APPLICATION OF CHAOTIC MUTATION STRATEGY BASED SINGLE CANDIDATE OPTIMIZER TO ENGINEERING DESIGN PROBLEMS

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 among the frequently used methods for solving complex optimization problems. These algorithms are usually based on swarm or population-based approaches and play an important role in solving complex problems by providing efficient and flexible approaches. These swarm-based approaches have disadvantages such as high computational cost, some of them have a large number of parameters, and early convergence problems.

Unlike swarm-based algorithms, the Single Candidate Optimizer (SCO) algorithm tries to reach the best solution by reducing the computational cost by using only a single candidate. The SCO algorithm is a simple, low-parameter, low-cost and high-performance heuristic algorithm. However, the SCO algorithm also has some disadvantages such as limited exploration ability of a single candidate solution, the risk of being caught in local optima and the possibility of getting stuck in suboptimal regions.

In this paper, we present a chaotic mutation strategy based Single Candidate Optimizer (CSCO) algorithm, which is constructed using a new mutation technique based on chaotic functions to overcome these drawbacks and improve the performance of the SCO algorithm.

In order to evaluate the performance of the proposed CSCO algorithm, some engineering design problems found in the literature are considered. Some of them are Welded Beam Design (WBD), Compression Spring Design (CSD) and Pressure Vessel Design (PVD). In addition to the original SCO algorithm, popular heuristic algorithms frequently used in the literature were used in the comparisons.

Keywords: Heuristic, Chaotic, Engineering Design Problems, CSCO

INTRODUCTION

Heuristic algorithms play a critical role in solving complex problems by providing efficient and flexible approaches. Inspired by natural processes or social interactions, these algorithms optimize the objective function and search for the optimal solution. Their applications range from engineering to business, from artificial intelligence to computer science. Heuristic algorithms fall into three main categories: evolutionary, swarm dynamics and based on physical principles. Evolutionary algorithms are based on Darwin's theories of natural selection and include Genetic Algorithm (GA) (Mirjalili, 2019), Differential Evolution (DE) (Price, 2013) and similar methods. Swarm-based algorithms are inspired by the collective actions of animal herds and include prominent examples such as Particle Swarm Optimization (PSO) (Wang, 2018), Artificial Bee Colony (ABC) (Karaboga, 2014), Cuckoo Calling (CS) (Song, 2020) algorithm, Bat Algorithm (BA) (Yang, 2013), Ant Colony Optimization (ACO) (Dorigo, 2019), Firefly Algorithm (FA) (Yang, 2020), Salp Swarm Algorithm (SSA) (Faris, 2020), Grey Wolf Optimization (GWO) (Mirjalili, 2014). Physics-based algorithms work by modelling the universal laws

of physics and include prominent examples such as Simulated Annealing (SA) (Delahaye, 2019), Electromagnetism-Like Search (EM) (Rocha, 2009), Gravitational Search Algorithm (GSA) (Rashedi, 2018).

The Single Candidate Optimization (SCO) algorithm proposes a new strategy as an alternative to traditional swarm-based heuristic search algorithms (Shami, 2022). Unlike swarm or population-based heuristic algorithms, SCO aims to produce superior solutions by working on only a single candidate solution throughout the entire optimization process. The optimization process consists of two separate phases that use different methodologies to update the position of the candidate solution. While single solution-oriented algorithms and two-phase approaches were developed as independent meta-heuristic optimization techniques, SCO combines these two concepts into an integrated and robust algorithm. One of the highlights of the algorithm is the use of a unique set of equations based on its current state to update the position of a candidate solution. By integrating these components, SCO provides an innovative and efficient solution to optimization problems.

Although the Single Candidate Optimization (SCO) algorithm offers certain advantages, its potential limitations need to be considered. SCO's focus on only a single candidate solution can limit its exploration capacity. In the absence of the diversity offered by multiple candidate solutions, SCO may struggle to efficiently scan the solution space and may become trapped in local optima. It may also tend to get stuck in suboptimal regions of the search space and struggle to detect the global optimum. The performance of SCO may vary depending on the initial location of the initial candidate solution. In addition, the SCO mechanism applied in the exploration phase can quickly lead the position of the candidate solution to zero, which can slow down the convergence process of the algorithm in problems with non-zero optimal solution points.

