Section two, a literature review on meta-heuristic algorithms is provided. Section three develops the mathematical model and the algorithm of the proposed approach. It also presents the complexity of the proposed single candidate optimizer. Sections four and five discuss the performance of the proposed optimization algorithm on thirty three benchmarking functions and four engineering problems, respectively. Finally, Section six concludes this work and provides some potential research directions.

## 2 Literature review

Meta-heuristic algorithms can be classified into four main different groups: swarm algorithms, evolutionary algorithms, physics-based approaches, and human-based algorithms [11]. Swarm intelligence algorithms are inspired by the behaviour of animals when they search for food in groups. In this category, the information of all or part of the particles is shared during an iterative optimization process. One of the most well-known swarm approaches is Particle Swarm Optimization (PSO) which was developed by Kennedy and Eberhart in 1995 [12]. PSO mimics birds flying in swarms where individuals in a swarm are guided by a leader who has the closest position to the target.

In PSO, a swarm of particles where each particle represents a potential solution flies in the search space with the aim of finding better positions that help to move toward the optimal solution. During the PSO iterative process, each particle is attracted to the global best position (gbest) which is the particle that has achieved the best fitness so far and it is also attracted to its best historical position (Pbest). Other widely known swarm algorithms are: Grey Wolf Optimization [9], Ant Colony Optimization [13], Salp Swarm Algorithm [14],Whale Optimization Algorithm [11], Krill Herd [15], Butterfly Optimization Algorithm [16], Seagull Optimization Algorithm [17], and Cuckoo Search [18].

Evolutionary algorithms as the second class of meta-heuristics are developed by imitating biological evolution such as mutation and crossover. The most well-known evolutionary algorithm is the Genetic Algorithm (GA) developed by Holland in 1992 [19]. A GA implements three main steps: selection, crossover, and mutation. In the selection process, some of the exiting candidate solutions (ones with better fitness) are selected to produce a second generation using the crossover concept. To maintain diversity, some dimensions of certain solutions are mutated with a mutation probability. Besides GA, Evolutionary Programming [20], Differential Evolution (DE) [21], Evolution Strategies [22] are three other widely used evolutionary approaches.

The third category of meta-heuristic algorithms utilizes the laws of physics such as Newton's gravitational law and Archimedes' principle to build interactions between candidate solutions. Simulated Annealing (SA) [23] and Gravitational Search Algorithm (GSA) [24] are two prominent approaches that belong to this class. SA is a single-solution-based algorithm that imitates the physical annealing process of metals. In GSA, candidate solutions are treated as a collection of masses that obey Newton's gravity and motion laws. Equilibrium Optimizer [25] and Henry Gas Solubility Optimization [26] are two recent state-of-the-art physics-based optimization algorithms. Other optimization algorithms that belong to this class are: Sine Cosine Algorithm [48], Water Cycle Algorithm [49], Black Hole algorithm [50], and Thermal Exchange Optimization [51].

The final group of meta-heuristic methods emulates the social behaviour of humans. For instance, Political Optimizer [36] and Parliamentary Optimization Algorithm [52] are two optimization algorithms inspired by the political process. Teaching-Learning-Based Optimization (TLBO) [53] is another social example where its mechanism is developed by mimicking the teaching-learning process in a classroom. Election Campaign Optimization [54], Brain Strom Optimization [55], Exchange Market Algorithm [56], Bus Transportation Algorithm [57], Group Teaching Optimization Algorithm [37], and Student Psychology Based Optimization [58] are other optimization algorithms that belong to this category.

Another classification of meta-heuristic algorithms is presented in [59] where meta-heuristic algorithms are divided into nine different categories: swarm-based, chemical-based, biology-based, physics-based, sports-based, musical-based, social-based, mathematical-based, and hybrid approaches. Besides the nine aforementioned categories, the authors in [60–62] added water-based, light-based and plant-based as three different classes of intelligent optimization algorithms.

Based on the number of candidate solutions involved in each iteration of the optimization process, meta-heuristic algorithms can also be classified into two categories: single-solution-based and population-based. In single-solution-based, only a single candidate solution is used to search for the optimal solution while in population-based methods a swarm of candidate solutions are needed. Most of the recent literature if not all as can be seen from Table 1 has focused on population-based methods as it is believed amongst the research community that single-solution-based algorithms always have poor performance compared with population-based approaches. This belief is because the performance of the three most common single-solution-based approaches, i.e, SA, Tabu Search [63], and Hill Climbing is very poor when compared with population-based algorithms. Lack of inspiration is another reason that has led to the development ignorance of single-solution-based algorithms. It is hard to find natural, physical or social phenomena that rely on a single object or creature. On the other hand, it is relatively easy to observe natural or physical behaviours generated by groups. These two main reasons have increased the popularity of population-based algorithms and at the same time neglected the development of new single-solution-based algorithms.

Many works have attempted to improve the optimization performance by introducing new ideas that can help to update the positions of potential solutions effectively. The authors in [9] proposed a Grey Wolf Optimizer (GWO) that implements a leadership hierarchy that consists of four wolves known as alpha, beta, delta, and omega. Moreover, GWO imitates the hunting behaviour of preys that is performed in three different steps: searching, encircling and attacking preys. The proposed hierarchy system of GWO provides diversity that can help to achieve good results. In [25], a novel optimization algorithm called Equilibrium Optimizer (EO) is

**Table 1** Some recent optimization algorithms

| Algorithm | Ref. | Inspiration | Year |
|---|---|---|---|
| Henry gas solubility optimization | [26] | Henry's law | 2019 |
| Harris hawks optimization (HHO) | [27] | Harris hawks attacking strategies | 2019 |
| Atom search optimization | [28] | Atomic motion model | 2019 |
| Pathfinder algorithm | [29] | Collective movements of swarms | 2019 |
| Sailfish Optimizer | [30] | Sailfish group hunting | 2019 |
| Equilibrium optimizer | [25] | Mass balance for a control volume | 2020 |
| Marine Predators Algorithm | [31] | foraging strategy of ocean predators | 2020 |
| Heap-based optimizer | [32] | Corporate rank hierarchy | 2020 |
| Gradient-based optimizer | [33] | Gradient-based Newton's approach | 2020 |
| Mayfly optimization algorithm | [34] | Flight behaviour of mayflies | 2020 |
| Bear smell search algorithm | [35] | Smelling mechanism of bears | 2020 |
| Political Optimizer | [36] | Multi-phased political process | 2020 |
| Group teaching optimization algorithm | [37] | Group teaching mechanism | 2020 |
| The Arithmetic Optimization Algorithm | [38] | Arithmetic operators | 2021 |
| Archimedes optimization algorithm | [39] | Archimedes' principle | 2021 |
| Aquila Optimizer | [40] | Aquila's behaviors | 2021 |
| Red fox optimization algorithm | [41] | Red fox hunting | 2021 |
| Horse herd optimization algorithm | [42] | Horses' herding behavior | 2021 |
| Remora optimization algorithm | [43] | Parasitic behavior of remora | 2021 |
| Dwarf Mongoose Optimization Algorithm | [44] | Foraging behavior of the dwarf mongoose | 2022 |
| Snake Optimizer | [45] | Mating behavior of snakes | 2022 |
| Ebola Optimization Algorithm | [46] | Propagation of the Ebola virus | 2022 |
| Reptile Search Algorithm | [47] | Hunting behaviour of Reptiles | 2022 |

proposed where it is inspired by mass balance for a control volume. The strength of EO mainly comes from a term called generation rate that is responsible to exploit the space and it sometimes plays an essential exploration role.

Inspired by the Archimedes' Principle, an Archimedes Optimization Algorithm (AOA) is proposed in [39]. The exploration phase of AOA is activated when objects collide with each other while exploitation takes place when no collision happens. The work in [64] developed a new nature-inspired algorithm called Moth-Flame Optimization (MFO) that mimics moths movements that rely on the moon's light to travel in a direct path. In MFO, moths are potential solutions whereas flames are the best solutions that have been obtained. Another difference between moths and flames is the updating mechanism. To promote exploration and avoid local optimum in MFO, each moth is assigned one flame only. Moreover, MFO attempts to balance exploration and exploitation by reducing the number of flames. In [65], a Honey Badger algorithm (HBA) is developed by formulating the digging behaviour to represent the exploration stage while exploitation is represented by the process of finding honey. HBA proposes a density factor that can help to smoothly switch from exploration to exploitation.

Another nature-inspired optimization algorithm called Ant Lion Optimizer (ALO) is proposed in [66]. In ALO, following the natural searching behaviour of ants, the movements of ants are modelled by a random walk. The roulette wheel operator is implemented for modelling the hunting behaviour of antlions. In addition, elitism is applied by ALO to store the best obtained solutions. The proposed random walk of ALO enhances its exploration abilities whereas elitism promotes exploitation particularly at the final stages of the ALO search process.

In [67], a novel intelligent optimization algorithm called War Strategy Optimization (WSO) is proposed where its mechanism mimics the strategical movements (defence or attack) of army troops when wars take place. Utilizing a war strategy, WSO develops a novel updating mechanism to update the position of soldiers. Moreover, a unique updating mechanism is proposed to update the positions of weak soldiers. Another interesting mechanism of WSO is to replace weak or injured soldiers with new ones or to relocate them. The proposed WSO strategies can contribute towards balancing exploration and exploitation and achieve good performances. Inspired by chemical reactions, an Artificial Chemical Reaction Optimization Algorithm (ACROA) is proposed in [68]. IN ACROA, atoms are treated as particles since they have positions and velocities. To enhance the global and local search capabilities, ACROA applies five chemical reactions: synthesis, bimolecular, redox2, displacement, and monomolecular reactions. One of the main advantages of ACROA is that it has a few parameters. The methodologies in [67, 68] and [5] can be integrated with SCO and other optimization algorithms to improve the optimization performance.

Although the state-of-the-art optimization algorithm have shown remarkable improvements in solving diverse problems, they achieve the best performance only on certain problems while their performances of different problems is far from optimal. The following summarizes six main disadvantages of existing metaheuristic algorithms:

- Many metaheuristic algorithms achieve strong exploration performances; however, their exploitation ability is weak. On the other hand, some metaheuristic algorithms can exploit the search space well; nevertheless, they have poor exploration capabilities. This results in undesired exploration-exploitation imbalance that degrades the overall optimization performance.
- Some intelligent swarm algorithms can easily coverage to local optima. These algorithms either have poor strategies that cannot avoid local optima or they do not have strong mechanisms that can help to redirect the search towards promising regions once the algorithm is trapped.
- A significant disadvantage of some algorithms is the requirement of massive number of function evaluations to achieve acceptable solutions.
- Many algorithms have various sensitive parameters where a slight change in a certain parameter can affect the performance significantly.
- Some optimization algorithms perform well on low dimensional problems; however, their performance substantially degrades as the number of dimensions increases.
- Although some algorithms can achieve promising results on unconstrained problems, they face difficulties in solving real-world constrained optimization problems.

According to the No Free Lunch (NFL) theorem [69], a meta-heuristic algorithm that performs well on a particular class of problems achieves degraded performance when it solves different sets of problems. In other words, there is no meta-heuristic algorithm that can provide the best solutions for all kind of problems. Many state-of-the-art meta-heuristic algorithms have shown promising results on a certain set of problems; however, they have demonstrated poor performance when applied to solve a different set of problems. This motivates researchers to develop novel meta-heuristic methods that achieve higher level of accuracy when applied to a wide range of optimization problems. Table 1 summarizes some recent optimization algorithms and their inspirations.

## 3 Proposed algorithm

This work proposes a novel approach that utilizes only a single candidate solution during the whole optimization process to find better solutions, unlike most of the existing searching algorithms that rely on a swarm of particles. In the proposed scheme, the overall optimization process that consists of $T$ function evaluations or iterations is divided into two phases where the candidate solution updates its position differently in each phase. Although single-solution-based algorithms and two-phase approaches are two established meta-heuristic optimization methods, they have been implemented separately. The developed approach integrates the single candidate approach with the two-phase strategy to form a single robust algorithm. Most importantly, the proposed algorithm implements a unique set of equations to update the position of the candidate solution with relying only on its information, i.e., its current position.

The purpose of the two-phase strategy is to provide diversity and balance between exploration and exploitation. The first phase in SCO terminates when $\alpha$ function evaluations are performed while the second phase consists of $\beta$ function evaluations where $\alpha + \beta = T$. In the first phase of SCO, the candidate solution updates its positions as follows:

$$x_j = \begin{cases} gbest_j + (w \mid gbest_j \mid) & \text{if } r_1 < 0.5 \\ gbest_j - (w \mid gbest_j \mid) & \text{otherwise} \end{cases} \tag{2}$$

where $r_1$ is a random variable in the range [0,1].

The mathematical definition of $w$ is given as follows:

$$w(t) = \exp^{-\left(\frac{bt}{T}\right)^b} \tag{3}$$

where $b$ is a constant, $t$ is the current function evaluation or iteration, and $T$ is the maximum number of function evaluations, respectively.

The second phase of SCO performs a deep search that starts by extensively exploring the space around the best position obtained in the first phase. The latter part of phase two reduces the space to be searched which helps to focus on promising regions only. The following shows how the candidate solution updates its position in the second phase:

$$x_j = \begin{cases} gbest_j + \left((r_2 w\left(ub_j - lb_j\right)\right) & \text{if } r_2 < 0.5 \\ gbest_j - \left((r_2 w\left(ub_j - lb_j\right)\right) & \text{otherwise} \end{cases} \tag{4}$$

where $r_2$ is another random variable in the range of [0,1], $ub_j$ and $lb_j$ are the boundary upper and lower bounds, respectively and $w$ is the most important parameter in SCO which is responsible to balance between exploration and exploitation. From (3), $w$ decreases exponentially as the number of function evaluations increase. This behaviour is crucial as a relatively high value of $w$ at the beginning of the search process helps to explore the search space effectively while a small value of $w$ strengthens the exploitation abilities at the latter stages of the optimization process. One of the main limitations of meta-heuristic algorithms is becoming trapped in local optima particularly at the latter phases of the searching process. In other words, continuous update of the positions of candidate solutions does not yield fitness improvement. SCO tackles this issue by updating the position of the candidate solution differently in the second phase if no fitness improvement is achieved in $m$ consecutive function evaluations. A counter $c$ is used to count the number of function evaluations $m$ that sequentially can not achieve fitness improvement. A binary parameter $p$ is used to determine whether the updated candidate can achieve a successful fitness or not where $p = 1$ indicates successful fitness improvement while $p = 0$ denotes fitness improvement failure. In the second phase of SCO, a candidate solution updates its position based on (4); however, if performing $m$ consecutive function evaluations does not improve the fitness value, the candidate solution updates its position as follows:

**Algorithm 1** Pseudo-code of SCO

```
1:  Set $c = 0$, $p = 0$ and define the values of $\alpha$ and $m$
2:  Generate the initial candidate solution based on equation (7) and calculate its fitness $f(gbest)$
3:  while $t <$maximum number of function evaluations do
4:      if $t < \alpha$ then
5:          Update the dimension position based on equation (2)
6:      else
7:          if $p = 0$ then
8:              $c = c + 1$
9:          end if
10:         if $c = m$ then
11:             reset the counter $c = 0$
12:             Update the dimension position based on equation (5)
13:         else
14:             Update the dimension position based on equation (4)
15:         end if
16:     end if
17:     Calculate the fitness of the new candidate solution $f(x)$
18:     if $f(x)$ is better than $f(gbest)$ then
19:         $gbest = x$
20:         $f(gbest)=f(x)$
21:         $p = 1$
22:     else
23:         $p = 0$
24:     end if
25:     t=t+1
26: end while
27: return $gbest$
```

$$x_j = \begin{cases} gbest_j + \left(\left(r_3\left(ub_j - lb_j\right)\right)\right) & \text{if } r_3 < 0.5 \\ gbest_j - \left(\left(r_3\left(ub_j - lb_j\right)\right)\right) & \text{otherwise} \end{cases} \tag{5}$$

where $r_3$ is a random number that can have a value in the range of [0,1]. The position update in (5) allows the candidate solution to shift from exploitation to exploration which is helpful to escape from local optimum.

