Knowledge-Based Systems 314 (2025) 113169

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Accelerated opposition learning based chaotic single candidate optimization algorithm: A new alternative to population-based heuristics

Ugur Yuzgec

Department of Computer Engineering, Bilecik Seyh Edebali University, Bilecik, 11100, Turkiye

ARTICLE INFO

Keywords:
Opposition learning
Chaotic
Single candidate
Engineering design
Benchmark

ABSTRACT

This study considers the Single Candidate Optimizer (SCO) as an alternative to population-based heuristics, that is faster than them. Although the SCO algorithm is a fast single-candidate-based heuristic, it has certain limitations. To overcome these limitations and enhance the search performance of SCO, several solutions were proposed in this study. First, owing to the single-candidate nature of the SCO, the initial solution position can play a critical role. To compensate for this, an accelerated opposition-learning mechanism was integrated into the SCO. In addition, instead of the equation that is active when the number of unsuccessful improvement attempts is reached in the SCO structure, a mutation operator including chaotic functions (Levy, Gauss, and Cauchy) has been incorporated into the algorithm. Again, equations based on new approaches were added to the SCO algorithm to update the position of the candidate solution during the exploration and exploitation phases. Finally, the standard boundary value control mechanism is replaced with a more effective one. The algorithm developed in this study is named Accelerated Opposition Learning based Chaotic Single Candidate Optimizer (AccOppCSCO), inspired by the accelerated opposition learning mechanism and the mutation operator involving chaotic behaviors. The search capability of the proposed AccOppCSCO algorithm was first analyzed using four different methods: convergence, search history, trajectory, and computational complexity. The effectiveness of the mechanisms used in the AccOppCSCO algorithm for four different two-dimensional benchmark problems from the IEEE Congress on Evolutionary Computation 2014 (CEC2014) package was demonstrated. Subsequently, the performance of the proposed AccOppCSCO algorithm was evaluated on the CEC2014 and IEEE Congress on Evolutionary Computation 2020 (CEC2020) benchmark problems with different dimensions. The results show that the AccOppCSCO algorithm works effectively in the CEC2014 and CEC2020 test sets and offers better optimization results than SCO. The AccOppCSCO algorithm ranked first in the overall evaluation of the 30-dimensional CEC2014 comparison results with State of the Art (SOTA) heuristics from the literature. Finally, for ten different engineering design problems, the AccOppCSCO algorithm was analyzed and compared with the original SCO and other SOTA heuristics. The results show that AccOppCSCO is effective for engineering design problems. This emphasizes that the algorithm can work effectively on a wide range of problems and can be used in various applications. The source code of the AccOppCSCO algorithm for the CEC2014 benchmark suite is publicly available at https://github.com/uguryuzgec/AccOppCSCO.

1. Introduction

Optimization is a crucial concept that plays a significant role in various real-world applications. It involves the process of making something as effective and functional as possible [1,2]. Whether in engineering, finance, or technology, optimization is used to maximize efficiency and performance. By minimizing costs in supply chain management to maximize profits in financial investment, the applications of optimization are numerous and wide ranging. [3].

One category of optimization algorithms is based on derivative techniques, which involve using the derivative of a function to determine the minimum or maximum value. Algorithms such as gradient descent and Newton's method are commonly used in machine learning and engineering applications to optimize parameters and functions [4,5]. Although derivative-based techniques are powerful tools for optimization, they present several challenges and difficulties. A major challenge is the need for a differentiable function. In real-world scenarios, many functions may not have easily computable derivatives, making it difficult to apply derivative-based optimization techniques [6,7].

Another challenge is the issue of local minima, where the optimization algorithm may obtain a suboptimal solution by not finding the global minimum. Derivative-based techniques may converge to

E-mail address: ugur.yuzgec@bilecik.edu.tr.

https://doi.org/10.1016/j.knosys.2025.113169
Received 14 November 2024; Received in revised form 4 February 2025; Accepted 9 February 2025
Available online 18 February 2025

a local optimum instead of a global optimum, particularly in non-convex functions. This can lead to suboptimal solutions and hinder the effectiveness of the optimization process [8]. Furthermore, the selection of the step size in the gradient descent and determination of the appropriate damping factor in Newton's method present additional challenges. These parameters must be carefully chosen to ensure optimal convergence and to avoid problems such as overshooting or slow progress [9].

In the context of large-scale optimization, the computational complexity of derivative-based techniques can become a significant obstacle. For high-dimensional problems, the computational complexity of derivative-based techniques can result in significant computational and temporal costs, which in turn can affect the overall efficiency of the optimization process [10]. Addressing these challenges in derivative-based optimization techniques is critical for their successful application in real-world scenarios. Researchers continue to explore and develop methods to overcome these difficulties, advancing the field of optimization and practical implementation. When addressing the challenges in derivative-based optimization techniques, it is essential to consider the connection with free derivative methods, such as heuristic algorithms.

Heuristic algorithms provide a powerful alternative to derivative-based techniques by offering flexibility in handling non-differentiable functions and navigating complex, non-convex optimization landscapes [7,8]. Heuristic optimization algorithms, such as genetic, simulated annealing, differential evolution, tabu search, and particle swarm optimization algorithms, have emerged as promising alternatives to derivative-based techniques. Heuristic algorithms can be classified into two groups: single-solution-based and population-based heuristics. The two groups differ in the number of candidates used to solve the optimization problem [11].

In single-solution-based heuristics, the solution is iteratively developed for a single candidate starting from a random search point to find the global solution. The most well-known examples are simulated annealing [12] and tabu search [13]. Population-based algorithms use a set of candidate solution individuals (population) to solve optimization problems. Each individual in the population is a candidate that represents a possible solution to the optimization problem. During the process of finding the solution to the optimization problem, the positions of individuals in the population are updated by different mechanisms [14,15].

Population-based heuristics can be categorized into four classes depending on their inspiration. Evolutionary, Physics-based, Human-based and Swarm-based heuristics [16]. Evolutionary algorithms, such as Genetic [17,18] and Differential Evolution [19,20] algorithms, are inspired by the process of natural selection and evolution, using operators such as selection, crossover, and mutation to develop a population over generations. Physics-based heuristics draw inspiration from physical laws and phenomena, thereby providing a framework for searching for optimal solutions [21]. Examples of such heuristics include Central Force Optimization [22], Gravitational Search [23], Intelligent Water Drops [24], Big Bang Big Crunch [25], and Flood [26] algorithms. The third group of population-based heuristics is based mainly on mechanisms that mimic human behavior. The examples that come to mind in this group are Teaching-Learning Based Optimization [27], Social Cognitive Optimization [28], Socio-Evolution and Learning Optimization [29], and Hiking Optimization [30] algorithms. The last group of population-based heuristics, swarm-based heuristics, is inspired by the collective behavior of natural swarms such as flocks of birds, schools of fish, or ant colonies. The most well-known swarm-based heuristics are Particle Swarm [31], Artificial Bee Colony [32,33], Ant Colony [34,35], Gray Wolf [36], Firefly [37], Bat [38], Whale Optimization [39], Electric Eel Foraging Optimization [40], Hippopotamus Optimization [41], and Polar Fox Optimization [42] algorithms.

Population-based heuristics are designed to balance exploration (diversifying searches to discover new regions) with exploitation (intensifying searches around known good solutions). This enables them to navigate complex solution spaces efficiently, thereby facilitating their search for optimal solutions [43]. Population-based heuristics are powerful tools for optimization problems; however, they have some weaknesses. One challenge is that they rely on the population size, which can result in time constraints, as larger populations require more computation time. There is also the risk that, when making intensive use of local areas of the search space, algorithms may not identify potentially superior global optima owing to premature convergence or getting stuck in non-optimal regions [44].

In recent years, new designs, improvements, and applications of heuristic algorithms have been widely reported. However, the No Free Lunch (NFL) theorem states that all optimization algorithms perform equivalently on average when applied to all possible optimization tasks. This theorem emphasizes that there is no general-purpose optimal optimization algorithm [45]. Therefore, the NFL theorem encourages research on the development of more efficient heuristic optimization algorithms. This shows that the research on heuristics is interesting.

This study considers a Single Candidate Optimizer (SCO) that reduces the long running time costs of population based heuristics and relies on a single candidate solution that is as good as their search performance. The SCO algorithm stands out for its simple structure and ability to reach successful solutions in a short time with a single candidate in the search space for optimization problems [46]. However, despite its speed and efficacy, the SCO algorithm has certain drawbacks and limitations. As it searches for a single candidate solution, the performance of this algorithm is highly sensitive to the location of the initial solution, which can affect the quality of the final solution. Additionally, the update mechanism employed in the exploration phase can result in the position of the candidate solution rapidly approaching zero, which may delay the convergence process when the optimal solution is found at non-zero points.

To overcome these drawbacks and improve the search performance of SCO algorithm, several innovations have been integrated. First, an accelerated opposition learning mechanism is added to the original SCO algorithm. Furthermore, a chaotic mutation operator was introduced to replace the equation that calculates the candidate's new position when the number of failed improvement attempts is reached. In the exploration phase, a new approach was applied to update the position of the candidate solution. In the exploitation phase, a new equation is added to update the candidate solution position. Finally, the boundary-checking mechanism of the SCO algorithm is updated to be more efficient. Owing to these innovations, the overall performance and quality of the algorithm solution are significantly improved.

The algorithm introduced in this paper is called the Accelerated Opposition Learning-based Chaotic Single Candidate Optimization (AccOppCSCO) algorithm. This term reflects the improvements made to the original SCO algorithm, which incorporates accelerated opposition learning and a chaotic mutation operator. The motivation for this study is summarized as follows:

1. To achieve problem-solving success of population-based heuristics with a single candidate in less time,
2. To eliminate the disadvantages of the single candidate optimizer structure,
3. To improve the search performance of the SCO algorithm with innovations such as opposition-based learning and the chaotic mutation operator.

These three main motivations played a critical role in the development of the proposed algorithm. First, the goal of achieving the success of population-based methods with a single candidate in a shorter time is necessary to speed up the optimization process by reducing high computational costs. The motivation to eliminate the disadvantages of single candidate optimization reduces the dependence of the algorithm on the initial solution, resulting in a more stable and reliable search process. Finally, the goal of improving the search performance of the

SCO algorithm is achieved through mechanisms such as opposition-based learning and chaotic mutation operators, which make it possible to explore a wider solution space and avoid getting stuck in local minima. The proposed improvements improved the overall performance of the algorithm by making both the exploration and exploitation phases more efficient, leading to more competitive results.

One of the key techniques in optimization is the Probability-directed random search algorithm (PD-RSA) [47], which combines random search techniques with an efficient routing mechanism to expedite convergence to the global optimum. However, PD-RSA exhibits limitations in its exploration process due to its random routing mechanism. The proposed AccOppCSCO algorithm differs from PD-RSA in several aspects. Firstly, PD-RSA emphasizes improving exploitation through a triangular routing mechanism, whereas AccOppCSCO enhances exploration utilizing chaotic mutation mechanisms. Secondly, PD-RSA dynamically updates routing parameters without adaptive components, while AccOppCSCO employs chaotic mutation and opposition learning for adaptive updates. Lastly, AccOppCSCO demonstrates superior performance on high-dimensional problems due to its opposition learning strategy and chaotic mutations, rendering it a viable alternative for real-world optimization tasks. These distinctions elucidate the advantages of AccOppCSCO over PD-RSA, substantiating its integration into contemporary optimization frameworks.

The remainder of this paper is organized as follows: Section 2 presents the equations and operations of the original single-candidate optimizer algorithm. Section 3 describes the AccOppCSCO algorithm, which includes structures and procedures integrated into the original SCO algorithm. Section 4 presents the results of the comparative analysis of the AccOppCSCO algorithm on different benchmark test suites (CEC2014 and CEC2020) and real-world problems. Finally, Section 5 presents the conclusions of this study.

2. Single Candidate Optimizer (SCO)

The Single Candidate Optimizer (SCO), a novel approach, differs from conventional search algorithms because it uses only a single candidate solution throughout the optimization process. Unlike population-based algorithms that rely on multiple search agents, SCO focuses solely on a single solution to improve optimization results. By dividing the optimization process into two distinct phases, SCO incorporates different strategies and mechanisms that facilitate exploration and exploitation capabilities. This unique blend of the single candidate method and two-phase strategy enables SCO to take advantage of the advantages of each approach, resulting in a robust and adaptive optimization algorithm [46]. In the first phase of SCO, the candidate solution position is updated according to the following equation:

$$X(j) = \begin{cases} S(j) + w(t)|S(j)|, \text{if } r_1 < 0.5 \\ S(j) - w(t)|S(j)|, \text{otherwise} \end{cases} \quad (1)$$

$$w(t) = e^{-\left(\frac{b.t}{t_{max}}\right)^b} \quad (2)$$

where, $S(j)$ stands for the best candidate solution at the end of each iteration, $X(j)$ is the updated value of $S(j)$, $w(t)$ is the iteration-dependent weighting factor, r_1 is a random number in the interval [0,1], t represents the iteration, t_{max} is the maximum number of iterations, b is a constant that is used as 2.4 in the algorithm. To move to the first phase, an α parameter equal to one-third of the total number of iterations in the algorithm was used. Until the number of iterations reaches α, SCO attempts to find the best solution in the search space in the exploration phase.

In the second stage of the SCO algorithm, a counter was used, unlike in the first stage. This counter increases in each iteration when there is no improvement in the cost value of the updated candidate solution. When the counter value reaches the number of unsuccessful attempts to improve the candidate solution (m=5), the counter is reset and the following mechanism updates the position of the candidate solution.

$$X(j) = \begin{cases} S(j) + r_3[u_b(j) - l_b(j)], \text{if } r_2 < 0.5 \\ S(j) - r_4[u_b(j) - l_b(j)], \text{otherwise} \end{cases} \quad (3)$$

Here, $u_b(j)$ and $l_b(j)$ are the boundary values (upper and lower), r_2, r_3, and r_4 are the numbers produced by the random number generator. If, in this phase, the counter value is less than the threshold value (m), SCO calculates the position of the candidate solution according to the updated equations given in the following equation:

$$X(j) = \begin{cases} S(j) + w(t)r_6[u_b(j) - l_b(j)], \text{if } r_5 < 0.5 \\ S(j) - w(t)r_6[u_b(j) - l_b(j)], \text{otherwise} \end{cases} \quad (4)$$

where r_5, and r_6 denote random numbers in the range 0-1. At the end of the update, there is a possibility that the candidate solution is outside the bounds of the problem dimensions. In this case, at the end of each iteration, the position of the updated candidate is checked to be outside the boundaries of the search space. If the candidate solution is out of bounds, the updated candidate solution is considered the best candidate solution. The equations for this standard checking mechanism are as follows.

$$X(j) = \begin{cases} S(j), \text{if } x(j) > u_b(j) \\ S(j), \text{if } x(j) < l_b(j) \end{cases} \quad (5)$$

The pseudocode of the SCO algorithm is presented in Algorithm 1.

3. Accelerated opposition learning based chaotic single candidate optimizer (AccOppCSCO)

Although the SCO algorithm is fast and successful, it has several limitations. Because SCO searches for a single candidate, its performance is sensitive to the position of the initial candidate solution, which may affect the quality of the final solution. Another disadvantage is that SCO's update mechanism in the exploration phase (presented in Eq. (1)) can cause the position of the candidate solution to rapidly converge to zero. This can potentially delay convergence when the optimal solution lies at non-zero solution points. To address these limitations and enhance the search performance of the SCO algorithm, I integrated the following operations into the SCO:

1. An accelerated opposition-based learning mechanism is integrated into the algorithm structure to improve the search performance of the original SCO algorithm.
2. A chaotic mutation operator was placed in the original SCO instead of Eq. (3), which calculates the new position of the candidate when the number of unsuccessful improvement attempts has been reached.
3. During the exploration phase, a new approach was implemented to update the candidate solution position.
4. A new update equation is added to the mechanism used to update the candidate solution position during the exploitation phase.
5. The SCO algorithm's boundary-checking mechanism was updated to be more effective.

The new algorithm proposed in this paper is called the Accelerated Opposition Learning-based Chaotic Single Candidate Optimization (AccOppCSCO) algorithm because the improvements in the original SCO algorithm are based on accelerated opposition learning and a mutation operator based on chaotic functions. Opposition-Based Learning (OBL) integrates the concept of opposition into optimization techniques, where solutions are assessed based not only on their original values but also on their opposites. Including opposing solutions introduces further diversity and enhances the exploration within the search space, potentially resulting in enhanced optimization results [48]. Dinkar and Deep [49] proposed enhancing the OBL strategy using an acceleration coefficient (e_{ac}) to expedite the convergence rate of the optimization process.

Algorithm 1 Pseudocode of Single Candidate Optimizer (SCO)

Initialize: l_b, u_b, dim, f_c, Max_iter, m ,α, b
$FitImp \leftarrow 0$ ▷ FitImp=0 indicates no fitness improvement
for $i = 1$ to dim **do**
  $S(i) \leftarrow lb(i) + \text{rand} \times (ub(i) - lb(i))$ ▷ Global best Position
end for
$BF \leftarrow f_c(S)$ ▷ BF denotes the best fitness
$Count \leftarrow 0$ ▷ Counter to count unsuccessful fitness improvements
for $t = 1$ to Max_iter **do**
  $w(t) \leftarrow \exp\left(-\left(\frac{b \cdot t}{Max\_iter}\right)^b\right)$
  if $t > \alpha$ and $FitImp = 0$ **then**
    $Count \leftarrow 1 + Count$ ▷ Counting the number of unsuccessful fitness improvements
  end if
  $K \leftarrow$ rand
  for $j = 1$ to dim **do**
    if $t < \alpha$ **then**
      if rand < 0.5 **then**
        $X(j) \leftarrow S(j) + w(t) \times |S(j)|$
      else
        $X(j) \leftarrow S(j) - w(t) \times |S(j)|$
      end if
    else
      if $Count = m$ **then**
        $Count \leftarrow 0$ ▷ Reset counter
        if rand < 0.5 **then**
          $X(j) \leftarrow S(j) + \text{rand} \times (u_b(j) - l_b(j))$
        else
          $X(j) \leftarrow S(j) - \text{rand} \times (u_b(j) - l_b(j))$
        end if
      else
        if rand < 0.5 **then**
          $X(j) \leftarrow S(j) + w(t) \times K \times (u_b(j) - l_b(j))$
        else
          $X(j) \leftarrow S(j) - w(t) \times K \times (u_b(j) - l_b(j))$
        end if
      end if
    end if
    if $x(j) > u_b(j)$ **then**
      $X(j) \leftarrow S(j)$
    end if
    if $x(j) < l_b(j)$ **then**
      $X(j) \leftarrow S(j)$
    end if
  end for
  if $f_c(X) < BF$ **then**
    $BF \leftarrow f_c(X)$, $S \leftarrow X$, $FitImp \leftarrow 1$
  else
    $FitImp \leftarrow 0$
  end if
  $gbest \leftarrow S$
end for

The opposition-based learning mechanism operates in two main phases: at the beginning and end of each iteration during optimization. During the opposition-based initialization phase, the current candidate generates its opposite counterpart. This approach produces an individual that is considered more favorable than that obtained via random initialization. By integrating the opposition-based method, this initialization step improves the suitability of the selected individual for the subsequent optimization processes. The procedure is outlined below.