Chaos theory plays a central role in the study of nonlinear dynamical systems due to their extreme sensitivity to minor variations in initial conditions. As pointed out by Liu (2005), such systems tend to exhibit unstable periodic motions that, although deterministic, lead to unpredictable outcomes. The development of optimization algorithms and the detailed analysis of chaotic systems provide a better understanding of nonlinear phenomena, and the work in this field has received increasing attention. A fundamental property of chaotic systems is that small changes in parameters or initial states can have significant and radical effects on the long-term trajectories of the systems. This sensitivity makes the analysis and control of chaotic systems essential and contributes to the diversification of applications in nonlinear disciplines.

In this study, in order to overcome the known limitations of the Single Candidate Optimization (SCO) algorithm and improve its search performance, the effectiveness of the Chaotic Mutation Strategy Based Single Candidate Optimization (CSCO) algorithm, which is developed based on chaotic functions, is evaluated in comparison with both the original SCO algorithm and existing popular heuristic algorithms. Comparative analyses on engineering design problems show that the CSCO algorithm outperforms the original SCO algorithm. In addition, compared to other popular heuristic algorithms used in the comparison, the CSCO algorithm was found to have competitive results.

METADOLOGY

The Single Candidate Optimization (SCO) Algorithm

Single Candidate Optimization (SCO) is an innovative approach that differs from traditional multi-solution search algorithms used in optimization processes. Unlike swarm intelligence-based algorithms, SCO focuses on maximizing the potential of a single solution by concentrating on a single candidate solution throughout the optimization process. SCO's optimization process consists of two main phases that use different methods to update the position of the candidate solution. These two phases allow the integration of different strategies and mechanisms to improve both the exploration and exploitation capabilities of the algorithm. By combining the advantages of a single candidate solution with a two-stage strategy, SCO aims to exploit the strengths of both approaches (Shami, 2022). Algorithm 1 shows the pseudocode of the SCO algorithm.

Algorithm 1. *Pseudo-code of SCO*

1. Initialize a random candidate solution S within the bounds.
2. Evaluate the objective function at S to get the Best Fitness.
3. Initialize counter for unsuccessful fitness improvements to 0.
4. Set number of allowed unsuccessful attempts to improve fitness m to 5.
5. Set alpha function evaluation number in the first phase to 1000.
6. Set parameter for the weight function to 2.4.
7. For each iteration t from 1 to maximum iteration:
7.1. Calculate the weight w(t) using the exponential decay function.
7.2. If iteration is beyond alpha:
7.2.1. If there is no fitness improvement, increment the counter.
7.3. Generate a random number.
7.4. For each dimension j from 1 to dim:
7.4.1. Calculate EE using the weight and range.
7.4.2. If iteration is less than alpha:
7.4.2.1. Generate a new position x(j) by adding or subtracting a weighted absolute value of S(j).
7.4.3. Else:
7.4.3.1. If the counter equals m, reset the counter.
7.4.3.2. Generate a new position x(j) by addition or subtraction of a random value within the range.
7.4.3.3. Or, if the counter is not m, adjust x(j) by EE.
7.4.4. Ensure x(j) stays within bounds.
end for
7.5. Evaluate the fitness of the new position.
7.6. If the new fitness is better than Best Fitness:
7.6.1. Update Best Fitness, S, and P.
7.7. Return the convergence curve, best fitness, best position
end for
8. End Function.

Chaotic Mutation Strategy Based Single Candidate Optimizer (CSCO) Algorithm

To enhance the overall performance of the Single Candidate Optimization (SCO) algorithm and overcome its existing limitations, we propose an innovative mutation operator. This operator is based on Cauchy, Gaussian, and Levy distributions that exhibit chaotic behaviour. This advanced mutation technique contributes significantly to the global optimization process by enhancing the algorithm's exploration capability. The proposed mutation operator provides access to larger areas of the search space when updating the positions of candidate solutions, avoiding local minima and thus enabling a more efficient scanning of the solution space. This methodology enables the algorithm to preserve diversity and explore potential solutions in various regions of the search space, thereby increasing the likelihood of achieving the global optimum.

