CHAOTIC MUTATION STRATEGY BASED SINGLE CANDIDATE OPTIMIZER ALGORITHM

Halil İbrahim EMEK
Bilecik Seyh Edebali University, Faculty of Engineering, Department of Computer Engineering, Bilecik, Türkiye.
ORCID ID: https://orcid.org/0000-0002-0836-6117
Prof. Dr. Uğur YÜZGEÇ
Bilecik Seyh Edebali University, Faculty of Engineering, Department of Computer Engineering, Bilecik, Türkiye.
ORCID ID: https://orcid.org/0000-0002-5364-6265

ABSTRACT

Heuristic search algorithms are widely used methods for solving complex optimization problems. These algorithms, usually based on swarm or population-based approaches, try to approach the best solution by moving multiple candidate solutions through the search space. However, these approaches have drawbacks such as requiring a large number of parameters, high computational cost, early convergence or loss of diversity.

In this paper, we consider the Single Candidate Optimizer (SCO) algorithm, a new heuristic search strategy that, unlike swarm-based methodologies, uses only a single candidate solution throughout the optimization. SCO uses a two-stage approach to update the position of the candidate solution. In the first stage, the candidate searches different regions of the search space based on its own knowledge. In the second stage, the candidate searches for the best solution in its region using local search methods. In this way, the SCO algorithm balances both exploration and exploitation capabilities.

The advantages of the SCO algorithm are simplicity, small number of parameters, low computational cost and high performance. However, the SCO algorithm also has some potential disadvantages. These are the limited exploration capability of a single candidate solution, the risk of getting caught in local optima, and the possibility of getting stuck in suboptimal regions. SCO's exploration mechanism can quickly lead the candidate solution to zero, but this can slow down SCO's convergence to the solution for problems with non-zero solutions. To overcome these drawbacks, various methods can be proposed to improve and refine the SCO algorithm.

In this study, a new mutation technique based on chaotic functions such as Chaucy, Gaussian and Levy is used to improve the performance of the SCO algorithm. This mutation operator enhances the exploration capability by allowing the candidate solution to jump to different regions of the search space while updating its position. The proposed Chaotic mutation strategy based Single Candidate Optimizer (CSCO) algorithm is compared with the original SCO algorithm for 23 different benchmark functions taken from the literature. The results show that different mutation techniques significantly improve the performance of the SCO algorithm.

Keywords: Heuristic, Mutation, Chaotic, Single Candidate Optimization.

INTRODUCTION

Heuristic algorithms play a vital role in solving complex problems by offering efficient and flexible approaches to manage complexity and find solutions. These algorithms optimize an objective function and aim to discover optimal solutions, drawing inspiration from natural processes or social interactions. Well-known heuristic algorithms include Particle Swarm Optimization (PSO) (Kennedy, 1995; Eberhart, 1995), Genetic Algorithm (GA) (Alhijawi & Awajan, 2023), Tabu Search Algorithm (Prajapati et al., 2020), Differential Evolution (DE) (Price et al., 2005), Harmony Search Algorithm (HS) (Yang, 2009), Simulated Annealing (SA) (Suman, B., & Kumar, 2006), Firefly Algorithm (FA)

(Yang & Slowik, 2020), Artificial Bee Colony (ABC) (Karaboga, 2014), Grey Wolf Optimizer (GWO) (Mirjalili et al, 2014), and Ant Colony Optimization (ACO) (Dorigo & Stützle, 2019). However, population-based heuristic algorithms like GWO and ABC may have certain drawbacks. These algorithms can be computationally expensive due to the maintenance and updating of a large pool of candidate solutions. Moreover, their performance heavily relies on the careful tuning of parameters, which can be a challenging task. Additionally, they may struggle to avoid local optima and cannot guarantee finding the global optimum in complex search spaces.

Single Candidate Optimizer (SCO) represents an innovative methodology that differs from traditional heuristic algorithms and especially from the applications of swarm-based algorithms. This approach, introduced by Shami (2022), differs from population-based algorithms by focusing on only one candidate solution during the full optimization process. The optimization process of SCO consists of two separate phases that follow different strategies to update the position of the candidate solution. This algorithm offers a robust solution in the field of heuristic optimization, integrating the advantages of single-solution algorithms and two-stage approaches. SCO is distinguished by employing a unique set of equations for updating candidate solution positions relative to their current state. Through the integration of these elements, SCO introduces a novel and effective approach to addressing optimization challenges.