Updating the positions of some variables can sometimes cause their values to go out of range or boundaries. To restrict variables from exceeding the boundaries, the updated positions are set as follows in case their values are higher than their upper bounds and lower bounds, respectively:

$$x_j = \begin{cases} gbest_j & \text{if } x_j > ub_j \\ gbest_j & \text{if } x_j < lb_j \end{cases} \tag{6}$$

In (6), the updated dimension of a candidate solution is assigned the same value as the global best value if the updated position goes out of boundaries.

In SCO, a single candidate solution $x$ is randomly generated and then it is iteratively updated in order to search for a better solution. The steps of the proposed algorithm are explained as follows. The process starts by randomly generating a candidate solution in the search space, evaluating its fitness, recording this candidate as *gbest* (global best position) and its fitness *f(gbest)* as the global best fitness. The initial candidate solution is generated as follows:

$$x_j = lb_j + r_4(ub_j - lb_j) \tag{7}$$

**Table 2** Unimodal test functions

| Function | Range | $f_{min}$ |
|---|---|---|
| $f_1(x) = \sum_{i=1}^{n} x_i^2$ | [−100,100] | 0 |
| $f_2(x) = \sum_{i=1}^{n} \mid x_i \mid + \prod_{i=1}^{n} \mid x_i \mid$ | [−10,10] | 0 |
| $f_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$ | [−100,100] | 0 |
| $f_4(x) = max_i\{\mid x_i \mid, 1 \le i \le n\}$ | [−100,100] | 0 |
| $f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ | [−30,30] | 0 |
| $f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$ | [−100,100] | 0 |
| $f_7(x) = \sum_{i=1}^{n} ix_i^4 + random[0, 1)$ | [−1.28, 1.28] | 0 |

**Table 3** Multimodal test functions

| Function | Range | $f_{min}$ |
|---|---|---|
| $f_8(x) = \sum_{i=1}^{n} -x_i sin(\sqrt{\mid x_i \mid})$ | [−500,500] | $-418.9829 \times Dim$ |
| $f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$ | [−5.12,5.12] | 0 |
| $f_{10}(x) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + e$ | [−32,30] | 0 |
| $f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ | [−600,600] | 0 |
| $f_{12}(x) = \frac{\pi}{n}\{10sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10sin^2(\pi y_i + 1)] + (y_n - 1)^2 + \sum_{i=1}^{n} u(x_i, 10, 100, 4)\}$ <br> $y_i = 1 + \frac{x_i+1}{4}$ <br> $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$ | [−50,50] | 0 |
| $f_{13}(x) = 0.1\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ | [−50,50] | 0 |

**Table 4** Fixed-dimension multimodal test functions

| Function | Dim | Range | $f_{min}$ |
|---|---|---|---|
| $f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$ | 2 | [−65,65] | 1 |
| $f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$ | 4 | [−5,5] | 0.00030 |
| $f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$ | 2 | [−5,5] | −1.0316 |
| $f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cosx_1 + 10$ | 2 | [−5,5] | 0.398 |
| $f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$ | 2 | [−2,2] | 3 |
| $f_{19}(x) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$ | 3 | [1,3] | −3.86 |
| $f_{20}(x) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$ | 6 | [0,1] | −3.32 |
| $f_{21}(x) = -\sum_{i=1}^{5}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$ | 4 | [0,10] | −10.1532 |
| $f_{22}(x) = -\sum_{i=1}^{7}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$ | 4 | [0,10] | −10.4028 |
| $f_{23}(x) = -\sum_{i=1}^{10}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$ | 4 | [0,10] | −10.5363 |

where $lb_j$ and $ub_j$ are the lower and upper boundaries of the search space, $r_4$ is a random number in the range of [0,1].

The repetitive process that terminates when it reaches $T$ function evaluations starts by updating the position of the candidate solution. The candidate solution $x$ updates its position in phase one and phase two based on (2) and (4), respectively. After updating the candidate position, the fitness of the newly generated candidate solution $f(x)$ is evaluated and

**Table 5** CEC2019 test functions

| No. | Function Name | Dim | Range | $f_{min}$ |
|---|---|---|---|---|
| $F_1$ | Storn's Chebyshev Polynomial Fitting Problem | 9 | [−8192,8192] | 1 |
| $F_2$ | Inverse Hilbert Matrix Problem | 16 | [−16384,16384] | 1 |
| $F_3$ | Lennard-Jones Minimum Energy Cluster | 18 | [−4,4] | 1 |
| $F_4$ | Rastrigin's Function | 10 | [−100,100] | 1 |
| $F_5$ | Griewangk's Function | 10 | [−100,100] | 1 |
| $F_6$ | Weierstrass Function | 10 | [−100,100] | 1 |
| $F_7$ | Modified Schwefel's Function | 10 | [−100,100] | 1 |
| $F_8$ | Expanded Schaffer's F6 Function | 10 | [−100,100] | 1 |
| $F_9$ | Happy Cat Function | 10 | [−100,100] | 1 |
| $F_{10}$ | Ackley Function | 10 | [−100,100] | 1 |

**Table 6** Parameter settings of all compared algorithms

| Algorithm | Parameter | Value |
|---|---|---|
| SCO | $\alpha$, b | 1000, 2.4 |
| EO | Generation probability , $a_1$, $a_2$ | 0.5, 2, 1 |
| GWO | a | Linearly decreasing from 2 to 0 |
| PSO | $C_1$, $C_2$, $w$ | 2,2,[0.9-0.4] |
| SSA | Position update probability | 0.5 |
| MA | Population size | 15 males and 15 females |
| | g, $a_1$, $a_2$, $a_3$, $\beta$, $d$, $fl$ | 0.8,1,1.5,1.5,2,0.1,0.1 |
| AOA | $C_1$, $C_2$ | 2, 6 |
| GSA | Alpha, G0, Rnorm, Rpower | 20, 100, 2, 1 |

compared with $f(gbest)$. If $f(x)$ is better than the gbest fitness $f(gbest)$, $gbest$ and $f(gbest)$ are replaced by $x$ and $f(x)$, respectively. The iterative process continues until the maximum number of function evaluations $T$ is reached. The pseudocode of the proposed algorithm is presented in Algorithm 1.

## 3.1 Complexity analysis

The computational complexity of population-based algorithms depends on four parameters: the number of candidate solutions and dimensions denoted as $N$ and $D$, respectively, the cost of evaluating the objective function $C$ and the maximum number of function evaluations $T$. In swarm algorithms, the maximum number of function evaluations is given as $T = Nt$ where $t$ is the maximum number of iterations. Generally, the minimal computational complexity of swarm algorithms including PSO, GWO and EO is composed of two main elements: initialization and main loop. Initialization involves generation of random candidate solutions and evaluating their fitness. The time complexity of generating candidate solutions is given as $O(ND)$ while $O(NC)$ represents the complexity of evaluating their fitness. Thus, the overall initialization complexity is give as $O(ND + NC)$.

The main loop is mainly composed of function evaluations and position updates. The computational complexity of evaluating the fitness for all candidate solutions in the main loop for all iterations is $O(tNC)$ which is equivalent to $O(TC)$ whereas the complexity of updating positions is $O(TD)$. The minimal computational complexity of population-based algorithms can be written as follows:

$$O(swarm_{min}) = O(ND + NC + TC + TD) \tag{8}$$

Besides O(initialization) and O(main loop), other operations such as memory savings as in PSO and EO are needed which contribute to increasing the complexity level.

In SCO, the initialization complexity is $O(D + C)$ which is lower than the initialization complexity of other algorithms ($O(ND + NC)$) as SCO has only one candidate solution. The main loop complexity of SCO is $O(TC + TD)$ which includes function evaluations and positions update with complexities of $O(TC)$ and $O(TD)$, respectively. Thus, the overall computational complexity of SCO is given as follows:

$$O(SCO) = O(D + C + TC + TD) \tag{9}$$

From (8) and (9), it is clear that the computational complexity of SCO is even lower than the minimal complexity of population-based algorithms.

## 4 Results and discussion

To validate the effectiveness of the proposed algorithm, it is tested first on a set of 23 classical benchmarking functions [9, 11, 70–72] that are divided into three different groups: unimodal, multimodal and fixed-dimension multimodal functions. Unimodal functions are used to test the exploitation ability of optimization algorithms since they have only one global optimum whereas multimodal functions assess the exploration efficiency as they have multiple local optima.

**Table 7** Results of average fitness of unimodal functions when D=30

| Fun | | SCO | EO | GWO | PSO | SSA | MA | AOA | GSA |
|---|---|---|---|---|---|---|---|---|---|
| $f_1$ | Mean | 0 | 4.73E-05 | 1.25E-02 | 2.90E+01 | 1.20E+03 | 9.38E+00 | 1.90E-14 | 2.81E+03 |
| | Std | 0 | 4.18E-05 | 8.20E-03 | 1.23E+01 | 4.83E+02 | 1.78E+01 | 4.11E-14 | 6.55E+02 |
| $f_2$ | Mean | 6.45E-258 | 8.89E-04 | 2.27E-02 | 1.82E+00 | 1.68E+01 | 1.37E+00 | 7.24E-09 | 1.33E+01 |
| | Std | 0 | 3.58E-04 | 8.41E-03 | 5.75E-01 | 4.35E+00 | 1.22E+00 | 1.44E-08 | 4.70E+00 |
| $f_3$ | Mean | 7.94E-123 | 6.25E+01 | 3.59E+02 | 2.77E+03 | 6.91E+03 | 2.36E+03 | 5.04E-09 | 5.61E+03 |
| | Std | 4.35E-122 | 9.10E+01 | 3.31E+02 | 1.03E+03 | 3.75E+03 | 1.27E+03 | 2.18E-08 | 1.95E+03 |
| $f_4$ | Mean | 3.28E-20 | 2.61E-01 | 1.39E+00 | 1.01E+01 | 2.15E+01 | 8.82E+00 | 2.89E-07 | 2.01E+01 |
| | Std | 1.80E-19 | 1.52E-01 | 4.39E-01 | 2.21E+00 | 4.39E+00 | 2.63E+00 | 8.95E-07 | 2.79E+00 |
| $f_5$ | Mean | 2.85E+01 | 2.83E+01 | 4.52E+01 | 1.42E+03 | 1.61E+05 | 2.42E+02 | 2.88E+01 | 2.61E+05 |
| | Std | 9.48E-02 | 4.28E-01 | 6.82E+01 | 9.71E+02 | 1.34E+05 | 1.82E+02 | 7.11E-02 | 1.72E+05 |
| $f_6$ | Mean | 2.63E-01 | 1.12E+00 | 3.18E+00 | 3.74E+01 | 1.16E+03 | 1.84E+01 | 5.65E+00 | 2.94E+03 |
| | Std | 9.25E-02 | 3.41E-01 | 6.90E-01 | 1.91E+01 | 4.55E+02 | 4.11E+01 | 3.28E-01 | 8.50E+02 |
| $f_7$ | Mean | 2.87E-04 | 6.87E-03 | 1.80E-02 | 8.15E-02 | 5.21E-01 | 1.14E-01 | 3.56E-03 | 4.26E-01 |
| | Std | 2.69E-04 | 2.94E-03 | 7.26E-03 | 2.55E-02 | 2.21E-01 | 5.32E-02 | 2.50E-03 | 2.30E-01 |
| Mean rank | | 1.14 | 2.57 | 3.85 | 5.85 | 7.57 | 5.14 | 2.42 | 7.42 |
| Rank | | 1 | 3 | 4 | 6 | 8 | 5 | 2 | 7 |

**Table 8** Results of average fitness of multimodal functions when D=30

| Fun | | SCO | EO | GWO | PSO | SSA | MA | AOA | GSA |
|---|---|---|---|---|---|---|---|---|---|
| $f_8$ | Mean | −8.25E+03 | −7.08E+03 | −5.34E+03 | −6.23E+03 | −6.31E+03 | −5.73E+03 | −3.62E+03 | −2.47E+03 |
| | Std | 5.86E+02 | 6.39E+02 | 1.19E+03 | 7.73E+02 | 8.54E+02 | 6.33E+02 | 4.06E+02 | 4.98E+02 |
| $f_9$ | Mean | 0 | 2.49E+00 | 3.62E+01 | 5.78E+01 | 9.20E+01 | 2.22E+01 | 2.02E+01 | 7.83E+01 |
| | Std | 0 | 2.97E+00 | 1.16E+01 | 1.52E+01 | 1.98E+01 | 7.01E+00 | 6.20E+01 | 2.46E+01 |
| $f_{10}$ | Mean | 8.88E-16 | 1.26E-03 | 2.60E-02 | 2.94E+00 | 9.45E+00 | 7.38E+00 | 1.87E-08 | 8.12E+00 |
| | Std | 0 | 6.91E-04 | 6.68E-03 | 4.40E-01 | 1.09E+00 | 1.61E+00 | 4.28E-08 | 9.34E-01 |
| $f_{11}$ | Mean | 0 | 5.57E-03 | 8.47E-02 | 1.28E+00 | 1.28E+01 | 9.59E+00 | 6.89E-03 | 4.10E+02 |
| | Std | 0 | 1.45E-02 | 6.36E-02 | 1.54E-01 | 5.05E+00 | 4.95E+00 | 3.77E-02 | 4.88E+01 |
| $f_{12}$ | Mean | 2.60E-02 | 3.44E-02 | 4.06E-01 | 1.89E+00 | 2.52E+01 | 7.40E+00 | 8.21E-01 | 4.61E+01 |
| | Std | 3.77E-02 | 1.37E-02 | 2.65E-01 | 1.21E+00 | 1.92E+01 | 3.09E+00 | 1.98E-01 | 9.15E+01 |
| $f_{13}$ | Mean | 1.45E+00 | 7.67E-01 | 2.31E+00 | 9.20E+00 | 4.40E+04 | 3.50E+01 | 2.92E+00 | 8.05E+04 |
| | Std | 5.91E-01 | 2.82E-01 | 5.36E-01 | 5.49E+00 | 8.36E+04 | 1.44E+01 | 8.94E-02 | 1.00E+05 |
| Mean rank | | 1.16 | 2.00 | 4.16 | 5.00 | 6.66 | 5.50 | 3.83 | 7.66 |
| Rank | | 1 | 2 | 4 | 5 | 7 | 6 | 3 | 8 |

The difference between the multimodal functions ($f_8 - f_{13}$) and the fixed-dimension multimodal functions ($f_{14} - f_{23}$) is that the number of variables of $f_{14} - f_{18}$ are unchangeable, unlike $f_8 - f_{13}$. In addition, the fixed-dimension multimodal functions spans different search range. Tables 2 , 3 and 4 list the mathematical representations of unimodal, multimodal and fixed-dimension multimodal functions, respectively.