The mutation mechanism is activated during function evaluations when no improvement is observed during m iterations. This approach aims to prevent the algorithm from getting stuck at local optima and overcome the limitations of the original SCO algorithm by exploiting the random and non-deterministic nature of chaotic functions, especially in the exploration phase of the algorithm. The algorithm's overall performance is improved, and the optimization process is made more efficient. Algorithm 2 presents the pseudo code for the CSCO algorithm.

Algorithm 2. *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

Engineering Design Problems

In this section, the performance of the CSCO algorithm is tested for engineering design problems. Three test problems are taken from the literature, namely tensile/compression spring, welded beam, and pressure vessel, and the problems are shown in Figure 1. For each design problem, the comparison of the CSCO algorithm with the original SCO is presented first, followed by the comparison results of the CSCO algorithm with popular heuristics in the literature.

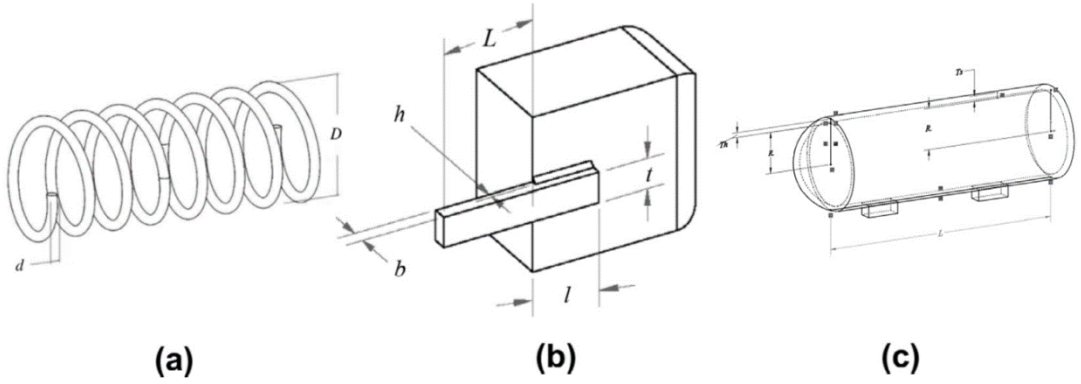

Figure 1. *Engineering Design Problems: (a) Tension/compression spring, (b) Welded beam, (c) Pressure vessel.*

Tension/Compression Spring Design

In the field of engineering optimization, one of the most common problems is the optimal design of coil springs, also known as tension/compression spring design (Arora, 2004). The optimization task aims to design a spring with the lowest possible mass that meets specific performance criteria and can support a given axial load without material failure (Tzanetos, 2023). The minimization procedure is constrained by several parameters, including the allowable shear stress, the threshold of surge frequency, and the requisite minimum deflection. The problem at hand involves three variables: wire diameter, mean coil diameter, and the number of active coils (Mirjalili, 2014).

Welded Beam Design

The objective of this problem is to minimize the fabrication cost of a welded beam. The problem involves constraints such as shear stress, bending stress, buckling load, end deflection, and side constraints. There are four variables to consider: weld thickness, length of the attached part of the bar, bar height, and bar thickness (Mirjalili, 2014).

Pressure Vessel Design

The objective of this problem is to minimise the total cost of a cylindrical vessel, which includes expenses for materials, forming, and welding. The vessel is capped at both ends, with the head having a

hemispherical shape. The problem involves four variables: the thickness of the shell, the thickness of the head, the inner radius, and the length of the cylindrical section without considering the head. This problem is constrained by four constraints (Mirjalili, 2014).

RESULTS

Comparison of the original SCO algorithm and the CSCO algorithm

The effectiveness of the CSCO algorithm was assessed by analysing three engineering design problems from literature: compression spring design problem, welded beam design problem, and pressure vessel design problem. The evaluation compared the performance of the original SCO algorithm to the enhanced CSCO version, as well as comparing the performance of CSCO to other commonly used heuristic algorithms. Each algorithm underwent 10 independent runs, with a maximum of 100 iterations per run. The best solutions obtained from these runs were then compared. Figure 2.a shows the comparison between the compression spring design problem solved using the both algorithms. Despite the initial disadvantage faced by the CSCO algorithm, both algorithms performed similarly, as shown by their closely matched convergence curves. Figure 2.b shows a comparison of the performance of the SCO and the CSCO for solving the welded beam design, with the CSCO outperforming the SCO. The disadvantages of the original SCO in the exploration phase affect its performance and cause a latency in convergence. Figure 2.c shows a comparison of the performance of the SCO and the CSCO to solve the pressure vessel design problem. The results show that the original SCO's internal limitations have a negative effect on its performance.

