




Article

An Enhanced Neural Network Algorithm with Quasi-Oppositional-Based and Chaotic Sine-Cosine Learning Strategies

Xuan Xiong ¹, Shaobo Li ^{2,*} and Fengbin Wu ^{2,*}

¹ State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China; gs.xxiong22@gzu.edu.cn

² State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China

* Correspondence: lishaobo@gzu.edu.cn (S.L.); wfaceboss@163.com (F.W.)

Abstract: Global optimization problems have been a research topic of great interest in various engineering applications among which neural network algorithm (NNA) is one of the most widely used methods. However, it is inevitable for neural network algorithms to plunge into poor local optima and convergence when tackling complex optimization problems. To overcome these problems, an improved neural network algorithm with quasi-oppositional-based and chaotic sine-cosine learning strategies is proposed, that speeds up convergence and avoids trapping in a local optimum. Firstly, quasi-oppositional-based learning facilitated the exploration and exploitation of the search space by the improved algorithm. Meanwhile, a new logistic chaotic sine-cosine learning strategy by integrating the logistic chaotic mapping and sine-cosine strategy enhances the ability that jumps out of the local optimum. Moreover, a dynamic tuning factor of piecewise linear chaotic mapping is utilized for the adjustment of the exploration space to improve the convergence performance. Finally, the validity and applicability of the proposed improved algorithm are evaluated by the challenging CEC 2017 function and three engineering optimization problems. The experimental comparative results of average, standard deviation, and Wilcoxon rank-sum tests reveal that the presented algorithm has excellent global optimality and convergence speed for most functions and engineering problems.

Keywords: neural network algorithm; quasi-oppositional-based learning; complex optimization; chaotic mapping; sine-cosine learning strategy; a new strategy; CEC 2017



Citation: Xiong, X.; Li, S.; Wu, F. An Enhanced Neural Network Algorithm with Quasi-Oppositional-Based and Chaotic Sine-Cosine Learning Strategies. *Entropy* **2023**, *25*, 1255. <https://doi.org/10.3390/e25091255>

Academic Editor: Fernando Morgado-Dias

Received: 30 July 2023

Revised: 17 August 2023

Accepted: 22 August 2023

Published: 24 August 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In contemporary practical applications, it is significant to imperative tackle a wide variety of optimization problems. These encompass the optimization of route planning [1,2], production scheduling [3,4], energy system [5], nonlinear programming [6], supply chain [7], facility layout [8], medical registration [9], and unmanned system [10], among others. These projects typically involve an enormous amount of information and constraints where conventional algorithms would struggle to find an optimal solution within a reasonable timeframe. Consequently, investigating efficient approaches to these intricate optimization processes has become an extremely challenging research domain. After relentless efforts, there are numerous optimization methods exploited by researchers, commonly employing deterministic and meta-heuristic approaches over intricate optimization issues.

Deterministic methods can be described as problem-solving approaches that rely on rigorous logic and mathematical models, effectively utilizing extensive gradient information to search for optimal or near-optimal solutions [11]. However, the strong dependence on the initial starting point makes it easy to produce identical results. In the real world, optimization problems are often highly intricate and exhibit nonlinear characteristics [12], which frequently involve multiple local optima within the objective function. Consequently,

deterministic methods often encounter difficulties in escaping local minima when dealing with complex optimization problems [13,14]. Instead, metaheuristics are inspired by phenomena observed in nature and simulate these phenomena to efficiently optimize and solve problems without relying on complex gradient information and mathematical principles thereby better exploring optimal solutions [15–17]. For instance, the grey wolf optimization (GWO) [18] replicates the social behavior of grey wolves during the search for optimization; the artificial immune algorithm (AIA) [19] mimics the evolutionary process of the human immune system to adaptively adjust the solution quality; the ant colony optimization (ACO) [20] emulates the pheromone-based foraging behavior of ants. It is noteworthy that the parameters of metaheuristic algorithms can be classified into two categories [21]: common parameters and special parameters. Common parameters encompass the foundational principles that govern the behavior of an algorithm, such as population size and termination criteria. On the other hand, specific parameters are tailored to the unique characteristics of individual algorithms. For instance, in simulated annealing (SA) [22] configuring the initial temperature and cooling rate is crucial for achieving optimal outcomes. Given the sensitivity of the algorithms for input data, any improper tuning of specific parameters may contribute to an augmented computational effort or the conundrum of local optimality when treating varying sorts of projects.

It is for heuristic algorithms featuring non-specific parameters that have gained immense relevance. Neural network algorithm (NNA) [23], which draws inspiration from artificial neural networks and biological nervous systems, emerged in 2018 as a promising method towards achieving globally optimal solutions. Additionally, a distinguishing trait of NNA from many famous heuristic algorithms is that it relies only on common parameters; hence, no extra parameters are required. This universality dramatically enhances its superior adaptability across a range of engineering applications. Nevertheless, NNA is confronted with two notable constraints: susceptibility to local optima and sluggish convergence speed. Therefore, a lot of improved optimization algorithms based on the scientific method have been offered to ameliorate the defects of NNA. For example, the competitive learning chaos neural network algorithm (CCLNNA) [24] is proposed by integrating NNA with competitive mechanisms and chaotic mapping; an effective hybrid algorithm TLNNA based on TLBO algorithm and NNA is proposed [25]; the gray wolf optimization neural network algorithm (GNNA) was created by combining GWO with NNA [26]; and the dropout strategy in the neural network was introduced and the elite selection strategy was proposed as a neural network algorithm with dropout using elite selection [27]. Moreover, by the no free lunch theorem [28], no one algorithm can be applied to all optimization questions. Thereby, it is fundamental for the ongoing refinement of existing to develop novel algorithms along with the integration of multiple algorithms for better results under practical applications. In this paper, the quasi-oppositional and chaotic sine-cosine neural network algorithm to boost the global search capability and refine the convergence performance of NNA is proposed. The main contributions of this work are listed below:

- To maintain the QOCSCNNA diversity of populations, a quasi-oppositional-based learning (QOBL) [29] is introduced, where quasi-opposite populations are randomly generated between the centers of solution space and opposite space, which contributes to better balance exploration and exploitation that make these populations closer to the most optimal ones more likely.
- By integrating logistic chaotic mapping [30] and sine-cosine strategy [31], a new logistic chaotic sine-cosine learning strategy (LCSC) is proposed that helps to escape from local optimum in the bias strategy phase.
- To improve the QOCSCNNA convergence performance, a dynamic tuning factor of piecewise linear chaotic mapping [32] is employed to adjust the chances of operation for the bias and transfer operators.

- The optimization performance of QOCSCNNA was verified through 29 numerical optimization problems based on the CEC 2017 test suite [33], as well as two real-world engineering constraint problems.

The remainder of this paper follows the following structure: a brief introduction of the original NNA is given in Section 2. Section 3 describes the proposed QOCSCNNA in detail. Section 4 validates the performance of the QOCSCNNA as well as explores the application of the QOCSCNNA to real-world engineering design problems using the CEC 2017 test suite. Finally, the main conclusions of this paper are summarized in Section 5 and further research directions are proposed.

2. NNA

Artificial neural networks (ANNs) are mathematical models that are based on the principles of biological neural networks, aiming to simulate the mechanisms of information processing in the human brain. ANNs are used for prediction primarily by receiving input data and output data which infer the relationship between these. The input data for ANN are typically obtained through experiments, computations, and other means, and the weights are iteratively adjusted to minimize the error between the predicted solution and the target solution, as shown in Figure 1. However, it might sometimes be unknown what the target solution is. Aiming to solve in this way, the authors of NNA treat the current best solution as the target solution and keep adjusting the weights of each neuron to achieve it. The NNA is a population-based evolutionary algorithm, which involves initializing the population, updating the weight matrices, and setting bias operators, and transferring operators.