To further validate the effectiveness of the proposed algorithm, its performance is evaluated on the CEC 2019 test suite. A summary of the CEC 2019 test functions that includes function name, dimension, and search range is provided in Table 5. The performance of the proposed approach is compared with three well-known optimization algorithms, i.e., PSO, GWO and GSA. It is also compared with four recent high-performance approaches: EO, AOA, MA, and SSA. EO, AOA, MA, and SSA have shown outstanding performance when applied to solve benchmarking functions and real-world engineering problems. Their results showed that they can outperform several optimization algorithms such as Success-History Based Parameter Adaptation Differential Evolution (SHADE) [73], LSHADE-SPACM, GA, DE, HHO, and L-SHADE.

The results of all compared algorithms are averaged over 30 independent runs. For all the test functions, the algorithms are compared in terms of the average fitness and the average standard deviation. All simulation results are

**Table 9** Results of average fitness of fixed-dimension multimodal functions

| Fun | | SCO | EO | GWO | PSO | SSA | MA | AOA | GSA |
|---|---|---|---|---|---|---|---|---|---|
| $f_{14}$ | Mean | 1.09E+00 | 1.48E+00 | 5.10E+00 | 4.34E+00 | 2.61E+00 | 5.60E+00 | 1.46E+00 | 9.13E+00 |
| | Std | 3.03E-01 | 1.86E+00 | 4.32E+00 | 2.64E+00 | 1.68E+00 | 3.73E+00 | 1.06E+00 | 4.42E+00 |
| $f_{15}$ | Mean | 4.53E-03 | 4.57E-03 | 5.33E-03 | 2.04E-03 | 8.67E-03 | 2.32E-03 | 1.31E-03 | 1.30E-02 |
| | Std | 1.15E-02 | 8.03E-03 | 8.48E-03 | 4.99E-03 | 1.34E-02 | 6.11E-03 | 1.09E-03 | 1.00E-02 |
| $f_{16}$ | Mean | −1.03E+00 | −1.03E+00 | −1.03E+00 | −1.03E+00 | −1.03E+00 | −1.03E+00 | −1.03E+00 | −1.03E+00 |
| | Std | 6.77E-08 | 4.70E-16 | 3.47E-07 | 4.49E-16 | 4.00E-13 | 5.45E-16 | 1.73E-03 | 9.82E-05 |
| $f_{17}$ | Mean | 3.97E-01 | 3.97E-01 | 3.97E-01 | 3.97E-01 | 3.97E-01 | 3.97E-01 | 4.18E-01 | 4.04E-01 |
| | Std | 6.00E-08 | 0 | 2.80E-05 | 0 | 1.21E-13 | 0 | 3.33E-02 | 3.27E-02 |
| $f_{18}$ | Mean | 3.00E+00 | 3.00E+00 | 3.00E+00 | 5.70E+00 | 3.00E+00 | 3.00E+00 | 4.29E+00 | 3.00E+00 |
| | Std | 8.16E-07 | 3.75E-15 | 1.93E-03 | 1.47E+01 | 1.50E-12 | 3.92E-15 | 5.02E+00 | 1.06E-14 |
| $f_{19}$ | Mean | -3.86E+00 | −3.86E+00 | −3.86E+00 | −3.86E+00 | −3.85E+00 | −3.86E+00 | −3.80E+00 | −3.84E+00 |
| | Std | 6.99E-05 | 1.43E-03 | 2.42E-03 | 1.97E-15 | 2.75E-02 | 2.43E-15 | 5.84E-02 | 2.35E-02 |
| $f_{20}$ | Mean | −3.30E+00 | −3.25E+00 | −3.25E+00 | −3.29E+00 | −3.24E+00 | −3.29E+00 | −2.73E+00 | −3.30E+00 |
| | Std | 4.98E-02 | 6.71E-02 | 7.55E-02 | 5.11E-02 | 9.35E-02 | 4.83E-02 | 2.86E-01 | 4.82E-02 |
| $f_{21}$ | Mean | −9.47E+00 | −7.80E+00 | −8.45E+00 | −4.81E+00 | −6.15E+00 | −6.52E+00 | −5.39E+00 | −4.67E+00 |
| | Std | 1.76E+00 | 3.22E+00 | 2.94E+00 | 3.12E+00 | 3.62E+00 | 3.71E+00 | 2.04E+00 | 3.36E+00 |
| $f_{22}$ | Mean | -9.61E+00 | −8.66E+00 | −9.86E+00 | −6.04E+00 | −7.43E+00 | −7.26E+00 | −4.70E+00 | −8.51E+00 |
| | Std | 2.07E+00 | 2.98E+00 | 1.86E+00 | 3.68E+00 | 3.48E+00 | 3.67E+00 | 1.70E+00 | 3.18E+00 |
| $f_{23}$ | Mean | −9.54E+00 | -8.78E+00 | −9.44E+00 | −7.43E+00 | −7.71E+00 | −6.72E+00 | −4.79E+00 | −8.73E+00 |
| | Std | 2.28E+00 | 3.24E+00 | 2.72E+00 | 3.68E+00 | 3.79E+00 | 3.66E+00 | 1.96E+00 | 3.07E+00 |
| Mean rank | | 1.40 | 2.60 | 2.70 | 3.70 | 3.80 | 3.40 | 4.30 | 4.00 |
| Rank | | 1 | 2 | 3 | 5 | 6 | 4 | 8 | 7 |

generated under equal conditions. MATLAB is used to produce the results for all algorithms on Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz with 8 GB RAM.

To provide a fair comparison, the maximum number of function evaluations for all algorithms is set to 3000. For all algorithms except SCO, the maximum number of iterations are 100 while the number of candidate solutions are 30 which is equivalent to 3000 function evaluations. Similar to all other algorithms, the maximum number of function evaluations in SCO is 3000; however, the difference is that SCO uses only one candidate solution instead of a swarm of particles. Candidate solutions are known as particles in PSO, search agents in EO, GWO, and SSA, solutions in MA, objects in AOA, and mass in GSA. Table 6 summarizes the parameter settings of all algorithms as recommended by their original papers.

Tables 7, 8, 9 show the results of the average fitness and standard deviation for the unimodal, multimodal, and fixed-dimension functions for the eight compared algorithms, respectively.

## 4.1 Exploitation analysis

As mentioned earlier, the purpose of unimodal benchmarking functions is to validate the exploitation ability of an optimization algorithm.

According to the statistical results of unimodal functions ($f_1 - f_7$) in Table 7, it is clear that SCO outperforms all other compared algorithms on all functions except $f_5$. For $f_5$, SCO is ranked second following EO and its performance is very close to the performance achieved by EO. The superior performance of the proposed approach is also shown in terms of standard deviation demonstrating that the proposed method is a more stable algorithm. The results in Table 7 shows that the SCO algorithm has strong exploitation ability.

## 4.2 Exploration analysis

The exploration ability of the proposed algorithm is validated by testing it on 16 multimodal functions that include high dimensional ($f_8 - f_{13}$) and fixed dimension ($f_{14} - f_{23}$) functions. Tables 8 and 9 provide the statistical results of all compared approaches for the $f_8 - f_{13}$ functions and for the $f_{14} - f_{23}$ functions, respectively. The results illustrate that SCO achieves better solution accuracy than other algorithms on functions $f_8 - f_{12}$, $f_{14}$, $f_{21}$, and $f_{23}$ while it achieves the best performance on $f_{16} - f_{19}$ equally with a few other algorithms such as EO and MA.

As Tables 8 and 9 show, SCO is able to achieve the optimal solutions for $f_9$, $f_{11}$, $f_{16}$, $f_{18}$ and $f_{19}$. It is also evident that SCO is the only algorithm that provides the optimal solutions for $f_9$ and $f_{11}$. For the rest of the multimodal functions,

**Table 10** Results of average fitness of $f_1$-$f_{13}$ functions when D=100

| Fun | | SCO | EO | GWO | PSO | SSA | MA | AOA | GSA |
|---|---|---|---|---|---|---|---|---|---|
| $f_1$ | Mean | 0 | 3.50E-02 | 6.36E+01 | 3.52E+03 | 1.78E+04 | 4.13E+03 | 2.45E-12 | 2.79E+03 |
| | Std | 0 | 1.65E-02 | 1.90E+01 | 6.12E+02 | 2.25E+03 | 1.23E+03 | 5.34E-12 | 7.85E+02 |
| $f_2$ | Mean | 4.73E-238 | 6.34E-02 | 3.98E+00 | 7.28E+01 | 1.04E+02 | 3.95E+01 | 3.31E-07 | 1.25E+01 |
| | Std | 0 | 1.96E-02 | 6.44E-01 | 5.18E+01 | 9.66E+00 | 7.12E+00 | 5.12E-07 | 3.38E+00 |
| $f_3$ | Mean | 6.25E-147 | 9.79E+03 | 4.29E+04 | 6.85E+04 | 8.62E+04 | 3.29E+04 | 3.10E-06 | 7.02E+03 |
| | Std | 3.42E-146 | 5.99E+03 | 1.14E+04 | 1.61E+04 | 4.61E+04 | 8.97E+03 | 1.32E-05 | 3.68E+03 |
| $f_4$ | Mean | 2.39E-17 | 1.84E+01 | 3.08E+01 | 3.54E+01 | 3.22E+01 | 2.27E+01 | 1.11E-06 | 1.93E+01 |
| | Std | 1.26E-16 | 7.19E+00 | 7.30E+00 | 2.99E+00 | 3.76E+00 | 2.05E+00 | 1.37E-06 | 2.78E+00 |
| $f_5$ | Mean | 9.88E+01 | 1.02E+02 | 4.05E+03 | 9.34E+05 | 6.30E+06 | 4.82E+04 | 9.89E+01 | 2.30E+05 |
| | Std | 5.58E-02 | 3.61E+00 | 2.88E+03 | 3.18E+05 | 1.60E+06 | 1.88E+04 | 4.14E-02 | 1.64E+05 |
| $f_6$ | Mean | 9.27E+00 | 1.65E+01 | 8.44E+01 | 3.59E+03 | 1.83E+04 | 3.90E+03 | 2.28E+01 | 2.77E+03 |
| | Std | 1.69E+00 | 1.00E+00 | 2.35E+01 | 6.90E+02 | 2.23E+03 | 8.38E+02 | 5.12E-01 | 9.53E+02 |
| $f_7$ | Mean | 6.89E-04 | 2.13E-02 | 1.86E-01 | 2.32E+00 | 1.17E+01 | 2.77E+00 | 3.11E-03 | 4.41E-01 |
| | Std | 5.42E-04 | 8.33E-03 | 6.18E-02 | 7.64E-01 | 4.41E+00 | 7.28E-01 | 2.17E-03 | 2.52E-01 |
| $f_8$ | Mean | −2.22E+04 | −1.61E+04 | −1.21E+04 | −1.69E+04 | -1.28E+04 | -1.04E+04 | −6.24E+03 | −2.56E+03 |
| | Std | 2.49E+03 | 1.81E+03 | 3.93E+03 | 1.88E+03 | 1.59E+03 | 1.74E+03 | 8.34E+02 | 5.12E+02 |
| $f_9$ | Mean | 0 | 2.46E+00 | 2.43E+02 | 4.81E+02 | 6.20E+02 | 2.24E+02 | 2.64E-12 | 7.68E+01 |
| | Std | 0 | 3.38E+00 | 5.12E+01 | 4.09E+01 | 4.27E+01 | 2.31E+01 | 5.52E-12 | 2.36E+01 |
| $f_{10}$ | Mean | 8.88E-16 | 2.77E-02 | 3.01E+00 | 8.48E+00 | 1.33E+01 | 1.10E+01 | 1.63E-07 | 7.87E+00 |
| | Std | 0 | 1.02E-02 | 3.09E-01 | 4.56E-01 | 4.43E-01 | 7.41E-01 | 3.14E-07 | 1.02E+00 |
| $f_{11}$ | Mean | 0 | 5.16E-02 | 1.63E+00 | 3.27E+01 | 1.55E+02 | 9.97E+01 | 3.29E-02 | 4.15E+02 |
| | Std | 0 | 6.69E-02 | 2.49E-01 | 5.06E+00 | 2.35E+01 | 2.09E+01 | 1.35E-01 | 5.21E+01 |
| $f_{12}$ | Mean | 1.87E-01 | 4.97E-01 | 5.66E+00 | 1.31E+04 | 1.71E+05 | 1.58E+01 | 1.04E+00 | 5.64E+01 |
| | Std | 4.56E-02 | 9.61E-02 | 2.19E+00 | 1.44E+04 | 2.08E+05 | 3.22E+00 | 7.04E-02 | 1.74E+02 |
| $f_{13}$ | Mean | 9.77E+00 | 9.08E+00 | 5.56E+01 | 3.99E+05 | 6.26E+06 | 8.17E+02 | 9.93E+00 | 1.01E+05 |
| | Std | 2.91E-01 | 5.93E-01 | 1.95E+01 | 1.88E+05 | 2.59E+06 | 1.47E+03 | 4.60E-02 | 1.47E+05 |
| Mean rank | | 1.07 | 2.76 | 4.53 | 6.23 | 7.53 | 5.84 | 2.61 | 5.38 |
| Rank | | 1 | 3 | 4 | 7 | 8 | 6 | 2 | 5 |

SCO provides a close to optimum solutions and its performance is competitive with other algorithms. Table 8 also shows that SCO is the best algorithm to solve $f_8$ where $f_8$ is considered to be one of the most difficult multimodal functions as it has a high number of local optimum.

## 4.3 Impact of high-dimensionality

Many metaheuristic algorithms achieve degraded performance when they solve high dimensional problems. Therefore, it is essential to test the high-dimensional performances of new metaheuristic algorithms. This subsection evaluates the high-dimensional performance of SCO on $f_1 - f_{13}$ by increasing the number of dimensions from 30 to 100 and 200. Tables 10 and 11 present the statistical results of SCO and the other comparative algorithms when $D = 100$ and $D = 200$, respectively. Table 10 shows that SCO outperforms all algorithms on all functions except $f_{13}$ when $D = 100$. It also shows that SCO is the only algorithm that can achieve the optimal solutions for $f_1$, $f_9$ and $f_{11}$. When $D = 200$, Table 11 shows that SCO performs better than all other algorithms on all functions except $f_5$ and $f_{13}$ where SCO is ranked second. Tables 10 and 11 have shown that SCO is not significantly affected by increasing the number of dimensions unlike other algorithms.