SCO has certain advantages, but also some potential limitations that need to be taken into account. The fact that the algorithm only works on a single candidate solution may limit the comprehensive exploration of the large solution space. This lack of diversity may reduce the effectiveness of escape mechanisms when local minima are encountered and limit the algorithm's capacity to reach the global optimum. Besides, SCO's dependence on the position of the initial candidate may negatively affect the quality of the results. The tendency of SCO's exploration mechanism to rapidly converge the candidate solution’s position to zero can slow down the convergence process when the optimal solution is found at non-zero values. This is seen as a significant drawback that limits the efficiency and applicability of the algorithm.

Chaos theory is a central concept in the analysis of nonlinear dynamical systems that exhibit extreme sensitivity to small changes of initial conditions and thus produce unpredictable results (Liu, 2005). Such systems exhibit infinitely unstable periodic motions, resulting in deterministic yet unpredictable behaviour. The development of optimization algorithms and the study of chaotic systems are critical to the understanding of nonlinear phenomena, and work in this area has recently received a great deal of attention. A characteristic feature of chaotic systems is that minor variations in parameters or initial states can have radical effects on the long-term behaviour of the systems. This sensitivity makes the analysis and control of chaotic systems essential and diversifies their applications in nonlinear sciences.

This research presents an innovative methodology to improve the search capacity and overcome the existing limitations of the Single Candidate Optimizer (SCO). The proposed method is called Chaotic Mutation Strategy Based Single Candidate Optimization Algorithm (CSCO) and integrates chaotic mutation functions based on Gaussian, Cauchy and Levy distributions. To measure the effectiveness of the CSCO algorithm, tests were performed on 23 different benchmarks from the literature. Findings indicate that the CSCO algorithm demonstrates superior performance compared to the standard SCO algorithm.

METHODOLOGY

The Single Candidate Optimizer (SCO)

The Single Candidate Optimizer (SCO) is an innovative methodology that differs from traditional multiple candidate solution approaches by focusing on only one agent in the optimization process. Unlike population-based heuristic algorithms, the SCO algorithm aims to improve optimization performance by concentrating on a single solution. SCO divides the optimization process into two phases: exploration and exploitation. Various strategies and mechanisms are applied to create synergistic effects in both phases. The single-candidate approach and the two-stage process enable SCO to

efficiently leverage the advantages of each methodology, creating a flexible and powerful optimization tool. In the first phase of SCO, the candidate solution's position undergoes updating through the application of a predefined formula. This formula as given in Eq. (1) balances the algorithm's ability to explore and exploit, contributing to the overall optimization process.

$$x(j) = \begin{cases} S(j) + (w|S(j)|), if r_1 < 0.5 \\ S(j) - (w|S(j)|), otherwise \end{cases}$$

(1)

$$w(i) = e^{-\left(\frac{b \cdot i}{i_{max}}\right)^b}$$

(2)

In this context, x(j) is the candidate solution's position in the jth dimension, w denotes the weight parameter, S(j) is the best solution obtained at the end of each iteration, b represents a fixed parameter, i stands for the current iteration, i_{max} is the maximum number of iterations and r_1 is a random number between 0 and 1. In the 2nd stage of SCO, a methodical search is conducted in the immediate vicinity of the best location identified in the first stage. This in-depth exploration process covers a large part of the search space and thus scans a large cross-section of the potential solution space. In later stages, the search process is narrowed down, focusing on regions where the most fruitful results are expected, and these regions are examined in more detail. The schematic illustration below explains how the location update of the candidate solution is performed step by step during the second phase.

$$x(j) = \begin{cases} S(j) + \left(wr_2\left(UB(j) - LB(j)\right)\right), if r_2 < 0.5 \\ S(j) - \left(wr_2\left(UB(j) - LB(j)\right)\right), otherwise \end{cases}$$

(3)

In this equation, UB and LB represent the upper and lower boundaries, respectively, and r_2 is a random number that can take a value between 0 and 1. SCO algorithm has an adaptive behaviour of the weight parameter (w), which decreases exponentially with an increasing number of function evaluations. Here, this mechanism plays a critical role in ensuring a balanced transition between exploitation and exploration activities in the optimization process. Initially, the parameter w is set to a high value, allowing the SCO to perform an extensive exploration of the search space. As the optimization process progresses, decreasing w allows for the improvement of the solution and a focus on exploitation activities. During the second phase, SCO modifies the location update mechanism to minimize the risk of becoming trapped in local optima. If there is no improvement after m consecutive function evaluations, the position of the candidate solution undergoes a specific adjustment process as outlined below.