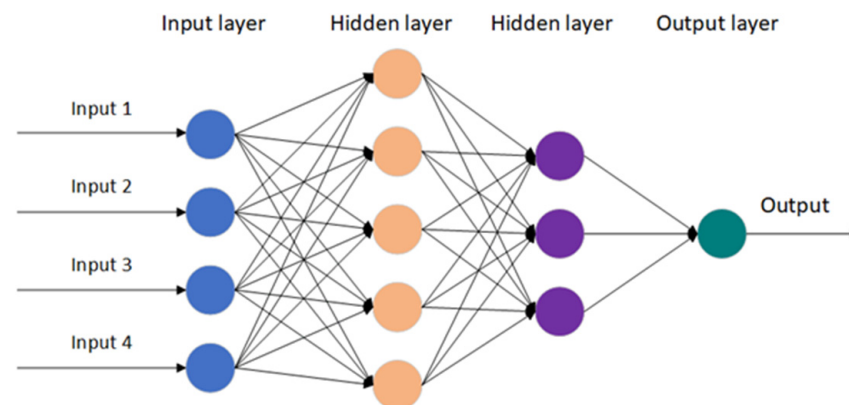


Figure 1. Structure of an artificial neural network.

2.1. Initial Population

In the NNA algorithm, the population is updated using a neural network model-like approach. In the search space, the initial population $X^r = [x_1^r, x_2^r, \dots, x_N^r]$ is updated through the weight matrix $W^r = [w_1^r, w_2^r, \dots, w_N^r]$, for any generation (r). Here, x_i^r represents the i^{th} individual vector and w_i^r represents the i^{th} weight vector, both with D dimensions. Thus, $x_i^r = [x_{i,1}^r, x_{i,2}^r, \dots, x_{i,D}^r]$ and $w_i^r = [w_{i,1}^r, w_{i,2}^r, \dots, w_{i,D}^r]$, where $i = 1, 2, \dots, N_p$.

It is desirable to impose constraints on the weights associated with new model solutions so that significant biases are prevented in the generation and transmission of these solutions. In this way, NNA was equipped to regulate its behavior through subtle deviations. After initializing the weights, the one corresponding to the desired solution (X_{target}), i.e., the target weight (W_{target}), that is chosen from the weight matrix W . Therefore, the summation of the weight matrix must adhere to the following conditions:

$$\sum_{j=1}^N w_{i,j}^r = 1, \quad i = 1, 2, \dots, N_p \quad (1)$$

where

$$w_{i,j} \in U[0,1], \quad i, j = 1, 2, \dots, N_p \tag{2}$$

In addition, the formula of generating a new population at the $(r + 1)^{th}$ iteration can be expressed by:

$$x_{i,new}^{r+1} = \sum_{i=1}^N w_{i,j}^r \times x_i^r, \quad i = 1, 2, \dots, N, \quad j = 1, 2, \dots, N_p \tag{3}$$

$$x_i^{r+1} = x_i^r + x_{i,new}^{r+1}, \quad i = 1, 2, \dots, N_p \tag{4}$$

where N_p is the population size, r is the current number of iterations, and $x_{i,new}^r$ is the weighted solution of the i^{th} individual at time r .

2.2. Update Weight Matrix

The weight matrix is then adjusted based on the desired target weight (W_{target}) using the following formula:

$$w_i^{r+1} = \left| w_i^r + 2 \times rand(0,1) \times (w_{target}^r - w_i^r) \right|, \quad i = 1, 2, \dots, N_p \tag{5}$$

where w_{target}^r is the vector of optimal target weights obtained in each iteration.

2.3. Bias Operator

To enhance the global search capability of NNA, a bias operator has been incorporated to fine-tune the probabilities of pattern solutions generated using the new population and updated weight matrices. A correction factor β is utilized to precisely define the probability of the adjusted pattern solution. Initially, β is initialized to 1 and progressively decreased in each iteration. The update process can be outlined as follows:

$$\beta^{r+1} = \beta^r \times 0.99, \quad r = 1, 2, \dots, T_{max} \tag{6}$$

The bias operator encompasses two components: the bias population and the bias weight matrix. To begin, a random number N_p and a set P are generated, where N_p is D multiplied by β^r . Let $L = (l_1, l_2, \dots, l_D)$ and $U = (u_1, u_2, \dots, u_D)$ be the lower and upper limits of the variables. Additionally, P denotes a set of N_p integers that are randomly selected from the range of 0 to D . Consequently, the definition of the bias population can be formulated as follows:

$$x_{i,P(s)}^r = l_{P(s)} + (u_{P(s)} - l_{P(s)}) \times \alpha_1, \quad s = 1, 2, \dots, N_p \tag{7}$$

where α_1 is a random number between 0 and 1 that obeys a uniform distribution. The bias weight matrix also involves two variables: a random number P_w , a stochastic number determined by the formula $N \times \beta^r$, and Q , a set of P_w integers randomly chosen between 0 and N . Therefore, the scientific representation for defining the bias weight matrix can be formulated as follows:

$$w_{i,Q(t)}^r = \alpha_2, \quad t = 1, 2, \dots, P_w \tag{8}$$

where α_2 is a random number between 0 and 1, following a uniform distribution.

2.4. Transfer Operator

There is an introduced transfer function operator (TF) that transfers the new mode solution at the current position to a new position in the search space proximal to the target solution (x_{target}^r). This operator can be denoted as:

$$x_i^{r+1} = x_i^r + 2 \times \alpha_3 \times (x_{target}^r - x_i^r), \quad i = 1, 2, \dots, N_p \tag{9}$$

where α_3 is a random number between 0 and 1 that follows a uniform distribution. Based on the above statements, the overall NNA framework can be seen in the pseudocode in Algorithm 1.

Algorithm 1: The pseudocode of the NNA algorithm

1. Initialize the population X^r and the weight matrix W^r .
2. Calculate the fitness value of each solution and then set X_{target} and W_{target}
3. for $i = 1 : N_p$
4. Generate the new solution x_i^r by Equation (3) and new weight matrix w_i^r by Equation (5)
5. if $\beta^r \geq \text{rand}$
6. Perform the bias operator for x_i^{r+1} by Equation (7) and the weight matrix w_i^{r+1} by Equation (8)
7. else
8. Perform the transfer function operator for x_i^r via Equation (9)
9. end if
10. end for
11. Generate the new modification factor β^{r+1} by Equation (6)
12. Calculate the fitness value of each solution and find the optimal solution and the optimal weight
13. Until(stop condition = false)
14. Post process results and visualization

3. Quasi-Optpositional-Based Chaotic Sine-Cosine Neural Network Algorithm

3.1. Quasi-Optpositional-Based Learning Strategy

Opposites-based learning (OBL) theory [34] has been proposed by Tizhoosh to synthesize the selection of existing solutions and their opposites to improve the quality of candidate solutions. The OBL strategy can provide more accurate candidate solutions. Moreover, the OBL theory evolved into quasi-oppositional-based learning (QOBL) approaches, which show a higher probability of approaching the unknown optimal solution compared to the candidate solutions generated by OBL in terms of achieving the global optimum [29]. To enhance the quality and convergence speed of the solutions, researchers integrated QOBL into metaheuristic methods.