## 4.4 Sensitivity analysis

The SCO parameters particularly $\alpha$ and $b$ are expected to significantly influence its optimization performance. The impact of $\alpha$ and $b$ on the SCO performance is investigated in this subsection. Three different cases are studied where the first phase of SCO in case one, case two and case three consists of 500, 1000 and 2000 function evaluations, respectively. In each case, eight different scenarios are considered where the value of $b$ decreases from 3 to 0.9. The statistical results of unimodal and multimodal functions for the first case, second case and third case are presented in Table 12, Table 13, and Table 14, respectively. From these Tables, it is evident that better exploitation is achieved in the first

**Table 11** Results of average fitness of $f_1$-$f_{13}$ functions when D=200

| Fun | | SCO | EO | GWO | PSO | SSA | MA | AOA | GSA |
|---|---|---|---|---|---|---|---|---|---|
| $f_1$ | Mean | 9.53E-301 | 5.18E-01 | 1.29E+03 | 2.46E+04 | 4.54E+04 | 1.69E+04 | 1.84E-11 | 2.89E+03 |
| | Std | 0 | 3.53E-01 | 2.77E+02 | 2.93E+03 | 4.20E+03 | 3.36E+03 | 4.42E-11 | 8.08E+02 |
| $f_2$ | Mean | 6.31E-221 | 2.71E-01 | 2.69E+01 | 4.15E+02 | 2.35E+02 | 1.18E+02 | 1.28E-06 | 1.22E+01 |
| | Std | 0 | 5.73E-02 | 2.90E+00 | 8.10E+01 | 1.10E+01 | 1.00E+01 | 1.68E-06 | 3.96E+00 |
| $f_3$ | Mean | 2.88E-167 | 7.98E+04 | 2.20E+05 | 2.96E+05 | 3.63E+05 | 1.38E+05 | 1.76E-04 | 6.33E+03 |
| | Std | 0 | 3.99E+04 | 4.99E+04 | 7.53E+04 | 1.83E+05 | 4.63E+04 | 9.11E-04 | 3.77E+03 |
| $f_4$ | Mean | 2.62E-14 | 4.17E+01 | 5.88E+01 | 4.71E+01 | 3.64E+01 | 2.79E+01 | 2.37E-06 | 1.92E+01 |
| | Std | 1.43E-13 | 7.81E+00 | 5.17E+00 | 2.63E+00 | 2.77E+00 | 2.65E+00 | 4.19E-06 | 2.44E+00 |
| $f_5$ | Mean | 1.98E+02 | 2.39E+02 | 1.90E+05 | 1.41E+07 | 2.21E+07 | 9.26E+05 | 1.98E+02 | 1.97E+05 |
| | Std | 5.91E-02 | 3.30E+01 | 7.04E+04 | 2.74E+06 | 5.11E+06 | 2.85E+05 | 3.45E-02 | 1.38E+05 |
| $f_6$ | Mean | 3.35E+01 | 4.19E+01 | 1.29E+03 | 2.48E+04 | 4.56E+04 | 1.55E+04 | 4.75E+01 | 2.78E+03 |
| | Std | 2.31E+00 | 1.50E+00 | 2.85E+02 | 2.84E+03 | 4.60E+03 | 3.00E+03 | 6.00E-01 | 6.64E+02 |
| $f_7$ | Mean | 5.05E-04 | 3.63E-02 | 1.26E+00 | 4.17E+01 | 6.95E+01 | 2.42E+01 | 3.17E-03 | 3.86E-01 |
| | Std | 4.31E-04 | 1.72E-02 | 3.90E-01 | 8.65E+00 | 1.76E+01 | 8.11E+00 | 2.70E-03 | 2.04E-01 |
| $f_8$ | Mean | −3.52E+04 | −2.33E+04 | −2.28E+04 | −2.68E+04 | −1.93E+04 | −1.39E+04 | −9.21E+03 | −2.54E+03 |
| | Std | 2.98E+03 | 2.53E+03 | 4.25E+03 | 3.17E+03 | 2.08E+03 | 2.36E+03 | 1.46E+03 | 4.48E+02 |
| $f_9$ | Mean | 0 | 4.85E+00 | 7.39E+02 | 1.33E+03 | 1.51E+03 | 8.03E+02 | 1.78E-11 | 7.80E+01 |
| | Std | 0 | 5.29E+00 | 7.72E+01 | 6.27E+01 | 8.35E+01 | 6.15E+01 | 4.16E-11 | 2.25E+01 |
| $f_{10}$ | Mean | 8.88E-16 | 5.83E-02 | 4.80E+00 | 1.26E+01 | 1.42E+01 | 1.18E+01 | 1.91E-07 | 8.14E+00 |
| | Std | 0 | 1.69E-02 | 3.82E-01 | 4.04E-01 | 4.59E-01 | 5.57E-01 | 3.89E-07 | 1.03E+00 |
| $f_{11}$ | Mean | 0 | 1.85E-01 | 1.21E+01 | 2.31E+02 | 4.14E+02 | 2.71E+02 | 6.89E-03 | 4.16E+02 |
| | Std | 0 | 1.08E-01 | 2.38E+00 | 1.96E+01 | 4.07E+01 | 4.57E+01 | 3.77E-02 | 5.40E+01 |
| $f_{12}$ | Mean | 3.23E-01 | 8.68E-01 | 2.16E+01 | 2.20E+06 | 2.29E+06 | 3.93E+01 | 1.11E+00 | 1.22E+03 |
| | Std | 4.52E-02 | 1.25E-01 | 7.55E+00 | 1.16E+06 | 1.34E+06 | 1.62E+01 | 4.13E-02 | 5.75E+03 |
| $f_{13}$ | Mean | 1.99E+01 | 2.32E+01 | 1.38E+03 | 1.94E+07 | 2.61E+07 | 1.02E+05 | 1.99E+01 | 1.02E+05 |
| | Std | 9.56E-02 | 2.13E+00 | 3.26E+03 | 7.59E+06 | 8.37E+06 | 8.19E+04 | 3.84E-02 | 1.52E+05 |
| Mean rank | | 1.15 | 3.15 | 4.69 | 6.53 | 7.38 | 5.61 2.38 | 5.07 | |
| Rank | | 1 | 3 | 4 | 7 | 8 | 6 | 2 | 5 |

case where the value of $\alpha$ is 500. However, better exploration can be obtained if the value of $\alpha$ increases from 500 to 1000 or 2000.

According to the results, a good balance between exploration and exploitation is achieved when the value of $\alpha$ is 1000. Considering the parameter $b$, Tables 12, 13 and 14 show that the exploration performance of SCO degrades when $b$ decreases from 3 to 0.9 while the exploitation of SCO improves. The best performance is achieved when the value of $b$ is 2.4. Overall, based on the Friedman mean rank, the best performance of SCO is achieved when the value of $\alpha$ is 1000 and the value of $b$ is 2.4.

## 4.5 Performance of SCO on the CEC 2019 suite

The performance of SCO on the CEC 2019 test suite is presented in Table 15. The results in Table 15 show that SCO outperforms all algorithms on functions $F_1$, $F_2$, $F_5$, and $F_{10}$. It is also clear from Table 15 that SCO achieves the optimal solution on $F_1$ while all other algorithms achieve poor performance when solving the same function. The proposed algorithm achieved competitive results on $F_3$ and $F_8$ that allows it to be ranked second. The performance of SCO on the rest of the CEC 2019 functions is close to the best performance achieved by other algorithms as Table 15 illustrates.

## 4.6 Statistical significance analysis

The Friedman test as one of the most famous statistical tests is used to statistically analyze the performance of SCO. The principle of this test is to rank all compared approaches for each problem individually. For each problem, the best, second best, and third best algorithms are ranked as 1, 2, and 3 and so on. The performance of each algorithm is averaged over all problems. In Friedman test,the best approach is the one that achieves the lowest average rank. The Friedman test results that provide the average rank for all algorithms considering all test functions in low and high dimensional cases are shown in Table 16. As Table 16 illustrates, SCO achieves the lowest average rank with a value of 1.4 demonstrating

**Table 12** Statistical results of case 1 when $\alpha$ is 500 and $b$ varies from 3 to 0.9

| Fun | | b=3 | b=2.7 | b=2.4 | b=2.1 | b=1.8 | b=1.5 | b=1.2 | b=0.9 |
|---|---|---|---|---|---|---|---|---|---|
| $f_1$ | Mean | 0 | 7.01E-292 | 2.43E-287 | 4.72E-264 | 2.04E-222 | 5.15E-205 | 1.89E-167 | 3.10E-138 |
| | Std | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.69E-137 |
| $f_3$ | Mean | 5.18E-197 | 8.65E-122 | 3.51E-135 | 2.35E-160 | 2.11E-108 | 7.79E-09 | 7.94E-47 | 1.29E-71 |
| | Std | 0 | 4.73E-121 | 1.79E-134 | 1.28E-159 | 1.15E-107 | 4.26E-08 | 4.35E-46 | 7.08E-71 |
| $f_5$ | Mean | 2.86E+01 | 2.85E+01 | 2.88E+01 | 2.89E+01 | 2.89E+01 | 2.89E+01 | 2.89E+01 | 2.89E+01 |
| | Std | 9.46E-02 | 1.54E-01 | 1.25E-01 | 4.06E-02 | 3.97E-02 | 9.74E-02 | 7.28E-02 | 9.06E-02 |
| $f_7$ | Mean | 5.01E-04 | 8.82E-04 | 8.20E-04 | 8.31E-04 | 1.41E-03 | 1.32E-03 | 9.32E-04 | 1.28E-03 |
| | Std | 4.32E-04 | 9.29E-04 | 6.72E-04 | 1.27E-03 | 1.40E-03 | 8.34E-04 | 9.60E-04 | 1.39E-03 |
| $f_9$ | Mean | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Std | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $f_{11}$ | Mean | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Std | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $f_{13}$ | Mean | 1.97E+00 | 1.69E+00 | 1.59E+00 | 2.20E+00 | 2.78E+00 | 2.82E+00 | 2.87E+00 | 2.88E+00 |
| | Std | 5.82E-01 | 6.66E-01 | 5.08E-01 | 4.53E-01 | 2.45E-01 | 2.89E-01 | 2.52E-01 | 1.56E-01 |
| $f_{15}$ | Mean | 1.85E-02 | 6.74E-03 | 1.34E-02 | 1.07E-02 | 6.34E-03 | 5.97E-03 | 1.09E-02 | 6.99E-03 |
| | Std | 3.92E-02 | 2.27E-02 | 2.87E-02 | 2.31E-02 | 2.00E-02 | 9.24E-03 | 2.20E-02 | 1.29E-02 |
| $f_{17}$ | Mean | 3.97E-01 | 3.97E-01 | 3.97E-01 | 3.97E-01 | 3.97E-01 | 3.98E-01 | 3.98E-01 | 3.98E-01 |
| | Std | 0 | 4.38E-13 | 3.35E-08 | 1.23E-05 | 1.02E-04 | 5.01E-04 | 5.69E-04 | 7.19E-04 |
| $f_{19}$ | Mean | −3.86E+00 | −3.86E+00 | −3.86E+00 | −3.86E+00 | −3.86E+00 | −3.86E+00 | −3.86E+00 | −3.86E+00 |
| | Std | 8.22E-05 | 2.66E-05 | 7.23E-05 | 4.48E-04 | 1.17E-03 | 7.46E-04 | 8.34E-04 | 8.20E-04 |
| $f_{21}$ | Mean | −8.80E+00 | −9.98E+00 | −9.81E+00 | −9.64E+00 | −9.85E+00 | −9.97E+00 | −9.45E+00 | −9.74E+00 |
| | Std | 2.54E+00 | 9.22E-01 | 1.29E+00 | 1.54E+00 | 1.36E+00 | 1.28E-01 | 1.86E+00 | 2.92E-01 |
| $f_{23}$ | Mean | −9.28E+00 | −8.87E+00 | −9.04E+00 | −8.89E+00 | −9.26E+00 | −9.54E+00 | −9.43E+00 | −9.59E+00 |
| | Std | 2.87E+00 | 3.10E+00 | 3.06E+00 | 3.33E+00 | 2.82E+00 | 2.41E+00 | 2.34E+00 | 1.93E+00 |
| Mean rank | | 2.75 | 2.50 | 3.08 | 4.00 | 4.66 | 4.16 | 5.25 | 4.91 |
| Rank | | 2 | 1 | 3 | 4 | 6 | 5 | 8 | 7 |

the superiority of the proposed approach. The second best algorithm is EO, followed by AOA, GWO, MA, PSO, GSA and SSA.

Wilcoxon rank-sum test is another prominent statistical test that is widely used to validate the effectiveness of novel metaheuristic algorithms. A pair-wise comparison between SCO and the comparative algorithms at 0.05 significance level is carried out. Table 17, Table 18, Table 19, and Table 20 show the *p*-values of the Wilcoxon test for $f_1 - f_{23}$ (D=30 for $f_1 - f_{13}$), $f_1 - f_{23}$ (D=100), $f_1 - f_{23}$ (D=200), and the ten CEC2019 test functions, respectively. From these Tables, it is obvious that SCO is significantly better as compared with the state-of-the-art algorithms.

## 4.7 Convergence behavior of SCO

One of the main problems faced by optimization algorithms is convergence to local optima. To tackle this undesired convergence behavior, it is essential to balance between exploration and exploitation which in turns ensures convergence to global optima. The balance between exploration and exploitation in SCO is achieved by the implementation of the two phases. Moreover, in the second phase, the parameter *w* is used to control the right amount of space to be explored or exploited. A relatively high *w* value allows it to perform extensive exploration while a low *w* value is needed to exploit promising regions. As a result, *w* is set to have a high value at the beginning of the search process to enable efficient exploration and it is decreased as the number of function evaluations increase in order to achieve successful exploitation.