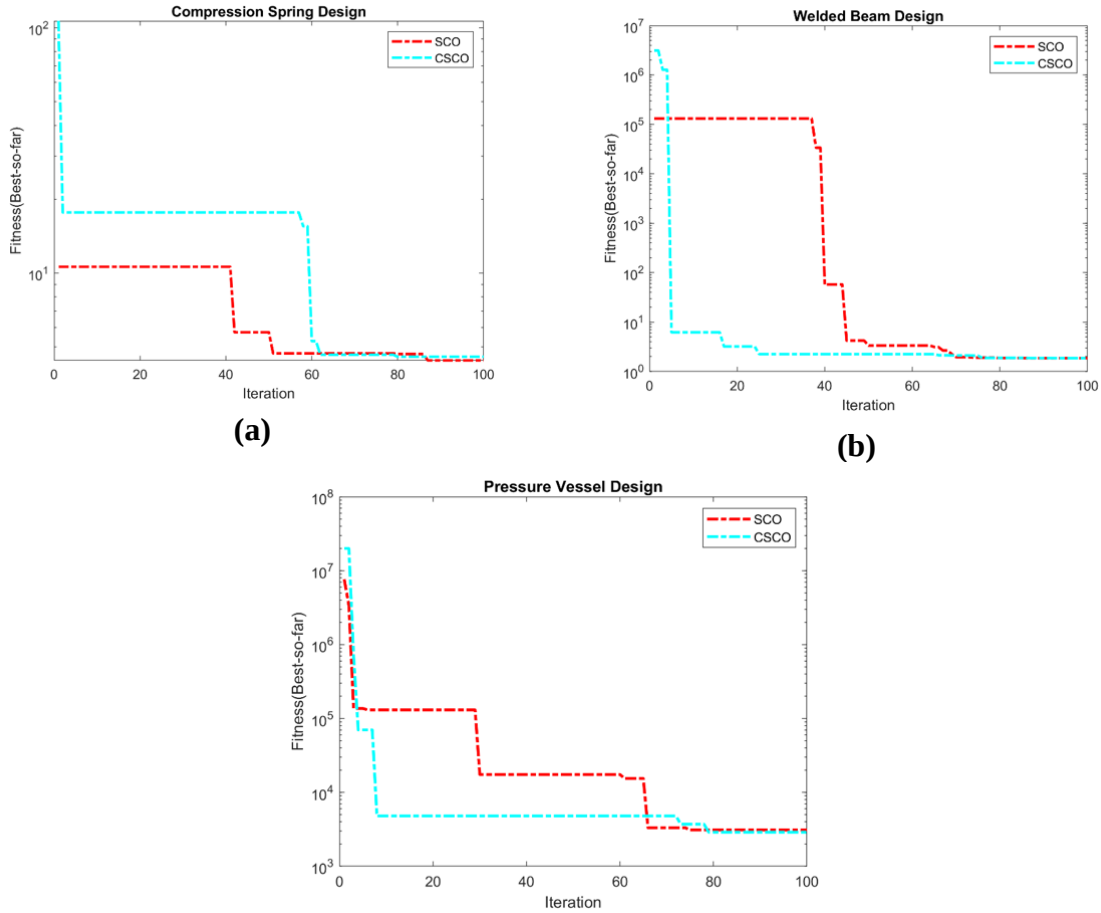

Figure 2. *Comparison of the original SCO and the CSCO for the engineering design problems: (a) Tension/compression spring, (b) Welded beam, (c) Pressure vessel.*

Comparison of the CSCO algorithm and frequently used heuristic algorithms

Secondly, we compare the proposed CSCO algorithm with popular heuristic algorithms for three engineering design problems. These are Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE) Algorithm, Artificial Bee Colony (ABC), Grey Wolf Optimization (GWO), Gravitational Search Algorithm (GSA), Biogeography-Based Optimization (BBO) Algorithm, Sine Cosine Algorithm (SCA), and Salp Swarm Algorithm (SSA) Algorithm. Table 1 presents the optimal outcomes achieved by each algorithm across ten runs for the compression spring design challenge. The algorithms PSO, DE, and GWO are discerned as more efficacious than their counterparts when evaluated against the best metric values. For the worst metric values, the GA algorithm emerges as the most proficient. The PSO algorithm is noted for its robust performance concerning the median metric values. In the context of mean metric values and standard deviation, the GWO algorithm's performance is unparalleled, surpassing other algorithms in these aspects. Table 2 summarises the best time results of CSCO and heuristics.