$$\hat{S} = (u_b + l_b) - S \tag{6}$$

where S denotes the initial candidate's position and $\hat{S}$ represents the opposite position of the initial candidate. After each optimization iteration, a candidate was selected using a predetermined jump probability. This candidate is then opposed, and the superior candidates for the next iteration are chosen by combining the current and opposing candidates. The opposition-based generation jump is executed only if the jump rate (J_r) is compared with a random number (r) within the range [0,1], as shown in the following expression:

$$\hat{X} = (u_b + l_b) - X, \text{if } r < J_r \tag{7}$$

where X denotes the updated candidate's position and $\hat{X}$ represents the opposite position of the updated candidate. While the OBL strategy expedites convergence in the optimization process, the resulting opposite candidate might display unexpected exploratory tendencies. While increased exploration may be advantageous in the early stages of generation, the algorithm should gradually move towards exploitation as the number of generations increases. Therefore, I introduced a secondary strategy to make the OBL adaptive by incorporating an acceleration coefficient (e_{ac}) to balance exploration and exploitation. The speedup coefficient e_{ac} approximates this balance and adaptively reduces as the number of iterations increases [49]. The calculation of this parameter (e_{ac}) can be formulated as follows:

$$e_{ac} = e_{ac,max} - \frac{t(e_{ac,max} - e_{ac,min})}{t_{max}} \tag{8}$$

where $e_{ac,max}$ is the maximum value (1), and $e_{ac,min}$ is the minimum value (10^{-5}). As can be seen from this equation, e_{ac} decreases from 1 to 10^{-5} depending on iteration during the optimization process. The new version of the opposition-based update mechanism given in Eq. (7) is as follows according to the coefficient e_{ac}:

$$\hat{X} = e_{ac} \cdot (r_1 \cdot u_b + r_2 \cdot l_b) - X \tag{9}$$

where r_1 and r_2 are random numbers in [0-1]. Second, a chaotic-based mutation operator was integrated into the SCO algorithm. The mutation operator replaces the update mechanism in Eq. (3) in the original algorithm. The mutation operator is based on three fundamental chaotic functions: Cauchy, Gaussian, and Levy functions. The pseudocode for the mutation operator is shown in Algorithm 2. The mutation operator utilizes only one of the three chaotic mutation operators, depending on the random number used in the pseudocode. The objective is to incorporate the power of three different chaotic functions into the mutation operator. Cauchy-based mutation, introduced by Yao et al. [50], addresses issues related to premature convergence and stagnation in local optima that commonly affect various optimization algorithms. The Cauchy-based mutation procedure is as follows:

$$X(j) = S(j)\left(1 + \tan\left(\pi(r - \frac{1}{2})\right)\right) \tag{10}$$

Algorithm 2 Pseudocode of mutation operator used in AccOppCSCO

for $j = 1$ to dim **do**
  / ***** /
  $r_1 \leftarrow$ rand()
  if $r_1 > 0.7$ **then**
    $X(j) \leftarrow$ Cauchy($S(j)$) ▷ Cauchy Mutation
  else if $r_1 > 0.3$ and $r_1 < 0.7$ **then**
    $X(j) \leftarrow$ Gauss($S(j)$) ▷ Gauss Mutation
  else
    $X(j) \leftarrow$ Lévy($S(j)$) ▷ Lévy Mutation
  end if
  / ***** /
end for

where r denotes a uniformly distributed random number. The use of a Gaussian distribution in classical evolutionary algorithms was first investigated by [50] to improve the efficiency of evolutionary

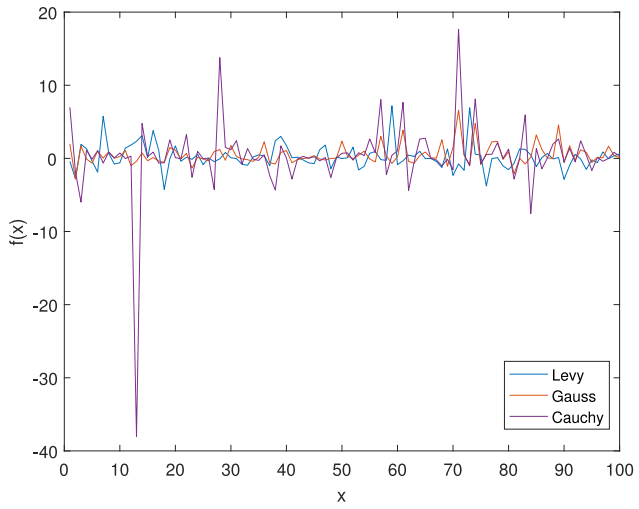


Fig. 1. The numbers generated by chaotic-based mutation operators.

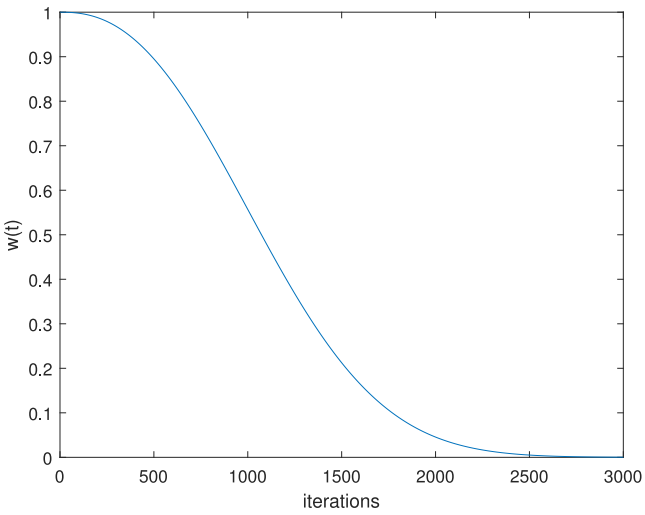


Fig. 2. w(t) curve using Eq. (2) for $t_{max} = 3000$ and $b = 2.4$.

processes. The standard Gaussian mutation process updates the population using the difference between two randomly selected solutions to update the search agent position. However, because the original SCO algorithm is based on a single candidate solution, normally distributed random numbers (randns) are used instead of these two solutions in the Gaussian mutation operation. The formula for updating the position of the candidate solution using a Gaussian mutation is as follows:

$$X(j) = S(j)(1 + \text{randn}(1)) \tag{11}$$

Finally, in the mutation operator, the position of the candidate solution is updated according to the Lévy function based on the Lévy flight mechanism as follows:

$$X(j) = S(j)(1 + L) \tag{12}$$

$$L = \frac{u}{|v|^{1/\beta}} \tag{13}$$

$$u \sim N(0, \sigma_u^2), \quad v \sim N(0, \sigma_v^2) \tag{14}$$

$$\sigma_u = \left(\frac{\Gamma(1+\beta)\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \cdot \beta \cdot 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}}, \quad \sigma_v = 1 \tag{15}$$

Here, u and v are the normally distributed variables. In this equation, β is a constant that is equal to 3/2. Fig. 1 shows the numbers generated by the chaotic functions used by the mutation operator.

While generating these numbers, the same initial values in the range [−10 10] were given for these functions. The difference in the numbers generated for the same initial values shows that using chaotic functions in the mutation operator increases the diversity. This study also discusses the update equations used in the exploration and exploitation phases of the SCO algorithm. The position update equation of the candidate solution used in the original SCO code during the exploration phase is, presented in Eq. (1), consists of two options and varies depending on the parameter w(t) (presented in Eq. (2)). Fig. 2 plots w(t) for 3000 maximum iterations and a constant b of 2.4. The w curve is a decreasing function from one to zero.

Referring back to Eq. (2), during the initial stages of the optimization process, the first of the two update equations $S(j) + w(t)|S(j)|$ causes the candidate solution to be $2S(j)$ and the second option $S(j) - w(t)|S(j)|$ causes the candidate solution to be 0. Notably $2S(j)$ may represent a different solution or exceed the boundary value. The candidate solution being at zero position presents two possibilities. If the solution point of the optimization problem being considered is at zero, i.e. at the center, the original SCO will converge very quickly. On the other hand, if the solution point is not at zero, the SCO is likely to get stuck at a fixed location during the exploration phase. To illustrate this shortcoming, consider the following two basic minimization problems.

$$\begin{aligned} &min f_1(x) = x_1^2 + x_2^2 \quad min \acute{f}_1(x) = (x_1 - 25)^2 + (x_2 - 25)^2 \\ &x_1, x_2 \in [-100, 100] \end{aligned} \tag{16}$$

Here, the solution of the f_1 function is at the (0,0) point, and $\acute{f}_1$ function is at (25,25) point. Fig. 3 shows the three-dimensional graphs of both functions and the convergence graphs obtained by the original SCO algorithm after 3000 iterations.

Looking at the convergence curve obtained for the function f_1, it can be seen that the SCO algorithm converges very quickly to the solution point of the problem. However, for function $\acute{f}_1$ whose solution point is not at the origin, it is clear that the SCO algorithm becomes stuck in the exploration phase and reaches the solution in the second phase. This is because of the update equations used by the SCO algorithm during the exploration phase. To overcome this shortcoming, the following equations were added to the existing equations:

$$X(j) = \begin{cases} S(j) + w(t) \cdot Sin(2\pi r_2) \cdot |S(j)|, \text{if } r_1 < 0.5 \\ S(j) + w(t) \cdot Cos(2\pi r_3) \cdot |S(j)|, \text{otherwise} \end{cases} \tag{17}$$

In this new equation, r_1, r_2, and r_3 are random numbers. In the updated equation, sinusoidal functions were utilized to overcome the handicap of the SCO in the exploration phase. By adding a new approach to the mechanism used to update the position of the candidate solution in the exploration phase, the AccOppCSCO algorithm becomes more effective for problems with solutions other than the origin. The pseudocode for the new update mechanism of the AccOppCSCO algorithm in the exploration stage is presented in Algorithm 3.

If $\acute{f}_1$ is reconsidered to demonstrate the effect of the new update mechanism, Fig. 4 shows the convergence curves of the SCO and AccOppCSCO algorithms for this function. In particular, during the exploration phase, when this mechanism is active, it is clear that the algorithm does not get stuck at a fixed value with the effect of the new update mechanism and achieves convergence in this phase.

Similarly, the innovative thinking proposed in the exploration phase was also considered for the candidate solution position update mechanism in the exploitation phase.

$$X(j) = \begin{cases} S(j) + w(t) r_5 [u_b(j) - l_b(j)] \cdot Sin(2\pi r_6), \text{if } r_4 < 0.5 \\ S(j) + w(t) r_5 [u_b(j) - l_b(j)] \cdot Cos(2\pi r_7), \text{otherwise} \end{cases} \tag{18}$$

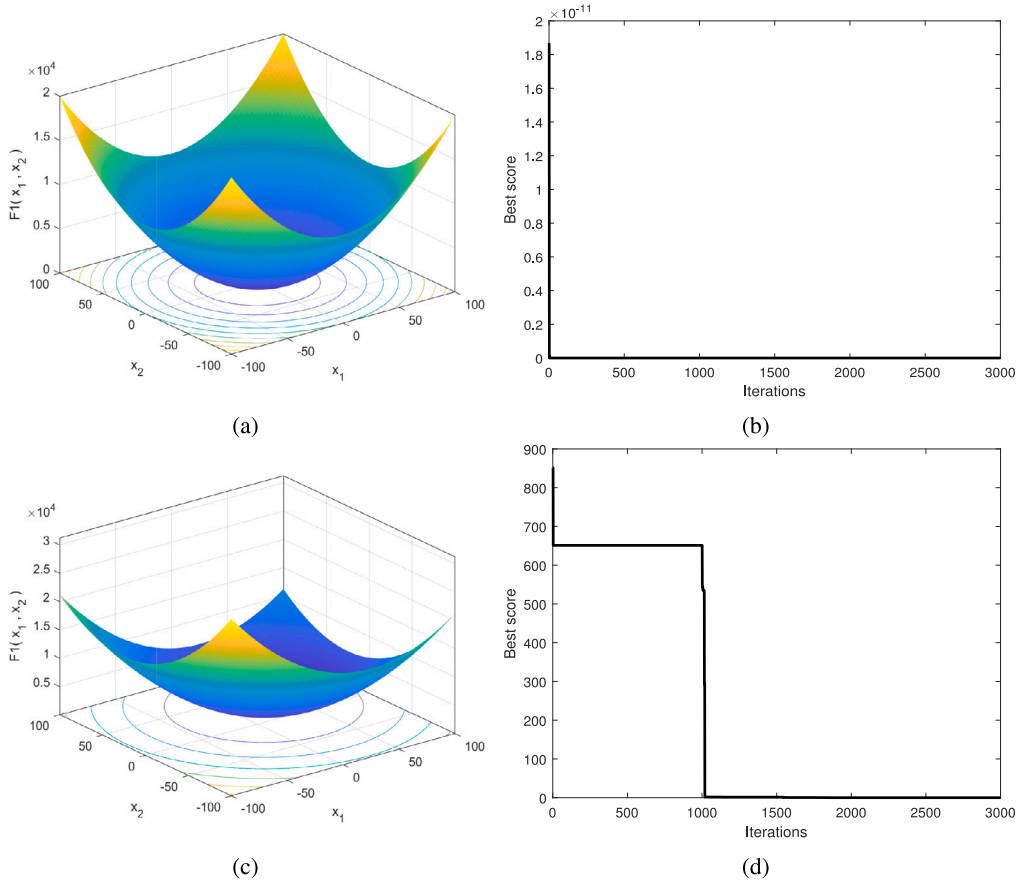


Fig. 3. 3D and convergence graphs of both functions.

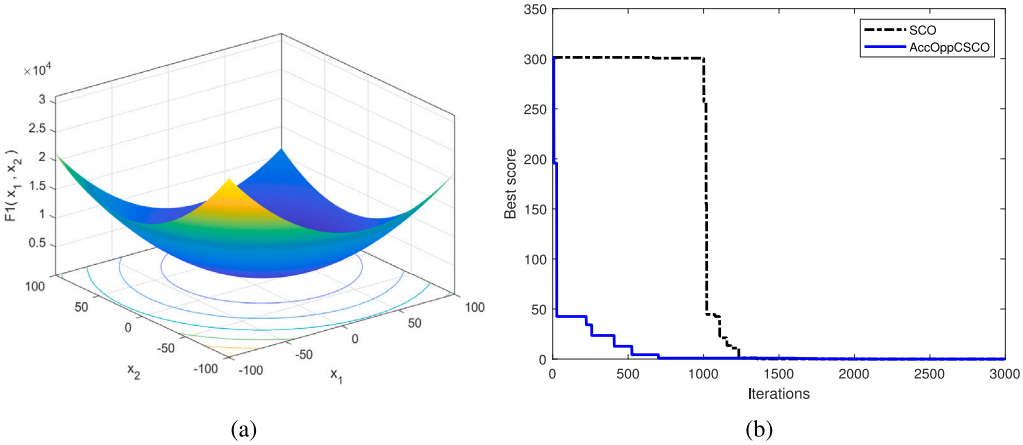


Fig. 4. Convergence analysis of SCO and AccOppCSCO algorithms for $\acute{f}_1$ function.

Here, random numbers are represented by r_4, r_5, r_6, and r_7. Finally, instead of the boundary value control mechanism in the original SCO algorithm, a more effective mechanism was taken from the literature and integrated into the algorithm. Existing checking mechanism (Eq. (5)) reverted to the previous candidate's solution if the boundary values exceeded after updating the candidate's position. This mechanism makes it difficult for the algorithm to converge in the search spaces of problems with solutions close to the boundary during the exploration phase. Therefore, the following mechanism was integrated into SCO by Nadimi-Shahraki et al. [51]:

$$X(j) = \begin{cases} \frac{1}{2} \cdot (S(j) + u_b(j)), \text{if } x(j) > u_b(j) \\ \frac{1}{2} \cdot (S(j) + l_b(j)), \text{if } x(j) < l_b(j) \end{cases} \tag{19}$$

The pseudocode of the AccOppCSCO algorithm is presented in Algorithm 4.

4. Results and discussion

This section presents analyses and results of the proposed AccOppCSCO algorithm. First, a comparative analysis of the AccOppCSCO

Algorithm 3 Pseudocode of new update mechanism used in AccOppCSCO

```
for j = 1 to dim do
    if t < α then
        if rand < 0.5 then
            if rand < 0.5 then
                X(j) ← S(j) + w(t) × |S(j)|   ▷ Update mechanism from original SCO
            else
                X(j) ← S(j) − w(t) × |S(j)|
            end if
        else
            if rand < 0.5 then
                X(j) ← S(j) + w(t) × Sin(2π rand) × |S(j)|   ▷ New update mechanism
            else
                X(j) ← S(j) + w(t) × Cos(2π rand) × |S(j)|
            end if
        end if
    end if
end for
```

Algorithm 4 Pseudocode of AccOppCSCO Algorithm

```
Define the values of l_b, u_b, dim, f_c, Max_iter, m ,α, b
FitImp ← 0, Count ← 0
Initialize the candidate solution (S_i) and calculate its cost value f_c(S_i)
Find opposite (Ŝ_i) of candidate solution by opposition-based learning and calculate its cost value f_c(Ŝ_i)
Select the fittest solution from S_i and Ŝ_i as the initial candidate solution (S) and BF ← f_c(S)
for t = 1 to Max_iter do
    Calculate w(t)
    if t < α then
        if rand < 0.5 then
            Update the position of the candidate (X) according to Eq. (1)   ▷ From the original SCO
        else
            Update the position of the candidate (X) according to Eq. (17)   ▷ Improved update equation
        end if
    else
        if FitImp = 0 then
            Count ← 1 + Count
        end if
        if Count = m then
            Count ← 0
            r_1 ← rand
            if r_1 > 0.7 then
                Update the candidate's position (X) by Cauchy Mutation
            else if r_1 > 0.3 and r_1 < 0.7 then
                Update the candidate's position (X) by Gauss Mutation
            else
                Update the candidate's position (X) by Lévy Mutation
            end if
        else
            if rand < 0.5 then
                Update the position of the candidate (X) according to Eq. (4)   ▷ From the original SCO
            else
                Update the position of the candidate (X) according to Eq. (18)   ▷ Improved update equation
            end if
        end if
    end if
    Boundary control mechanism for the updated candidate
    if rand < J_r then   ▷ J_r represents the jumping rate
        Calculate the opposite candidate (X̂) using accelerating opposition learning
        if f_c(X̂) < f_c(X) then
            X ← X̂
        end if
    end if
    if f_c(X) < BF then
        BF ← f_c(X), S ← X, FitImp ← 1
    else
        FitImp ← 0
    end if
    gbest ← S
end for
```

algorithm with the original SCO algorithm is performed, and the advantages and superiority of the new structures proposed in AccOppCSCO are shown. Then, the performance of the AccOppCSCO algorithm was evaluated for the benchmark problems of the IEEE Congress on Evolutionary Computation (CEC)2014, which are widely known in the literature. The CEC2014 test suite comprises 30 single-objective minimization benchmark problems [52]. These problems are divided into four categories: unimodal, simple multimodal, hybrid, and compositional functions. Each category in the CEC2014 test suite presents unique challenges and characteristics. Liang et al. [52] provided a comprehensive description of the CEC2014 test suite in their report.

To provide a more comprehensive evaluation, the proposed AccOppCSCO and the original SCO are evaluated based not only on the CEC2014 benchmarks but also on ten minimization problems taken from the CEC2020 benchmark problems. This benchmark represents one of the most recent sets of test problems in the rapidly evolving field of numerical optimization [53]. A comparison of these two algorithms on the CEC2020 benchmark package allows for a more updated and relevant assessment of their performance.

In this study, several statistical metrics were used to evaluate the performance of optimization algorithms. The best metric refers to the optimal objective function value obtained from multiple runs, while the worst metric represents the least optimal solution. The mean metric provides an average performance measure for all runs and the median metric helps mitigate the impact of extreme values. In addition, the standard deviation (SD) metric indicates the consistency of the algorithm, with lower values indicating higher stability. These metrics allow for a thorough comparison between AccOppCSCO and SCO across different benchmark problems.

To closely examine the performance of both algorithms, the computational complexity was presented for the F1 function from the CEC2020 benchmarks. Finally, the proposed AccOppCSCO algorithm was tested using real-world engineering design problems. Ten different optimization problems, taken from the existing literature, were used for the testing. The results were compared to those produced by the original SCO and other popular heuristic algorithms.