The opposite point is the symmetric point of a given point concerning the center point in the solution space. Figure 2 shows the positions of the current point, the opposite point \tilde{X} , and the quasi-opposite \tilde{QH} within the one-dimensional space [A, B]. If $X = (x_1, x_2, \dots, x_n)$ represents a point in an n-dimensional space, where each coordinate $x_i \in [a_i, b_i]$ for $i = 1, 2, \dots, n$. The opposite point $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_n)$ corresponding to the generated X is as follows:

$$\tilde{x}_i = a_i + b_i - x_i \tag{10}$$

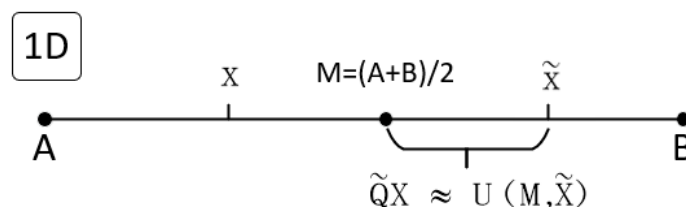


Figure 2. One-dimensional space opposites points and quasi-opposite points.

Furthermore, the quasi-opposite point $\tilde{QH} = (\tilde{q}x_1, \tilde{q}x_2, \dots, \tilde{q}x_n)$ is randomly generated between the inverse point and the center point $M = (A + B)/2$ of the solution space. The quasi-opposite point \tilde{QH} of \tilde{X} can be generated as follows [29]:

$$\tilde{q}x_i = \begin{cases} m_i + (\tilde{x}_i - m_i) \times k, & m_i < \tilde{x}_i \\ \tilde{x}_i + (m_i - \tilde{x}_i) \times k, & m_i > \tilde{x}_i \end{cases} \tag{11}$$

where k is a uniformly distributed random number between 0 and 1.

In this study, QOBL performs the initialization and generation of jumps for QOCSCNNA. The initialization phase through which randomly generated initial populations of quasi-opposite populations is created. The ridiculously generated initial population is taken to define the optimal solution of the inception phase; the generation jumping phase drives the algorithm jumps during the selection process to the solution with a better fitness function value. In this process, a greedy strategy is used to decide whether to keep the current solution or leap to a quasi-opposite solution. The pseudocode for the QOBL strategy is presented in Algorithm 2.

Algorithm 2: QOBL Strategy

```

for  $i = 1 : N_p$ 
  for  $j = 1 : N$ 
     $\tilde{x}_i = \alpha_i + \gamma_i - x_i$ 
     $m_i = (\alpha_i + \gamma_i) / 2$ 
    If  $m_i < \tilde{x}_i$ 
       $\tilde{q}\tilde{x}_i = m_i + (\tilde{x}_i - m_i) \times k$ 
    else
       $\tilde{q}\tilde{x}_i = \tilde{x}_i + (m_i - \tilde{x}_i) \times k$ 
    end if
  end for
end for

```

3.2. Chaotic Sine-Cosine Learning Strategy

3.2.1. Sine-Cosine Learning Strategy

For the performance improvement of meta-heuristic algorithms, Mirjalili introduced the sine-cosine learning strategy (SCLS) in his research [31]. It is the core idea of this strategy that the current solution is updated using the sine and cosine functions which effectively refrain the algorithm from falling into a local optimum. The definition of the algorithm is given below [31]:

$$x_{i,j}^{r+1} = \begin{cases} x_{i,j}^r + u_1 \times \sin(u_2) \times |u_3 \times x_{target}^r - x_{i,j}^r|, & u_4 < 0.5 \\ x_{i,j}^r + u_1 \times \cos(u_2) \times |u_3 \times x_{target}^r - x_{i,j}^r|, & u_4 \geq 0.5 \end{cases} \quad (12)$$

where r is the current iteration number; $x_{i,j}^r$ is the position of the i^{th} individual in the j^{th} dimension in the r iteration; and x_{target}^r is the optimal solution of the previous generation. u_2 is a range greater than 0 and less than $\sqrt{2}$ as the radius of the circles. u_3 is set to be a random number between 0 and 2, to control the distance of the optimal solution and maintain the diversity of the population. The value of u_4 is a random number between 0 and 1. u_1 is the cosine amplitude adjustment factor, set as follows:

$$u_1 = \frac{r}{R} \quad (13)$$

where R is the maximum number of iterations.

3.2.2. Logistic Chaos Mapping

The exploratory potential of chaos optimization algorithms can be further enhanced by leveraging the traversing traits and stochastic attributes of chaotic variables to optimize the diversity within the population [35]. In this work, the well-known logistic chaos mapping (LCM) was chosen to generate chaotic candidate solutions. Logistic mapping is formulated as follows [30]:

$$c_{z+1} = p \times c_z \times (1 - c_z) \quad (14)$$

where $c_z \in (0, 1) \forall z \in \{0, 1, \dots, N_p\}$ and $p = 3.8$.

Based on the candidate solutions generated by the logistic chaos, the generation is as follows:

$$\delta_z = l_z + (u_z - l_z) \times c_z \quad (15)$$

In biased operators, a novel strategy, i.e., logistic chaotic sine-cosine learning strategy (LCSC), is generated by integrating the LCM with the SCLS to align the candidate solutions to be that more chaotic to explore the design space. This mechanism serves as a preventive measure against premature convergence in subsequent iterations. The new solution is generated as follows:

$$x_{i,j}^{r+1} = \begin{cases} \delta_z + u_1 \times \sin(u_2) \times |u_3 \times c_z - x_{i,j}^r|, & u_4 < 0.5 \\ \delta_z + u_1 \times \cos(u_2) \times |u_3 \times c_z - x_{i,j}^r|, & u_4 \geq 0.5 \end{cases} \quad (16)$$

where r is the current iteration number; $x_{i,j}^r$ is the position of the i^{th} individual in the r^{th} iteration. u_1 is the positive cosine amplitude adjustment factor, defined as shown in Equation (12). u_2 is set to be more than 0 and smaller than a circle with a radius of $\sqrt{2}$, u_3 is set to be a random number between 0 and 2, and u_4 is set to be a random number between 0 and 1. The pseudocode of the bias operator changed by the CSCL is given in Algorithm 3.

Algorithm 3: The Bias Operator

P_n signifies the number of biased variables in the population of the new pattern solution

P_w signifies the number of biased variables in the updated weight matrix

for $i = 1: N_p$

 if $\text{rand} \leq \beta$

 %Bias for new pattern solution %

$P_n = \text{round}(N \times \beta)$

 Update the chaotic sequence δ_z using Equation (14)

 for $j = 1: P_n$

 Update the new pattern solution $x_{i, \text{Integer rand}[0, N]}^{r+1}$ by Equation (16)

 end for

 %Bias for updated weight %

$P_w = \text{round}(N_p \times \beta)$

 for $j = 1: P_w$

 Update the weight $w_{j, \text{Integer rand}[0, N_p]}^r$ by Equation (8)

 end for

 end if

end for

3.3. The Dynamic Tuning Factor

Since the bias operator decreases as the number of iterations increases, a piecewise linear chaotic map (PWLCM) [32] is introduced, for which the chances of running different learning strategies are dynamically tuned to help QOCSCNNA converge faster as more iterations are added. As well, the definition of PWLCM is denoted in Equation (17):

$$Z_{r+1} = \begin{cases} Z_r/k, & Z_r \in (0, k) \\ (1 - Z_r)/(1 - k), & Z_r \in [k, 1) \end{cases} \quad (17)$$

where r represents the function mapping value for the r^{th} iteration; k is a control parameter, with k between 0 and 1.

In this study, the improved algorithm by fusing the original NNA with the bias operators of QOBL, LCSC strategy, and PWLCM factor is called QOCSCNNA. The detailed flowchart as shown in Figure 3 and the pseudocode for QOCSCNNA can be found in Algorithm 4.