$$x(j) = \begin{cases} S(j) + \left(r_4\left(UB(j) - LB(j)\right)\right), if r_3 < 0.5 \\ S(j) - \left(r_5\left(UB(j) - LB(j)\right)\right), otherwise \end{cases}$$

(4)

In this case, $r_{3\text{-}5}$ represent randomly generated numbers. The updating equation (Eq. 4) is crucial in the SCO framework as it helps transition candidate solutions from exploitation to exploration during optimization. This transition improves the algorithm's ability to avoid local optima, where candidate solutions get stuck in suboptimal regions of the search space. By modifying the mechanism for updating positions, SCO can efficiently explore diverse regions of the search space. This mitigates the risk of being trapped in suboptimal solutions and ultimately leads to superior optimization outcomes.

Chaotic Mutation Strategy Based Single Candidate Optimizer (CSCO)

In order to overcome some limitations of the SCO algorithm and improve its overall performance, a new mutation operator based on Chaucy, Gaussian and Levy distributions, which exhibit chaotic behaviour, is proposed. This innovative mutation technique contributes to the global optimization process by increasing the exploration capacity of the algorithm. The proposed mutation operator makes it possible to avoid local minima by accessing larger regions of the search space while updating the positions of candidate solutions, thus enabling a more efficient scanning of the solution space. This approach allows the algorithm to maintain diversity and discover potential solutions in different regions of the search space, thus increasing the probability of reaching the global optimum.

In order to maximize the advantages of the SCO algorithm and increase the exploration capacity of the algorithm, the integration of a chaotic mutation structure is adopted as a strategic approach. When no improvement is observed during the m iterations of function evaluations, the chaotic mutation mechanism is activated, allowing the algorithm to avoid disadvantages such as getting stuck in local optima. This method aims to overcome the limitations of the original SCO algorithm by exploiting the random and non-deterministic nature of chaotic functions, especially during the exploitation phase of the algorithm. This has a positive impact on the overall performance of the algorithm and improves the efficiency of the optimization process. Algorithm 1 shows the pseudo code of CSCO algorithm.

Algorithm 1. *Pseudo-code of CSCO*

1. Initialize a random candidate solution (S) within the given bounds
2. Initialize the best fitness (BF) to the fitness of S
3. Set the counter for unsuccessful fitness improvements to 0
4. Define the number of unsuccessful attempts
5. For each iteration from 1 to maximum iteration:
  a. Adaptively change the b parameter
  b. Calculate the weight w(t) using the adaptive b parameter
  c. If there has been no improvement, increase the counter
  d. Generate a random number
  e. Calculate the exploration-exploitation parameter p1
  f. For each dimension j:
    i. Calculate the range 'EE'
    ii. Generate a random number p0
    iii. Calculate the mutation factor beta
    iv. If counter equals m, reset counter and apply a chaotic mutation based on a random selection
    v. Otherwise, apply a mutation based on the value of 'beta'
  end for
  g. Ensure the candidate solution x is within bounds
  h. Evaluate the fitness of x
  i. If the fitness of x is better than BF, update BF, S, and P
end for
6. Return the convergence curve, best fitness, best position

Benchmark Functions

In this study, the effectiveness of the proposed CSCO algorithm is evaluated in comparison with the original SCO algorithm on 23 benchmark problems that are widely accepted in the literature. All benchmarks are the single minimization problems. Figure 1 visualizes the problems selected for this comparison in two-dimensional space.