4.1. Analysis of the AccOppCSCO algorithm

In this section, I compare the search capability of the proposed AccOppCSCO with that of the original SCO. The comparison was based on four aspects: (1) convergence analysis, (2) search history analysis, (3) trajectory analysis, and (4) computational complexity analysis. These analyses aimed to contrast the search behaviors exhibited by

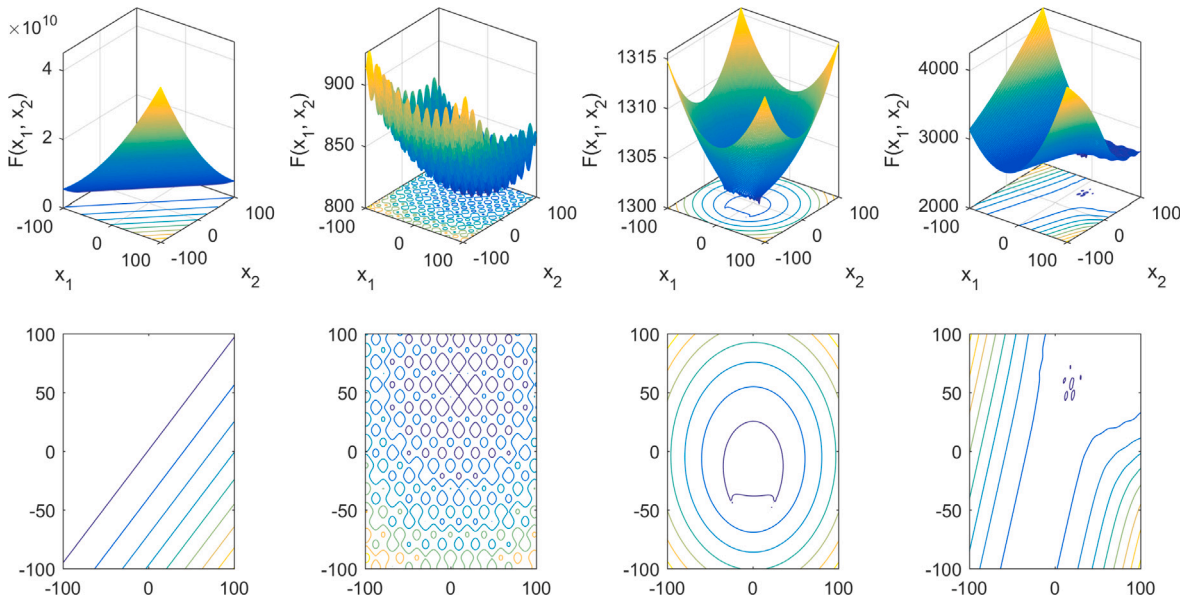


Fig. 5. Benchmark functions selected from CEC2014 test suite for analysis of the AccOppCSCO algorithm.

the proposed AccOppCSCO and the original SCO while attempting to locate the global optimum within the search space. Convergence analysis provides insight into the evolution of the fitness value of the best candidate solution obtained during the optimization. Search history analysis examines the path followed by the candidate solutions through the search space. Trajectory analysis focuses on the position of the elite candidate solution and determines the path that it follows to reach the global solution point in the search space. Computational complexity analysis determines the running time of an algorithm, which is expressed using the notation O(.). For the AccOppCSCO analysis, four benchmark functions from the CEC2014 test suite were chosen. These functions included the Rotated Bent Cigar function (F_2), Shifted Rastrigin's function (F_8), Shifted and Rotated HappyCat function (F_{13}), and Composition Function 3 (N = 3) (F_{25}). Three-dimensional and contour plots of the benchmark functions selected for the evaluation of AccOppCSCO are shown in Fig. 5.

4.1.1. Convergence analysis

Convergence analysis is a vital technique for evaluating the effectiveness of an optimization algorithm. It assesses the closeness of the optimized objective function to the solution value. This technique allows for monitoring the progression of the algorithm towards the problem solution by recording the fitness values of the best solution obtained during optimization. The convergence analysis results provide information on how far and how close the algorithm is to the optimal value of the objective function. This analysis is critical for evaluating and enhancing the performance of the algorithms.

Convergence analyzes of the proposed AccOppCSCO and original SCO algorithms were performed for four benchmarks obtained from the CEC2014 test problems. In these benchmark problems, the problem dimension is two, and both algorithms are initialized with the same initial candidate solution. In Fig. 6, the convergence curves obtained by both algorithms for the four benchmark functions on a logarithmic scale are shown. The convergence curves of the proposed AccOppCSCO show that it has a better convergence ability than the original SCO algorithm for all selected benchmarks.

For the F_1, F_{13}, and F_{25} benchmarks, the original SCO algorithm was stuck at a fixed solution point during the exploration phase and exhibited constant flat convergence. For F_8 the benchmark function, the original SCO appears to be stuck at the local minima in the second phase. The successful convergence curves of the proposed AccOppCSCO algorithm were due to the integration of the innovative techniques presented in this work. These include the accelerated opposition-based mechanism, chaotic-based mutation operator, and developed candidate solution update equations, which were incorporated into the original algorithm.

4.1.2. Search history analysis

Search history analysis shows the solutions discovered by the optimization algorithm at each step within the search space. This analysis aids in understanding how the algorithm explores the problem, where it focuses, and how it advances. Fig. 7 illustrates the solution points obtained by the SCO and AccOppCSCO algorithms during the optimization process.

Search history analyzes were performed for both algorithms, with consideration given to the same initial positions. The updated positions of the candidate solutions are displayed in different colors on the contour surfaces of the selected benchmarks. The proposed AccOppCSCO algorithm demonstrates a high search capability owing to the movement of candidate solutions in the most promising regions of the search space.

4.1.3. Trajectory analysis

This analysis focused on the algorithm's best solution, known as the elite solution, and examined the elite's behavior in the search space. In this analysis, in each benchmark problem, the initial positions of the candidate solutions of both algorithms were the same. Fig. 8 shows the evolution of the elite candidate solution during the optimization in the search space. Here, blue indicates SCO, and red indicates AccOppCSCO. For benchmark F_2 with a flat search surface, the analysis showed that the original SCO elite solution followed a longer trajectory than AccOppCSCO.

However, of the two different solution points resulting from the optimization, the AccOppCSCO solution is closer to the solution of the problem. Although AccOppCSCO and SCO started the solution from the same position, the AccOppCSCO algorithm achieved a better solution with a shorter path. The curves in the convergence analysis show that the proposed AccOppCSCO converges rapidly to the solution point at the beginning of the optimization. The trajectory analysis of benchmark F_8, which has many local minima, clearly shows that AccOppCSCO reaches the final solution point with a shorter path than that of SCO. The algorithms arrive at two different solution points close to each other when solving the optimization problem. However, the cost value

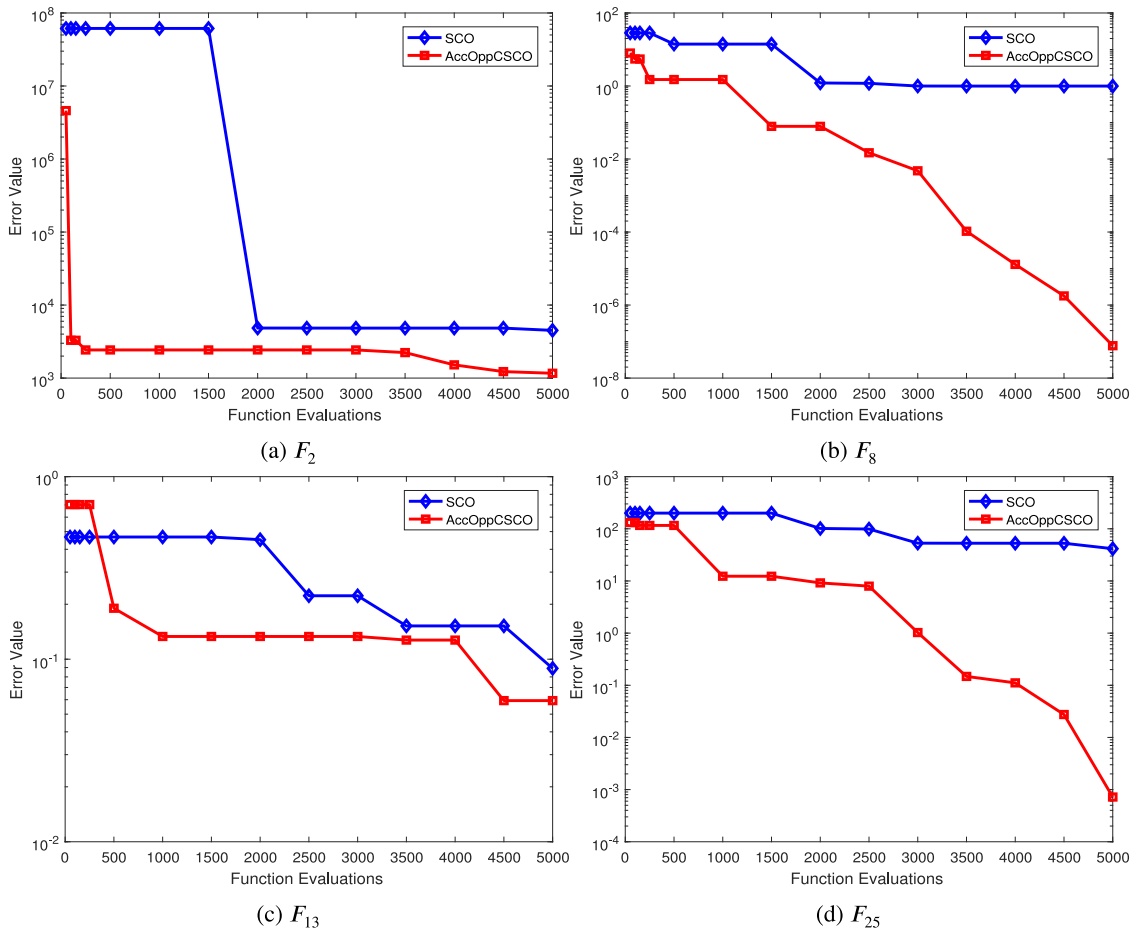


Fig. 6. Convergence analysis of SCO and AccOppCSCO algorithms.

of the AccOppCSCO solution point was better than that of SCO. This indicates that the SCO algorithm is trapped at a local minimum point for this benchmark function and cannot escape.

The analysis results obtained for the F_{13} problem show that both algorithms reach almost the same final solution from the same starting point. The analysis results for the final benchmark (F_{25}) indicate that AccOppCSCO and SCO do not have the same starting point. This is because AccOppCSCO uses an opposition-learning-based initialization, which provides a better initial solution point for optimization. Although the final solution points obtained appear to be close to each other, the fitness value of the final elite solution point of AccOppCSCO was more successful than that of SCO. As shown in the figure, AccOppCSCO reached a greater number of solution points during the exploitation phase.

In Fig. 9, the changes in the positions of the elite candidate solution for both axes during optimization are shown. The variables x_1 and x_2 represent the positions of the candidate solutions. The positions of the final elite solutions of AccOppCSCO and the original SCO algorithms differed for the other three benchmark functions, except for benchmark function F_{13}. In the benchmark function (F_2), the positions of the algorithm solutions differ significantly. The elite candidate identified by the proposed AccOppCSCO is generally more stable and reaches the final solution point faster.

4.1.4. Computational complexity analysis

The computational complexity estimates the difficulty of solving a problem computationally. However, it does not provide information on the computation time required to solve a specific problem or instance, as this depends on other factors, such as hardware and software. Therefore, the complexity is an estimate of the number of computational operations required. The computational complexity of the AccOppCSCO algorithm is as follows.

1. Because it is not population-based and searches with a single candidate, the computational complexity of initially randomizing the locations of candidate solutions is given by $O(d)$. Here, d denotes the problem size.
2. The fitness values are calculated for the initial position of the candidate solution, and the complexity of this process is also $O(d)$.
3. The complexity of finding the opposite candidate position and its fitness value is $O(2d)$.
4. In the main loop of the algorithm, the computational complexity is $O(\frac{T}{3}(2d+4)+\frac{2T}{3m}((d+2)(m-1)+(3d+2))+T(4d+7))$ for the procedures of updating the candidate solution location, chaotic-based mutation, boundary checking, and accelerated opposition learning-based candidate location generation. Here, T denotes the maximum iteration and m is the number of unsuccessful attempts.

Consequently, the computational complexity of the AccOppCSCO algorithm is $O(4d)+O(\frac{T}{3}(2d+4)+\frac{2T}{3m}((d+2)(m-1)+(3d+2))+T(4d+7))$. Therefore, the computational complexity of AccOppCSCO is O(T x d), which is almost identical to that of the original SCO algorithm.

4.2. Performance results for CEC2014 test suite

In this subsection, the results for 30 different minimization benchmark problems, known as CEC2014, are presented. The statistical results of the original SCO and AccOppCSCO for different problem

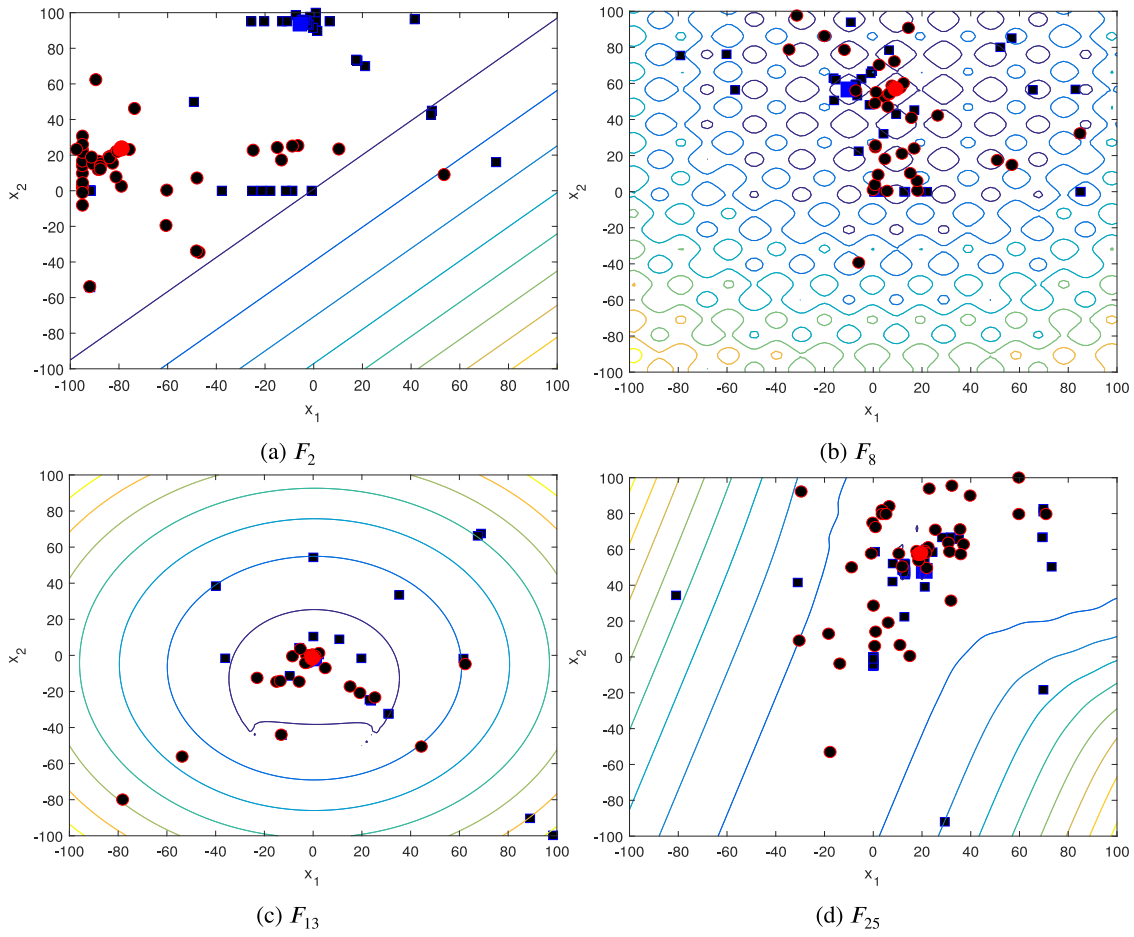


Fig. 7. Search history analysis of SCO and AccOppCSCO algorithms. ● denotes the candidate's position for AccOppCSCO, ■ denotes the candidate's position for SCO, ● denotes the best solution for AccOppCSCO, and ■ denotes the best solution for SCO.

sizes (10, 30, 50, and 100) with 51 runs are presented first. Table 1 summarizes the results of the CEC2014 benchmarks for 10D.

This table contains the best, worst, median, mean, and standard deviation of the repeated results. The AccOppCSCO algorithm achieved a 70% higher success rate for the best metric values, 60% higher success rate for the worst metric values, 70% higher for the median, 66.67% higher success rate for the mean and 53.33% higher success rate for the standard deviation in 30 benchmark problems over 51 runs compared to the original SCO.

Table 2 presents the statistical results of the 30D CEC2014 benchmarks for both the algorithms. AccOppCSCO outperformed SCO in 24 out of 30 benchmark problems, achieving superior results in the worst-case, mean, and median evaluations.

In Table 3, the statistical results of the AccOppCSCO and SCO algorithms are presented for the 50D CEC2014 benchmark problems. Upon analysis of the results, it was found that AccOppCSCO outperformed SCO in 26 out of 30 problems according to the best metric, 21 out of 30 problems according to the worst and median metrics, and 17 out of 30 problems according to the mean and standard deviation metrics. These results demonstrate that AccOppCSCO generally performs more effectively than SCO on 50-dimensional CEC2014 problems.

Table 4 summarizes the results of the 100D CEC2014 benchmarks obtained by both algorithms over 51 runs. The success rate of AccOppCSCO was 70% based on the best performance criterion. In the worst performance measure, the difference was even more pronounced, with AccOppCSCO achieving a success rate of 73.33%. The median performance measure indicated that AccOppCSCO achieved a success rate of 66.67%. Based on the mean performance measure, AccOppCSCO outperformed SCO with a success rate of 56.67%. In terms of the standard deviation, AccOppCSCO achieved a success rate of 73.33%. These results indicate that AccOppCSCO is generally a more effective solution than SCO for the 100-dimensional CEC2014 problems.

The results of the SCO and AccOppCSCO algorithms for 30 different CEC2014 benchmark test suites with different problem sizes are summarized in Table 5. Here, the number of more successful benchmark functions of the two algorithms is presented for each dimension and for each statistical metric value. The bottom row of the table shows the sum of the number of benchmark functions for which the algorithms were better for each metric. This summary table clearly shows that the proposed AccOppCSCO algorithm outperforms the original SCO algorithm on different dimensional benchmarks.

Finally, the AccOppCSCO algorithm was compared statistically with various state-of-the-art heuristic algorithms for the 30D CEC2014 benchmarks. The heuristic algorithms used in the comparison works are the Ageist Spider Monkey Optimizer (ASMO) [54], Chaotic Cuckoo Search optimizer (CCS) [55], Coyote Optimization Algorithm (COA) [56], Dragonfly Algorithm (DA) [57], Elephant Herding Optimizer (EHO) [58], Human Felicity Algorithm (HFA) [59], Moth–Flame Optimizer (MFO) [60], Particle Swarm Optimizer (PSO) [31], Red Fox Optimizer (RFO) [61],Sine Cosine Algorithm (SCA) [62], Variable Neighborhood Bat Algorithm (VNBA) [63], and Whale Optimization Algorithm (WOA) [39]. The parameters of the heuristics are listed in Table 6.

The comparison results for the 30-dimensional CEC2014 benchmarks are presented in the two tables. Table 7 shows the results of the first group of heuristics (ASMO, CCS, COA, DA, EHO, EHO, and HFA) and the proposed AccOppCSCO algorithm in terms of the mean metric. The ranking results between AccOppCSCO and the first group

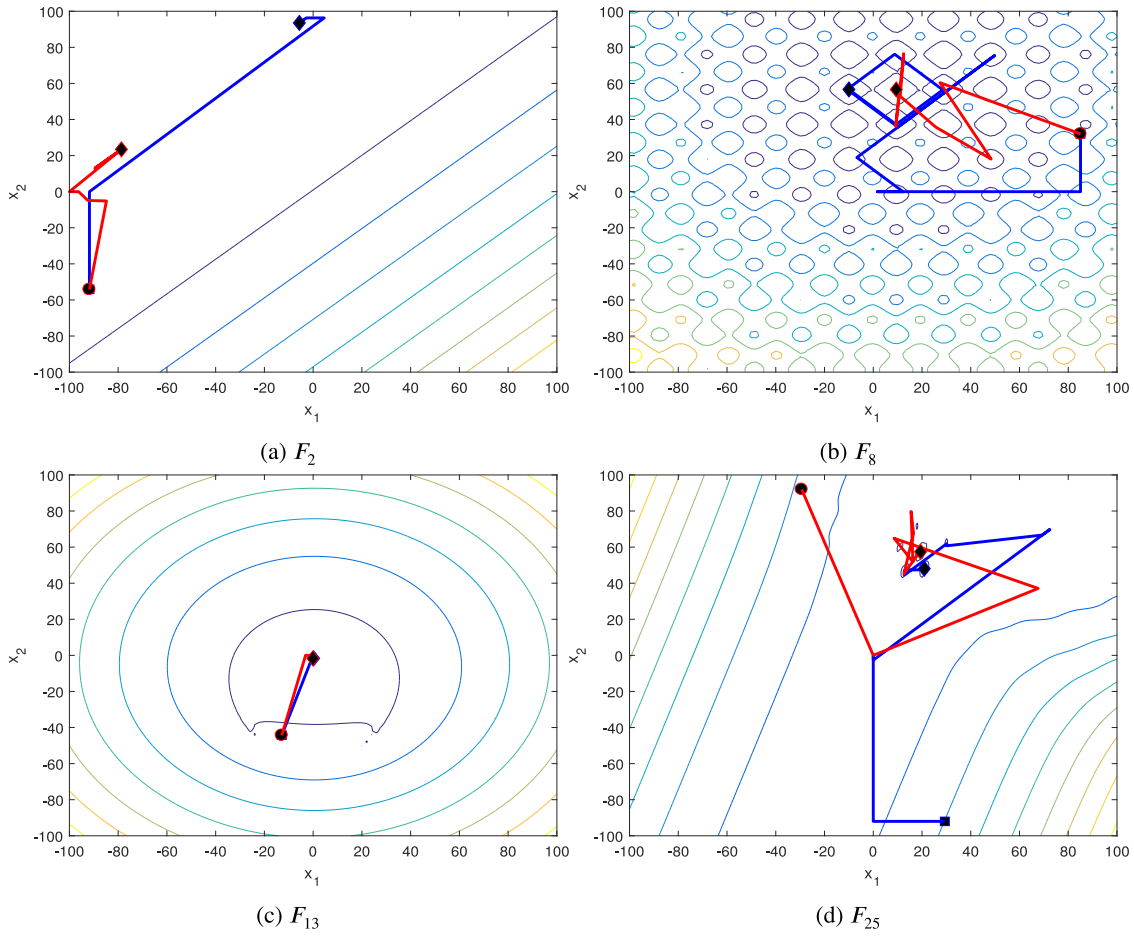


Fig. 8. Elite candidate's trajectory analysis for SCO (blue line) and AccOppCSCO (red line) algorithms. ● denotes the elite's initial position for AccOppCSCO, ■ denotes the elite's initial position for SCO. ◆ denotes the elite's final position for AccOppCSCO, and ◆ denotes the elite's final position for SCO.

are provided in the table at the end of each benchmark. The second row from the bottom presents the average ranking results obtained after 30 benchmarks, titled 'Average rank'. The overall ranking is given in the bottom row based on these averages. According to the overall ranking results, AccOppCSCO ranked first among the 30-dimensional CEC2014 benchmarks. Table 8 presents the mean metric results of the AccOppCSCO algorithm using the second group of heuristics (MFO, PSO, RFO, SCA, VNBA, and WOA) for the 30-dimensional CEC2014 problems. In the analysis of ranking results, the AccOppCSCO algorithm achieved the highest average ranking score of 2.07.