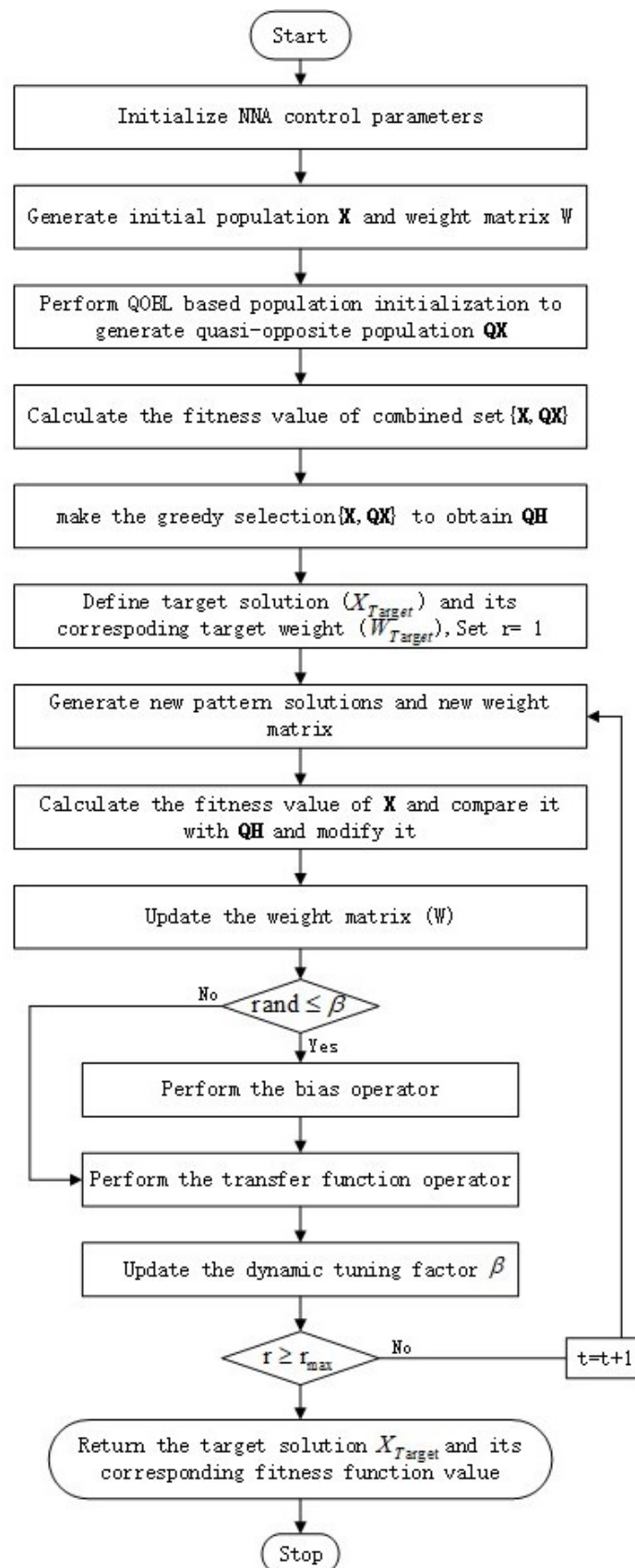


Figure 3. Flowchart of QOCSCNNA.

Algorithm 4: QOCSCNNA algorithm

1. Initialize the number of iterations r ($r = 1$), the dynamic tuning factor β^r ($\beta^1 = 1$)
2. Randomly generate an initial population X
3. Generate quasi-opposite solutions \widetilde{QH}_i using algorithm 2
4. Calculate the fitness value of combined set $\{X, \widetilde{QH}\}$ then make the greedy selection to obtain \widetilde{QH}
5. Randomly generate the weight matrix considering the imposed constraints in Equations (1) and (2)
6. Set the optimal solution x_{target}^r and the optimal weight w_{target}^r
7. While $r < r_{max}$
8. Generate new pattern solution x_i^r by Equations (3)–(4), new weight matrix w_i^r by Equation (5)
9. Calculate the fitness value of X
10. if $\text{fit}(\widetilde{QH}_i) < \text{fit}(X_i)$
11. $X_i = \widetilde{QH}_i$
12. $\text{fit}(X_i) = \text{fit}(\widetilde{QH})$
13. end if
14. if $\text{rand} \leq \beta^r$
15. Perform the bias operator using algorithm 3
16. else
17. Perform the transfer function operator for x_i^r via Equation (7)
18. end if
19. Calculate the fitness value of each solution and find the optimal solution x_{target}^{r+1} and the optimal weight w_{target}^{r+1}
20. Update the current number of iterations by $r = r + 1$
21. Update the dynamic tuning factor β^{r+1} by Equation (17)
22. End while

4. Numerical Experiments and Result Analysis

This section examines the properties of the proposed QOCSCNNA numerical optimization problems. This chapter is divided into three subsections. Section 4.1 details the CEC 2017 test function and the experimental environment that ensures the reliability of the experimental results. Section 4.2 provides a comparative analysis between QOCSCNNA and eight other metaheuristics on the CEC 2017 function which validates the effectiveness of the improved algorithms. Finally, the performance of the algorithm is compared with other algorithms through three engineering projects of practical significance in Section 4.3.

4.1. Experiment Setup

It is a broadly used CEC 2017 test suite [33] specifically dedicated to evaluating the performance of complex optimization algorithms. The test suite consists of 30 test functions covering a wide range of test requirements to obtain a more comprehensive insight into the performance characteristics of optimization algorithms. Unfortunately, for unavoidable reasons, the F2 test functions could not be tested, resulting in only 29 functions being tested. These functions could be categorized into four types, each with diverse levels of complexity and characteristics. Firstly, there are the single-peaked functions (F1,F3), which have a clear optimal solution and are suitable for assessing the behavior of the algorithm when dealing with simple problems. Secondly, there are simple multimodal functions (F4–F9), which have multiple partial optimal solutions and can be used to test the robustness and convergence of the algorithm during local search. The third category is hybrid functions (F11–F20), which combine the characteristics of single-peak and multimodal and are closer to the situation of complex problems in reality, enabling a comprehensive assessment of the overall global and local search capability of algorithms. Finally, the synthesized functions (F21–F30) are combined with other functions. The specific functions are shown in Table 1.

Table 1. The definition of the CEC2017 test suite.

	No.	Function	Optimum
Unimodal function	F1	Shifted and Rotated Bent Cigar Function	100
	F3	Shifted and Rotated Zakharov Function	300
Simple multimodal function	F4	Shifted and Rotated Rosenbrock's Function	400
	F4	Shifted and Rotated Rastrigin's Function	500
	F6	Shifted and Rotated Expanded Scaffer's F6 Function	600
	F7	Shifted and Rotated Lunacek BiRastriginFunction	700
	F8	Shifted and Rotated Non-Continuous Rastrigin's Function	800
	F9	Shifted and Rotated Levy Function	900
	F10	Shifted and Rotated Schwefel's Function	1000
Hybrid function (HF)	F11	Hybrid Function 1 (N = 3)	1100
	F12	Hybrid Function 2 (N = 3)	1200
	F13	Hybrid Function 3 (N = 3)	1300
	F14	Hybrid Function 4 (N = 4)	1400
	F15	Hybrid Function 5 (N = 4)	1500
	F16	Hybrid Function 6 (N = 4)	1600
	F17	Hybrid Function 6 (N = 5)	1700
Hybrid function (HF)	F18	Hybrid Function 6 (N = 5)	1800
	F19	Hybrid Function 6 (N = 5)	1900
Composition function (CF)	F20	Hybrid Function 6 (N = 6)	2000
	F21	Composition Function 1 (N = 3)	2100
	F22	Composition Function 2 (N = 3)	2200
	F23	Composition Function 3 (N = 4)	2300
	F24	Composition Function 4 (N = 4)	2400
	F25	Composition Function 5 (N = 5)	2500
	F26	Composition Function 6 (N = 5)	2600
	F27	Composition Function 7 (N = 6)	2700
	F28	Composition Function 8 (N = 6)	2800
	F29	Composition Function 9 (N = 3)	2900
F30	Composition Function 10 (N = 3)	3000	

Search range : $[-100, 100]^D$ (D is the population dimension)

Furthermore, it was necessary to place all algorithms under the same test conditions to ensure fairness, and experiments were conducted using MATLAB R2022a software under MacOS 12.3 M1. In the CEC 2017 suite, the population size was set to 50 and the dimensionality was set to 10 D. To fully evaluate the performance of the algorithms, the maximum number of function evaluations was set to 20,000 times the population size. This setup ensures a thorough exploration of the search space, thus improving the optimization results. It is noted that the other parameters required to compare the algorithms were extracted directly from the original references to keep the consistency of the results. Moreover, there were 30 independent runs of each algorithm execution to get reliable results, and the average value (AVG) and standard deviation (STD) of the obtained results were logged.