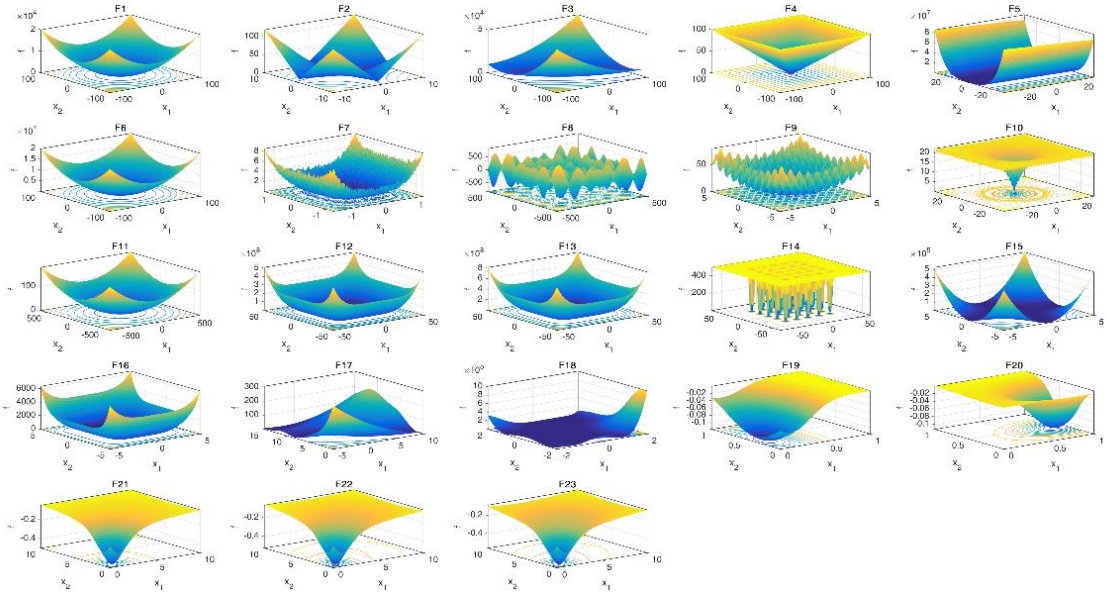

Figure 1. *Benchmark functions in 2D*

In the first step of the analysis process, two-dimensional analyses of certain functions were performed to evaluate the performance of both algorithms. In this context, four different benchmark problems numbered 14, 18, 21 and 23 were chosen as examples among the tested problems.

The analysis processes involve several methodologies, including search history analysis, trajectory analysis, fitness evolution analysis, and convergence analysis. Search history analysis tracks the evolution of candidate solutions in the search space throughout the optimization phase. Trajectory analysis describes the change over time in the first component of the candidate's position vector. Fitness evolution analysis observes variations in the candidate's fitness value over the optimization iterations. Convergence analysis evaluates the progression of candidate solutions towards the optimum. Figure 2 presents in detail the results of the comparative analysis for selected problems.