To further demonstrate the superiority of AccOppCSCO, I emphasized its performance on Function F24 from the CEC2014 benchmark suite. This function poses significant challenges owing to its highly multimodal nature, non-separability, and varying characteristics around numerous local optima. These properties make it particularly difficult for traditional optimization algorithms to effectively navigate large and complex search space. However, the benchmark results revealed that AccOppCSCO achieved the highest performance ranking, significantly surpassing the alternative heuristic methods. This shows that AccOppCSCO is particularly capable of handling optimization problems characterized by complex surfaces and variable difficulty levels. These findings confirm the robustness of the proposed enhancements and their valuable contribution to the field of numerical optimization.

4.3. Performance results for CEC2020 test suite

In this subsection, the proposed AccOppCSCO and original SCO algorithms are evaluated using the CEC2020 benchmark suite, one of the most recently introduced test suites in the field of optimization. This benchmark suite consists of 10 extremely difficult minimization problems. One of these problems is a unimodal function; three are multimodal functions, three are hybrid functions and three are composite functions [53]. The ten benchmarks here were solved 30 times by both algorithms for different problem sizes (D = 5, 10, 15, and 20). The search space is the same for all benchmarks and is between −100 and 100. The stopping criterion of the algorithms was chosen to reach the maximum number of function evaluations (5e4 for all dimensions). Again, problems F6 and F7 were excluded from the results obtained for the 5-dimensional benchmarks.

In Table 9, the statistical results of AccOppCSCO and the original SCO algorithms are summarized for the CEC2020 benchmarks with 5D, 10D, 15D, and 20D. The table lists the best, worst, median, mean, and standard deviation values of each algorithm for each problem. When the 5D results were examined, the AccOppCSCO algorithm performed better than the SCO algorithm.

AccOppCSCO achieved more successful results than SCO: with 62.50% for the best results, 87.50% for the worst results, 50.00% for the median, 62.50% for the mean, and 75.00% for the standard deviation. In the 10D benchmark results, AccOppCSCO outperformed SCO in 7 out of 10 functions for the best metric, 8 out of 10 functions for the worst, median, standard deviation metric, and 9 out of 10 functions for the mean metric.

The results obtained for the 15D benchmarks show that, among the 10 benchmark problems, AccOppCSCO outperforms SCO by 60.00% for the best metric, 80.00% for the worst, 60.00% for the median, 100.00% for the mean and 70.00% for the standard deviation. In the 20D benchmarks, AccOppCSCO consistently outperformed SCO in the best, mean, and median metrics, demonstrating higher stability and

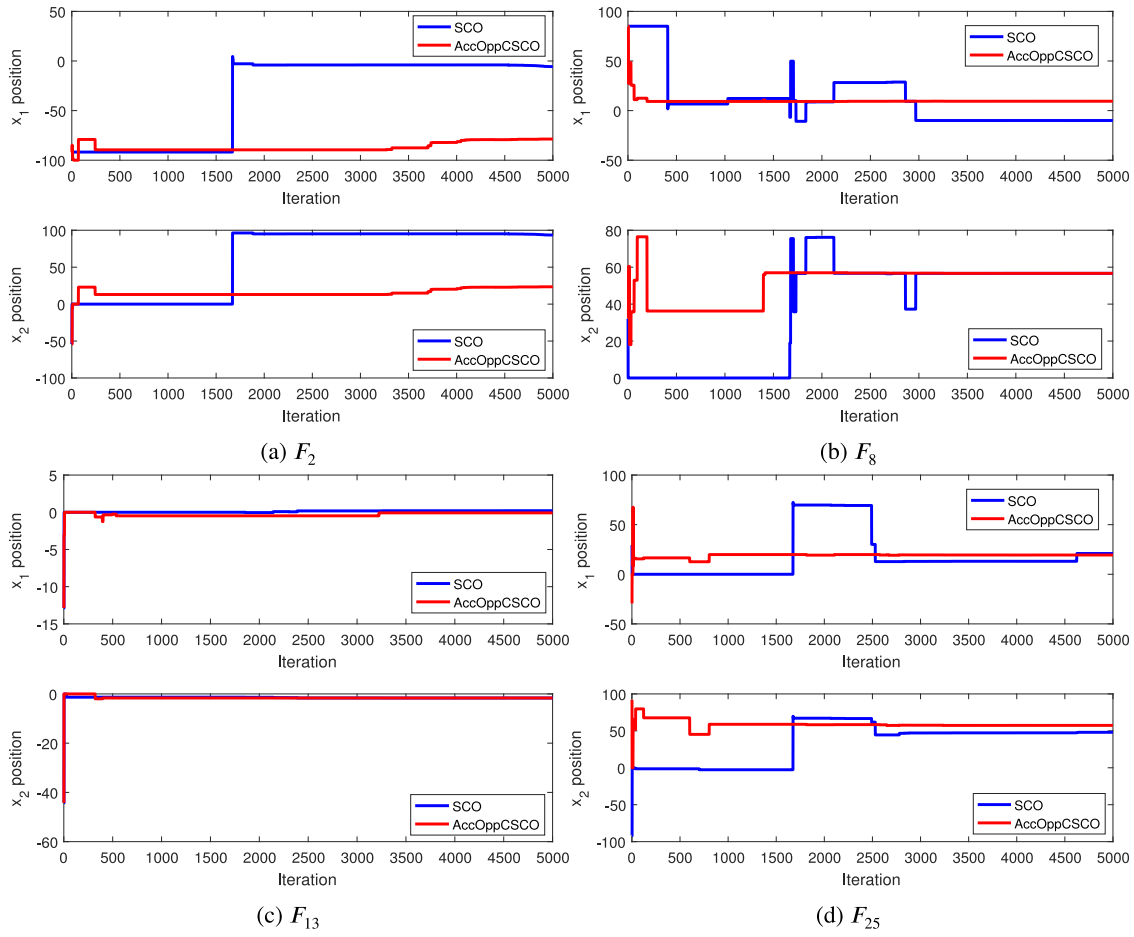


Fig. 9. Elite candidate's first position history for SCO and AccOppCSCO algorithms.

Table 1
Results of SCO and AccOppCSCO algorithms for 10D CEC2014 Benchmark.

No	SCO					AccOppCSCO				
	Best	Worst	Median	Mean	Std	Best	Worst	Median	Mean	Std
1	1.44E + 5	1.19E + 7	1.80E + 6	2.41E + 6	2.30E + 6	2.70E + 5	7.49E + 6	1.73E + 6	2.31E + 6	1.68E + 6
2	627.72	12 917	2964.70	3990.70	2883.50	452.83	11 873	2172.10	3138.40	2582.90
3	8032.90	6.59E + 4	1.79E + 4	2.07E + 4	9333.10	4127.10	1.77E + 4	1.24E + 4	1.16E + 4	3410.10
4	0.12	69.39	34.85	29.21	18.56	0.11	66.01	34.83	27.73	14.89
5	20	20.03	20.01	20.01	0.01	20	20.48	20.01	20.02	0.07
6	0.97	10.45	6.12	5.87	1.89	1.60	9.36	5.81	5.47	1.90
7	0.06	0.46	0.16	0.19	0.10	0.02	0.73	0.16	0.18	0.12
8	1.99	14.93	5.97	6.66	2.68	6.97	60.69	17.91	19.61	9.49
9	8.96	68.65	32.84	35.10	13.76	15.92	54.72	39.80	38.67	11.07
10	161.23	1269.80	796.14	804.29	274.62	255.72	1388.20	918.42	900.30	270.48
11	370.85	1995	1069.20	1037.70	339.76	133.57	1569.20	972.14	922.91	312.02
12	0.06	1.09	0.41	0.47	0.25	0.04	1.02	0.37	0.41	0.23
13	0.17	1.10	0.45	0.45	0.19	0.18	0.72	0.39	0.43	0.15
14	0.13	1.10	0.29	0.35	0.21	0.07	1.29	0.36	0.48	0.31
15	0.64	4.99	2.53	2.50	0.96	0.54	5.37	2.23	2.39	1.05
16	2.57	3.99	3.46	3.41	0.31	2.43	3.81	3.41	3.34	0.29
17	1.31E + 5	2.17E + 6	4.49E + 5	5.07E + 5	3.07E + 5	1.93E + 3	3.11E + 5	5.20E + 4	7.90E + 4	8.14E + 4
18	2380.60	1.83E + 4	6727.60	8191.30	3551.30	153.27	2.87E + 4	1.29E + 4	1.12E + 4	4533.50
19	2.74	10.03	5.33	5.35	1.55	1.85	7.32	3.55	3.87	1.33
20	85.87	8.21E + 4	7438.70	1.11E + 4	1.29E + 4	445.98	2.62E + 4	8153.40	8134.40	5361.90
21	922.32	5.57E + 5	1.07E + 4	2.81E + 4	8.03E + 4	225.38	2.79E + 4	5302.40	6551.30	5203.80
22	24.02	368.97	171.30	159.31	83.12	30.91	241.94	169.26	156.99	39.05
23	200	200	200	200	0	200	331.40	200	205.13	25.62
24	126.61	200	200	191.61	21.71	111.05	200	200	173.18	32.53
25	148.51	200	200	196.75	11.43	124.60	200	200	192.61	18.80
26	100.08	100.91	100.40	100.45	0.20	100.15	100.83	100.41	100.43	0.18
27	200	200	200	200	0	4.26	434.51	200	186.12	96.72
28	200	200	200	200	0	200	582.33	200	249.93	112.44
29	200	200	200	200	0	200	1580.20	200	385.06	308.02
30	200	200	200	200	0	200	3775.20	1881	1856.40	1173.50

Table 2
Results of SCO and AccOppCSCO algorithms for 30D CEC2014 Benchmark.

No	SCO					AccOppCSCO				
	Best	Worst	Median	Mean	Std	Best	Worst	Median	Mean	Std
1	1.45E + 7	1.13E + 8	3.31E + 7	3.97E + 7	2.03E + 7	1.33E + 7	6.72E + 7	2.96E + 7	3.09E + 7	1.16E + 7
2	1.34E + 5	9.86E + 5	3.53E + 5	3.70E + 5	1.56E + 5	1.12E + 5	7.08E + 5	2.80E + 5	3.03E + 5	1.15E + 5
3	7.06E + 4	3.55E + 5	9.26E + 4	1.13E + 5	5.45E + 4	5.57E + 4	1.26E + 5	8.01E + 4	7.97E + 4	1.22E + 4
4	72.74	295.73	150.60	153.16	45.62	5.75	280.93	155.05	157.90	45.06
5	20.07	20.46	20.24	20.24	0.08	20.10	20.44	20.21	20.22	0.08
6	20.90	37.65	30.15	30.09	3.75	21.49	36.57	28.90	28.79	3.28
7	0.33	0.76	0.55	0.56	0.10	0.37	0.82	0.58	0.57	0.10
8	21.01	119.47	73.68	75.95	20.75	66.76	242.83	123.42	132.68	35.70
9	114.55	249.08	183.21	182.57	29.88	113.53	275.93	178.17	175	28.26
10	2361.10	4795.90	3593.90	3589.20	588.62	2084.90	5023.40	3643.30	3577.70	645.36
11	2691	5684.20	4201.50	4127.30	657.47	2475.90	5210.60	4130.50	4057	725.56
12	0.46	2.57	1.27	1.29	0.43	0.26	2.88	0.99	1.24	0.67
13	0.37	0.97	0.66	0.66	0.14	0.36	0.96	0.63	0.62	0.13
14	0.20	1.09	0.34	0.41	0.21	0.18	1.08	0.34	0.40	0.21
15	12.07	57.94	24.64	25.35	8.29	9.89	60.87	25.17	25.57	8.65
16	11.49	13.76	12.66	12.60	0.58	11.01	13.35	12.57	12.52	0.52
17	2.87E + 5	6.17E + 6	2.06E + 6	2.35E + 6	1.62E + 6	1.68E + 5	3.80E + 6	1.20E + 6	1.42E + 6	9.13E + 5
18	769.98	2.64E + 4	3909.10	5528.70	5240.70	440.97	1.36E + 4	2832.10	3763	3019.90
19	13.11	80.32	19.13	24.63	18.01	14.63	146.03	19.08	26.06	26.24
20	2.24E + 4	4.43E + 5	1.25E + 5	1.39E + 5	7.80E + 4	7.26E + 3	1.03E + 5	5.81E + 4	5.93E + 4	2.41E + 4
21	1.04E + 5	1.89E + 6	5.54E + 5	7.11E + 5	4.91E + 5	7.97E + 4	3.05E + 6	7.09E + 5	9.08E + 5	6.08E + 5
22	402.34	1302.40	885.39	857.70	184.23	185.08	1092.80	686.64	679.62	232.23
23	200	200	200	200	0	200	200	200	200	0
24	200	200	200	200	0	200	200	200	200	0
25	200	200	200	200	0	200	200	200	200	0
26	100.35	200	100.69	114.29	34.53	100.41	200	100.85	131.88	46.52
27	200	1268.20	200	297.06	284.94	200	1251.70	418.39	443.62	220.37
28	200	2382	200	242.78	305.54	200	3170.10	200	419.88	662.67
29	200	200	200	200	0	200	2.69E + 7	200	9.29E + 5	4.68E + 6
30	200	5.09E + 4	200	1194.80	7104.30	200	2.63E + 5	2.73E + 4	5.21E + 4	6.96E + 4

Table 3
Results of SCO and AccOppCSCO algorithms for 50D CEC2014 Benchmark.

No	SCO					AccOppCSCO				
	Best	Worst	Median	Mean	Std	Best	Worst	Median	Mean	Std
1	2.98E + 7	1.76E + 8	8.83E + 7	8.98E + 7	3.27E + 7	1.92E + 7	1.42E + 8	7.21E + 7	7.49E + 7	2.85E + 7
2	1.60E + 6	2.57E + 8	5.33E + 6	1.27E + 7	3.62E + 7	3.38E + 6	6.26E + 7	7.76E + 6	1.28E + 7	1.51E + 7
3	1.20E + 5	4.42E + 5	2.15E + 5	2.32E + 5	8.05E + 4	9.08E + 4	1.94E + 5	1.34E + 5	1.34E + 5	1.96E + 4
4	141.08	590.59	249.92	266.08	89.43	123.66	517.12	263.99	272.55	84.93
5	20.33	20.72	20.55	20.53	0.08	20.28	20.65	20.47	20.48	0.10
6	43.26	68.96	58.49	58.69	5.65	45.25	67.29	57.31	57.10	5.45
7	0.98	1.29	1.10	1.11	0.06	1.02	1.56	1.12	1.15	0.11
8	135.03	315.65	221.94	221.13	42.41	188.98	459.08	277.04	286.36	54.11
9	264.01	547.42	384.28	388.38	58.78	246.67	465.67	336.19	345.88	51.84
10	4240.60	8722.20	6965.40	6881.10	1019.60	5543.50	9736.50	6958.20	6990.70	887.42
11	5978.60	9688.90	7934.70	7930.90	850.32	5151.80	9581.60	7285.80	7379.30	1018.20
12	0.95	3.50	1.85	1.95	0.64	0.89	3.72	1.78	1.86	0.61
13	0.44	1	0.68	0.70	0.13	0.45	0.94	0.66	0.68	0.12
14	0.23	1.11	0.37	0.48	0.24	0.24	1.03	0.38	0.48	0.23
15	51.39	269.19	97.51	108.98	48.87	49.50	202.97	95.69	105.39	37.10
16	20.41	23.20	21.88	21.88	0.62	19.46	23.19	21.99	21.92	0.65
17	1.10E + 6	1.75E + 7	5.18E + 6	6.01E + 6	3.81E + 6	2.49E + 5	8.40E + 6	3.73E + 6	3.94E + 6	1.92E + 6
18	3180	1.73E + 4	6832.40	7560.50	3249.60	2264.10	5.27E + 4	6911.80	8133.80	7316
19	31.41	105.68	63.52	66.96	24.84	29.43	102.99	54.22	57.77	22.36
20	3.46E + 4	6.30E + 5	1.11E + 5	1.33E + 5	9.58E + 4	3.36E + 4	2.19E + 5	8.07E + 4	8.04E + 4	2.69E + 4
21	7.76E + 5	1.28E + 7	3.49E + 6	4.13E + 6	2.67E + 6	7.46E + 5	1.48E + 7	2.98E + 6	3.70E + 6	2.50E + 6
22	288.10	2441.80	1556	1562.50	441.87	590.14	2215	1591.40	1576.20	345.73
23	200	200	200	200	0	200	200	200	200	0
24	200	200	200	200	0	200	200	200	200	0
25	200	200	200	200	0	200	200	200	200	0
26	100.44	200	200	176.61	42.58	100.81	200	200	190.28	29.77
27	200	2215.60	200	844.61	837.34	200	2066.10	1867.40	1773.90	324.51
28	200	3370	200	262.16	443.89	200	6731.30	200	584.89	1361.60
29	200	8.78E + 7	200	1.73E + 6	1.23E + 7	200	6.03E + 7	200	2.22E + 6	1.11E + 7
30	200	3.27E + 5	200	2.72E + 4	7.90E + 4	200	5.93E + 5	1.25E + 5	1.47E + 5	1.34E + 5

accuracy.

Finally, a computational complexity analysis of both algorithms was performed in the CEC2020 benchmarks. The function F_1 was chosen for the calculations. The computational complexity of both algorithms was accurately calculated using a PC equipped with an Intel(R) Core(TM) i7-8550U CPU @ 1.80 GHz and 20 GB RAM. First, the computing time (T_0) of the code given below is calculated.

```
for i=1:1000000 do
    x = 0.55 + (double) i;
    x=x+x; x=x/2;
```

Table 4
Results of SCO and AccOppCSCO algorithms for 100D CEC2014 Benchmark.