4.2. QOCSCNNA for Unconstrained Benchmark Functions

To evaluate the performance of the improved algorithm, QOCSCNNA was compared with eight other well-known optimization algorithms, including NNA, CSO [36], SA [22], HHO [37], WOA [38], SCA [31], WDE [39], and RSA [40]. Based on the experimental settings outlined in Section 4.1, the average (AVG) and standard deviation (STD) of the minimum fitness values obtained on the CEC 2017 benchmark functions are presented in Table A1, with the smallest average and standard deviation highlighted in bold. When compared to other algorithms, QOCSCNNA demonstrated significant superiority in terms of both AVG and STD results in the 2017 CEC functions. Moreover, given the limited evaluation budget, the QOCSCNNA algorithm had relatively minor means and standard deviations for a range of functions including F1, F4, F5, F7, F8, F10-F17, F19-F21, F27, F29, and F30. These results highlight QOCSCNNA's superior ability to effectively tackle optimization problems characterized by complexity and hybridity.

The results of the Wilcoxon rank-sum test (“+”, “=”, and “−” indicate that QOCSCNNA performs better, the same, or worse, respectively, compared to the other algorithms) are shown in Table A1 to better compare the performance of the different algorithms. As can be seen in the last row of Table A2, QOCSCNNA achieved significantly superior results to SA, SCA, RSA, and CSO on more than 28 test functions, while QOCSCNNA beats HHO and WOA for more than 26 functions and exceeds WDE and NNA for 23 functions. In other words, the average superiority rate of QOCSCNNA over 29 functions is 92.24% ($\sum_{i=1}^8 \frac{+1}{29 \times 8} \times 100\%$). These results indicate that adopting the CSCL can effectively improve the optimization capability of NNA.

Nine convergence plots of QOCSCNNA with the comparison algorithm on the CEC 2017 test set including F1, F8, F10, F12, F16, F21, F24, F29, and F30 are given in Figure 4, where the vertical axis takes the logarithm of the function’s minimum value, and the horizontal axis denotes the number of times the function was evaluated. It can be noticed that although sometimes QOCSCNNA does not perform the best in the initial phase, as the number of function iterations increases, smaller fitness values can be searched for by constantly jumping out of the local optimum. The good performance of this algorithm is because the exploration of QOBL enhances the global search capability.

4.3. Real-World Engineering Design Problems

Furthermore, to validate the feasibility of the QOCSCNNA for actual engineering applications, multiple algorithms were utilized to address the critical engineering design problems of cantilever beam structures (CB) [41], car side impact (CSI) [41], and tension spring (TS) [41]. For three problems, a population size of 50 was set with an iteration count of 2000 times the population size. Moreover, each algorithm was independently run 30 times to obtain reliable results. Such settings ensured thorough exploration of the search space, leading to improved optimization results. Additionally, the solution provided by QOCSCNNA was compared to well-known algorithms to better evaluate its performance.

4.3.1. CB Engineering Design Problem

The weight optimization of a square cross-section cantilever beam is involved in the CB structural engineering design. The beam has a rigid support at one extremity, while vertical forces act on the free nodes of the cantilever. A model of the CB design problem is illustrated in Figure 5. The beam consists of five hollow squares of equal thickness, with the height (or width) of each square being the decision variable. Meanwhile, the thickness of these squares remains constant at 2/3. The objective function of this design problem can be represented by Equation (18).

$$F(x)_{min} = 0.0624 (x_1 + x_2 + x_3 + x_4 + x_5) \quad (18)$$

Subject to:

$$G(x) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0 \quad (19)$$

Variable range:

$$0.01 \leq x_i \leq 100, i = 1, \dots, 5. \quad (20)$$

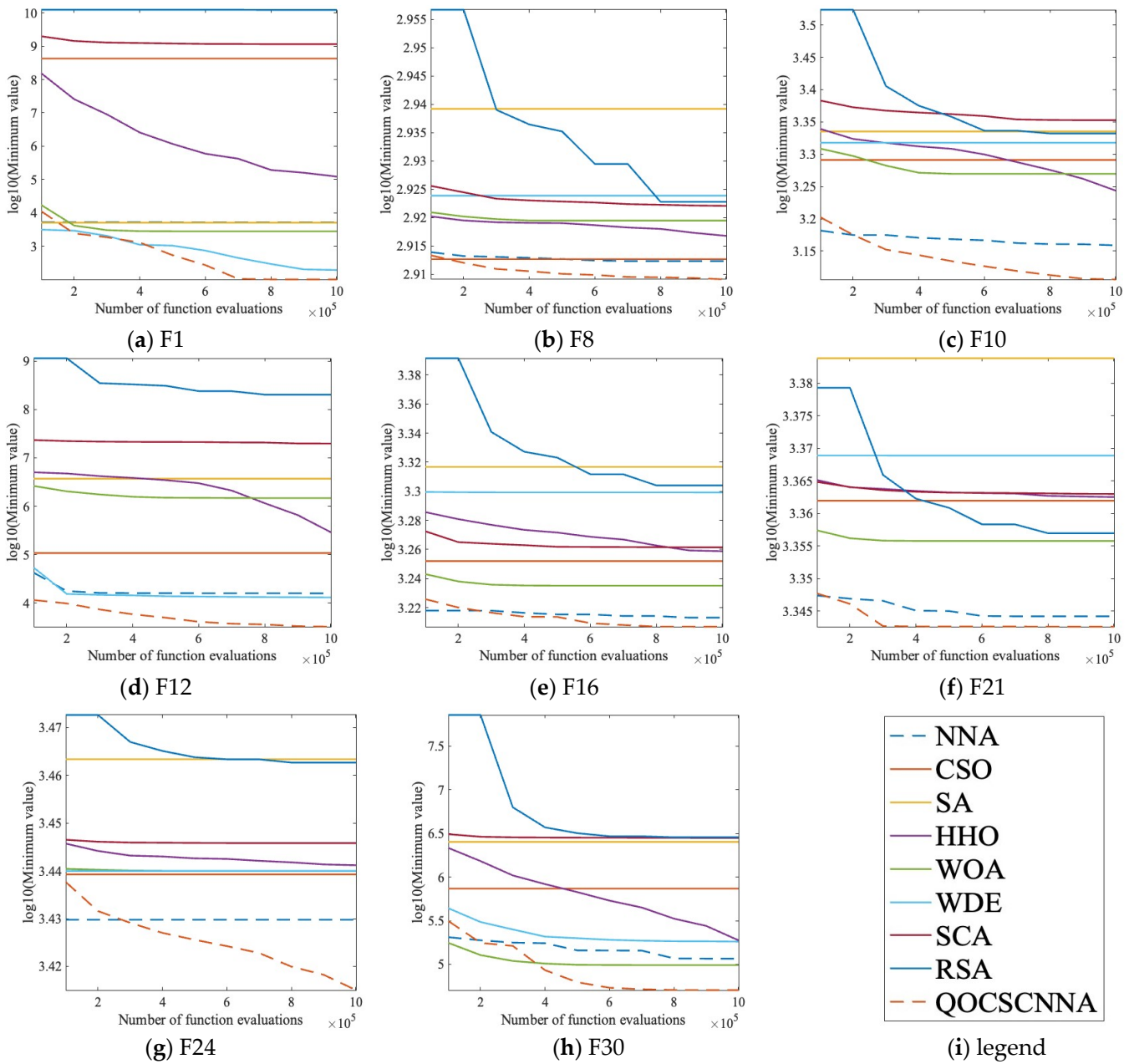


Figure 4. Convergence graph of QOCSC and its competitors.