To provide a fair and concise comparison, the convergence behaviour of the proposed algorithm is compared with the best four existing algorithms, i.e, EO, GWO, AOA, and PSO. The selection of these algorithms is based on their rank as presented in Table 16. The convergence curves of SCO and the best four existing algorithms for some unimodal, multimodal and CEC 2019 functions are shown in Figure 1. From Fig. 1, it is clear that SCO outperforms all algorithms. The convergence behaviour of SCO on unimodal functions ($f_1$, $f_3$, $f_4$, $f_6$ and $f_7$) shows the superiority of SCO to rapidly exploit promising regions. From the same figure, it is evident that SCO requires only a few function evaluations to reach near optimal solutions while other algorithms require

**Table 13** Statistical results of case 2 when $\alpha$ is 1000 and $b$ varies from 3 to 0.9

| Fun | | b=3 | b=2.7 | b=2.4 | b=2.1 | b=1.8 | b=1.5 | b=1.2 | b=0.9 |
|---|---|---|---|---|---|---|---|---|---|
| $f_1$ | Mean | 0 | 0 | 0 | 0 | 3.39E-319 | 5.15E-291 | 1.28E-250 | 2.37E-195 |
| | Std | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $f_3$ | Mean | 9.61E-237 | 1.17E-213 | 7.94E-123 | 1.13E-124 | 6.24E-173 | 1.91E-58 | 9.67E-148 | 3.72E-94 |
| | Std | 0 | 0 | 4.35E-122 | 6.19E-124 | 0 | 1.04E-57 | 5.29E-147 | 2.03E-93 |
| $f_5$ | Mean | 2.86E+01 | 2.85E+01 | 2.85E+01 | 2.89E+01 | 2.89E+01 | 2.89E+01 | 2.89E+01 | 2.89E+01 |
| | Std | 9.11E-02 | 1.61E-01 | 9.48E-02 | 5.20E-02 | 4.46E-02 | 6.99E-02 | 5.15E-02 | 7.78E-02 |
| $f_7$ | Mean | 5.94E-04 | 4.77E-04 | 2.87E-04 | 8.19E-04 | 1.04E-03 | 8.15E-04 | 7.10E-04 | 9.32E-04 |
| | Std | 7.76E-04 | 5.91E-04 | 2.69E-04 | 5.96E-04 | 8.34E-04 | 6.59E-04 | 8.39E-04 | 8.56E-04 |
| $f_9$ | Mean | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Std | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $f_{11}$ | Mean | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Std | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $f_{13}$ | Mean | 1.91E+00 | 1.73E+00 | 1.45E+00 | 2.25E+00 | 2.76E+00 | 2.84E+00 | 2.93E+00 | 2.88E+00 |
| | Std | 5.19E-01 | 5.25E-01 | 5.91E-01 | 4.93E-01 | 2.79E-01 | 3.16E-01 | 1.20E-01 | 2.04E-01 |
| $f_{15}$ | Mean | 1.90E-02 | 1.85E-02 | 4.53E-03 | 8.81E-03 | 9.92E-03 | 1.05E-02 | 1.07E-02 | 7.75E-03 |
| | Std | 3.87E-02 | 3.22E-02 | 1.15E-02 | 2.37E-02 | 1.73E-02 | 2.28E-02 | 2.60E-02 | 1.32E-02 |
| $f_{17}$ | Mean | 3.97E-01 | 3.97E-01 | 3.97E-01 | 3.97E-01 | 3.98E-01 | 3.98E-01 | 3.98E-01 | 3.98E-01 |
| | Std | 0 | 3.69E-13 | 6.00E-08 | 1.94E-05 | 1.88E-04 | 3.80E-04 | 8.23E-04 | 6.59E-04 |
| $f_{19}$ | Mean | −3.76E+00 | −3.86E+00 | −3.86E+00 | −3.86E+00 | −3.86E+00 | −3.86E+00 | −3.86E+00 | −3.86E+00 |
| | Std | 5.22E-01 | 2.54E-05 | 6.99E-05 | 7.39E-04 | 2.47E-03 | 3.06E-03 | 2.17E-03 | 3.18E-03 |
| $f_{21}$ | Mean | −7.94E+00 | −8.79E+00 | −9.47E+00 | −9.38E+00 | −9.92E+00 | −9.76E+00 | −9.31E+00 | −9.41E+00 |
| | Std | 2.56E+00 | 2.29E+00 | 1.76E+00 | 2.00E+00 | 9.13E-01 | 9.16E-01 | 1.84E+00 | 1.37E+00 |
| $f_{23}$ | Mean | −7.92E+00 | −8.45E+00 | −9.54E+00 | −9.55E+00 | −1.02E+01 | −9.09E+00 | −9.61E+00 | −9.03E+00 |
| | Std | 2.88E+00 | 3.09E+00 | 2.28E+00 | 2.57E+00 | 1.47E+00 | 2.87E+00 | 2.09E+00 | 2.59E+00 |
| Mean rank | | 3.83 | 2.83 | 2.16 | 3.50 | 3.50 | 4.50 | 4.75 | 4.75 |
| Rank | | 5 | 2 | 1 | 3 | 4 | 6 | 7 | 8 |

higher numbers of function evaluations. For instance, SCO requires only 199 function evaluations to achieve a value of $10^{-10}$ (a near-optimal value) when solving $f_1$ while all other algorithms are not able to reach this value with 3000 function evaluations except AOA that requires 1553 function evaluations. This demonstrates the efficient convergence behaviour of SCO when dealing with unimodal functions.

Similarly, SCO has shown its ability to escape from local optima quickly as illustrated in Fig. 1 ($f_9$, $f_{10}$, $f_{11}$, $f_{21}$). Figure 1 also shows that SCO is a fast optimization algorithm. Overall, the convergence curves shown in Fig. 1 illustrate the superiority of SCO in terms of convergence speed when it is used to solve different kind of optimization problems.

## 5 Engineering problems

The proposed algorithm is tested on four widely used real-world engineering problems in order to further validate its effectiveness. Real-world optimization problems usually have a number of constraints that must be satisfied. The presence of constraints divides particles or candidate solutions into two groups: valid and invalid candidate solutions. A valid candidate solution is a one that can satisfy all constraints whereas a candidate solution that violates one or more constraints is invalid. To penalize an invalid candidate solution in minimization problems, its fitness is assigned a large value, for example $10^9$. For all engineering problems, the parameter setting of all algorithms are the same parameters presented in Table 6 except that the maximum number of function evaluations is 15000. The following presents the engineering problems and the obtained results for all compared algorithms while their mathematical formulations are provided in [9, 74].

### 5.1 Welded beam design (WBD)

Welded beam design is one of the most well-known real-world engineering problems that serves as a benchmark to validate the effectiveness of meta-heuristic algorithms. This problem aims to minimize the fabrication cost when designing a welded beam. The welded beam design problem has four variables and five constraints as shown in Appendix A.

**Table 14** Statistical results of case 3 when $a$ is 2000 and $b$ varies from 3 to 0.9

| Fun | | b=3 | b=2.7 | b=2.4 | b=2.1 | b=1.8 | b=1.5 | b=1.2 | b=0.9 |
|---|---|---|---|---|---|---|---|---|---|
| $f_1$ | Mean | 0 | 0 | 0 | 0 | 0 | 0 | 3.57E-319 | 5.32E-291 |
| | Std | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $f_3$ | Mean | 1.96E-239 | 1.21E-290 | 2.11E-159 | 4.15E-240 | 2.20E-157 | 7.73E-109 | 1.49E-177 | 6.47E-149 |
| | Std | 0 | 0 | 1.15E-158 | 0 | 1.20E-156 | 4.23E-108 | 0 | 3.32E-148 |
| $f_5$ | Mean | 2.86E+01 | 2.86E+01 | 2.88E+01 | 2.89E+01 | 2.89E+01 | 2.89E+01 | 2.89E+01 | 2.89E+01 |
| | Std | 1.12E-01 | 1.25E-01 | 8.36E-02 | 3.56E-02 | 6.56E-02 | 7.99E-02 | 3.91E-02 | 6.10E-02 |
| $f_7$ | Mean | 3.56E-04 | 2.19E-04 | 4.37E-04 | 3.50E-04 | 6.42E-04 | 4.71E-04 | 4.63E-04 | 5.11E-04 |
| | Std | 4.36E-04 | 2.17E-04 | 4.03E-04 | 3.33E-04 | 5.90E-04 | 4.61E-04 | 4.78E-04 | 5.12E-04 |
| $f_9$ | Mean | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Std | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $f_{11}$ | Mean | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Std | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $f_{13}$ | Mean | 2.93E+00 | 2.44E+00 | 1.68E+00 | 2.24E+00 | 2.86E+00 | 2.92E+00 | 2.96E+00 | 2.94E+00 |
| | Std | 3.86E-02 | 4.22E-01 | 5.55E-01 | 4.70E-01 | 2.93E-01 | 2.72E-01 | 1.41E-01 | 7.76E-02 |
| $f_{15}$ | Mean | 5.26E-02 | 1.80E-02 | 3.37E-02 | 1.83E-02 | 8.86E-03 | 1.16E-02 | 1.59E-02 | 1.42E-02 |
| | Std | 4.97E-02 | 3.60E-02 | 4.87E-02 | 3.97E-02 | 2.21E-02 | 2.42E-02 | 3.39E-02 | 2.71E-02 |
| $f_{17}$ | Mean | 4.1E+00 | 1.26E+00 | 3.97E-01 | 3.97E-01 | 3.98E-01 | 3.98E-01 | 3.99E-01 | 3.99E-01 |
| | Std | 2.20E+00 | 1.05E+00 | 9.30E-08 | 2.12E-05 | 3.20E-04 | 1.43E-03 | 3.03E-03 | 3.68E-03 |
| $f_{19}$ | Mean | −3.41E+00 | −3.00E+00 | −3.09E+00 | −3.67E+00 | −3.00E+00 | −3.76E+00 | −3.85E+00 | −3.85E+00 |
| | Std | 1.09E+00 | 1.33E+00 | 1.28E+00 | 7.25E-01 | 1.33E+00 | 5.21E-01 | 4.53E-03 | 6.29E-03 |
| $f_{21}$ | Mean | −9.42E-01 | −5.04E+00 | −5.05E+00 | −5.05E+00 | −5.02E+00 | −8.45E+00 | −9.42E+00 | −9.01E+00 |
| | Std | 1.38E+00 | 3.11E-02 | 4.58E-06 | 1.92E-03 | 3.64E-02 | 2.16E+00 | 5.99E-01 | 9.37E-01 |
| $f_{23}$ | Mean | −1.36E+00 | −5.08E+00 | −5.12E+00 | −5.03E+00 | −6.17E+00 | −8.92E+00 | −8.55E+00 | −8.12E+00 |
| | Std | 1.70E+00 | 1.68E-01 | 3.74E-06 | 4.93E-01 | 2.16E+00 | 2.03E+00 | 2.45E+00 | 2.21E+00 |
| Mean rank | | 4.50 | 3.33 | 3.25 | 3.33 | 4.16 | 3.25 | 3.66 | 3.75 |
| Rank | | 8 | 3 | 1 | 4 | 7 | 2 | 5 | 6 |

Table 21 shows the best solutions obtained by all compared algorithms including the best variables and the best fitness. From Table 21, it is clear that SCO provides the best fitness besides EO, PSO, MA, and AOA. It is also evident from Table 21 that SCO requires fewer number of function evaluations.

The statistical results of all algorithms for the welded beam design problem are presented in Table 22.

### 5.2 Speed reducer design problem (SRD)

SRD deals with designing a speed reducer for small aircraft engine with the objective of minimizing the weight of the speed reducer. As shown in Appendix B [75], the number of constraints and variables of the SRD minimization problem are 11 and 7, respectively. Table 23 shows the best obtained solutions in terms of best variables, best weight, and the number of function evaluations for the proposed and the state-of-the-art algorithms. From Table 23, it is shown that SCO outperforms GWO, SSA and GSA algorithms in terms of the best obtained weight while it achieves the same performance as EO, PSO, MA and AOA. The statistical results of all compared algorithms are presented in Table 24.

### 5.3 Pressure vessel design problem (PVD)

The main aim of the PVD problem is to minimize the total cost when designing a pressure vessel. As shown in Appendix C, four constrains must be satisfied to solve the PVD problem while four variables are involved to compute the objective function. The best achieved solutions and the statistical results of all algorithms are presented in Tables 25 and 26, respectively. The results in Table 25 demonstrates the superiority of SCO in terms of achieving the best cost. Moreover, SCO requires fewer number of function evaluations to achieve better cost compared with other algorithms.

### 5.4 Tension/compression spring design problem (TSDP)

TSDP involves designing a tension/compression spring where the main objective is to minimize weight. The TSDP problem contains 3 variables and 4 constraints as illustrated in Appendix D. Table 27 compares the performance of all compared algorithms in terms of best achieved variables, best achieved solution, and the number of function

**Table 15** Results of average fitness of CEC 2019 benchmarking functions

| Fun | | SCO | EO | GWO | PSO | SSA | MA | AOA | GSA |
|---|---|---|---|---|---|---|---|---|---|
| $F_1$ | Mean | 1 | 1.29E+05 | 9.79E+05 | 1.33E+06 | 6.03E+06 | 1.63E+07 | 1.60E+01 | 1.88E+09 |
| | Std | 8.00E-05 | 3.23E+05 | 1.90E+06 | 1.04E+06 | 5.05E+06 | 1.73E+07 | 6.38E+01 | 8.34E+08 |
| $F_2$ | Mean | 4.97E+00 | 5.64E+02 | 1.42E+03 | 1.37E+03 | 3.40E+03 | 2.88E+03 | 6.77E+00 | 3.28E+04 |
| | Std | 6.06E-01 | 4.48E+02 | 6.12E+02 | 6.52E+02 | 1.56E+03 | 1.16E+03 | 9.56E+00 | 9.77E+03 |
| $F_3$ | Mean | 4.29E+00 | 4.37E+00 | 5.2779E+00 | 5.22E+00 | 6.19E+00 | 2.52E+00 | 6.45E+00 | 8.48E+00 |
| | Std | 1.56E+00 | 1.53E+00 | 2.85E+00 | 2.19E+00 | 1.88E+00 | 1.81E+00 | 1.14E+00 | 1.36E+00 |
| $F_4$ | Mean | 4.25E+01 | 1.91E+01 | 2.81E+01 | 3.12E+01 | 4.28E+01 | 2.63E+01 | 7.69E+01 | 1.11E+02 |
| | Std | 2.30E+01 | 8.07E+00 | 1.59E+01 | 1.03E+01 | 2.34E+01 | 8.57E+00 | 1.27E+01 | 1.68E+01 |
| $F_5$ | Mean | 1.14E+00 | 1.23E+00 | 2.79E+00 | 1.29E+00 | 1.22E+00 | 1.45E+00 | 5.43E+01 | 1.20E+02 |
| | Std | 9.60E-02 | 1.31E-01 | 2.20E+00 | 2.42E-01 | 1.23E-01 | 5.00E-01 | 1.89E+01 | 3.10E+01 |
| $F_6$ | Mean | 6.17E+00 | 2.13E+00 | 3.94E+00 | 2.69E+00 | 7.36E+00 | 6.27E+00 | 9.15E+00 | 1.23E+01 |
| | Std | 1.67E+00 | 7.63E-01 | 1.48E+00 | 1.31E+00 | 1.68E+00 | 1.23E+00 | 1.04E+00 | 9.23E-01 |
| $F_7$ | Mean | 1.14E+03 | 9.03E+02 | 1.20E+03 | 8.99E+02 | 1.24E+03 | 1.19E+03 | 1.79E+03 | 2.39E+03 |
| | Std | 3.81E+02 | 3.12E+02 | 5.06E+02 | 3.01E+02 | 3.01E+02 | 3.76E+02 | 2.09E+02 | 2.85E+02 |
| $F_8$ | Mean | 4.32E+00 | 4.12E+00 | 4.33E+00 | 4.43E+00 | 4.61E+00 | 4.84E+00 | 4.69E+00 | 5.38E+00 |
| | Std | 4.80E-01 | 4.27E-01 | 3.06E-01 | 3.67E-01 | 4.02E-01 | 2.63E-01 | 2.58E-01 | 1.44E-01 |
| $F_9$ | Mean | 1.37E+00 | 1.22E+00 | 1.32E+00 | 1.26E+00 | 1.48E+00 | 1.24E+00 | 3.12E+00 | 4.27E+00 |
| | Std | 1.29E-01 | 7.26E-02 | 1.32E-01 | 7.83E-02 | 1.93E-01 | 1.22E-01 | 5.56E-01 | 6.57E-01 |
| $F_{10}$ | Mean | 2.03E+01 | 2.16E+01 | 2.16E+01 | 2.15E+01 | 2.10E+01 | 2.14E+01 | 2.14E+01 | 2.11E+01 |
| | Std | 3.28E+00 | 1.10E-01 | 1.14E-01 | 1.53E-01 | 1.04E-01 | 6.13E-01 | 2.95E-01 | 2.02E-01 |
| Mean rank | | 2.50 | 2.60 | 4.50 | 3.70 | 5.20 | 4.30 | 5.70 | 7.50 |
| Rank | | 1 | 2 | 5 | 3 | 6 | 4 | 7 | 8 |

**Table 16** Friedman test result

| | SCO | EO | GWO | PSO | SSA | MA | AOA | GSA |
|---|---|---|---|---|---|---|---|---|
| Mean Rank | 1.40 | 2.61 | 4.07 | 5.17 | 6.36 | 4.96 | 3.54 | 6.17 |
| Rank | 1 | 2 | 4 | 6 | 8 | 5 | 3 | 7 |

evaluations required to reach the value of the best weight. It is clear from Table 27 that SCO achieves the best solution with requiring only 1445 function evaluations while other algorithms require more than 4170 function evaluations (Table 28).