No	SCO					AccOppCSCO				
	Best	Worst	Median	Mean	Std	Best	Worst	Median	Mean	Std
1	1.86E + 8	7.58E + 8	3.80E + 8	3.86E + 8	1.11E + 8	1.64E + 8	5.36E + 8	3.25E + 8	3.24E + 8	7.31E + 7
2	1.53E + 9	6.15E + 9	3.05E + 9	3.21E + 9	1.24E + 9	1.07E + 9	5.79E + 9	2.87E + 9	3.06E + 9	9.86E + 8
3	2.54E + 5	8.91E + 5	4 E+5	4.42E + 5	1.59E + 5	2.46E + 5	4.65E + 5	2.93E + 5	3 E+5	3.51E + 4
4	593.81	1776.30	913.68	934.16	211.76	679.75	1599.80	979.18	999.69	190.33
5	20.73	21	20.86	20.87	0.07	20.71	20.98	20.88	20.87	0.07
6	124.82	163.85	136.93	138.24	8.36	114.74	146.55	135.30	134.49	7.03
7	12.94	38.77	24.22	24.65	6.73	12.58	55.99	31.11	31.01	9.85
8	548.24	850.50	653.84	660.21	68.99	539.85	876.73	639.36	663.05	81.34
9	722.34	940.94	860.38	857.96	48.52	662.18	906.63	816.93	819.62	50.49
10	1.37E + 4	1.83E + 4	1.64E + 4	1.62E + 4	1190.60	1.36E + 4	1.92E + 4	1.64E + 4	1.64E + 4	1314.10
11	1.28E + 4	2.15E + 4	1.61E + 4	1.62E + 4	1469.30	1.30E + 4	2.03E + 4	1.60E + 4	1.63E + 4	1654.80
12	1.96	4.50	2.79	2.90	0.59	1.24	4.37	2.61	2.66	0.56
13	0.52	1.04	0.69	0.71	0.10	0.50	0.89	0.68	0.69	0.09
14	0.32	5.75	0.40	0.52	0.75	0.34	2.58	0.41	0.53	0.35
15	581.28	1.80E + 4	1593.20	2518.60	2587	336.34	3084.80	1087.40	1337.40	708.92
16	43.86	47.28	45.52	45.57	0.70	43.46	46.80	45.07	45.10	0.74
17	1.17E + 7	1.04E + 8	3.57E + 7	3.80E + 7	1.97E + 7	1.12E + 7	6.80E + 7	3.23E + 7	3.37E + 7	1.40E + 7
18	6.06E + 4	8.84E + 7	1 E+5	4.39E + 6	1.49E + 7	3.52E + 4	1.06E + 8	7.77E + 4	4.30E + 6	1.77E + 7
19	154.12	419.07	231.13	251.43	64.90	142.58	437.71	252.69	252.63	61.64
20	1.51E + 5	5.93E + 5	2.66E + 5	2.81E + 5	9.47E + 4	1.46E + 5	4.88E + 5	2.40E + 5	2.46E + 5	6.79E + 4
21	6.84E + 6	2.48E + 7	1.31E + 7	1.38E + 7	4.17E + 6	5.43E + 6	3.25E + 7	1.33E + 7	1.46E + 7	6.21E + 6
22	2291.30	4988.60	3331.30	3440.40	536.02	2173.60	4931.40	3395.70	3378.50	563.92
23	200	200	200	200	0	200	200	200	200	0
24	200	200	200	200	0	200	200	200	200	0
25	200	200	200	200	0	200	200	200	200	0
26	200	200	200	200	0	200	200	200	200	0
27	200	4515	3925.90	3543.30	1253.30	3286.50	4439.90	3999.30	3968.70	261.28
28	200	12 980	200	450.58	1789.50	200	1.42E + 4	200	916.77	2924.20
29	200	3.64E + 4	200	910.58	5074.60	200	1.16E + 5	200	3.15E + 3	1.68E + 4
30	200	200	200	200	0	200	4.93E + 6	200	8.06E + 5	1.49E + 6

Table 5
Summary score table of the benchmark results for SCO and AccOppCSCO.

D	SCO					AccOppCSCO				
	Best	Worst	Median	Mean	Std	Best	Worst	Median	Mean	Std
10	9	12	9	10	14	21	18	21	20	16
30	6	11	10	11	13	24	19	20	19	17
50	4	9	9	13	13	26	21	21	17	17
100	9	8	10	13	8	21	22	20	17	22
Total	28	40	38	47	48	**92**	**80**	**82**	**73**	**72**

Table 6
Parameters of heuristics used in this study.

Algorithm's name	Acronym	Parameters
Ageist Spider Monkey Optimizer	ASMO	GLL = 20, LLL = 500, MG = 4, SS = 32
Chaotic Cuckoo Search optimizer	CCS	β = 1.5, N = 10, p_a = 0.025
Coyote Optimization Algorithm	COA	Number of coyotes in each pack = 10
Dragonfly Algorithm	DA	r = Δ_{max} = 0.1(ub − lb)
Elephant Herding Optimizer	EHO	n_c = 8, β = 0.1, α = 0.3, γ = 0.05
Human Felicity Algorithm	HFA	$N_{elite} = N_{pers} = N_{und}$ = 2,
Moth–Flame Optimizer	MFO	b = 1
Particle Swarm Optimizer	PSO	$c_1 = c_2$ = 1.5, w = 1
Red Fox Optimizer	RFO	a = 0.2, θ_0 = 1
Sine Cosine Algorithm	SCA	a = 2
Variable Neighborhood Bat Algorithm	VNBA	A = 0.5, r = 0.5, ϵ = 0.001
Whale Optimization Algorithm	WOA	b = 1

```
    x=x*x; x=sqrt(x);
    x=log(x); x=exp(x); x=x/(x+2);
end for
```

Then, the computing time (T_1) of Function 1 was calculated for 200 000 evaluations of a certain dimension D (5,10, and 15). Finally, the computation times (T_2) for 200 000 evaluations were obtained using the AccOppCSCO and SCO algorithms of F_1 for the same problem dimensions. This process was iterated five times to derive the $T2$ values and compute the average $\widehat{T_2}$. The values of T_0, T_1, $\widehat{T_2}$, and $(\widehat{T_2} - T_1)/T_0$ for the computational complexity of both algorithms are summarized in Table 10.

In this table, the computational complexities of AccOppCSCO and SCO are presented according to problem size. The $\widehat{T_2}$ results in this table show that the computation times for the proposed AccOppCSCO are greater than those of the original SCO algorithm for all problem sizes. The main reason for this increase in computation time is the addition of new mechanisms to the original algorithm. However, despite the increased computation time, it is still superior to most population-based heuristics.

Table 7
Comparison Results of AccOppCSCO and the first group of heuristics for 30D CEC2014 benchmarks.

		ASMO	CCS	COA	DA	EHO	HFA	AccOppCSCO
F1	*Mean*	1.20E + 8	2.93E + 9	1.03E + 8	4.49E + 8	4.83E + 8	3.34E + 7	3.09E + 7
	Rank	4	7	3	5	6	2	1
F2	*Mean*	1.95E + 9	9.48E + 10	2.06E + 9	3.67E + 10	3.82E + 10	2.14E + 9	3.03E + 5
	Rank	2	7	3	5	6	4	1
F3	*Mean*	2.45E + 5	1.01E + 7	1.78E + 5	1.81E + 5	5.64E + 4	1.66E + 5	7.97E + 4
	Rank	6	7	4	5	1	3	2
F4	*Mean*	711	2.43E + 4	398	4860	5450	488	157.90
	Rank	4	7	2	5	6	3	1
F5	*Mean*	511	503	488	20.80	505	498	20.22
	Rank	7	5	3	2	6	4	1
F6	*Mean*	622	604	437	37.20	618	599	28.79
	Rank	7	5	3	2	6	4	1
F7	*Mean*	704	1660	701	354	989	700	0.57
	Rank	5	7	4	2	6	3	1
F8	*Mean*	1060	1160	421	289	1050	801	132.68
	Rank	6	7	3	2	5	4	1
F9	*Mean*	1170	1330	902	301	1210	902	175
	Rank	4	6	3	2	5	3	1
F10	*Mean*	9010	8060	1870	6570	7690	1000	3577.70
	Rank	7	6	2	4	5	1	3
F11	*Mean*	9800	8830	4980	6920	7960	1100	4057
	Rank	7	6	3	4	5	1	2
F12	*Mean*	1210	1210	1210	2.21	1210	1210	1.24
	Rank	3	3	3	2	3	3	1
F13	*Mean*	1310	1310	1310	5.14	1310	1310	0.62
	Rank	4	4	4	2	3	4	1
F14	*Mean*	1410	1670	1410	131	1470	1410	0.40
	Rank	3	5	3	2	4	3	1
F15	*Mean*	1530	1.81E + 6	3.84E + 4	2.85E + 5	2.89E + 5	3.40E + 4	25.57
	Rank	2	7	4	5	6	3	1
F16	*Mean*	1610	1610	1610	1.32E + 1	1610	1610	12.52
	Rank	3	3	3	2	3	3	1
F17	*Mean*	8.77E + 6	3.82E + 8	6.41E + 6	1.61E + 7	1.15E + 7	6.53E + 6	1.42E + 6
	Rank	4	7	2	6	5	3	1
F18	*Mean*	9.30E + 7	1.21E + 10	2.18E + 8	6.11E + 8	3.83E + 8	2.02E + 8	3763
	Rank	2	7	4	6	5	3	1
F19	*Mean*	1900	2730	2540	268	2090	1910	26.06
	Rank	3	7	6	2	5	4	1
F20	*Mean*	7.12E + 5	5.99E + 6	8.81E + 4	2.78E + 5	2.28E + 4	9870	5.93E + 4
	Rank	6	7	4	5	2	1	3
F21	*Mean*	4.34E + 6	1.85E + 8	2.51E + 5	8.23E + 6	2.74E + 6	2.05E + 5	9.08E + 5
	Rank	5	7	2	6	4	1	3
F22	*Mean*	3370	3.43E + 6	8.21E + 4	1.12E + 3	3270	5.60E + 4	679.62
	Rank	4	7	6	2	3	5	1
F23	*Mean*	2600	3730	2630	576	2430	2320	200
	Rank	5	7	6	2	4	3	1
F24	*Mean*	2610	2530	2410	281	2520	2420	200
	Rank	7	6	3	2	5	4	1
F25	*Mean*	2670	2660	5290	240	2620	2520	200
	Rank	6	5	7	2	4	3	1
F26	*Mean*	2650	2790	2620	104	2630	2620	131.88
	Rank	5	6	3	1	4	3	2
F27	*Mean*	4170	7070	2720	1090	2810	2720	443.62
	Rank	5	6	3	2	4	3	1
F28	*Mean*	4160	1.41E + 4	2820	2240	2910	2820	419.88
	Rank	5	6	3	2	4	3	1
F29	*Mean*	1.81E + 6	1.10E + 9	1.26E + 7	2.15E + 7	3.62E + 5	1.80E + 7	9.29E + 5
	Rank	3	7	4	6	1	5	2
F30	*Mean*	8.86E + 4	2.35E + 7	3.88E + 5	4.79E + 5	5.41E + 5	3.91E + 5	5.21E + 4
	Rank	2	7	3	5	6	4	1
Average rank		**4.53**	**6.13**	**3.53**	**3.33**	**4.40**	**3.10**	**1.33**
Overall rank		**6**	**7**	**4**	**3**	**5**	**2**	**1**

Table 8
Comparison Results of AccOppCSCO and the second group of heuristics for 30D CEC2014 benchmarks.

		MFO	PSO	RFO	SCA	VNBA	WOA	AccOppCSCO
F1	*Mean*	7.59E + 7	8.19E + 7	1.93E + 8	2.24E + 8	1.21E + 8	3.24E + 7	3.09E + 7
	Rank	3	4	6	7	5	2	1
F2	*Mean*	1.36E + 10	2.81E + 4	1.35E + 10	1.60E + 10	2.81E + 8	5.05E + 6	3.03E + 5
	Rank	6	1	5	7	4	3	2
F3	*Mean*	8.99E + 4	20.9	8.73E + 4	3.79E + 4	2.63E + 4	2.97E + 4	7.97E + 4
	Rank	7	1	6	4	2	3	5
F4	*Mean*	1140	9010	2180	1020	629	176	157.90
	Rank	5	7	6	4	3	2	1
F5	*Mean*	20.40	219	499	20.90	515	20.40	20.22
	Rank	2	4	5	3	6	2	1
F6	*Mean*	24	6780	600	34.40	616	34.80	28.79
	Rank	1	7	5	3	6	4	2
F7	*Mean*	117	1.61	701	133	702	1.03	0.57
	Rank	4	3	6	5	7	2	1
F8	*Mean*	143	1100	802	236	815	176	132.68
	Rank	2	7	5	4	6	3	1
F9	*Mean*	223	2.30E + 6	903	267	1030	213	175
	Rank	3	7	5	4	6	2	1
F10	*Mean*	3470	44.30	3180	5880	1200	3770	3577.70
	Rank	4	1	3	7	2	6	5
F11	*Mean*	4150	6.50E + 5	4430	7040	4960	4500	4057
	Rank	2	7	3	6	5	4	1
F12	*Mean*	0.43	373	1210	2.45	1210	1.67	1.24
	Rank	1	5	6	4	6	3	2
F13	*Mean*	2.21	214	1310	2.89	1310	0.50	0.62
	Rank	3	5	6	4	6	1	2
F14	*Mean*	35.40	905	1410	41.10	1410	0.28	0.40
	Rank	3	5	6	4	6	1	2
F15	*Mean*	2.23E + 5	1.04E + 7	1.09E + 5	2820	1520	70	25.57
	Rank	6	7	5	4	3	2	1
F16	*Mean*	1.27E + 1	2.49E + 7	1610	1.28E + 1	1610	1.26E + 1	12.52
	Rank	3	6	5	4	5	2	1
F17	*Mean*	3.39E + 6	1.26E + 4	9E+6	6.61E + 6	1.47E + 7	4.33E + 6	1.42E + 6
	Rank	3	1	6	5	7	4	2
F18	*Mean*	5.19E + 6	0.05	2.03E + 8	1.85E + 8	1.24E + 7	1.55E + 4	3763
	Rank	4	1	7	6	5	3	2
F19	*Mean*	73.60	29.10	1910	90.80	1940	44.80	26.06
	Rank	4	2	6	5	7	3	1
F20	*Mean*	5.67E + 4	13.40	5.87E + 4	1.31E + 4	3.70E + 4	2.04E + 4	5.93E + 4
	Rank	5	1	6	2	4	3	7
F21	*Mean*	7.83E + 5	388	2.51E + 6	1.48E + 6	3.67E + 6	9.48E + 5	9.08E + 5
	Rank	2	1	6	5	7	4	3
F22	*Mean*	8.67E + 4	0.74	2.28E + 4	754	2910	749	679.62
	Rank	7	1	6	4	5	3	2
F23	*Mean*	371	2940	2320	370	2630	331	200
	Rank	4	7	5	3	6	2	1
F24	*Mean*	276	1.05E + 6	2420	201	2600	206	200
	Rank	4	7	5	2	6	3	1
F25	*Mean*	214	16.40	2520	227	2690	225	200
	Rank	3	1	6	5	7	4	2
F26	*Mean*	103	2.83E + 5	2620	102	2670	100	131.88
	Rank	3	7	5	2	6	1	4
F27	*Mean*	921	22.80	2720	722	4230	905	443.62
	Rank	5	1	6	3	7	4	2
F28	*Mean*	1120	2.59	2820	2000	4460	2150	419.88
	Rank	3	1	6	4	7	5	2
F29	*Mean*	3.06E + 6	271	6.79E + 6	1.34E + 7	3.11E + 6	4.38E + 6	9.29E + 5
	Rank	3	1	6	7	4	5	2
F30	*Mean*	5.89E + 4	1.18E + 7	3.29E + 5	2.48E + 5	4.92E + 4	8.20E + 4	5.21E + 4
	Rank	3	7	6	5	1	4	2
Average rank		**3.60**	**3.87**	**5.50**	**4.40**	**5.23**	**3**	**2.07**
Overall rank		**3**	**5**	**7**	**4**	**6**	**2**	**1**

Table 9
Comparison Results of AccOppCSCO and SCO Algorithms for CEC2020 benchmarks.

Dim	No	SCO					AccOppCSCO				
		Best	Worst	Median	Mean	Std	Best	Worst	Median	Mean	Std
5	1	262.09	1.19E + 4	3658.8	4459.6	3260.2	1125.6	1.09E + 4	3961.5	4663.8	2281.6
	2	900.41	1752.6	1326.5	1306.4	197.96	900.28	1637.4	1254.7	1231.5	155.74
	3	407.46	436.57	415.46	417.53	7.5551	407.69	428.52	422.65	419.3	6.9133
	4	1500.2	1502.5	1500.8	1500.9	0.7091	1500.1	1502.4	1500.6	1500.7	0.5223
	5	1249.5	2882.1	1790.4	1811.7	333.15	1413.8	22 931	7612.8	7344.5	5073.2
	6	–	–	–	–	–	–	–	–	–	–
	7	–	–	–	–	–	–	–	–	–	–
	8	1411.1	1510.4	1502.3	1463.9	44.187	1400	1509.8	1432.5	1458.3	44.848
	9	1501.7	1850.4	1820.7	1733.5	134.5	1501	1600.1	1600.1	1590.8	28.47
	10	1847.4	1892	1847.4	1854.5	16.096	1847.4	1889.6	1847.5	1854.4	15.871
10	1	1.80E + 4	2.79E + 7	6.78E + 4	1.04E + 6	5.07E + 6	1.07E + 4	2.70E + 6	7.86E + 4	1.91E + 5	4.80E + 5
	2	1303.3	2618	2085.1	1993.6	336.17	1293.1	2351.6	1992.8	1938.9	302.44
	3	432.6	538.25	465.48	470.4	24.923	419.55	520.61	449.6	452.56	21.579
	4	1501.3	1509.5	1503.5	1504.2	2.1035	1501.3	1521.7	1503.4	1504.4	3.6651
	5	1.61E + 4	1.16E + 6	5.60E + 5	5.39E + 5	2.39E + 5	3660.2	5.90E + 5	2.69E + 5	2.87E + 5	1.94E + 5
	6	1000.8	1240.9	1019.2	1030.8	44.95	1001.4	1066.7	1019.1	1018.6	11.569
	7	26 833	2.65E + 6	5.20E + 5	8.33E + 5	8.13E + 5	2337.4	1.79E + 5	8587.8	1.88E + 4	3.27E + 4
	8	1461.5	2777	1506.6	1574.7	270.55	1447.9	1540.5	1509.9	1510.3	14.878
	9	1600.7	1898.8	1860.5	1837.5	76.786	1600.5	1909.4	1856.9	1793.5	116.48
	10	1899	2029.1	1947.7	1942.9	27.625	1899.7	1957.8	1930.5	1926	23.079
15	1	3.03E + 5	7.21E + 7	3.51E + 6	7.64E + 6	1.33E + 7	2.82E + 5	8.37E + 7	1.66E + 6	7.19E + 6	1.59E + 7
	2	2006.30	3491.70	2788.00	2811.50	376.30	1544.70	3318.00	2639.30	2591.00	445.10
	3	456.20	611.83	519.70	521.85	41.61	455.45	619.97	512.63	520.30	40.19
	4	1506.60	1586.30	1510.10	1515.70	16.03	1506.80	1554.70	1511.70	1514.00	9.14
	5	6.62E + 4	5.85E + 6	8.15E + 5	1.35E + 6	1.38E + 6	3.96E + 5	2.92E + 6	8.39E + 5	1.16E + 6	7.02E + 5
	6	1008.20	1696.90	1349.90	1351.30	260.53	1008.20	1696.60	1349.8	1351.2	260.48
	7	8.15E + 4	6.00E + 6	1.05E + 6	1.56E + 6	1.61E + 6	1.54E + 4	1.97E + 6	2.24E + 5	3.17E + 5	3.81E + 5
	8	1506.10	3.48E + 3	1511.00	1576.60	359.19	1508.10	1519.70	1511.8	1512	2.2651
	9	1907.90	2113.00	1949.20	1975.10	54.25	1678.00	2061.30	1972.2	1940.1	105.38
	10	2056.80	2297.30	2168.50	2171.50	62.73	2022.30	2283.60	2140.7	2141.8	45.315
20	1	1.36E + 6	4.24E + 8	6.72E + 7	1.01E + 8	1.01E + 8	1.37E + 7	8.41E + 8	9.46E + 7	1.44E + 8	1.63E + 8
	2	2967.90	4663.80	3613.10	3686.00	485.80	2150.70	4641.70	3686.40	3695.30	579.38
	3	502.85	681.14	582.45	587.03	47.60	496.47	707.32	595.15	596.75	49.15
	4	1513.70	1828.00	1544.40	1574.70	76.27	1507.50	1763.40	1526.80	1544.90	48.44
	5	4.88E + 5	4.24E + 6	1.54E + 6	1.60E + 6	7.84E + 5	2.69E + 5	7.09E + 6	1.22E + 6	1.78E + 6	1.63E + 6
	6	1027.80	1611.50	1257.40	1290.60	256.37	1027.80	1611.50	1257.40	1290.60	256.37
	7	3.87E + 4	4.79E + 6	7.53E + 5	9.82E + 5	1.07E + 6	5.76E + 4	2.57E + 6	2.08E + 5	5.23E + 5	6.85E + 5
	8	1514.10	5277.70	1538.40	1775.80	895.70	1513.30	1563.00	1527.40	1531.60	14.55
	9	1709.60	2257.80	2037.80	2035.20	83.72	1661.30	2194.00	2031.30	2024.30	91.25
	10	1965.00	2120.90	2030.90	2038.20	37.24	1937.70	2061.10	2009.20	2007.20	26.32

Table 10
Computational complexity of AccOppCSCO and SCO algorithms.