This problem is being solved by several researchers using different metaheuristic methods, such as NNA, WOA, SCA, SA and PSO used in [42]. Table 2 reveals that the optimal result of QOCSCNNA is 1.3548, as well as the optimal constraints obtained from QOCSCNNA, satisfy Equation (19), which proves the validity of the optimal solutions obtained by QOCSCNNA. In addition, the optimum solutions of WOA and SA are 1.3567 and 1.3569, respectively, which are very close to the best results of QOCSCNNA. In contrast, the NNA, PSO, and SCA algorithms have poor optimum solutions, which indicates that these three algorithms are not suitable for the problem. Furthermore, by comparing the results of the Wilcoxon rank-sum test (+, =, and – indicating better, equal, or worse performance of QOCSCNNA compared to other algorithms), it is possible to discover that QOCSCNNA outperforms NNA, PSO, and SCA in terms of performance. Hence, it can be summarized that the proposed QOCSCNNA demonstrates superior feasibility compared to other algorithms.

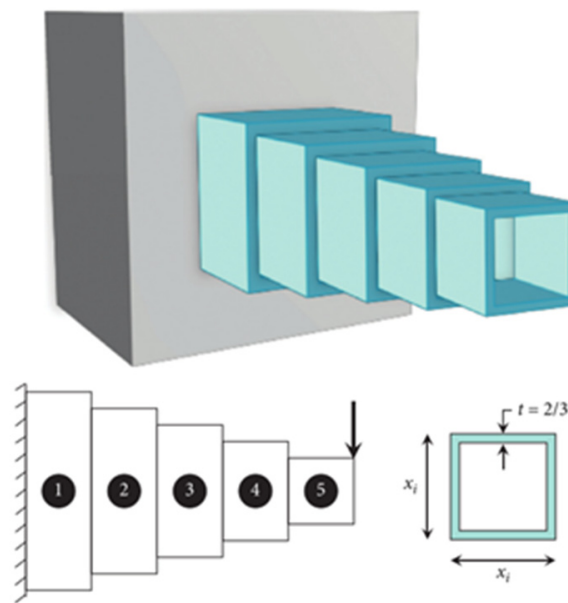


Figure 5. The model for the CB design problem [41].

Table 2. Comparison results between QOCSCNNA and its competitor CB design problems.

	QOCSCNNA	NNA	PSO	WOA	SCA	SA
x_1	5.8487	16.0042	5.7184	5.3875	5.6072	5.7227
x_2	5.2207	4.6257	8.1930	5.7766	6.0038	6.1674
x_3	4.6131	7.0074	5.9331	4.6485	4.1554	4.3927
x_4	3.8433	2.3777	2.8983	3.7243	3.9320	3.4333
x_5	2.2120	6.6237	1.6369	2.2048	2.1331	2.0293
$G(x)$	-0.0258	-0.0319	-1.6287×10^{-5}	-1.2506×10^{-11}	-1.6002×10^{-5}	0
$F(x)_{min}$	1.3564	2.2863	1.5213	1.3567	1.3623	1.3569
+/-/=		+	+	=	+	=

4.3.2. CSI Engineering Design Problem

As shown in Table 3, 11 parameters should be considered when minimizing the impact of a side impact on a vehicle. Figure 6 illustrates the model of the CSI crash design problem. The objective function of this design problem can be expressed as Equation (21):

$$F(x)_{min} = 1.98 + 4.90x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 2.73x_7 \quad (21)$$

Table 3. Influence parameters of the weight of the door.

No.	Variables	Description of Variables
1	x_1	Thicknesses of Bpillar Inner
2	x_2	Bpillar Reinforcement
3	x_3	Floor Side Inner
4	x_4	Cross Members
5	x_5	Door Beam
6	x_6	Door Beltline Reinforcement
7	x_7	Roof Rail
8	x_8	Materials of Bpillar Inner
9	x_9	Floor Side Inner
10	x_{10}	Barrier Height
11	x_{11}	Hitting Position



Figure 6. The model for the CSI problem [41].

Subject to:

$$G_1(x) = 1.16 - 0.3717x_2x_4 - 0.00931x_2x_{10} - 0.484x_3x_9 + 0.01343x_6x_{10} - 1 \leq 0 \quad (22)$$

$$G_2(x) = 46.36 - 9.9x_2 - 12.9x_1x_2 + 0.1107x_3x_{10} - 32 \leq 0 \quad (23)$$

$$G_3(x) = 33.86 + 2.95x_3 + 0.1792x_3 - 5.057x_1x_2 - 11.0x_2x_8 - 0.0215x_5x_{10} - 9.98x_7x_8 + 22.0x_8x_9 - 32 \leq 0 \quad (24)$$

$$G_4(x) = 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.0207x_5x_{10} + 6.63x_6x_9 - 7.7x_7x_8 + 0.32x_9x_{10} - 32 \leq 0 \quad (25)$$

$$G_5(x) = 0.261 - 0.0159x_1x_2 - 0.188x_1x_8 - 0.019x_2x_7 + 0.0144x_3x_5 + 0.0008757x_5x_{10} + 0.08045x_6x_9 + 0.00139x_8x_{11} + 0.00001575x_{10}x_{11} - 32 \leq 0 \quad (26)$$

$$G_6(x) = 0.214 + 0.00817x_5 - 0.131x_1x_8 - 0.0704x_1x_9 + 0.03099x_2x_6 - 0.018x_2x_7 + 0.0208x_3x_8 - 0.02x_2^2 + 0.121x_3x_9 - 0.00364x_5x_6 + 0.0007715x_5x_{10} - 0.0005354x_6x_{10} + 0.00121x_8x_{11} + 0.00184x_9x_{10} - 0.32 \leq 0 \quad (27)$$

$$G_7(x) = 0.74 - 0.61x_2 - 0.163x_3x_8 + 0.001232x_3x_{10} - 0.166x_7x_9 + 0.227x_2^2 - 0.32 \leq 0 \quad (28)$$

$$G_8(x) = 4.72 - 0.5x_4 - 0.19x_2x_3 - 0.0122x_4x_{10} + 0.009325x_6x_{10} + 0.000191x_{11}^2 - 4 \leq 0 \quad (29)$$

$$G_9(x) = 10.58 - 0.674x_1x_2 - 1.95x_2x_8 + 0.02054x_3x_{10} - 0.0198x_4x_{10} + 0.028x_6x_{10} - 9.9 \leq 0 \quad (30)$$

$$G_{10}(x) = 16.45 - 0.489x_3x_7 - 0.843x_5x_6 + 0.0432x_9x_{10} - 0.0556x_9x_{11} - 0.000786x_{11}^2 - 15.7 \leq 0 \quad (31)$$

Variable range:

$$0.5 \leq x_1, x_2, x_3, x_4, x_5, x_6, x_7 \leq 1.5, \quad x_8, x_9 \in \{0.192, 0.345\}, \quad -30 \leq x_{10}, x_{11} \leq +30 \quad (32)$$

The CSI problem is a widely studied classical engineering design problem and many heuristics have been proposed to solve it over the years. The methods include NNA, SA, WOA, PSO, and SCA. According to the comparative experimental results (Table 4), the presented QOCSNNA achieves the optimal fitness value of 23.4538 and makes the optimal constraints satisfy Equations (22)–(32). This validates the efficacy of the optimal results

obtained by QOCSCNNA. In addition, the optimal results of NNA, SA, WOA, PSO, and SCA are significantly higher than those of QOCSCNNA. This indicates that QOCSCNNA holds a clear advantage among the five algorithms for solving the problem. The analysis results by the Wilcoxon rank-sum test showed that QOCSCNNA was superior to the other algorithms. It further confirms the feasibility of the obtained QOCSCNNA.