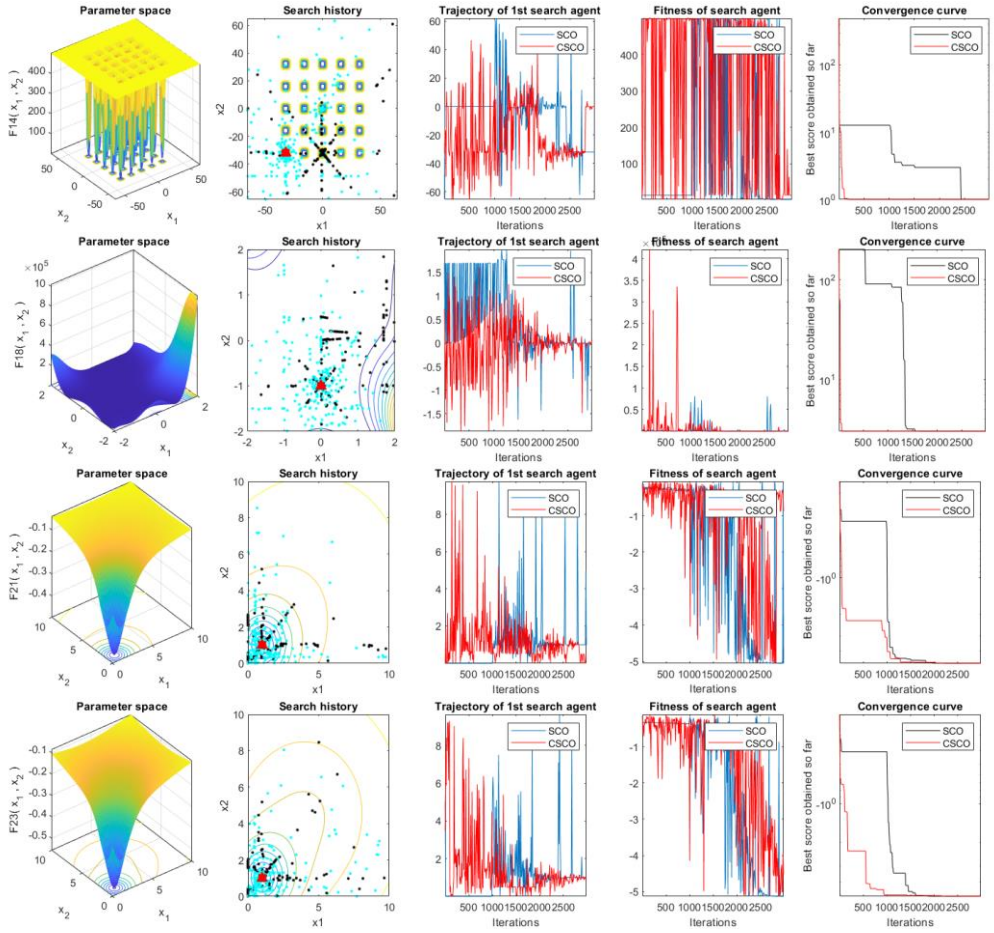

Figure 2. *Convergence Curves of SCO and CSCO*

RESULTS

After analysing four different problems, it was observed that the CSCO algorithm outperformed the original SCO algorithm. Specifically, when analysing the convergence curve of the 14th function, it was found that the SCO algorithm was unable to make progress during the first 1000 iterations of the exploration phase due to internal limitations. After the initial 1000 iterations, the algorithm became trapped at the local optimum and remained so until the 2400th iteration. Only after this point was it able to progress towards the global solution. In contrast, the CSCO algorithm overcomes these limitations of SCO and performs more efficiently by avoiding local optima and converging to the global solution. The convergence curve for the 18th function indicates that the SCO algorithm is limited internally, which negatively affects its performance. The SCO algorithm is known to produce fast results for problems with solutions at (0,0) for the 21st and 23rd functions. However, even in these cases, the CSCO algorithm outperforms it. The analysis indicates that the CSCO algorithm has a significant advantage over the original SCO algorithm, particularly in the exploration phase and in avoiding local optima. These results demonstrate that the CSCO algorithm is more efficient and effective in optimizing overall. The comparative analysis of SCO and CSCO algorithms 30 runs on 23 benchmark functions is summarized in Table 1.

Table 1. *Statistical results for 30 runs of the SCO and CSCO algorithms.*

Fn.	Original SCO					CSCO				
No	Best	Worst	Median	Mean	Std	Best	Worst	Median	Mean	Std
1	0	0	0	0	0	0	0	0	0	0
2	1.9E-307	2E-250	9.7E-276	1.3E-251	0	0	3E-181	1.8E-202	1E-182	0
3	0	3.5E-242	0	1.2E-243	0	0	0	0	0	0
4	5.5E-184	1.5E-35	9.4E-137	4.8E-37	2.7E-36	0	2.9E-171	2.2E-196	1.1E-172	0
5	29	29	29	29	0.11	29	29	29	29	0.097
6	0.15	0.65	0.21	0.25	0.11	2.4	5.9	4.2	4.2	0.86
7	2.0E-05	3.0E-03	3.1E-04	4.6E-04	5.4E-04	7.6E-05	1.0E-02	1.3E-03	1.9E-03	2.0E-03
8	-10000	-7300	-8200	-8300	700	-8600	-5800	-7400	-7300	740
9	0	0	0	0	0	0	0	0	0	0
10	8.9E-16	8.9E-16	8.9E-16	8.9E-16	4E-31	8.9E-16	8.9E-16	8.9E-16	8.9E-16	4E-31
11	0	0	0	0	0	0	0	0	0	0
12	0.02	0.26	0.054	0.073	0.051	0.096	0.35	0.21	0.21	0.069
13	0.33	3	1.5	1.5	0.7	1.8	3	2.9	2.8	0.24
14	1	2	1	1.2	0.4	1	1	1	1	4.5E-16
15	3.5E-04	1.2E-01	1.4E-03	7.1E-03	2.3E-02	3.5E-04	2.5E-03	8.7E-04	1.0E-03	5.6E-04
16	-1	-1	-1	-1	0	-1	-1	-1	-1	0
17	0.4	0.4	0.4	0.4	0	0.4	0.4	0.4	0.4	1.8E-05
18	3	3	3	3	0	3	3	3	3	0
19	-3.9	-3.9	-3.9	-3.9	0	-3.9	-3.9	-3.9	-3.9	4.3E-04
20	-3.3	-3.1	-3.3	-3.3	0.077	-3.3	-3.1	-3.3	-3.3	0.067
21	-10	-2.6	-10	-8.6	2.6	-10	-2.6	-5.1	-6.2	2.5
22	-10	-2.8	-10	-8.3	2.9	-10	-2.8	-5.1	-6.8	2.6
23	-11	-5.1	-11	-9.3	2.3	-11	-1.7	-5.1	-6.4	2.9