SCO has achieved significant and remarkable optimization improvements because it implements a unique set of equations that can effectively update the position of the candidate solution throughout the entire optimization process. The proposed unique set of updating equations allows SCO to extensively explore the search space in the early stages of the SCO optimization process. In this exploration phase, SCO updates its position based on an equation that allows the candidate solution to visit as many new locations as possible. In other words, SCO broadly yet effectively explores the search space to discover places where the optimal solution might be found.

The integration of SCO with the two-phase strategy has shown its effectiveness in balancing exploration and exploitation. According to the results, when SCO performs 500 function evaluations only during the exploration phase and 2500 function evaluations for exploitation, SCO achieves promising exploitation performances. However, the exploration performance of SCO degrades. This happens because SCO did not spend enough time to explore the space. As a result, SCO skips some regions where the optimal solution might be located. On the other hand, increasing the number of function evaluations from 500 to 1000 for the exploration phase has demonstrated that SCO performs well on unimodal and multimodal functions. This happens as a result of giving SCO enough time for exploration without affecting the exploitation process as SCO still spends two-thirds of the optimization process searching around the discovered promising areas. Overall, the best SCO performance is achieved when one-third of the optimization process is dedicated for exploration while the remaining SCO process focuses on exploitation.

**Table 17** The *p*-values of a pair-wise comparison between SCO and the other comparative algorithms at 0.05 significance level for $f_1 - f_{13}$ when D=30 and $f_{14} - f_{23}$

| Fun | EO | GWO | PSO | SSA | MA | AOA | GSA |
|---|---|---|---|---|---|---|---|
| $f_1$ | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 |
| $f_2$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 |
| $f_3$ | 1.6179E-11 | 1.6179E-11 | 1.6179E-11 | 1.6179E-11 | 1.6179E-11 | 1.6179E-11 | 1.6179E-11 |
| $f_4$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 |
| $f_5$ | 1.2870E-09 | 4.6159E-10 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | **1.9073E-01** | 3.0199E-11 |
| $f_6$ | 3.3384E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0180E-11 |
| $f_7$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 1.3289E-10 | 3.0199E-11 |
| $f_8$ | 2.8314E-08 | 3.0199E-11 | 6.0658E-11 | 1.2057E-10 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 |
| $f_9$ | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 5.6375E-09 | 1.2118E-12 |
| $f_{10}$ | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 |
| $f_{11}$ | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.6435E-11 | 1.2118E-12 |
| $f_{12}$ | 1.3272E-02 | 2.1544E-10 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 |
| $f_{13}$ | 8.1975E-07 | 1.7290E-06 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.6897E-11 | 3.0199E-11 |
| $f_{14}$ | 5.0572E-06 | 7.3846E-11 | 1.0804E-08 | **5.7338E-02** | 9.9186E-11 | 2.0152E-08 | 3.0199E-11 |
| $f_{15}$ | **2.7071E-01** | **7.8446E-01** | **6.5671E-02** | **1.4128E-01** | 1.3562E-07 | **4.7335E-01** | 1.3594E-07 |
| $f_{16}$ | 2.3638E-12 | 5.0922E-08 | 3.1578E-12 | 3.0199E-11 | 1.2455E-11 | 3.0199E-11 | 1.6865E-10 |
| $f_{17}$ | 1.2118E-12 | 3.0199E-11 | 1.2118E-12 | 3.0066E-11 | 1.2118E-12 | 3.0199E-11 | 1.2384E-09 |
| $f_{18}$ | 2.9654E-11 | 6.6955E-11 | 5.5392E-10 | 3.0199E-11 | 2.9580E-11 | 3.0199E-11 | 3.0066E-11 |
| $f_{19}$ | 4.3543E-10 | 2.8314E-08 | 1.6933E-11 | 2.0523E-03 | 1.2455E-11 | 3.0199E-11 | 4.1997E-10 |
| $f_{20}$ | **7.3940E-01** | 8.3520E-08 | 7.6588E-05 | **4.8252E-01** | 1.1706E-05 | 4.0772E-11 | 1.3242E-02 |
| $f_{21}$ | **8.8830E-01** | 1.7294E-07 | 2.8389E-04 | 4.8252E-03 | 9.5853E-03 | 4.1825E-09 | 1.8515E-03 |
| $f_{22}$ | 3.0339E-03 | 6.5261E-07 | 2.9047E-02 | 2.7071E-02 | 2.8789E-03 | 1.1023E-08 | 1.2643E-03 |
| $f_{23}$ | 5.2978E-03 | 2.4913E-06 | 2.3985E-03 | 5.7460E-02 | 8.7607E-01 | 2.6015E-08 | 1.3954E-03 |
| + | 18 | 22 | 19 | 20 | 18 | 22 | 21 |
| ≈ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| – | 5 | 1 | 4 | 3 | 5 | 1 | 2 |

Bold face show the *p*-values that are higher than 0.05. '+', '≈', and '−' show when SCO achieves significant improvements, statically similar performances, and achieves significant degradation compared with other algorithms, respectively

**Table 18** The *p*-values of a pair-wise comparison between SCO and the other comparative algorithms at 0.05 significance level for $f_1 - f_{13}$ when D=100

| Fun | EO | GWO | PSO | SSA | MA | AOA | GSA |
|---|---|---|---|---|---|---|---|
| $f_1$ | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 |
| $f_2$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 |
| $f_3$ | 2.8646E-11 | 2.8646E-11 | 2.8646E-11 | 2.8646E-11 | 2.8646E-11 | 2.8646E-11 | 2.8646E-11 |
| $f_4$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 |
| $f_5$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 1.0315E-02 | 3.0199E-11 |
| $f_6$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 |
| $f_7$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 9.0632E-08 | 3.0199E-11 |
| $f_8$ | 2.3715E-10 | 3.3384E-11 | 1.1737E-09 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 |
| $f_9$ | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 5.7786E-09 | 1.2118E-12 |
| $f_{10}$ | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 |
| $f_{11}$ | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 |
| $f_{12}$ | 3.6897E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 |
| $f_{13}$ | 3.3242E-06 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | **8.0727E-01** | 3.0199E-11 |
| + | 12 | 13 | 13 | 13 | 13 | 13 | 13 |
| ≈ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| – | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

Bold face show the *p*-values that are higher than 0.05

**Table 19** The *p*-values of a pair-wise comparison between SCO and the other comparative algorithms at 0.05 significance level for $f_1 - f_{13}$ when D=200

| Fun | EO | GWO | PSO | SSA | MA | AOA | GSA |
|---|---|---|---|---|---|---|---|
| $f_1$ | 4.1110E-12 | 4.1110E-12 | 4.1110E-12 | 4.1110E-12 | 4.1110E-12 | 4.1110E-12 | 4.1110E-12 |
| $f_2$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 |
| $f_3$ | 3.0010E-11 | 3.0010E-11 | 3.0010E-11 | 3.0010E-11 | 3.0010E-11 | 3.0010E-11 | 3.0010E-11 |
| $f_4$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 |
| $f_5$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.6439E-02 | 3.0199E-11 |
| $f_6$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0180E-11 |
| $f_7$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 2.9215E-09 | 3.0199E-11 |
| $f_8$ | 3.0199E-11 | 3.0199E-11 | 5.5727E-10 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 |
| $f_9$ | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.9254E-09 | 1.2118E-12 |
| $f_{10}$ | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 |
| $f_{11}$ | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 | 1.2118E-12 |
| $f_{12}$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 |
| $f_{13}$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 2.4327E-05 | 3.0199E-11 |
| + | 13 | 13 | 13 | 13 | 13 | 11 | 13 |
| ≈ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| – | 0 | 0 | 0 | 0 | 0 | 2 | 0 |

**Table 20** The *p*-values of a pair-wise comparison between SCO and the other comparative algorithms at 0.05 significance level for the CEC2019 test functions

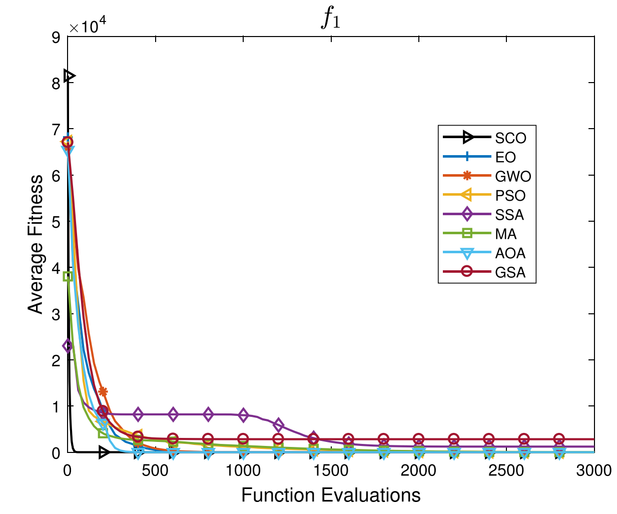
| Fun | EO | GWO | PSO | SSA | MA | AOA | GSA |
|---|---|---|---|---|---|---|---|
| $F_1$ | 3.8202E-10 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 1.4085E-07 | 3.0199E-11 |
| $F_2$ | 2.9822E-11 | 2.9822E-11 | 2.9822E-11 | 2.9822E-11 | 2.9822E-11 | 9.1168E-01 | 2.9822E-11 |
| $F_3$ | **9.8231E-01** | 3.1830E-02 | 9.3341E-02 | 1.7836E-04 | 1.0407E-04 | 1.1937E-06 | 1.7769E-10 |
| $F_4$ | 3.0103E-07 | 7.9590E-03 | 9.9258E-03 | **8.8830E-01** | 1.1738E-03 | 4.8011E-07 | 1.4643E-10 |
| $F_5$ | 4.0330E-03 | 3.0199E-11 | 1.3272E-02 | 1.0315E-02 | 7.2951E-04 | 3.0199E-11 | 3.0199E-11 |
| $F_6$ | 4.5043E-11 | 3.3242E-06 | 8.8910E-10 | 7.2884E-03 | **3.6322E-01** | 3.6459E-08 | 3.0199E-11 |
| $F_7$ | 2.6077E-02 | **8.5338E-01** | 1.6955E-02 | 4.1191E-02 | 4.8252E-02 | 4.5726E-09 | 4.0772E-11 |
| $F_8$ | 8.5000E-02 | **9.7052E-01** | 2.8378E-02 | 2.8129E-02 | 1.0188E-05 | 1.0576E-03 | 4.0772E-11 |
| $F_9$ | 2.6784E-06 | 7.4827E-02 | 3.7704E-04 | 2.4157E-02 | 3.3679E-04 | 6.0658E-11 | 3.0199E-11 |
| $F_{10}$ | 3.0199E-11 | 3.0199E-11 | 3.0199E-11 | 5.5999E-07 | 4.5726E-09 | 5.0723E-10 | 1.5178E-03 |
| + | 5 | 7 | 6 | 10 | 7 | 10 | 10 |
| ≈ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| – | 5 | 3 | 4 | 0 | 3 | 0 | 0 |

Bold face show the *p*-values that are higher than 0.05

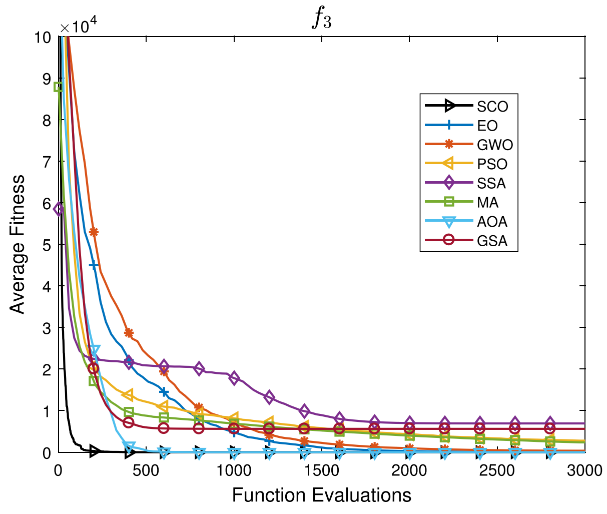
SCO does not only avoid local optima entrapment, it also implements the escape from local optima strategy to smoothly switch from exploration into exploitation in case SCO is stagnated. The integration of SCO and this strategy can help to enhance the performance particularly for problems that have many local optima as results have shown. In addition, the exploration and exploitation abilities of SCO heavily relies on the parameter *w*. This parameter can help to balance exploration and exploitation if its values during the iterative SCO process is chosen properly. The parameter *w* should have a relatively high value at the beginning of the SCO process to promote exploration. As the number of function evaluations increase, the value of *w* should decrease to strength the exploitation abilities.

To summarize, the proposed unique set of updating equations, the two-phase strategy, the escape from local optima strategy, and the parameter *w* are the main contributors that have supported SCO to achieve promising results.