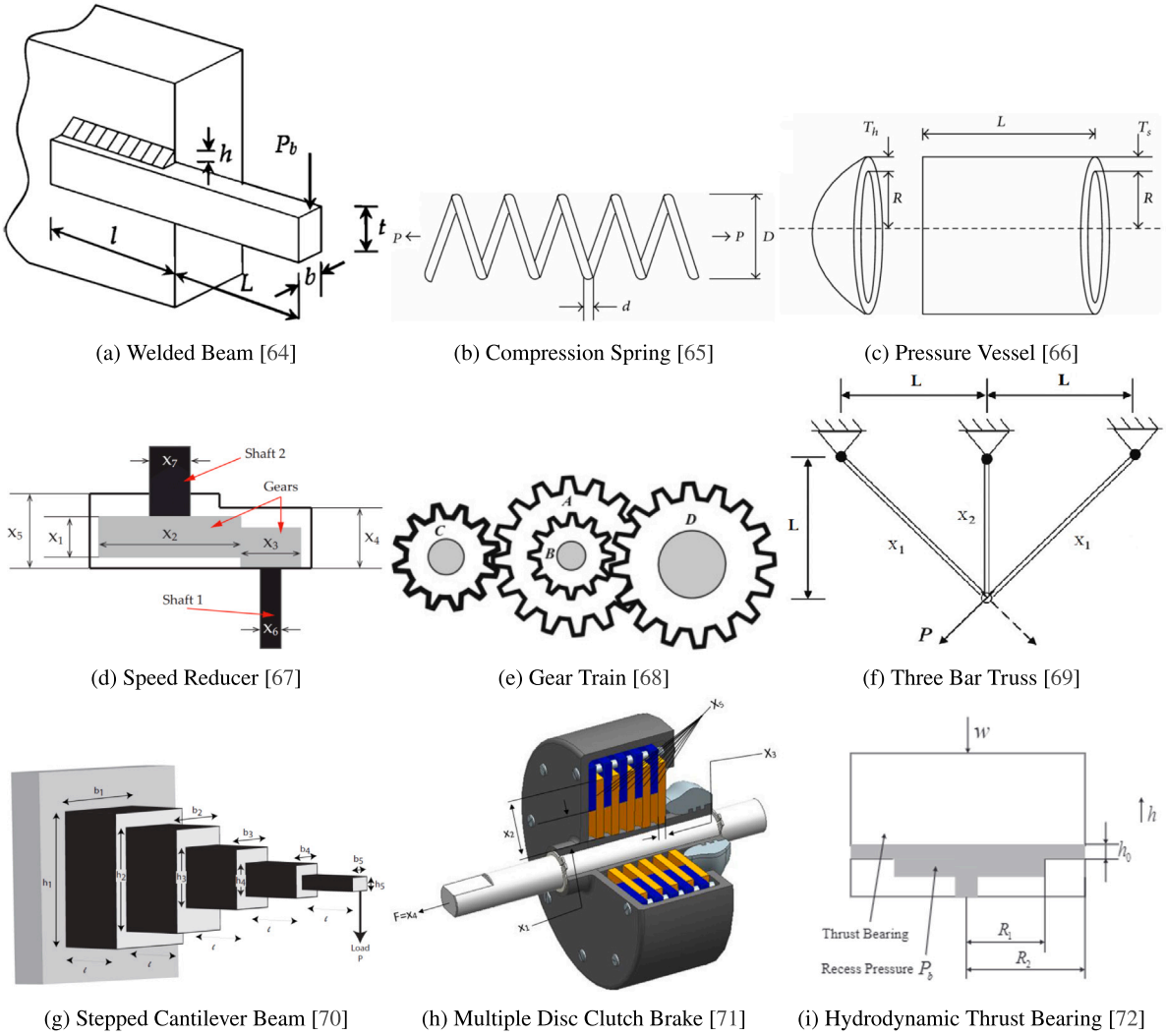
D	T_0	T_1	SCO		AccOppCSCO	
			$\widehat{T}_2$	$(\widehat{T}_2 - T_1)/T_0$	$\widehat{T}_2$	$(\widehat{T}_2 - T_1)/T_0$
5	6.95E − 2	2.28E − 1	9.68E − 1	1.06E + 1	6.06E + 0	8.39E + 1
10	6.95E − 2	2.35E − 1	1.05E + 0	1.18E + 1	6.55E + 0	9.09E + 1
15	6.95E − 2	4.20E − 1	1.13E + 0	1.03E + 1	6.93E + 0	9.37E + 1

4.4. Performance results for engineering design problems

To evaluate the performance of the AccOppCSCO algorithm proposed in this study, it was tested with ten engineering design problems in this subsection. In Fig. 10, the engineering design problems utilized in this study are illustrated, except Himmelblau's problem. These problems include Welded Beam Design (WBD) [64], Compression Spring Design (CSD) [65], Pressure Vessel Design (PVD) [66], Speed Reducer Design (SRD) [67], Gear Train Design (GTD) [68], Himmelblau's Problem (HP), Three Bar Truss Design (TBTD) [69], Stepped Cantilever Beam Design (SCBD) [70], Multiple Disc Clutch Brake Design (MDCBD) [71], and Hydrodynamic Thrust Bearing Design (HTBD) [72] respectively.

The WBD problem involves determining the optimal dimensions and shape of a welded beam to satisfy specific design criteria. The WBD problem involves determining the optimal dimensions and shape of a welded beam to meet specific design criteria, typically cost or weight. Factors such as the beam and weld dimensions, as well as the load-bearing capacity, were considered in the problem formulation. In the CSD problem, compression springs are designed to achieve a specific spring stiffness or length when subjected to a given load. PVD is based on the design of containers capable of safely holding liquids or gases under pressure in such a way as to withstand the internal pressure exerted by the substance they contain and to ensure the integrity and safety of the container and its surroundings. This is an engineering design problem consisting of four constraints and four variables.

SRD involves minimizing the weight of the speed reducer to reduce the speed of rotation from a motor to an output that is, commonly used in machinery. GTD is a four-variable problem based on the design of gear combinations to achieve the desired speed ratios and power transmission. Himmelblau's problem is a mathematical optimization problem that aims to determine the minimum or maximum points of multi-variable functions. In the TBTD problem, three-bar trusses were designed to provide the optimum strength against a given load. The SCBD problem involves the optimization of the dimensions and structure of a cantilever beam that has multiple steps or changes in the cross-sectional area along its length. The objective is usually to



(a) Welded Beam [64] (b) Compression Spring [65] (c) Pressure Vessel [66]

(d) Speed Reducer [67] (e) Gear Train [68] (f) Three Bar Truss [69]

(g) Stepped Cantilever Beam [70] (h) Multiple Disc Clutch Brake [71] (i) Hydrodynamic Thrust Bearing [72]

Fig. 10. Engineering design problems used in this study.

maximize the load-bearing capacity while minimizing the deflection or stress under specified constraints, such as material properties, load conditions, and geometric limitations.

The MDCBD problem concerns the design of a clutch or brake mechanism that uses multiple discs to transmit torque or efficiently provide braking action. The objective is to optimize the dimensions, materials, and configuration of the clutch or brake system to achieve the desired performance characteristics such as torque capacity, response time, and durability, while meeting specific operational requirements. This problem is crucial in automotive, industrial, and machinery applications, where precise control of torque transmission or braking is essential for safe and reliable operation. The field of HTBD is concerned with the creation of bearings that utilize fluid lubrication to support axial loads and reduce friction between rotating components. These bearings function by generating a fluid film between surfaces, thereby minimizing friction and wear. The key design aspects considered include bearing geometry, material choice, and design of the lubrication system to achieve optimal performance.

Table 11 presents the number of design variables and their boundary values for the engineering design problems considered in this study. For all problems, the original SCO and the proposed AccOppCSCO algorithms as well as popular heuristic algorithms from the literature were used. These heuristic algorithms include the Genetic Algorithm (GA) [18], Particle Swarm Optimizer (PSO) [31], Differential Evolution (DE) Algorithm [19], Artificial Bee Colony (ABC) Algorithm [32,33], Grey Wolf Optimizer (GWO) [36], Gravitational Search Algorithm (GSA) [23], Biogeography-based Optimization (BBO) Algorithm [73], Sine Cosine Algorithm (SCA) [62], and Salp Swarm Algorithm (SSA) [74].

In the real world, engineering design problems frequently involve a specific number of constraint functions. Constraint functions determine whether the solutions are suitable. A valid candidate solution must satisfy all constraint functions. In minimization problems, an invalid solution is penalized by multiplying the constraint functions by a large value (e.g., 10^6) and adding the result to the cost value. For all population-based heuristic algorithms, the number of populations was set to 10 and the maximum number of iterations was set to 100. The heuristic algorithms were executed 10 times for each engineering design problem. The parameters of the heuristics are listed in Table 12.

Fig. 11 shows the best convergence curves obtained by the original SCO and proposed AccOppCSCO algorithms among the 10 runs for the Welded Beam and Compression Spring design problems. Despite the comparable cost values obtained at the end of the optimization process for both problems, it is evident that the convergence speed of the AccOppCSCO algorithm is superior to that of SCO.

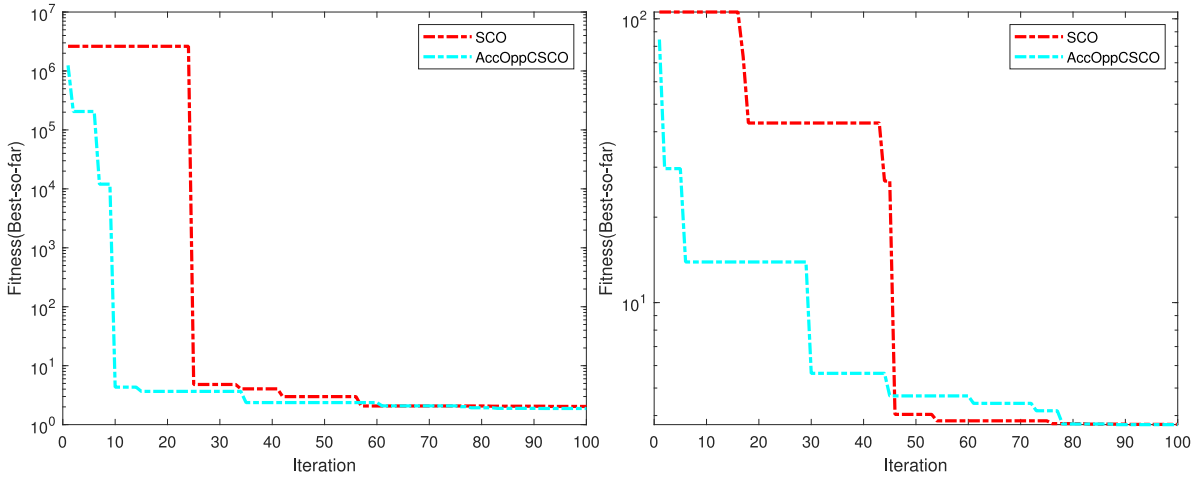
In Table 13, the results of AccOppCSCO and other heuristics are summarized. The best solutions for the 4-dimensional WBD problem,

Table 11
Engineering design problems used in this study.

No	Problem's name	D	Lower boundaries	Upper boundaries
1	Welded Beam Design	4	0.1,0.1,0.1,0.1	2,10,10,2
2	Compression Spring Design	3	0.05,0.25,2	2,1.3,15
3	Pressure Vessel Design	4	0,0,10,10	99,99,200,200
4	Speed Reducer Design	7	2.6,0.7,17,7.3,7.8,2.9,5	3.6,0.8,28,8.3,8.3,3.9,5.5
5	Gear Train Design	4	12,12,12,12	60,60,60,60
6	Himmelblau's Problem	5	78,33,27,27,27	102,45,45,45,45
7	Three Bar Truss Design	2	0,0	1,1
8	Stepped Cantilever Beam Design	10	1,1,1,1,1,30,30,30,30,30	5,5,5,5,5,65,65,65,65,65
9	Multiple Disc Clutch Brake Design	5	60,90,1.5,600,2	80,110,3,1000,9
10	Hydrodynamic Thrust Bearing Design	4	1,1,1E6,1	16,16,16E6,16

Table 12
Parameters of heuristics used for Engineering design problems.

Algorithm's name	Acronym	Parameters
Artificial Bee Colony Algorithm	ABC	$L = (0.6 \cdot DN)$
Biogeography-based Optimization Algorithm	BBO	KeepRate = 0.2, $\alpha = 0.9$, $p_m = 0.1$
Differential Evolution Algorithm	DE	$beta_{min} = 0.2$, $beta_{max} = 0.8$, $p_c = 0.2$
Genetic Algorithm	GA	$p_c = 0.95$, $p_m = 0.1$
Gravitational Search Algorithm	GSA	$R_{norm} = 2$, ElitistCheck = 1, $R_{power} = 1$
Grey Wolf Optimizer	GWO	–
Particle Swarm Optimizer	PSO	$w_{max} = 0.9$, $w_{min} = 0.2$, $c_1 = 2$, $c_2 = 2$
Sine Cosine Algorithm	SCA	a = 2
Salp Swarm Algorithm	SSA	–



(a) Welded Beam (b) Compression Spring

Fig. 11. Best convergence curves of AccOppCSCO and SCO algorithms for (a) Welded Beam and (b) Compression Spring design problems.

Table 13
Results for Welded Beam Design problem.

Algorithm	Best solution				Cost					Best run
	x_1	x_2	x_3	x_4	Best	Worst	Median	Mean	SD	time (s)
GA	0.2954	2.8668	6.2943	0.4271	2.4577	734.68	3.0095	76.238	231.35	4.35E − 2
PSO	0.1983	3.3931	9.0406	0.2058	1.7046	2.6054	1.7955	1.9194	0.2903	3.53E − 2
DE	0.2403	3.2138	7.7826	0.2774	1.9927	2.9609	2.3771	2.4632	0.3474	1.14E − 2
ABC	0.2217	4.9593	7.9384	0.2679	2.2093	3.1156	2.4620	2.5069	0.2472	4.12E − 2
GWO	0.1999	3.4274	9.1321	0.2053	1.7235	1.8265	1.7460	1.7560	0.0317	3.93E − 3
GSA	0.2147	3.7241	8.0633	0.2584	1.9663	3.9361	2.6140	2.6450	0.5826	8.31E − 3
BBO	0.2232	3.4976	7.2523	0.3196	2.1440	3.8833	2.6259	2.7538	0.5504	1.81E − 2
SCA	0.1629	4.0860	9.9173	0.2137	1.9641	2.5193	2.2001	2.2321	0.1854	3.79E − 3
SSA	0.1985	3.1806	9.8610	0.2020	1.7845	2.7603	2.3682	2.3109	0.3108	3.96E − 3
SCO	0.1871	5.6996	9.3631	0.2045	2.0348	181.80	3.0534	42.220	72.235	2.02E − 4
AccOppCSCO	0.1514	4.2863	9.9484	0.2023	1.8792	3.8296	2.2203	2.4375	0.6913	1.87E − 3

the statistical results of the cost values obtained following repeated runs, and the best running times obtained among the 10 runs are given in the table. Table 14 presents the results of the AccOppCSCO algorithm and other algorithms for the CSD problem with three dimensions.

The mean cost values in both tables indicate that the GWO algorithm is the most successful for the WBD problem and the PSO algorithm is the most successful for the CSD problem. The proposed AccOppCSCO algorithm was ranked 5th according to the mean metric for both problems. In terms of running times, the original SCO ranked

Table 14
Results for Compression Spring Design problem.

Algorithm	Best solution			Cost					Best run
	x_1	x_2	x_3	Best	Worst	Median	Mean	SD	time (s)
GA	0.1366	1.2881	11.615	3.7808	21.384	6.7096	8.6473	5.6329	3.73E − 2
PSO	0.1392	1.3000	11.896	3.6619	3.6953	3.6715	3.6756	0.0144	3.04E − 2
DE	0.1391	1.3000	11.888	3.6619	100.04	3.7170	13.689	30.361	1.07E − 2
ABC	0.1376	1.2579	12.436	3.6938	3.8710	3.7459	3.7642	0.0565	3.65E − 2
GWO	0.1392	1.3000	11.883	3.6625	3.7202	3.6791	3.6862	0.0205	3.40E − 3
GSA	0.1397	1.2660	12.188	4.2134	22.863	11.038	11.736	5.0200	7.01E − 3
BBO	0.1387	1.2996	11.767	3.6687	100.04	6.8892	16.492	29.589	1.40E − 2
SCA	0.1319	1.1223	15.000	3.7594	4.9420	4.2325	4.2821	0.4302	3.21E − 3
SSA	0.1376	1.2742	12.114	3.6847	14.265	8.3222	8.3932	3.7892	3.38E − 3
SCO	0.1373	1.2197	13.514	3.7232	29.591	7.9284	13.082	10.916	1.56E − 4
AccOppCSCO	0.1348	1.1803	13.959	3.7157	20.509	5.7749	7.8700	5.4028	1.52E − 3

Table 15
Results for Pressure Vessel Design problem.

Algorithm	Best solution				Cost					Best run
	x_1	x_2	x_3	x_4	Best	Worst	Median	Mean	SD	time (s)
GA	1.0024	0	64.248	15.430	2513.1	51 681	3785.1	8473.2	15 201	3.78E − 2
PSO	0.9141	0.0014	59.458	37.470	2906.2	3638.4	3577.9	3445.4	283.09	3.04E − 2
DE	1.0932	0	65.225	10.000	2302.6	6058.1	2302.7	2692.2	1183.3	1.05E − 2
ABC	1.1672	0	66.057	10.000	2424.5	3593.3	2652.2	2843.1	454.16	3.64E − 2
GWO	1.0997	0.0018	65.125	10.792	2345.8	3637.7	2504.2	2766.2	494.14	3.30E − 3
GSA	0.3813	35.521	53.131	100.50	1.84E + 5	1.59E + 6	4.15E + 5	5.83E + 5	4.51E + 5	6.84E − 3
BBO	1.0808	0	64.803	12.681	2390.2	5.E+4	3395.4	9002.6	1.E+4	1.56E − 2
SCA	1.0836	0	66.221	10.000	2404.6	6723.3	6068.8	4742.0	1914.4	3.09E − 3
SSA	1.0856	0	64.927	11.292	2336.1	4770.5	3628.1	3562.8	686.16	3.24E − 3
SCO	1.1087	0	66.191	11.268	2457.6	29 723	4481.9	6912.9	8117.2	1.40E − 4
AccOppCSCO	1.1020	0	66.729	13.146	2605.6	5767.8	3900.6	4036.3	901.46	1.47E − 3

Table 16
Results for Speed Reducer Design problem.

Algorithm	Best solution							Best	Best run
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	cost	time (s)
GA	2.9921	0.800	17.00	7.300	7.800	3.900	5.500	2.97E + 15	4.24E − 2
PSO	2.8902	0.800	17.00	7.929	7.800	3.900	5.500	2.97E + 15	3.29E − 2
DE	3.6000	0.800	17.00	7.300	7.800	3.900	5.500	2.97E + 15	1.14E − 2
ABC	3.6000	0.800	17.00	7.300	7.800	3.900	5.500	2.97E + 15	3.90E − 2
GWO	3.6000	0.800	17.00	7.300	7.800	3.900	5.500	2.97E + 15	4.45E − 3
GSA	3.1567	0.799	17.19	7.709	7.935	3.431	5.434	3.58E + 15	8.32E − 3
BBO	3.2134	0.800	17.00	8.144	7.800	3.782	5.500	2.97E + 15	2.16E − 2
SCA	3.6000	0.800	17.00	7.300	7.800	3.900	5.500	2.97E + 15	4.12E − 3
SSA	3.6000	0.800	17.00	7.300	7.800	3.900	5.500	2.97E + 15	4.11E − 3
SCO	3.3895	0.798	17.00	8.211	7.801	3.116	5.491	3.03E + 15	2.25E − 4
AccOppCSCO	2.6740	0.800	17.00	8.011	7.800	3.592	5.500	2.97E + 15	1.64E − 3

first, followed by AccOppCSCO. Population-based heuristics, except for SCO and the proposed AccOppCSCO, reach a theoretical number of 1000 function evaluations for 10 candidates and 100 maximum iterations. However, the SCO and AccOppCSCO algorithms have been observed to find solutions to engineering problems with 100 function evaluations for 100 iterations with only a single candidate. The mean cost and standard deviation of AccOppCSCO are superior to those of the SCO for both problems (WBD and CSD). This indicates that the proposed algorithm consistently finds closely matched values at different execution times.