Table 4. Comparison results between QOCSCNNA and its competitor CSI design problems.

	QOCSCNNA	NNA	SA	WOA	PSO	SCA
x_1	0.5000	0.5000	0.5846	0.5676	0.5000	0.5000
x_2	1.0702	0.9389	0.8214	0.8164	0.9749	1.0438
x_3	0.5000	0.5000	0.5000	0.5282	0.5000	0.5000
x_4	1.2346	1.4497	1.3295	1.3847	1.4484	1.3642
x_5	0.5000	0.5000	0.5386	0.5000	0.5000	0.5000
x_6	1.3072	0.7040	1.2600	0.6988	1.5000	0.9223
x_7	0.9359	1.1196	1.3245	1.2616	1.1604	0.9757
x_8	0.9200	0.9200	0.9200	0.9200	0.9200	0.9200
x_9	0.9200	0.9200	0.9200	0.9200	0.9200	0.9200
x_{10}	1.5200	0.5787	-0.6113	-5.1295	-26.1752	-7.1364
x_{11}	1.2847	15.7543	10.0525	2.1609	1.4852	-1.9988
$G_1(x)$	-0.5422	-0.5681	-0.4742	-0.5046	-0.8772	-0.6109
$G_2(x)$	-3.0534	-0.9586	0	-3.2758×10^{-4}	-3.0287	-3.1005
$G_3(x)$	-0.1003	-0.1158	-0.8491	-3.9851×10^{-9}	-0.6587	-0.0384
$G_4(x)$	-1.5518	-6.5431	-5.0120	-9.1875	-10.2060	-6.7642
$G_5(x)$	-0.0703	-0.0967	-0.0787	-0.1282	-0.0703	-0.1067
$G_6(x)$	-0.1258	-0.1282	-0.1378	-0.1600	-0.1578	-0.1548
$G_7(x)$	-0.1898	-0.1982	-0.2055	-0.2019	-0.2273	-0.1978
$G_8(x)$	-0.0030	-0.0531	-7.3880×10^{-4}	-1.6522×10^{-4}	-2.5206×10^{-9}	-0.0031
$G_9(x)$	-1.5665	-1.3500	-1.1290	-1.1124	-2.0150	-1.6091
$G_{10}(x)$	-0.0364	-0.7984	-0.7851	-0.1884	-1.2840	-0.0618
$F(x)_{min}$	23.4538	23.9420	23.7545	23.7803	24.2888	23.9060
+/-/=		+	+	+	+	+

4.3.3. TS Engineering Design Problem

The goal of the TS problem is to reduce the weight of the spring, illustrated in Figure 7. Minimum deflection, shear stress, surge frequency, outer diameter limits, and limitations on design variables need to be considered in the design process. The parameter settings include the average coil diameter D (denoted as x_1), the wire diameter d (denoted as x_2), and the effective number of coils N (denoted as x_3). The issue is described as:

$$F(x)_{min} = (x_3 + 2)x_2x_1^2 \tag{33}$$

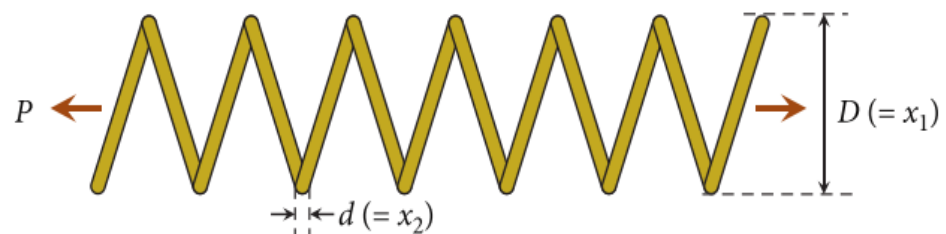


Figure 7. The model for the TS problem [41].

Subject to:

$$G_1(x) = 1 - x_1^3x_2/71785x_1^4 \leq 0 \tag{34}$$

$$G_2(x) = (4x_2^2 - x_1x_2)/12566(x_1^3x_2 - x_1^4) + 1/5108x_1^2 - 1 \leq 0 \tag{35}$$

$$G_3(x) = 1 - 140.45x_1/x_2^2x_3 \leq 0 \tag{36}$$

$$G_4(x) = (x_1 + x_2)/1.5 - 1 \leq 0 \tag{37}$$

Variable range:

$$0.05 \leq x_1 \leq 2, \quad 0.25 \leq x_2 \leq 1.3, \quad 2 \leq x_3 \leq 15 \tag{38}$$

Several researchers have tried various meta-heuristics to solve this problem, including NNA, SA, WOA, PSO, and HHO. Table 5 demonstrates the optimal solutions obtained by QOCSCNNA and the comparative algorithms, and it can be seen that the proposed QOCSCNNA obtains the optimal solution, i.e., 0.127. Also, it is given that the constraints on the optimal cost achieved with QOCSCNNA meet Equations (34)–(38), which implies that the best solution provided by QOCSCNNA is valid. In addition, HHO has an optimal fitness value of 0.0129, which is nearly the same as the optimal result of QOCSCNNA. On the contrary, the optimal solutions of NNA, SA, WOA, and PSO are inferior, which means QOCSCNNA and HHO have significant advantages. Moreover, through a comparison of the results of the Wilcoxon rank-sum test, it is observed that QOCSCNNA outperforms NNA, SA, WOA, and PSO in the aspect of performance. Therefore, it can be drawn that QOCSCNNA is a more efficient and feasible method compared with other algorithms.

Table 5. Comparison results between QOCSCNNA and its competitor TS design problems.

	QOCSCNNA	NNA	SA	WOA	PSO	HHO
x_1	0.5028	0.0659	0.0556	0.0612	0.0649	0.0552
x_2	0.3850	0.8040	0.4592	0.6309	0.7659	0.4478
x_3	9.8066	5.2301	10.0143	4.0040	6.8458	7.4345
$G_1(x)$	-1.1266×10^{-10}	-1.0099	-0.4100	-2.7309×10^{-6}	-1.4141	-1.3636×10^{-8}
$G_2(x)$	-1.1102×10^{-16}	-1.2845×10^{-11}	-1.1266×10^{-7}	-1.4795×10^{-10}	-5.3960×10^{-6}	-3.3307×10^{-16}
$G_3(x)$	-490.0553	-73.8673	-370.1311	-85.4248	-105.3872	-286.5735
$G_4(x)$	-0.7081	-0.4201	-0.6568	-0.5386	-0.4461	-0.6647
$F(x)_{min}$	0.0127	0.0252	0.0171	0.0142	0.0285	0.0129
+/-/=		+	+	+	+	=

5. Conclusions and Future Works

This paper reports on the NNA based on the quasi-oppositional-based strategy, piecewise linear chaotic mapping operator, and logistic chaotic sine-cosine learning strategy proposed to enhance global search capability and convergence. More specifically, QOBL allows the generation of quasi-opposite solutions between opposite solutions and the center of the solution space during the initialization phase, helping to balance exploration and exploitation in the generation jump. A new LCSC strategy by integrating LCM and SCLS is proposed which facilitates the algorithm to control at the bias strategy stage to jump out of the local optimum. Moreover, a dynamic adjustment factor that varies with the number of evaluations is presented, which facilitates tuning the search space and accelerates the convergence speed. To demonstrate the validity of QOCSCNNA, the performance of numerical optimization problems is investigated by solving challenging CEC 2017 functions. The results of the average and standard deviation of the comparison experiments in 29 test functions show that the QOCSCNNA algorithm outperforms the NNA algorithm in 23 functions and beats the other 7 algorithms in more than half of the test functions. Meanwhile, the Wilcoxon rank-sum test and convergence analysis indicate that the QOCSCNNA algorithm significantly outperforms the other algorithms. Furthermore, QOCSCNNA and other comparative algorithms are applied to three real-world engineering design problems, and the results further evidence the applicability of the algorithms in solving practical projects.