The empirical results indicate a pronounced enhancement in the performance of CSCO in contrast to SCO. Specifically, CSCO surpasses SCO by achieving superior best metric values in 17 out of the 23 functions, reflecting a 73.91% improvement. Similarly, in the context of the worst metric values, CSCO demonstrates a 56.52% enhancement with a success rate of 13/23. Additionally, CSCO's median metric performance is 56.52% more efficacious than SCO in 13 functions, and its mean metric performance is 43.48% more effective in 10 functions. Notably, CSCO also exhibits a more uniform and stable behaviour across the test functions, as evidenced by a 69.57% lower standard deviation compared to SCO. These observations collectively affirm that CSCO is more adept at optimizing test functions than SCO.

CONCLUSION

The paper introduces the Single Candidate Optimizer (SCO) algorithm, which differs from traditional swarm-based methods in that it uses only one candidate solution in the optimisation process. SCO uses a position update strategy with a two-stage method that balances between exploration and exploitation. Although SCO offers simplicity, few parameters, low computational cost and high performance, it may face some limitations such as limited exploration, stuck in local optima and sensitivity to sub-optimal regions. To address these issues, the study proposes a new mutation technique based on chaotic functions to enhance the exploration ability. The proposed CSCO algorithm is compared with SCO for 23 benchmark test sets. The results of the study show that different mutation techniques are effective in improving the optimisation capabilities of the SCO algorithm.

The results of the comparative study demonstrate the effectiveness of the CSCO algorithm as a robust optimization tool compared to the original SCO. The integration of chaotic mutation functions into CSCO has helped to achieve a balance between exploration and exploitation, resulting in superior optimisation results. Empirical evidence shows that CSCO outperforms SCO across various evaluative metrics, including Best, Worst, Median, and standard deviation measures. Additionally, CSCO demonstrates increased stability and reliability in addressing test functions, indicating its proficiency in navigating complex optimization landscapes. The algorithm's ability to systematically explore the search space while avoiding premature convergence makes CSCO a promising candidate for use in a wide range of real-world scenarios and complex optimization problems. Continuous research and iterative improvements are expected to further enhance CSCO's status as an essential optimization tool in various industries.

REFERENCES

Alhijawi, B., & Awajan, A. (2023). Genetic algorithms: Theory, genetic operators, solutions, and applications. Evolutionary Intelligence, 1-12.

Dorigo, M., & Stützle, T. (2019). Ant colony optimization: overview and recent advances (pp. 311-351). Springer International Publishing.

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artificial intelligence review, 42, 21-57.

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

Liu, B., Wang, L., Jin, Y. H., Tang, F., & Huang, D. X. (2006). Directing orbits of chaotic systems by particle swarm optimization. Chaos, Solitons & Fractals, 29(2), 454-461.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in engineering software, 69, 46-61.

Prajapati, V. K., Jain, M., & Chouhan, L. (2020, February). 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) (pp. 1-8). IEEE.

Price, K. V., Storn, R. M., & Lampinen, J. A. (2005). The differential evolution algorithm. Differential evolution: a practical approach to global optimization, 37-134.

Shami, T. M., Grace, D., Burr, A., & Mitchell, P. D. (2022). Single candidate optimizer: a novel optimization algorithm. Evolutionary Intelligence, 1-25.

Suman, B., & Kumar, P. (2006). A survey of simulated annealing as a tool for single and multi-objective optimization. Journal of the operational research society, 57, 1143-1160.

Yang, X. S. (2009). Harmony search as a metaheuristic algorithm. Music-inspired harmony search algorithm: theory and applications, 1-14.

Yang, X. S., & Slowik, A. (2020). Firefly algorithm. In Swarm intelligence algorithms (pp. 163-174). CRC Press.