Table 1. *Statistical results of the CSCO and other heuristics for the compression spring design problem.*

Algorithm	Best	Worst	Median	Mean	Std
GA	3.79	20.14	5.91	7.48	4.96
PSO	3.66	6.83	3.68	4.00	1.00
DE	3.66	4.94	3.70	3.82	0.40
ABC	3.67	4.03	3.77	3.79	0.10
GWO	3.66	3.74	3.67	3.69	0.03
GSA	4.91	23.75	10.36	10.81	5.65
BBO	3.70	22.14	5.85	7.50	5.54
SCA	3.69	4.96	4.09	4.15	0.36
SSA	3.76	33.40	5.94	9.87	9.34
SCO	4.41	30.61	8.96	12.82	9.30
CSCO	4.56	41.20	14.38	15.66	10.84

Table 2. *Best time results of the CSCO and heuristics for the compression spring design problem.*

Algorithm	Best	Worst	Median	Mean	Std
GA	0.0616750	0.1359000	0.0918690	0.0927340	0.0249560
PSO	0.0499910	0.1049700	0.0527880	0.0596920	0.0170510
DE	0.0170310	0.0230530	0.0177430	0.0185030	0.0019633
ABC	0.0594420	0.0731790	0.0629050	0.0643450	0.0040185
GWO	0.0052232	0.0088239	0.0055446	0.0059139	0.0010696
GSA	0.0112360	0.0183640	0.0114550	0.0123780	0.0022117
BBO	0.0231790	0.0379550	0.0238750	0.0259630	0.0046562
SCA	0.0052480	0.0079362	0.0053892	0.0059739	0.0010726
SSA	0.0057431	0.0092176	0.0058695	0.0064807	0.0011114
SCO	0.0002431	0.0003984	0.0002483	0.0002654	0.0000472
CSCO	0.0016326	0.0026542	0.0017230	0.0018843	0.0003620

The comparative analysis presented in the second table elucidates the temporal efficiency of various algorithms in identifying the optimal solution. It is apparent that the SCO algorithm outperforms its counterparts in terms of speed, as demonstrated by its superior performance across all evaluated metrics. While the CSCO algorithm does not match the velocity of the SCO algorithm, it still surpasses other well-known heuristic algorithms in speed. This discrepancy can be attributed to the chaotic mutation

mechanism employed by the CSCO algorithm during both the exploration and exploitation stages, which inherently decelerates its convergence relative to the SCO algorithm.

Figure 5 shows a comparison between the CSCO algorithm and popular heuristic algorithms for solving the compression spring design problem. Although in this particular problem, popular heuristic algorithms outperformed the proposed CSCO algorithm, overall, the CSCO algorithm's performance is better than other popular heuristic algorithms.

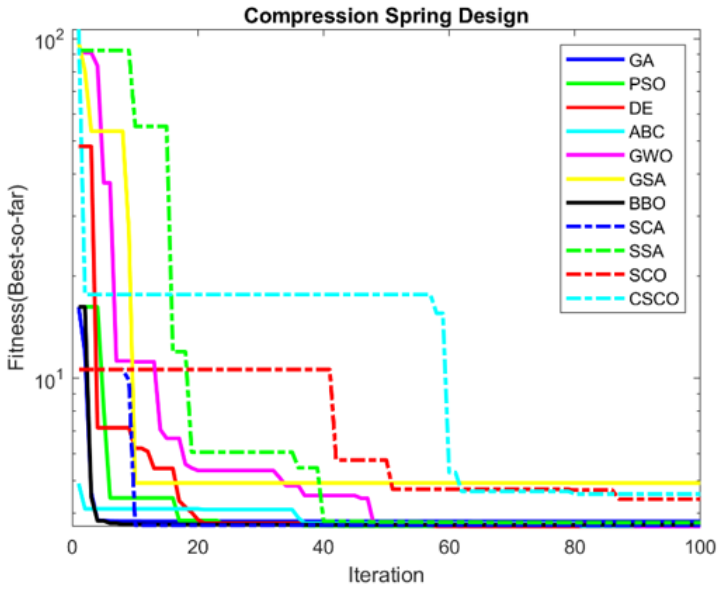

Figure 5. *Comparison of the CSCO and heuristics for the compression spring design problem.*