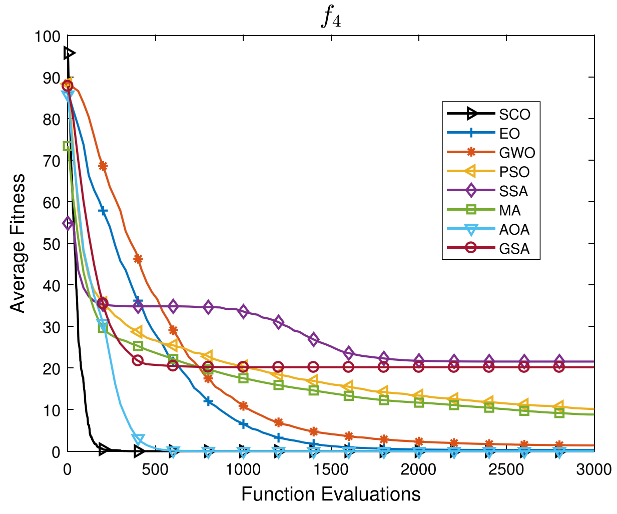
One of the main strengths of SCO is its strong exploration abilities that is achieved by effectively updating the position of the candidate solution. In addition, SCO starts a deep exploitation search after an extensive exploration



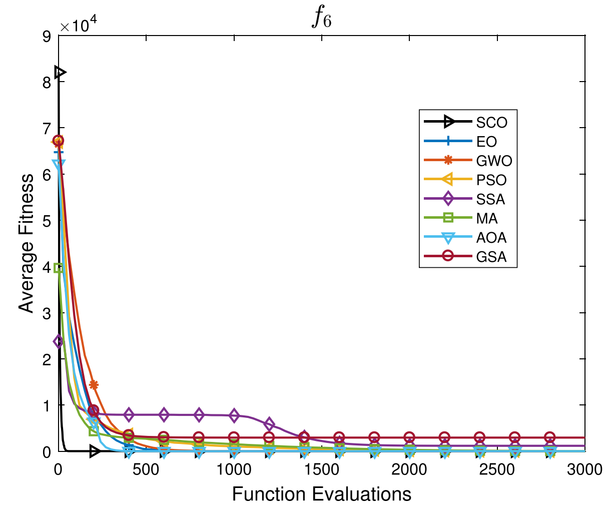
**(a)** $f_1$



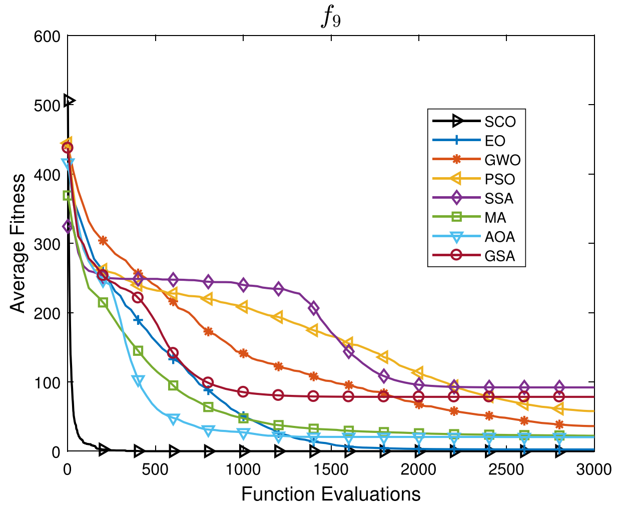
**(b)** $f_3$



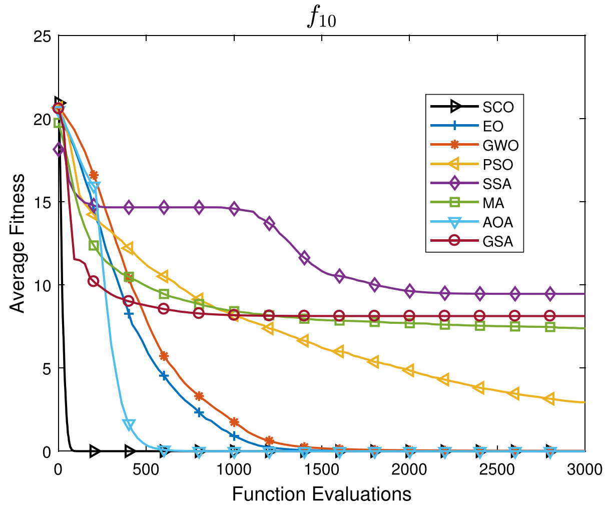
**(c)** $f_4$



**(d)** $f_6$



**(e)** $f_9$



**(f)** $f_{10}$

**Fig. 1** Convergence curves for some benchmarking functions

**Table 21** Best results of the comparative algorithms for the welded beam design problem

| Algorithm | $x_1$ | $x_2$ | $x_3$ | $x_4$ | Optimal cost | FEs |
|---|---|---|---|---|---|---|
| SCO | 0.1988 | 3.3374 | 9.1921 | 0.1988 | 1.6702 | 7502 |
| EO | 0.1988 | 3.3374 | 9.1920 | 0.1988 | 1.6702 | 9660 |
| GWO | 0.1981 | 3.3490 | 9.2019 | 0.1988 | 1.6720 | 14910 |
| PSO | 0.1988 | 3.3374 | 9.1920 | 0.1988 | 1.6702 | 11070 |
| SSA | 0.1909 | 3.4946 | 9.1920 | 0.1988 | 1.6789 | 13320 |
| MA | 0.1988 | 3.3374 | 9.1922 | 0.1988 | 1.6702 | 14610 |
| AOA | 0.1988 | 3.3374 | 9.1920 | 0.1988 | 1.6702 | 8820 |
| GSA | 0.1762 | 3.6965 | 9.6205 | 0.1969 | 1.7396 | 10620 |

**Table 22** Statistical results of the comparative algorithms for the welded beam design problem

| Algorithm | Mean | Best | Worst | Std |
|---|---|---|---|---|
| SCO | 1.7407 | 1.6702 | 2.1182 | 0.0888 |
| EO | 1.6720 | 1.6702 | 1.6875 | 0.0042 |
| GWO | 1.6764 | 1.6720 | 1.6863 | 0.0036 |
| PSO | 1.6754 | 1.6702 | 1.8100 | 0.0255 |
| SSA | 1.8305 | 1.6789 | 2.2434 | 0.1472 |
| MA | 1.6744 | 1.6702 | 1.7316 | 0.0118 |
| AOA | 1.6915 | 1.6702 | 1.8235 | 0.0354 |
| GSA | 2.3702 | 1.7396 | 3.1031 | 0.3133 |

**Table 23** Best results of the comparative algorithms for the speed reducer problem

| Algorithm | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | Optimal weight | FEs |
|---|---|---|---|---|---|---|---|---|---|
| SCO | 3.5000 | 0.7000 | 17.0000 | 7.3018 | 7.7153 | 3.3505 | 5.2867 | 2994.4 | 7203 |
| EO | 3.5000 | 0.7000 | 17.0000 | 7.3000 | 7.7153 | 3.3505 | 5.2867 | 2994.4 | 6720 |
| GWO | 3.5012 | 0.7000 | 17.0036 | 7.3527 | 7.8575 | 3.3542 | 5.2919 | 3003.4 | 15000 |
| PSO | 3.5000 | 0.7000 | 17.0000 | 7.3000 | 7.7153 | 3.3505 | 5.2867 | 2994.4 | 10080 |
| SSA | 3.5044 | 0.7000 | 17.0000 | 7.3018 | 7.8773 | 3.3531 | 5.2867 | 3000.4 | 12720 |
| MA | 3.5000 | 0.7000 | 17.0000 | 7.3000 | 7.7153 | 3.3505 | 5.2867 | 2994.4 | 9870 |
| AOA | 3.5000 | 0.7000 | 17.0000 | 7.3000 | 7.7153 | 3.3505 | 5.2867 | 2994.4 | 7410 |
| GSA | 3.5266 | 0.7052 | 18.1755 | 8.2003 | 7.8907 | 3.5628 | 5.3968 | 3384.4 | 6510 |

**Table 24** Statistical results of the comparative algorithms for the speed reducer problem

| Algorithm | Mean | Best | Worst | Std |
|---|---|---|---|---|
| SCO | 2995.8 | 2994.4 | 2999.3 | 1.0883 |
| EO | 2995.9 | 2994.4 | 3007.4 | 3.9136 |
| GWO | 3416.8 | 3003.4 | 5238 | 705.2673 |
| PSO | 3026.6 | 2994.4 | 3149.3 | 31.4321 |
| SSA | 3038.4 | 3000.4 | 3115.6 | 24.2014 |
| MA | 13334 | 2994.4 | 1.0000e+09 | 3.4574e+08 |
| AOA | 3000.3 | 2994.4 | 3034.6 | 8.6124 |
| GSA | 4287.3 | 3384.4 | 5598.8 | 653.4043 |

**Table 25** Best results of the comparative algorithms for the pressure vessel design problem

| Algorithm | $x_1$ | $x_2$ | $x_3$ | $x_4$ | Optimal cost | FEs |
|---|---|---|---|---|---|---|
| SCO | 0.7784 | 0.3848 | 40.3323 | 199.8267 | 5885.8 | 8644 |
| EO | 0.8163 | 0.4035 | 42.2963 | 174.1997 | 5953.8 | 9690 |
| GWO | 0.7823 | 0.3896 | 40.5307 | 197.0943 | 5901.3 | 15000 |
| PSO | 0.7909 | 0.3909 | 40.9773 | 191.0422 | 5903.4 | 11280 |
| SSA | 0.7876 | 0.3893 | 40.8104 | 196.3365 | 5968.9 | 14220 |
| MA | 0.8196 | 0.4051 | 42.4661 | 172.1333 | 5960.1 | 8340 |
| AOA | 0.7801 | 0.3856 | 40.4173 | 198.6448 | 5888.6 | 8250 |
| GSA | 1.7248 | 0.8526 | 89.3684 | 69.5338 | 24708 | 8370 |

**Table 26** Statistical results of the comparative algorithms for the pressure vessel design problem

| Algorithm | Mean | Best | Worst | Std |
|---|---|---|---|---|
| SCO | 6534.0 | 5885.8 | 7299.0 | 505.5225 |
| EO | 6607.2 | 5953.8 | 7319 | 522.7266 |
| GWO | 6075.8 | 5901.3 | 7263.1 | 335.0595 |
| PSO | 6346.8 | 5907.4 | 7319.0 | 386.8015 |
| SSA | 9430.1 | 5968.9 | 71768 | 11908 |
| MA | 6320.9 | 5960.1 | 6914.6 | 278.9480 |
| AOA | 6513.6 | 5888.6 | 7319 | 516.5846 |
| GSA | 2.2155E+05 | 2.4708E+04 | 4.9272E+05 | 1.0874E+05 |

**Table 27** Best results of the comparative algorithms for the tension/compression spring design problem

| Algorithm | $x_1$ | $x_2$ | $x_3$ | Optimal weight | FEs |
|---|---|---|---|---|---|
| SCO | 0.0530 | 0.3885 | 9.6450 | 0.0127 | 1445 |
| EO | 0.0550 | 0.4427 | 7.5922 | 0.0129 | 4170 |
| GWO | 0.0527 | 0.3827 | 9.9155 | 0.0127 | 14610 |
| PSO | 0.0505 | 0.3286 | 13.1501 | 0.0127 | 4350 |
| SSA | 0.0521 | 0.3663 | 10.7795 | 0.0127 | 8040 |
| MA | 0.0516 | 0.3552 | 11.3775 | 0.0127 | 4440 |
| AOA | 0.0527 | 0.3827 | 9.9138 | 0.0127 | 5280 |
| GSA | 0.0567 | 0.4732 | 7.7796 | 0.0149 | 2580 |

**Table 28** Statistical results of the comparative algorithms for the tension/compression spring design problem

| Algorithm | Mean | Best | Worst | Std |
|---|---|---|---|---|
| SCO | 0.0159 | 0.0127 | 0.0178 | 0.0017 |
| EO | 0.0138 | 0.0129 | 0.0178 | 0.0011 |
| GWO | 0.0129 | 0.0127 | 0.0136 | 0.0002 |
| PSO | 0.0136 | 0.0127 | 0.0177 | 0.0011 |
| SSA | 0.0143 | 0.0127 | 0.0220 | 0.0026 |
| MA | 0.0131 | 0.0127 | 0.0178 | 0.0009 |
| AOA | 0.0133 | 0.0127 | 0.0172 | 0.0010 |
| GSA | 0.0235 | 0.0149 | 0.0406 | 0.0070 |

search is performed. SCO also have a higher potential to avoid local optimum. However, it might be stuck at local optima. SCO can tackle this problem by the implementation of the escape from local optima strategy which allows SCO to smoothly switch from the exploitation process into an exploration mission. In terms of computational complexity, SCO has lower complexity compared with swarm algorithms as shown in (9). Moreover, SCO does not require a massive number of function evaluations to achieve good performances. In this work, only 3000 function evaluations are needed to produce optimal and near-optimal solutions.

Although SCO has shown superior performances on most of the investigated unconstrained and constrained problems, it still suffers from two main limitations. Similar to swarm algorithms, SCO replaces the entire global best position once a new candidate solution can achieve a better fitness even if a few dimensions of the newly candidate solution obtained at iteration $t$are worse than their corresponding dimensions of the global best position found at iteration $t-1$. In some cases, combining the best dimensions (not necessarily all) of the candidate solution at iteration $t$ and the best dimensions of the candidate solutions at iteration $t-1$ might generate a new candidate solution that can achieve a better fitness compared with the fitness of the best candidate solution found so far. This combination has been investigated in [76] for PSO and its implementation has shown significant improvements. However, the approach in [76] is computationally expensive. Therefore, it is needed to develop new methods to tackle this issue for SCO and swarm algorithms. Another limitation of SCO is that careful selections of the parameters $b$ and $\alpha$ are required to achieve the best performances.

## 6 Conclusions and future research

This paper proposes a novel optimization algorithm called Single Candidate Optimizer (SCO) that implements a unique set of equations to effectively update the position of the candidate solution. To balance between exploration and exploitation, the two-phase strategy is applied where the candidate solution updates its position differently eat each phase. SCO also implements a escape from local optima strategy which permits the candidate solution to shift from an exploitation mode into an exploration mode in the second phase of SCO. The integration of SCO with the two-phase strategy and the escape from local optimum method allows the candidate solution to explore and exploit the search space well. The effectiveness of SCO is validated by testing it on thirty-three classical benchmarking functions and four real-world engineering problems. The performance of SCO is compared with 7 well-known and recent optimization algorithms including PSO, GWO, EO and AOA. Results of unimodal and multimodal functions have demonstrated that SCO can effectively explore the search space in the first phase and it then switches to the second phase to perform deep exploitation. Moreover, SCO has shown that it can avoid and escape from local optima particularly when it solves functions with multiple optima. For most of the studied problems, results have shown that SCO can achieve optimal and near-optimal solutions and its performance is significantly better than other algorithms in terms of solution accuracy and convergence speed. According to the results, the best SCO performance is achieved when it spends one-third of the optimization process exploring the search space while the remaining SCO process focuses on exploitation. In addition, it has been demonstrated that the computational complexity of SCO is lower than the complexity of swarm algorithms. Another advantage of SCO is that it does not require massive number of functions evaluations to achieve significant performances. This work has shown that single-solution-based algorithms can outperform population-based algorithms if designed well.

Further work is needed to further improve the performance of single-solution-based algorithms. The following present some potential research directions that can further improve the performance of SCO:

- SCO can be hybridized with other algorithms such as PSO, GWO and EO.
- A new version of SCO can be developed to solve multi-objective problems.
- A binary version of SCO can be developed to solve binary problems such as the problem of feature selection.
- SCO can be applied to solve a wide range of real-world optimization problems such as lot-sizing optimization [77, 78], data clustering [79], optimizing the hyper-parameters of convolutional neural networks [80], designing supply-chain network [81], and maintenance scheduling [82].
- SCO can be integrated with chaotic maps and levy flight random walk [83].
- SCO be applied to solve well-known constrained optimization problems such as cantilever beam design and three-bar truss design.