Fig. 12 shows the best convergence curves of the AccOppCSCO and SCO algorithms for the Pressure Vessel and Speed Reducer design problems. Although the AccOppCSCO algorithm demonstrates convergence similar to that of the original SCO for the PVD problem, its convergence curve is clearly more successful, particularly for the SRD problem. Tables 15 and 16 present the results of AccOppCSCO and other heuristic algorithms. PVD is an engineering design problem with four dimensions, whereas the SRD problem consists of seven design parameters.

Considering the results for the PVD problem, the DE algorithm was the most successful according to the mean cost metric. Although the mean cost of the proposed AccOppCSCO algorithm (4036.3) was better than that of the original SCO (6912.9) ranks sixth among all heuristics for the PVD problem. In the best running times for the PVD problem, the original SCO had the best score, whereas AccOppCSCO ranked second.

Returning to the SRD problem, this problem has 7 design parameters, and to present the results properly in the table, we have only added the best cost and best running time along with the 7 parameter values. The other heuristics have the same best cost value for this problem, except for the GSA and SCO. Therefore, in terms of running time, the proposed AccOppCSCO algorithm obtains the same result in a shorter time.

Fig. 13 illustrates the convergence curves, which demonstrate the optimal performance of the AccOppCSCO and SCO algorithms in addressing the design challenges posed by the Gear Train Design problem and Himmelblau's problem. The convergence curves demonstrate that the proposed AccOppCSCO algorithm outperformed the original SCO for GTD and Himmelblau's problems. It is evident that the SCO algorithm was stuck in the exploration phase for these two problems. For the GTD problem, the results consisting of the best solutions, best run time values, and statistical metrics of the costs are summarized in Table 17. Table 18 presents the results of Himmelblau's problem using AccOppCSCO and the other heuristic algorithms.

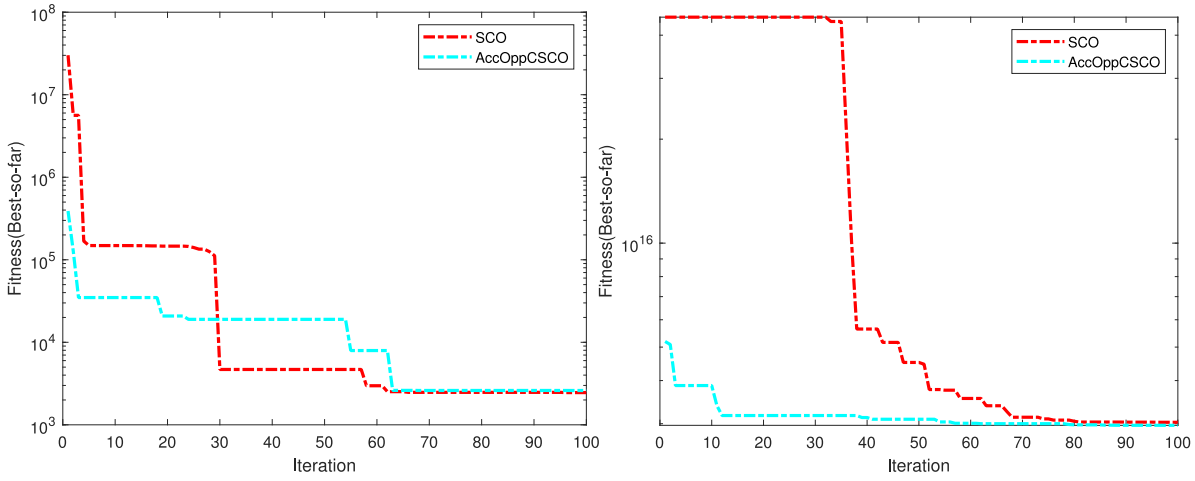


Fig. 12. Best convergence curves of AccOppCSCO and SCO algorithms for (a) Pressure Vessel and (b) Speed Reducer design problems.

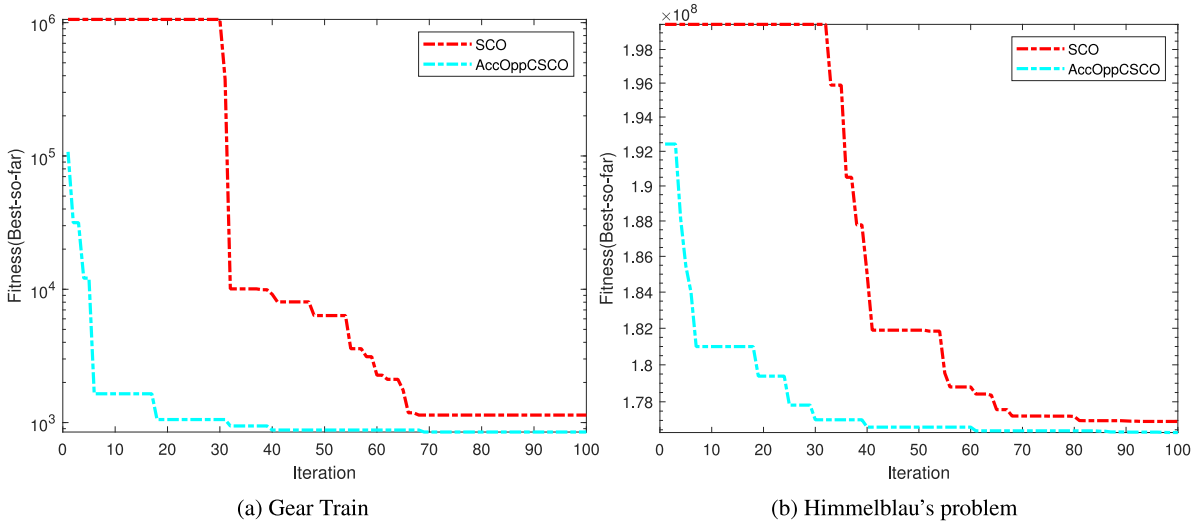


Fig. 13. Best convergence curves of AccOppCSCO and SCO algorithms for (a) Gear Train Design problem and (b) Himmelblau's problem.

The mean metric values obtained for the GTD problem indicate that the AccOppCSCO algorithm achieved a second rank with a value of 917.54, demonstrating superior performance to the original SCO (6080.6). In addition, as can be seen from the best running times, the AccOppCSCO algorithm, which ranked second after SCO, exhibited a faster solution time than the other population-based heuristics. Himmelblau's Problem is a five-dimensional engineering design problem, and the AccOppCSCO algorithm ranked first among the best cost values, together with eight population-based heuristics. In terms of the mean cost values, the AccOppCSCO algorithm is the second-best performer, and it is superior to SCO. The solution obtained by AccOppCSCO, which exhibited the second-best running time and employed a single candidate search strategy, demonstrated performance that was comparable to that of population-based heuristics that required substantial memory and processing power.

Fig. 14 shows the best convergence curves of the AccOppCSCO and SCO algorithms for the Three Bar Truss and Stepped Cantilever Beam design problems. In the convergence curve results for the TBTD problem, the SCO algorithm initiated a solution with rapid convergence. At the end of the optimization, both algorithms reached similar cost values that were comparable to each other.

As can be seen from the convergence curves for the SCBD problem, the original SCO algorithm performed worse than the proposed AccOppCSCO algorithm because of its handicap in the first phase. Table 19 summarizes the results of AccOppCSCO and the other heuristic algorithms for the TBTD problem. Here, according to the mean metric results, the best result (186.39) belongs to the PSO algorithm and the proposed AccOppCSCO algorithm ranks 5th with a score of 187.25. It can be seen that the AccOppCSCO algorithm achieves a better score than the mean score of the original SCO (188.29). When running times are analyzed, it becomes evident that the original SCO ranks first, with AccOppCSCO ranking second. The results of the SCBD problem were presented as two tables because of the 10-dimensional of problem. In Table 20, the best solutions found by the algorithms are given. Table 21 shows the cost metrics and best run time scores for the SCBD problem.

In the SCBD problem, while all algorithms achieve similar results, AccOppCSCO demonstrates superior stability and achieves the second-best runtime of 1.93E−3 s.

Finally, Fig. 15 shows a comparison of the convergence curves of the best run of the AccOppCSCO and SCO algorithms for the Multiple Disc Clutch Brake and Hydrodynamic Thrust Bearing design problems. From the convergence curves presented for the MDCBD problem, it is understood that the proposed AccOppCSCO reaches the solution point of the problem faster than SCO. It can be seen that the SCO algorithm gets stuck in the exploration phase and cannot minimize the candidate solution in this phase. Both algorithms used in the HTBD problem

Table 17
Results for Gear Train Design problem.

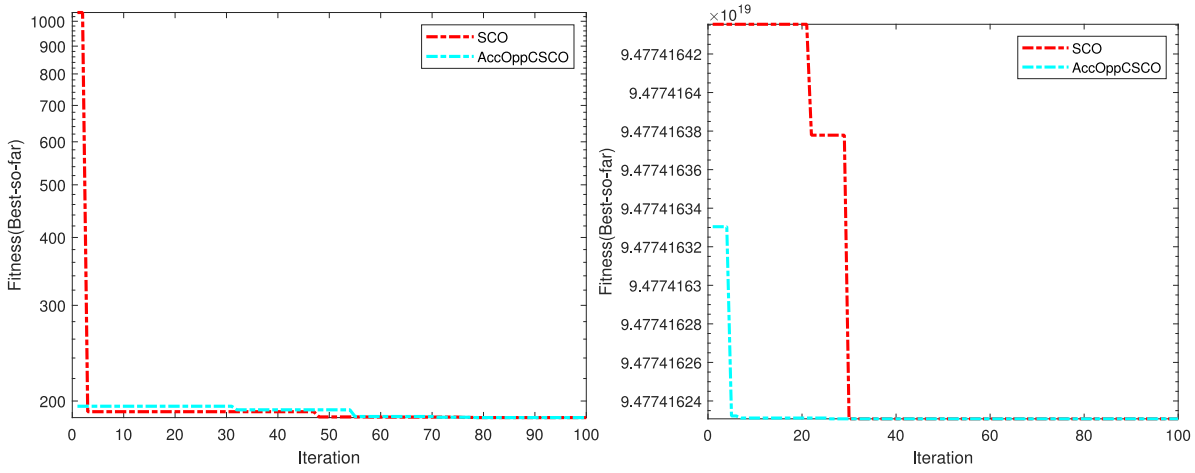
Algorithm	Best solution				Cost					Best run
	x_1	x_2	x_3	x_4	Best	Worst	Median	Mean	SD	time (s)
GA	12.000	12.000	60.000	12.463	821.15	821.15	821.15	821.15	2.40E − 13	3.77E − 2
PSO	12.000	12.000	60.000	12.000	821.15	821.15	821.15	821.15	2.40E − 13	3.01E − 2
DE	12.679	12.000	60.000	12.187	821.15	821.15	821.15	821.15	2.40E − 13	9.96E − 3
ABC	12.000	12.610	60.000	12.137	821.15	821.15	821.15	821.15	2.40E − 13	3.84E − 2
GWO	12.000	12.388	60.000	12.000	821.15	821.15	821.15	821.15	2.40E − 13	3.23E − 3
GSA	12.868	12.612	56.050	12.780	943.28	1.81E + 4	6157.3	8149.1	5.46E + 3	6.87E − 3
BBO	12.874	12.669	60.000	12.888	821.15	4496.5	850.10	1343.3	1.16E + 3	1.58E − 2
SCA	12.000	12.000	60.000	12.000	821.15	821.15	821.15	821.15	2.40E − 13	3.03E − 3
SSA	12.000	12.000	60.000	12.000	821.15	821.15	821.15	821.15	2.40E − 13	3.12E − 3
SCO	12.550	12.410	51.099	12.021	1138.3	2.32E + 4	3377.20	6080.6	6.77E + 3	1.41E − 4
AccOppCSCO	12.103	12.044	59.115	12.053	849.36	1184.40	879.05	917.54	1.07E + 2	1.50E − 3

Table 18
Results for Himmelblau's Problem.

Algorithm	Best solution					Cost					Best run
	x_1	x_2	x_3	x_4	x_5	Best	Worst	Median	Mean	SD	time (s)
GA	78.00	33.00	45.00	27.00	27.00	1.76E + 8	1.77E + 8	1.76E + 8	1.76E + 8	1.62E + 5	4.22E − 2
PSO	78.00	33.00	45.00	27.00	27.00	1.76E + 8	1.76E + 8	1.76E + 8	1.76E + 8	4.35E + 3	3.40E − 2
DE	78.00	33.00	45.00	27.00	27.00	1.76E + 8	1.76E + 8	1.76E + 8	1.76E + 8	0	1.18E − 2
ABC	78.00	33.00	45.00	27.00	27.00	1.76E + 8	1.76E + 8	1.76E + 8	1.76E + 8	0	4.01E − 2
GWO	78.00	33.00	45.00	27.00	27.00	1.76E + 8	1.76E + 8	1.76E + 8	1.76E + 8	0	3.62E − 3
GSA	80.71	34.14	42.43	34.51	27.20	1.80E + 8	1.84E + 8	1.82E + 8	1.82E + 8	1.49E + 6	7.71E − 3
BBO	78.00	33.00	45.00	27.00	27.00	1.76E + 8	1.78E + 8	1.77E + 8	1.77E + 8	6.49E + 5	1.91E − 2
SCA	78.00	33.00	45.00	27.00	27.00	1.76E + 8	1.76E + 8	1.76E + 8	1.76E + 8	0	3.52E − 3
SSA	78.00	33.00	45.00	27.00	27.00	1.76E + 8	1.76E + 8	1.76E + 8	1.76E + 8	0	3.29E − 3
SCO	78.02	33.05	44.66	34.80	27.00	1.77E + 8	1.87E + 8	1.79E + 8	1.80E + 8	3.23E + 6	1.58E − 4
AccOppCSCO	78.00	33.01	44.82	27.35	27.00	1.76E + 8	1.78E + 8	1.77E + 8	1.77E + 8	7.94E + 5	1.81E − 3

Table 19
Results for Three Bar Truss Design problem.

Algorithm	Best solution		Cost					Best run
	x_1	x_2	Best	Worst	Median	Mean	SD	time (s)
GA	0.7681	0.3269	186.580	202.160	191.460	192.470	5.460	4.07E − 2
PSO	0.7869	0.2879	186.390	186.390	186.390	186.390	0.000	3.17E − 2
DE	0.7867	0.2883	186.390	197.560	186.390	187.670	3.515	1.07E − 2
ABC	0.7892	0.2835	186.390	186.690	186.510	186.520	0.106	3.81E − 2
GWO	0.7882	0.2852	186.390	188.400	186.400	186.600	0.634	3.04E − 3
GSA	0.7679	0.3279	186.590	189.640	187.050	187.630	1.183	6.43E − 3
BBO	0.7730	0.3170	186.490	202.850	188.960	190.550	5.158	1.23E − 2
SCA	0.7943	0.2736	186.410	200.000	193.780	193.500	6.816	2.82E − 3
SSA	0.7868	0.2881	186.390	186.660	186.390	186.430	0.087	3.43E − 3
SCO	0.7793	0.3040	186.420	192.470	187.770	188.290	1.925	1.39E − 4
AccOppCSCO	0.7973	0.2680	186.440	188.730	186.910	187.250	0.744	1.65E − 3



(a) Three Bar Truss (b) Stepped Cantilever Beam

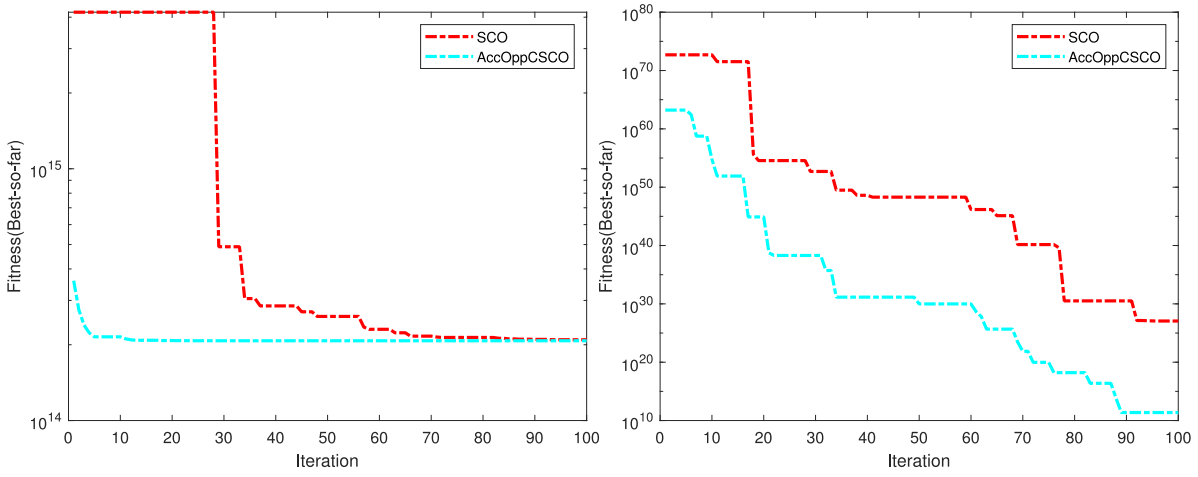
Fig. 14. Best convergence curves of AccOppCSCO and SCO algorithms for (a) Three Bar Truss and (b) Stepped Cantilever Beam design problems.

Table 20
Best solution results for Stepped Cantilever Beam Design problem.

Algorithm	Best solution									
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
GA	3.9167	3.2975	3.7064	4.8008	2.7639	55.567	51.934	47.971	51.082	30.959
PSO	3.1806	3.2333	2.5565	2.8266	1.9640	58.149	57.563	51.269	39.486	38.018
DE	3.0970	3.1846	3.9030	3.9965	2.6346	63.323	58.406	43.908	43.580	41.589
ABC	3.1770	2.8625	3.2705	2.9888	2.7562	61.331	57.988	44.375	44.163	34.258
GWO	5.0000	5.0000	2.4885	2.6465	3.2563	47.701	53.103	51.694	54.679	30.667
GSA	3.7753	4.1763	3.4204	3.3898	2.6567	53.802	46.795	54.426	50.036	44.420
BBO	3.5574	3.4886	3.5599	3.1326	4.5911	62.059	60.117	45.190	52.412	31.667
SCA	5.0000	5.0000	4.7289	5.0000	4.0975	50.297	65.000	44.426	30.000	30.000
SSA	3.8496	3.4900	3.4280	3.8565	3.3023	54.464	59.880	65.000	39.094	31.469
SCO	3.1112	3.9509	3.0933	2.9969	2.9502	60.263	54.414	52.483	58.841	52.404
AccOppCSCO	3.5488	3.6313	4.7691	2.8091	2.3535	61.381	57.347	49.780	44.947	46.371

Table 21
Cost & best run time results for Stepped Cantilever Beam Design problem.

Algorithm	Cost					Best run
	Best	Worst	Median	Mean	SD	time (s)
GA	9.48E + 19	9.48E + 19	9.48E + 19	9.48E + 19	23 805	5.83E − 2
PSO	9.48E + 19	9.48E + 19	9.48E + 19	9.48E + 19	12 212	3.35E − 2
DE	9.48E + 19	9.48E + 19	9.48E + 19	9.48E + 19	17 270	1.07E − 2
ABC	9.48E + 19	9.48E + 19	9.48E + 19	9.48E + 19	5461.3	3.93E − 2
GWO	9.48E + 19	9.48E + 19	9.48E + 19	9.48E + 19	7723.5	4.29E − 3
GSA	9.48E + 19	9.48E + 19	9.48E + 19	9.48E + 19	17 270	8.53E − 3
BBO	9.48E + 19	9.48E + 19	9.48E + 19	9.48E + 19	10 923	2.71E − 2
SCA	9.48E + 19	9.48E + 19	9.48E + 19	9.48E + 19	13 377	4.14E − 3
SSA	9.48E + 19	9.48E + 19	9.48E + 19	9.48E + 19	10 923	3.73E − 3
SCO	9.48E + 19	9.48E + 19	9.48E + 19	9.48E + 19	2.13E + 5	2.03E − 4
AccOppCSCO	9.48E + 19	9.48E + 19	9.48E + 19	9.48E + 19	21 845	1.93E − 3



(a) Multiple Disc Clutch Brake (b) Hydrodynamic Thrust Bearing

Fig. 15. Best convergence curves of AccOppCSCO and SCO algorithms for (a) Multiple Disc Clutch Brake and (b) Hydrodynamic Thrust Bearing design problems.

attempt to improve the global solution during optimization. However, AccOppCSCO has shown to be more effective in this regard. Table 22 summarizes the results of the MDCBD problem for the SOTA heuristic algorithms and proposed AccOpCSCO algorithm.