Upon examining the data in Table 3 pertaining to the welded beam design problem, it is observed that the PSO and GWO algorithms excel, as evidenced by the best metric values. The SCO and CSCO algorithms also demonstrate commendable performance, closely trailing the aforementioned algorithms. Notably, the GWO algorithm is distinguished as the most proficient in addressing this problem. Moreover, the efficacy of the CSCO algorithm is superior to most of the remaining algorithms under comparison.

The analysis of Table 4, which focuses on the resolution of the welded beam design problem, reveals the temporal efficiency of the algorithms. The SCO algorithm emerges as the most rapid across all metric values, significantly outpacing its counterparts. Subsequently, the CSCO algorithm, while not as swift as the SCO, still maintains a velocity advantage over other frequently used heuristic algorithms. Figure 6 compares the performance of the CSCO algorithm with popular heuristic algorithms for solving the welded beam design problem. The CSCO algorithm outperforms most of the popular heuristic algorithms.

Table 3. *Statistical results of the CSCO and heuristics for the welded beam design problem.*

Algorithm	Best	Worst	Median	Mean	Std
GA	2.07	3.70	3.19	3.09	0.54
PSO	1.71	2.52	1.82	1.88	0.24
DE	1.84	2.49	2.37	2.26	0.26
ABC	2.09	2.78	2.37	2.43	0.20
GWO	1.71	1.75	1.72	1.73	0.02
GSA	2.63	4.31	3.15	3.26	0.50
BBO	1.89	4.24	3.32	3.16	0.81
SCA	1.94	3.15	2.06	2.25	0.44

SSA	1.91	3.26	2.18	2.34	0.48
SCO	1.85	2818.40	2.84	284.86	890.21
CSCO	1.85	3.64	2.50	2.57	0.57

Table 4. *Best time results of the CSCO and heuristics for the welded beam design problem.*

Algorithm	Best	Worst	Median	Mean	Std
GA	0.0618960	0.1143400	0.0766410	0.0788180	0.0148760
PSO	0.0516930	0.0877810	0.0689900	0.0686000	0.0111100
DE	0.0173060	0.0297730	0.0179450	0.0198130	0.0040111
ABC	0.0624060	0.0794010	0.0682260	0.0687020	0.0048719
GWO	0.0059972	0.0103220	0.0062223	0.0072458	0.0017289
GSA	0.0118230	0.0198590	0.0127160	0.0135620	0.0024517
BBO	0.0265100	0.0480490	0.0303410	0.0341520	0.0078402
SCA	0.0055226	0.0099640	0.0063057	0.0070072	0.0016557
SSA	0.0060111	0.0097634	0.0065783	0.0071368	0.0013583
SCO	0.0002832	0.0038958	0.0003546	0.0007149	0.0011198
CSCO	0.0018357	0.0086371	0.0023083	0.0031494	0.0020645