## Appendix A: Welded beam design problem

$$\begin{aligned}
&\min_x \; f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4\left(14 + x_2\right)\\
&\text{s.t.}\quad g_1(x) = \tau(x) - \tau_{\max} \le 0\\
&\qquad g_2(x) = \sigma(x) - \sigma_{\max} \le 0\\
&\qquad g_3(x) = x_1 - x_4 \le 0\\
&\qquad g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4\left(14 + x_2\right)\\
&\qquad\qquad - 5 \le 0\\
&\qquad g_5(x) = 0.125 - x_1 \le 0\\
&\qquad g_6(x) = \delta(x) - \delta_{\max} \le 0\\
&\qquad g_7(x) = P - P_c(x) \le 0\\
&\text{range}\quad 0.1 \le x_i \le 2 \quad i = 1,4\\
&\qquad 0.1 \le x_i \le 10 \quad i = 2,3\\
&\text{where}\quad \tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\tfrac{x_2}{2R} + (\tau'')^2}\\
&\qquad \tau' = \frac{P}{\sqrt{2}x_1x_2}, \quad \tau'' = \frac{MR}{J}\\
&\qquad M = P\left(L + \frac{x_2}{2}\right)\\
&\qquad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}\\
&\qquad J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}\\
&\qquad \sigma(x) = \frac{6PL}{x_4x_3^2}, \quad \delta(x) = \frac{4PL^3}{Ex_3^3x_4}\\
&\qquad P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)\\
&\qquad \tau_{\max} = 13600psi \quad \sigma_{\max} = 30000psi\\
&\qquad \delta_{\max} = 0.25in, \quad P = 6000lb\\
&\qquad E = 30 \times 10^6 psi, \quad L = 14in\\
&\qquad G = 12 \times 10^6 psi
\end{aligned}$$

## Appendix B: Speed reducer design problem

$$\begin{aligned}
\min_x f(x) \quad & = 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right. - 1.508x_1\left(x_6^2 + x_7^2\right) \\
& + 7.4777\left(x_6^3 + x_7^3\right) + 0.7854\left(x_4x_6^2 + x_5x_7^2\right) \\
\text{s.t.} \quad & g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \le 0 \\
& g_2(x) = \frac{397.5}{x_1x_2^2x_3} - 1 \le 0 \\
& g_3(x) = \frac{1.93x_4^3}{x_2x_6^4x_3} - 1 \le 0 \\
& g_4(x) = \frac{1.93x_5^3}{x_2x_7^4x_3} - 1 \le 0 \\
& g_5(x) = \frac{\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9\times10^6}}{110x_6^3} - 1 \le 0 \\
& g_6(x) = \frac{\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5\times10^6}}{85x_7^3} - 1 \le 0 \\
& g_7(x) = \frac{x_2x_3}{40} - 1 \le 0 \\
& g_8(x) = \frac{5x_2}{x_1} - 1 \le 0 \\
& g_9(x) = \frac{x_1}{12x_2} - 1 \le 0 \\
& g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0 \\
& g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0 \\
\text{range} \quad & 2.6 \le x_1 \le 3.6, \quad 0.7 \le x_2 \le 0.8 \\
& 17 \le x_3 \le 28, \quad 7.3 \le x_4 \le 8.3 \\
& 7.3 \le x_5 \le 8.3, \quad 2.9 \le x_6 \le 3.9 \\
& 5.0 \le x_7 \le 5.5
\end{aligned}$$

## Appendix C: Pressure vessel design problem

$$\begin{aligned}
\min_x \quad & f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 \\
& 19.84x_1^2x_3 \\
\text{s.t.} \quad & g_1(x) = -x_1 + 0.0193x_3 \le 0 \\
& g_2(x) = -x_2 + 0.00954x_3 \le 0 \\
& g_3(x) = x_4 - 240 \le 0 \\
& g_4(x) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0 \\
\text{range} \quad & 0 \le x_i \le 100, \quad i = 1,2 \\
& 10 \le x_i \le 200, \quad i = 3,4
\end{aligned}$$

## Appendix D: Tension/compression spring design problem

$$\begin{aligned}
\min_x \quad & f(x) = x_1^2x_2\left(x_3 + 2\right) \\
\text{s.t.} \quad & g_1(x) = \frac{x_1 + x_2}{1.5} - 1 \le 0 \\
& g_2(x) = 1 - \frac{x_2^3x_3}{71785x_1^4} \le 0 \\
& g_3(x) = \frac{4x_2^2 - x_1x_2}{12566\left(x_2x_1^3 - x_1^4\right)} + \frac{1}{5108x_1^2} - 1 \le 0 \\
& g_4(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0 \\
\text{range} \quad & 0.05 \le x_1 \le 2.00 \\
& 0.25 \le x_2 \le 1.30 \\
& 2.00 \le x_3 \le 15.00
\end{aligned}$$

**Funding** Not applicable.

**Availability of data and materials** My manuscript has no associated data.

**Declaration**

**Conflict of interest** The authors declare that they have no financial or personal relationships related to this work.

## References

1. Afshar M, Faramarzi A (2010) Size optimization of truss structures by cellular automata. J Comput Sci Eng 3(1):1–9
2. Faramarzi A, Afshar M (2014) A novel hybrid cellular automata-linear programming approach for the optimal sizing of planar truss structures. Civil Eng Environ Syst 31(3):209–228
3. Shami TM, Grace D, Burr A, Vardakas JS (2019) Load balancing and control with interference mitigation in 5G heterogeneous networks. EURASIP J Wireless Commun Netw 2019(1):1–12
4. Feng S, Chen Y, Zhai Q, Huang M, Shu F (2021) Optimizing computation offloading strategy in mobile edge computing based on swarm intelligence algorithms. EURASIP J Adv Signal Process 2021(1):1–15
5. Pham Q-V, Mirjalili S, Kumar N, Alazab M, Hwang W-J (2020) Whale optimization algorithm with applications to resource allocation in wireless networks. IEEE Trans Vehicular Technol 69(4):4285–4297
6. Al-Tashi Q, Akhir EAP, Abdulkadir SJ, Mirjalili S, Shami TM, Alhusssian H, Alqushaibi A, Alwadain A, Balogun AO, Al-Zidi N (2021) Classification of reservoir recovery factor for oil and gas reservoirs: a multi-objective feature selection approach. J Marine Sci Eng 9(8):888
7. Moayedi H, Nguyen H, Kok Foong L (2021) Nonlinear evolutionary swarm intelligence of grasshopper optimization algorithm and gray wolf optimization for weight adjustment of neural network. Eng Comput 37(2):1265–1275
8. Singh P, Chaudhury S, Panigrahi BK (2021) Hybrid mpso-cnn: multi-level particle swarm optimized hyperparameters of convolutional neural network. Swarm Evol Comput 63:100863
9. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
10. Shami TM, El-Saleh AA, Alswaitti M, Al-Tashi Q, Summakieh MA, Mirjalili S (2022) Particle swarm optimization: a comprehensive survey. IEEE Access
11. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
12. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks 4:1942–1948 . IEEE
13. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1(4):28–39
14. 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
15. Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845
16. Arora S, Singh S (2019) Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput 23(3):715–734
17. Dhiman G, Kumar V (2019) Seagull optimization algorithm: theory and its applications for large-scale industrial engineering problems. Knowl Based Syst 165:169–196
18. Yang XS, Deb S (2009) Cuckoo search via lévy flights. In: 2009 World congress on nature and biologically inspired computing (NaBIC), pp 210–214. IEEE
19. Holland J (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor
20. Fogel DB (1998) Artificial intelligence through simulated evolution. Wiley-IEEE Press, London
21. Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Glob Opt 11(4):341–359
22. Hansen N, Müller SD, Koumoutsakos P (2003) Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol Comput 11(1):1–18
23. Kirkpatrick S, Gelatt Jr CD, Vecchi MP (1987) Optimization by simulated annealing. In: Readings in computer vision, pp 606–615. Elsevier
24. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248
25. Faramarzi A, Heidarinejad M, Stephens B, Mirjalili S (2020) Equilibrium optimizer: a novel optimization algorithm. Knowl Based Syst 191:105190
26. Hashim FA, Houssein EH, Mabrouk MS, Al-Atabany W, Mirjalili S (2019) Henry gas solubility optimization: a novel physics-based algorithm. Future Generation Comput Syst 101:646–667
27. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Future Generation Comput Syst 97:849–872
28. Zhao W, Wang L, Zhang Z (2019) Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. Knowl Based Syst 163:283–304
29. Yapici H, Cetinkaya N (2019) A new meta-heuristic optimizer: pathfinder algorithm. Appl Soft Comput 78:545–568
30. Shadravan S, Naji H, Bardsiri VK (2019) The sailfish optimizer: a novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. Eng Appl Artif Intell 80:20–34
31. Faramarzi A, Heidarinejad M, Mirjalili S, Gandomi AH (2020) Marine predators algorithm: a nature-inspired metaheuristic. Exp Syst Appl 152:113377
32. Askari Q, Saeed M, Younas I (2020) Heap-based optimizer inspired by corporate rank hierarchy for global optimization. Exp Syst Appl 161:113702
33. Ahmadianfar I, Bozorg-Haddad O, Chu X (2020) Gradient-based optimizer: a new metaheuristic optimization algorithm. Inf Sci 540:131–159
34. Zervoudakis K, Tsafarakis S (2020) A mayfly optimization algorithm. Comput Ind Eng 145:106559
35. Ghasemi-Marzbali A (2020) A novel nature-inspired meta-heuristic algorithm for optimization: bear smell search algorithm. Soft Comput 24(17):13003–13035
36. Askari Q, Younas I, Saeed M (2020) Political optimizer: a novel socio-inspired meta-heuristic for global optimization. Knowl Based Syst 195:105709
37. Zhang Y, Jin Z (2020) Group teaching optimization algorithm: a novel metaheuristic method for solving global optimization problems. Exp Syst Appl 148:113246
38. Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH (2021) The arithmetic optimization algorithm. Comput Methods Appl Mech Eng 376:113609
39. 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
40. Abualigah L, Yousri D, Abd Elaziz M, Ewees AA, Al-qaness MA, Gandomi AH (2021) Aquila optimizer: a novel meta-heuristic optimization algorithm. Comput Ind Eng 157:107250
41. Połap D, Woźniak M (2021) Red fox optimization algorithm. Exp Syst Appl 166:114107
42. MiarNaeimi F, Azizyan G, Rashki M (2021) Horse herd optimization algorithm: a nature-inspired algorithm for high-dimensional optimization problems. Knowl Based Syst 213:106711
43. Jia H, Peng X, Lang C (2021) Remora optimization algorithm. Exp Syst Appl 185:115665
44. Agushaka JO, Ezugwu AE, Abualigah L (2022) Dwarf mongoose optimization algorithm. Comput Methods Appl Mech Eng 391:114570

45. Hashim FA, Hussien AG (2022) Snake optimizer: a novel metaheuristic optimization algorithm. Knowl Based Syst 242:108320
46. Oyelade ON, Ezugwu AE-S, Mohamed TI, Abualigah L (2022) Ebola optimization search algorithm: a new nature-inspired metaheuristic optimization algorithm. IEEE Access 10:16150–16177
47. Abualigah L, Abd Elaziz M, Sumari P, Geem ZW, Gandomi AH (2022) Reptile search algorithm (rsa): a nature-inspired metaheuristic optimizer. Exp Syst Appl 191:116158
48. Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl Based Syst 96:120–133
49. Eskandar H, Sadollah A, Bahreininejad A, Hamdi M (2012) Water cycle algorithm—a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct 110:151–166
50. Hatamlou A (2013) Black hole: a new heuristic optimization approach for data clustering. Inf Sci 222:175–184
51. Kaveh A, Dadras A (2017) A novel meta-heuristic optimization algorithm: thermal exchange optimization. Adv Eng Softw 110:69–84
52. Borji A (2007) A new global optimization algorithm inspired by parliamentary political competitions. In: Mexican international conference on artificial intelligence, pp 61–71 . Springer
53. Rao RV, Savsani VJ, Vakharia D (2011) Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315
54. Lv W, He C, Li D, Cheng S, Luo S, Zhang X (2010) Election campaign optimization algorithm. Proc Comput Sci 1(1):1377–1386
55. El-Abd M (2017) Global-best brain storm optimization algorithm. Swarm Evol Comput 37:27–44
56. Ghorbani N, Babaei E (2014) Exchange market algorithm. Appl Soft Comput 19:177–187
57. Bodaghi M, Samieefar K (2019) Meta-heuristic bus transportation algorithm. Iran J Comput Sci 2(1):23–32
58. Das B, Mukherjee V, Das D (2020) Student psychology based optimization algorithm: a new population based optimization algorithm for solving optimization problems. Adv Eng Softw 146:102804
59. Akyol S, Alatas B (2017) Plant intelligence based metaheuristic optimization algorithms. Artif Intell Rev 47(4):417–462
60. Alatas B, Bingol H (2020) Comparative assessment of light-based intelligent search and optimization algorithms. Light Eng **28**(6)
61. Alatas B, Bingol H (2019) A physics based novel approach for travelling tournament problem: optics inspired optimization. Inf Technol Control 48(3):373–388
62. Bingol H, Alatas B (2020) Chaos based optics inspired optimization algorithms as global solution search approach. Chaos Solitons Fractals 141:110434
63. Glover F, Laguna M (1998) Tabu search. In: Handbook of combinatorial optimization, pp 2093–2229. Springer
64. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl Based Syst 89:228–249
65. Hashim FA, Houssein EH, Hussain K, Mabrouk MS, Al-Atabany W (2022) Honey badger algorithm: New metaheuristic algorithm for solving optimization problems. Math Comput Simul 192:84–110
66. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98
67. Ayyarao TS, RamaKrishna N, Elavarasan RM, Polumahanthi N, Rambabu M, Saini G, Khan B, Alatas B (2022) War strategy optimization algorithm: a new effective metaheuristic algorithm for global optimization. IEEE Access 10:25073–25105
68. Alatas B (2011) Acroa: artificial chemical reaction optimization algorithm for global optimization. Exp Syst Appl 38(10):13170–13180
69. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evolut Comput 1(1):67–82
70. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102
71. Digalakis JG, Margaritis KG (2001) On benchmarking functions for genetic algorithms. Int J Comput Math 77(4):481–506
72. Mirjalili S, Lewis A (2013) S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evol Comput 9:1–14
73. Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for differential evolution. In: 2013 IEEE congress on evolutionary computation, pp 71–78 . IEEE
74. He X, Zhou Y (2018) Enhancing the performance of differential evolution with covariance matrix self-adaptation. Appl Soft Comput 64:227–243
75. Pant M, Thangaraj R, Singh V (2009) Optimization of mechanical design problems using improved differential evolution algorithm. Int J Recent Trends Eng 1(5):21
76. Van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239
77. Gharaei A, Hoseini Shekarabi SA, Karimi M (2020) Modelling and optimal lot-sizing of the replenishments in constrained, multi-product and bi-objective EPQ models with defective products: Generalised cross decomposition. Int J Syst Sci Oper Logist 7(3):262–274
78. Gharaei A, Karimi M, Hoseini Shekarabi SA (2020) Joint economic lot-sizing in multi-product multi-level integrated supply chains: Generalized benders decomposition. Int J Syst Sci Oper Logist 7(4):309–325
79. Alswaitti M, Albughdadi M, Isa NAM (2018) Density-based particle swarm optimization algorithm for data clustering. Exp Syst Appl 91:170–186
80. Wang Y, Zhang H, Zhang G (2019) cPSO-CNN: An efficient pso-based algorithm for fine-tuning hyper-parameters of convolutional neural networks. Swarm Evol Comput 49:114–123
81. Rabbani M, Foroozesh N, Mousavi SM, Farrokhi-Asl H (2019) Sustainable supplier selection by a new decision model based on interval-valued Fuzzy sets and possibilistic statistical reference point systems under uncertainty. Int J Syst Sci Oper Logist 6(2):162–178
82. Duan C, Deng C, Gharaei A, Wu J, Wang B (2018) Selective maintenance scheduling under stochastic maintenance quality with multiple maintenance actions. Int J Prod Res 56(23):7160–7178
83. Houssein EH, Saad MR, Hashim FA, Shaban H, Hassaballah M (2020) Lévy flight distribution: a new metaheuristic algorithm for solving engineering optimization problems. Eng Appl Artif Intell 94:103731