In terms of cost values, the AccOppCSCO algorithm ranked first with seven heuristics on all metrics. Nevertheless, the standard deviation values indicate that the ABC, GWO, and SSA algorithms exhibit greater stability. As was the case in the previous results, SCO ranked first and AccOppCSCO second in terms of running times.

Table 23 presents the results obtained by AccOppCSCO and the other heuristics for the HTBD problem. This problem consisted of four optimized design parameters. The AccOppCSCO algorithm demonstrated superior performance in terms of mean cost values compared to the SCO algorithm, ranking second among all the heuristics. In the best cost results, the AccOppCSCO algorithm ranked first with six heuristics. In terms of running time, the proposed algorithm ranked second after SCO.

Table 24 summarizes the results of AccOppCSCO and other SOTA heuristics for the engineering design problems. In this table, there are ranking results for the mean cost and best run-time. The average ranking scores of the AccOppCSCO algorithm are presented at the bottom of the table. The average ranking results indicate that AccOppCSCO has a ranking score of 3.5 among 11 algorithms in terms of mean cost and a ranking score of 2 in terms of best running time. It can be concluded that the new mechanisms added to the SCO algorithm increase the running time of AccOppCSCO, but also enable the algorithm to find better solutions than SCO. Furthermore, AccOppCSCO could compete with other population-based heuristics with shorter run times.

Table 22
Results for Multiple Disc Clutch Brake Design Problem.

Algorithm	Best solution					Cost					Best run
	x_1	x_2	x_3	x_4	x_5	Best	Worst	Median	Mean	SD	time (s)
GA	60.00	90.00	3.00	600.00	2.00	2.07E + 14	2.15E + 14	2.07E + 14	2.08E + 14	2.25E + 12	6.07E − 2
PSO	60.00	90.00	3.00	600.00	2.00	2.07E + 14	2.07E + 14	2.07E + 14	2.07E + 14	2.79E + 7	4.52E − 2
DE	60.00	90.00	3.00	600.00	2.00	2.07E + 14	2.07E + 14	2.07E + 14	2.07E + 14	1.77E + 4	1.57E − 2
ABC	60.00	90.00	3.00	600.00	2.00	2.07E + 14	2.07E + 14	2.07E + 14	2.07E + 14	3.29E − 2	5.89E − 2
GWO	60.00	90.00	3.00	600.00	2.00	2.07E + 14	2.07E + 14	2.07E + 14	2.07E + 14	3.29E − 2	6.33E − 3
GSA	63.18	99.91	1.63	633.06	2.17	3.24E + 14	4.97E + 14	4.28E + 14	4.20E + 14	6.31E + 13	1.30E − 2
BBO	60.00	90.00	2.78	600.00	2.00	2.07E + 14	2.62E + 14	2.10E + 14	2.21E + 14	1.84E + 13	2.22E − 2
SCA	60.00	90.00	3.00	600.00	2.00	2.07E + 14	2.07E + 14	2.07E + 14	2.07E + 14	9.02E + 5	5.42E − 3
SSA	60.00	90.00	3.00	600.00	2.00	2.07E + 14	2.07E + 14	2.07E + 14	2.07E + 14	3.29E − 2	5.21E − 3
SCO	60.38	90.34	2.84	600.09	2.00	2.09E + 14	4.42E + 14	2.35E + 14	2.68E + 14	7.73E + 13	3.37E − 4
AccOppCSCO	60.00	90.00	1.56	600.00	2.00	2.07E + 14	2.07E + 14	2.07E + 14	2.07E + 14	1.60E + 5	3.12E − 3

Table 23
Results for Hydrodynamic Thrust Bearing Design problem.

Algorithm	Best solution				Cost					Best run
	x_1	x_2	x_3	x_4	Best	Worst	Median	Mean	SD	time (s)
GA	2.76	2.76	1.00E + 06	16.00	1.50E + 14	1.05E + 49	6.95E + 41	2.58E + 48	4.26E + 48	4.27E − 02
PSO	1.00	1.00	1.00E + 06	16.00	2.37E + 11	2.54E + 47	2.37E + 11	2.54E + 46	8.04E + 46	3.50E − 02
DE	1.00	1.00	1.00E + 06	16.00	2.37E + 11	2.37E + 11	2.37E + 11	2.37E + 11	6.43E − 05	1.22E − 02
ABC	1.00	1.00	1.00E + 06	16.00	2.37E + 11	2.37E + 11	2.37E + 11	2.37E + 11	6.43E − 05	4.27E − 02
GWO	1.00	1.00	1.00E + 06	16.00	2.37E + 11	2.37E + 11	2.37E + 11	2.37E + 11	6.43E − 05	3.84E − 03
GSA	2.92	2.05	1.13E + 07	2.17	2.15E + 63	2.10E + 73	3.41E + 68	2.11E + 72	6.63E + 72	7.91E − 03
BBO	1.68	1.68	4.15E + 06	13.45	3.59E + 11	1.37E + 50	9.85E + 29	1.37E + 49	4.33E + 49	2.56E − 02
SCA	1.00	1.00	1.00E + 06	16.00	2.37E + 11	2.37E + 11	2.37E + 11	2.37E + 11	6.43E − 05	3.82E − 03
SSA	1.00	1.00	1.00E + 06	16.00	2.37E + 11	2.37E + 11	2.37E + 11	2.37E + 11	6.43E − 05	3.91E − 03
SCO	1.07	1.07	1.09E + 06	11.01	1.15E + 27	1.34E + 35	7.39E + 29	1.34E + 34	4.24E + 34	2.05E − 04
AccOppCSCO	1.00	1.00	1.14E + 06	2.37	2.37E + 11	2.74E + 12	2.74E + 12	2.49E + 12	7.91E + 11	1.79E − 03

Table 24
The ranking results of AccOppCSCO algorithm for Engineering Design problems.

No	Problem's name	Mean cost	Best run time (s)
1	Welded Beam Design	5	2
2	Compression Spring Design	5	2
3	Pressure Vessel Design	6	2
4	Speed Reducer Design	4	2
5	Gear Train Design	2	2
6	Himmelblaus Problem	4	2
7	Three Bar Truss Design	5	2
8	Stepped Cantilever Beam Design	1	2
9	Multiple Disc Clutch Brake Design	1	2
10	Hydrodynamic Thrust Bearing Design	2	2
	Average ranking	**3.5**	**2**

5. Conclusion

The AccOppCSCO algorithm presented in this study integrates an accelerated opposition-learning mechanism and chaotic mutation operators into the SCO algorithm. In addition, two novel mechanisms were incorporated into the update phases of the original SCO algorithm to enhance exploration and exploitation. A more effective boundary control method was also introduced to prevent the solutions from exceeding the search space limits. These modifications address the limitations of single candidate optimization structures while improving the efficacy of the SCO algorithm.

The search capability of the proposed AccOppCSCO algorithm is evaluated using four independent analyzes: convergence, search history, trajectory, and computational complexity. These, conducted using four types of two-dimensional CEC2014 benchmark problems, highlight the benefits of new mechanisms. The AccOppCSCO algorithm was further compared with the original SCO algorithm using 51 independent runs on the CEC2014 benchmarks with varying problem dimensions (10, 30, 50, and 100). The results summarized in Table 5 indicate that AccOppCSCO significantly outperformed SCO, particularly in the best (76.67% improvement) and median (68.33% improvement) metrics. Furthermore, compared to 12 state-of-the-art (SOTA) heuristics from the literature on 30-dimensional CEC2014 benchmarks, AccOppCSCO ranks first according to the mean metric.

The CEC2020 benchmark results reinforce the effectiveness of AccOppCSCO, demonstrating its superiority over SCO for multiple problem dimensions (5D, 10D, 15D, and 20D). Additionally, the performance of the algorithm was tested on ten engineering design problems, where it consistently achieved a better solution quality than the original SCO. Comparisons with 10 SOTA heuristics indicate that AccOppCSCO ranks third in the mean cost value and second in the best runtime. Although the computation time of AccOppCSCO is higher than that of the original SCO due to the additional mechanisms, its superior performance justifies the increase in the computational effort, making it competitive with population-based heuristics.

In general, the findings demonstrate that AccOppCSCO is an effective optimization algorithm that produces consistent and high-quality solutions to various optimization problems. Its new mechanisms contribute to improved stability and efficiency, positioning AccOppCSCO as a competitive approach in heuristic optimization.

Future research directions include investigating the performance of AccOppCSCO in large-scale and complex optimization problems. Further studies on parameter optimization and comparative evaluations across different problem types will provide valuable insights.

Additionally, exploring the potential of the algorithm in parallel and distributed computing environments, adapting it to real-time and dynamic optimization scenarios, and assessing its industrial applications are promising areas for further exploration. These efforts will help to maximize the impact of AccOppCSCO in optimization research and applications.

Declaration of competing interest

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

Data availability

Data will be made available on request.

References

[1] P. Adby, Introduction to Optimization Methods, Springer Science & Business Media, 2013.
[2] P. Pedregal, Introduction to Optimization, vol. 46, Springer, 2004.
[3] A. Antoniou, W.-S. Lu, Practical Optimization: Algorithms and Engineering Applications, vol. 19, Springer, 2007.
[4] D. Dvinskikh, A. Ogaltsov, A. Gasnikov, P. Dvurechensky, V. Spokoiny, On the line-search gradient methods for stochastic optimization, IFAC- Pap. 53 (2) (2020) 1715–1720.
[5] E. Gobet, M. Grangereau, Newton method for stochastic control problems, SIAM J. Control Optim. 60 (5) (2022) 2996–3025.
[6] S. Sun, Z. Cao, H. Zhu, J. Zhao, A survey of optimization methods from a machine learning perspective, IEEE Trans. Cybern. 50 (8) (2019) 3668–3681.
[7] A.R. Conn, K. Scheinberg, L.N. Vicente, Introduction to Derivative-Free Optimization, SIAM, 2009.
[8] L.M. Rios, N.V. Sahinidis, Derivative-free optimization: a review of algorithms and comparison of software implementations, J. Global Optim. 56 (3) (2013) 1247–1293.
[9] A. Samson, et al., Nonlinear programming: Theories and algorithms of some unconstrained optimization methods (steepest descent and Newton's method), Int. J. Eng. Manag. Res. E- ISSN 10 (2) (2020) 1–12.
[10] A. Dereventsov, C.G. Webster, J. Daws, An adaptive stochastic gradient-free approach for high-dimensional blackbox optimization, in: Proceedings of International Conference on Computational Intelligence: ICCI 2020, Springer, 2022, pp. 333–348.
[11] K. Rajwar, K. Deep, S. Das, An exhaustive review of the metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges, Artif. Intell. Rev. 56 (11) (2023) 13187–13257.
[12] R.W. Eglese, Simulated annealing: a tool for operational research, European J. Oper. Res. 46 (3) (1990) 271–281.
[13] V.K. Prajapati, M. Jain, L. Chouhan, Tabu search algorithm (TSA): A comprehensive survey, in: 2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things, ICETCE, IEEE, 2020, pp. 1–8.
[14] D. Simon, Evolutionary Optimization Algorithms, John Wiley & Sons, 2013.
[15] R. Sarker, M. Mohammadian, X. Yao, Evolutionary optimization, vol. 48, Springer Science & Business Media, 2002.
[16] E.A. Silver, An overview of heuristic solution methods, J. Oper. Res. Soc. 55 (2004) 936–956.
[17] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–73.
[18] S. Mirjalili, S. Mirjalili, Genetic algorithm, Evol. Algorithms Neural Networks: Theory Appl. (2019) 43–55.
[19] K.V. Price, R.M. Storn, J.A. Lampinen, The differential evolution algorithm, Differ. Evol.: A Pr. Approach Glob. Optim. (2005) 37–134.
[20] A.W. Mohamed, H.Z. Sabry, M. Khorshid, An alternative differential evolution algorithm for global optimization, J. Adv. Res. 3 (2) (2012) 149–165.
[21] S. Sridharan, R.K. Subramanian, A.K. Srirangan, Physics based meta heuristics in manufacturing, Mater. Today: Proc. 39 (2021) 805–811.
[22] R.A. Formato, Central force optimization: a new deterministic gradient-like optimization metaheuristic, Opsearch 46 (1) (2009) 25–51.
[23] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Inform. Sci. 179 (13) (2009) 2232–2248.
[24] H. Shah-Hosseini, The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm, Int. J. Bio- Inspired Comput. 1 (1–2) (2009) 71–79.
[25] O.K. Erol, I. Eksin, A new optimization method: big bang–big crunch, Adv. Eng. Softw. 37 (2) (2006) 106–111.
[26] M. Ghasemi, K. Golalipour, M. Zare, S. Mirjalili, P. Trojovskỳ, L. Abualigah, R. Hemmati, Flood algorithm (FLA): an efficient inspired meta-heuristic for engineering optimization, J. Supercomput. 80 (15) (2024) 22913–23017.
[27] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Computer- Aided Des. 43 (3) (2011) 303–315.
[28] X.-F. Xie, W.-J. Zhang, Z.-L. Yang, Social cognitive optimization for nonlinear programming problems, in: Proceedings. International Conference on Machine Learning and Cybernetics, vol. 2, IEEE, 2002, pp. 779–783.
[29] M. Kumar, A.J. Kulkarni, S.C. Satapathy, Socio evolution & learning optimization algorithm: A socio-inspired optimization methodology, Future Gener. Comput. Syst. 81 (2018) 252–272.
[30] S.O. Oladejo, S.O. Ekwe, S. Mirjalili, The hiking optimization algorithm: A novel human-based metaheuristic approach, Knowl.-Based Syst. 296 (2024) 111880.
[31] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, vol. 4, IEEE, 1995, pp. 1942–1948.
[32] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Appl. Math. Comput. 214 (1) (2009) 108–132.
[33] B. Akay, D. Karaboga, Artificial bee colony algorithm for large-scale problems and engineering design optimization, J. Intell. Manuf. 23 (2012) 1001–1014.
[34] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE Comput. Intell. Mag. 1 (4) (2006) 28–39.
[35] M. Dorigo, T. Stützle, Ant Colony Optimization: Overview and Recent Advances, Springer, 2019.
[36] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[37] X.-S. Yang, Firefly algorithms for multimodal optimization, in: International Symposium on Stochastic Algorithms, Springer, 2009, pp. 169–178.
[38] X.-S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature Inspired Cooperative Strategies for Optimization, NICSO 2010, Springer, 2010, pp. 65–74.
[39] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[40] W. Zhao, L. Wang, Z. Zhang, H. Fan, J. Zhang, S. Mirjalili, N. Khodadadi, Q. Cao, Electric eel foraging optimization: A new bio-inspired optimizer for engineering applications, Expert Syst. Appl. 238 (2024) 122200.
[41] M.H. Amiri, N. Mehrabi Hashjin, M. Montazeri, S. Mirjalili, N. Khodadadi, Hippopotamus optimization algorithm: a novel nature-inspired optimization algorithm, Sci. Rep. 14 (1) (2024) 5032.
[42] A. Ghiaskar, A. Amiri, S. Mirjalili, Polar fox optimization algorithm: a novel meta-heuristic algorithm, Neural Comput. Appl. 36 (33) (2024) 20983–21022.
[43] A. Prügel-Bennett, Benefits of a population: Five mechanisms that advantage population-based algorithms, IEEE Trans. Evol. Comput. 14 (4) (2010) 500–517.
[44] X.-S. Yang, Nature-inspired optimization algorithms: Challenges and open problems, J. Comput. Sci. 46 (2020) 101104.
[45] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[46] T.M. Shami, D. Grace, A. Burr, P.D. Mitchell, Single candidate optimizer: a novel optimization algorithm, Evol. Intell. 2022 (2022) 1–25.
[47] M.J. Al-Muhammed, R.A. Zitar, Probability-directed random search algorithm for unconstrained optimization problem, Appl. Soft Comput. 71 (2018) 165–182.
[48] H.R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in: International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Vol. 1, CIMCA-IAWTIC'06, IEEE, 2005, pp. 695–701.
[49] S.K. Dinkar, K. Deep, Accelerated opposition-based antlion optimizer with application to order reduction of linear time-invariant systems, Arab. J. Sci. Eng. 44 (2019) 2213–2241.
[50] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (2) (1999) 82–102.
[51] M.H. Nadimi-Shahraki, S. Taghian, S. Mirjalili, An improved grey wolf optimizer for solving engineering problems, Expert Syst. Appl. 166 (2021) 113917.
[52] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report 635 (2), Nanyang Technological University, Singapore, 2013, p. 2014.
[53] C. Yue, K. Price, P.N. Suganthan, J. Liang, M.Z. Ali, B. Qu, N.H. Awad, P.P. Biswas, Problem Definitions and Evaluation Criteria for The CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization, Tech. Rep. 201911, Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China, 2019.
[54] A. Sharma, A. Sharma, B.K. Panigrahi, D. Kiran, R. Kumar, Ageist spider monkey optimization algorithm, Swarm Evol. Comput. 28 (2016) 58–77.
[55] G.-G. Wang, S. Deb, A.H. Gandomi, Z. Zhang, A.H. Alavi, Chaotic cuckoo search, Soft Comput. 20 (9) (2016) 3349–3362.
[56] J. Pierezan, L.D.S. Coelho, Coyote optimization algorithm: a new metaheuristic for global optimization problems, in: 2018 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2018, pp. 1–8.

[57] S. Mirjalili, Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, Neural Comput. Appl. 27 (4) (2016) 1053–1073.
[58] M.A. Elhosseini, R.A. El Sehiemy, Y.I. Rashwan, X. Gao, On the performance improvement of elephant herding optimization algorithm, Knowl.-Based Syst. 166 (2019) 58–70.
[59] E.F. Veysari, et al., A new optimization algorithm inspired by the quest for the evolution of human society: human felicity algorithm, Expert Syst. Appl. 193 (2022) 116468.
[60] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, Knowl.-Based Syst. 89 (2015) 228–249.
[61] D. Połap, M. Woźniak, Red fox optimization algorithm, Expert Syst. Appl. 166 (2021) 114107.
[62] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowl.-Based Syst. 96 (2016) 120–133.
[63] G.-G. Wang, M. Lu, X.-J. Zhao, An improved bat algorithm with variable neighborhood search for global optimization, in: 2016 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2016, pp. 1773–1778.
[64] F. Zitouni, S. Harous, R. Maamri, The solar system algorithm: a novel metaheuristic method for global optimization, IEEE Access 9 (2020) 4542–4565.
[65] S. Barshandeh, M. Haghzadeh, A new hybrid chaotic atom search optimization based on tree-seed algorithm and Levy flight for solving optimization problems, Eng. Comput. 37 (4) (2021) 3079–3122.
[66] S.K. Azad, O. Hasançebi, O. Erol, Evaluating efficiency of big bang-big crunch algorithm in benchmark engineering optimization problems, Int. J. Optim. Civ. Eng 3 (2011) 495–505.
[67] M. Hammadi, S. Patalano, Automatic generation of simulation workflows for system verification using XDSM representation, in: 2017 IEEE International Systems Engineering Symposium, ISSE, IEEE, 2017, pp. 1–6.
[68] S. Gupta, K. Deep, H. Moayedi, L.K. Foong, A. Assad, Sine cosine grey wolf optimizer to solve engineering design problems, Eng. Comput. 37 (2021) 3123–3149.
[69] A.E. Yildirim, A. Karci, Application of three bar truss problem among engineering design optimization problems using artificial atom algorithm, in: 2018 International Conference on Artificial Intelligence and Data Processing, IDAP, IEEE, 2018, pp. 1–5.
[70] N. Singh, L.H. Son, F. Chiclana, J.-P. Magnot, A new fusion of salp swarm with sine cosine for optimization of non-linear functions, Eng. Comput. 36 (2020) 185–212.
[71] B.S. Yildiz, N. Pholdee, S. Bureerat, A.R. Yildiz, S.M. Sait, Enhanced grasshopper optimization algorithm using elite opposition-based learning for solving real-world engineering problems, Eng. Comput. 38 (5) (2022) 4207–4219.
[72] Y. Sha, C. Lu, W. Pan, S. Chen, P. Ge, Nonlinear control system design for active lubrication of hydrostatic thrust bearing, Coatings 10 (4) (2020) 426.
[73] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (6) (2008) 702–713.
[74] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191.