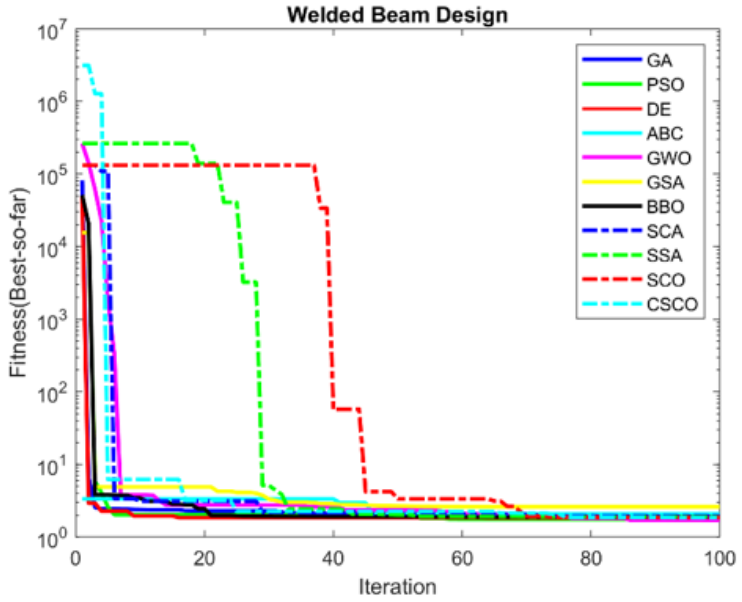

Figure 6. *Comparison of the CSCO and heuristics for the welded beam design problem.*

Table 5 shows the DE algorithm appears to be the most efficient and reliable among the evaluated algorithms, with the lowest Best and Mean metric values and a modest standard deviation. The GSA and BBO algorithms, while capable of finding good solutions, exhibit a high degree of variability and risk of poor performance. CSCO's results fall within a reasonable range, making it a viable option depending on the specific requirements and constraints of the optimization problem at hand.

When compared to other algorithms in the Table 6, the CSCO algorithm shows a balanced performance in terms of execution time. It does not achieve the best metric time, which is held by the SCO. However, CSCO is still among the faster algorithms. CSCO's performance is generally reliable, despite some variability. While CSCO has slightly higher variability than the leading algorithm SCO, it remains a competitive choice for pressure vessel design problem where execution time is critical.

Table 5. *Statistical results of the CSCO and heuristics for the pressure vessel design problem.*

Algorithm	Best	Worst	Median	Mean	Std
GA	2584.8	4905.5	3076.5	3327.5	700.7
PSO	2408.8	3638.7	3616.9	3426.8	395.6
DE	2302.5	3349.7	2309.4	2563.0	401.9
ABC	2403.0	3733.0	2552.7	2821.2	492.3
GWO	2309.1	3685.0	2372.6	2629.7	493.2
GSA	418280.0	1341700.0	819320.0	849760.0	344710.0
BBO	2697.8	168830.0	3533.0	19872.0	52338.0
SCA	2482.6	6335.9	4494.2	4424.8	1849.4
SSA	3494.0	4600.0	3923.9	3988.8	349.2
SCO	3086.5	29122.0	3944.2	9262.7	9207.7
CSCO	2887.8	7586.3	4023.1	4665.8	1583.5

Table 6. *Best time results of the CSCO and heuristics for the pressure vessel design problem.*

Algorithm	Best	Worst	Median	Mean	Std
GA	0.0616680	0.1252100	0.0812410	0.0857710	0.0216260
PSO	0.0505280	0.0592190	0.0519090	0.0531200	0.0028523
DE	0.0171420	0.0290100	0.0176450	0.0192780	0.0037015
ABC	0.0613900	0.0978440	0.0626500	0.0685590	0.0131080
GWO	0.0054781	0.0064982	0.0055442	0.0057175	0.0004056
GSA	0.0113780	0.0134050	0.0115310	0.0119310	0.0007819
BBO	0.0261480	0.0386830	0.0264590	0.0289530	0.0041730
SCA	0.0051390	0.0084059	0.0051807	0.0056852	0.0010373
SSA	0.0054838	0.0088132	0.0055446	0.0059871	0.0010466
SCO	0.0002098	0.0003442	0.0002180	0.0002398	0.0000437
CSCO	0.0016343	0.0026480	0.0016802	0.0018294	0.0003196

Figure 7 compares the performance of the CSCO algorithm with popular heuristic algorithms for solving the pressure vessel design problem. The performance of the CSCO algorithm is one of the best among given heuristic algorithms.

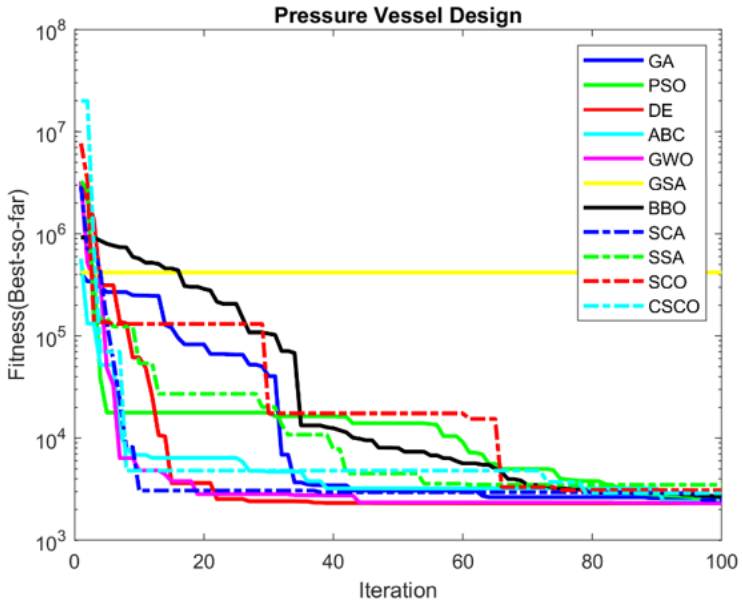

Figure 7. *Comparison of the CSCO and heuristics for the pressure vessel design problem.*

CONCLUSION

The article describes the inception of the Single Candidate Optimizer (SCO) algorithm, which differs from conventional swarm-based methods by relying on a single candidate solution for optimization efforts. The SCO algorithm relies on a bifurcated position update mechanism that judiciously allocates effort between exploration and exploitation phases. Despite its advantages, which include minimalism in design, a small number of parameters, reduced computational requirements, and robust performance, SCO is not without its drawbacks. These include a propensity for limited exploration, a susceptibility to getting trapped in local optima, and an increased sensitivity to suboptimal regions. In an effort to overcome these challenges, the research introduces an innovative mutation strategy, underpinned by chaotic functions, to significantly increase the algorithm's exploration capacity.

This study presents a comparative analysis of the Chaotic Mutation Strategy Based Single Candidate Optimizer (CSCO) algorithm and its predecessor, the SCO, along with established heuristic algorithms in the context of three engineering design problems. The empirical results show that the integration of different mutation methodologies significantly improves the optimization proficiency of the SCO algorithm. The performance of the CSCO algorithm is high compared to popular heuristic algorithms.

The CSCO algorithm's ability to systematically search the domain and avoid early convergence makes it a strong candidate for use in various practical environments and complex optimization problems. Ongoing research and improvements are expected to further enhance the CSCO's effectiveness as a key optimization tool across different sectors.

REFERENCES

Arora, J. S. (2004). Introduction to optimum design. Elsevier.

Delahaye, D., Chaimatanan, S., & Mongeau, M. (2019). Simulated annealing: From basics to applications. Handbook of metaheuristics, 1-35.

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

Faris, H., Mirjalili, S., Aljarah, I., Mafarja, M., & Heidari, A. A. (2020). Salp swarm algorithm: theory, literature review, and application in extreme learning machines. Nature-inspired optimizers: theories, literature reviews and applications, 185-199.

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

Mirjalili, S., & Mirjalili, S. (2019). Genetic algorithm. Evolutionary algorithms and neural networks: Theory and applications, 43-55.

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

Price, K. V. (2013). Differential evolution. In Handbook of optimization: From classical to modern approach (pp. 187-214). Berlin, Heidelberg: Springer Berlin Heidelberg.

Rashedi, E., Rashedi, E., & Nezamabadi-Pour, H. (2018). A comprehensive survey on gravitational search algorithm. Swarm and evolutionary computation, 41, 141-158.

Rocha, A. M. A., & Fernandes, E. M. (2009). Hybridizing the electromagnetism-like algorithm with descent search for solving engineering design problems. International Journal of Computer Mathematics, 86(10-11), 1932-1946.

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

Song, P. C., Pan, J. S., & Chu, S. C. (2020). A parallel compact cuckoo search algorithm for three-dimensional path planning. Applied Soft Computing, 94, 106443.

Tzanetos, A., & Blondin, M. (2023). A qualitative systematic review of metaheuristics applied to tension/compression spring design problem: Current situation, recommendations, and research direction. Engineering Applications of Artificial Intelligence, 118, 105521.

Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview. Soft computing, 22, 387-408.

Yang, X. S., & He, X. (2013). Bat algorithm: literature review and applications. International Journal of Bio-inspired computation, 5(3), 141-149.

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