For future research, we concentrate on the next two areas. First, QOCSCNNA will continue to be improved to address more complex real-world engineering optimal problems, which include intelligent traffic management, supply chain optimization, and large-scale unmanned aircraft systems. Second, even though QOCSCNNA can greatly enhance the

Table A1. *Cont.*

No.	Metric	QOCSCNNA	NNA	CSO	SA	HHO	WOA	WDE	SCA	RSA
F24	AVG	2.6003×10^3	2.6901×10^3	2.7495×10^3	2.9065×10^3	2.7617×10^3	2.7542×10^3	2.7540×10^3	2.7913×10^3	2.9018×10^3
	STD	1.2995×10^2	1.1722×10^2	3.9062×10^1	8.9372×10^1	9.2439×10^1	7.1181×10^1	7.1111×10^1	6.0177×10^1	8.8390×10^1
F25	AVG	2.9134×10^3	2.9334×10^3	2.9497×10^3	2.9915×10^3	2.9278×10^3	2.9324×10^3	2.9303×10^3	2.9939×10^3	3.4258×10^3
	STD	2.1652×10^1	2.2886×10^1	2.0807×10^1	9.7024×10^1	2.3714×10^1	2.4213×10^1	2.4698×10^1	8.0498×10^1	1.9369×10^2
F26	AVG	2.9770×10^3	2.9559×10^3	3.1403×10^3	4.0366×10^3	3.2685×10^3	3.1671×10^3	3.5374×10^3	3.3500×10^3	4.1195×10^3
	STD	8.3544×10^1	4.3782×10^1	1.5730×10^2	6.8739×10^2	5.2972×10^2	3.0177×10^2	6.1491×10^2	3.3099×10^2	3.4946×10^2
F27	AVG	3.0921×10^3	3.0932×10^3	3.1089×10^3	3.2479×10^3	3.1213×10^3	3.1193×10^3	3.1127×10^3	3.1621×10^3	3.2831×10^3
	STD	1.9594×10^0	2.8759×10^0	1.3637×10^1	6.4546×10^1	2.4699×10^1	3.2729×10^1	1.9635×10^1	3.8743×10^1	7.4340×10^1
F28	AVG	3.3480×10^3	3.2893×10^3	3.3576×10^3	3.6733×10^3	3.3046×10^3	3.3119×10^3	3.3573×10^3	3.5270×10^3	3.5508×10^3
	STD	1.0843×10^2	8.3349×10^1	1.3117×10^2	1.9834×10^2	1.5539×10^2	1.5543×10^2	1.1627×10^2	1.9909×10^2	1.0554×10^2
F29	AVG	3.1627×10^3	3.1851×10^3	3.2150×10^3	3.4427×10^3	3.2798×10^3	3.2644×10^3	3.3194×10^3	3.2993×10^3	3.4197×10^3
	STD	2.0458×10^1	3.4660×10^1	3.7003×10^1	1.5928×10^2	7.1812×10^1	6.0127×10^1	7.5910×10^1	7.9825×10^1	1.5867×10^2
F30	AVG	5.0572×10^4	1.1594×10^5	7.3684×10^5	2.5244×10^6	1.8827×10^5	9.7804×10^4	1.8197×10^5	2.8045×10^6	2.8547×10^6
	STD	1.5162×10^5	2.8149×10^5	1.1491×10^6	2.6234×10^6	3.9800×10^5	1.6309×10^5	3.6619×10^5	6.4617×10^6	6.5786×10^6

Table A2. The Wilcoxon rank-sum test results.

No.	NNA	CSO	SA	HHO	WOA	WDE	SCA	RSA
F1	+	+	+	+	+	+	+	+
F3	+	+	+	+	+	=	+	+
F4	+	+	+	+	+	+	+	+
F5	+	+	+	+	+	+	+	+
F6	−	+	+	+	+	+	+	+
F7	+	+	+	+	+	+	+	+
F8	+	+	+	+	+	+	+	+
F9	−	+	+	+	+	+	+	+
F10	+	+	+	+	+	+	+	+
F11	+	+	+	+	+	+	+	+
F12	+	+	+	+	+	+	+	+
F13	+	+	+	+	+	+	+	+
F14	+	+	+	+	+	+	+	+
F15	+	+	+	+	+	+	+	+
F16	+	+	+	+	+	+	+	+
F17	+	+	+	+	+	+	+	+
F18	+	+	+	+	+	=	+	+
F19	+	+	+	+	+	+	+	+
F20	+	+	+	+	+	+	+	+
F21	+	+	+	+	+	+	+	+
F22	−	+	+	=	+	=	+	+
F23	+	+	+	+	=	+	+	+
F24	+	+	+	+	+	+	+	+
F25	+	+	+	=	+	=	+	+
F26	=	+	+	+	+	+	+	+
F27	=	+	+	+	+	+	+	+
F28	−	=	+	=	−	=	+	+
F29	+	+	+	+	+	+	+	+
F30	+	+	+	+	+	=	+	+
Statistics Number (+/−/=)	23/4/2	28/0/1	29/0/0	26/0/3	27/1/1	23/0/6	29/0/0	29/0/0

References

- Huang, L. A Mathematical Modeling and an Optimization Algorithm for Marine Ship Route Planning. *J. Math.* **2023**, *2023*, 5671089. [[CrossRef](#)]
- Ali, Z.A.; Zhangang, H.; Zhengru, D. Path planning of multiple UAVs using MMACO and DE algorithm in dynamic environment. *Meas. Control* **2023**, *56*, 459–469. [[CrossRef](#)]
- Fontes, D.B.M.M.; Homayouni, S.M.; Gonçalves, J.F. A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources. *Eur. J. Oper. Res.* **2023**, *306*, 1140–1157. [[CrossRef](#)]

4. Chen, D.; Zhang, Y. Diversity-Aware Marine Predators Algorithm for Task Scheduling in Cloud Computing. *Entropy* **2023**, *25*, 285. [[CrossRef](#)] [[PubMed](#)]
5. Wang, R.; Zhang, R. Techno-economic analysis and optimization of hybrid energy systems based on hydrogen storage for sustainable energy utilization by a biological-inspired optimization algorithm. *J. Energy Storage* **2023**, *66*, 107469. [[CrossRef](#)]
6. Ta, N.; Zheng, Z.; Xie, H. An interval particle swarm optimization method for interval nonlinear uncertain optimization problems. *Adv. Mech. Eng.* **2023**, *15*, 16878132231153266. [[CrossRef](#)]
7. Aljabhan, B.; Obaidat, M.A. Privacy-Preserving Blockchain Framework for Supply Chain Management: Perceptive Craving Game Search Optimization (PCGSO). *Sustainability* **2023**, *15*, 6905. [[CrossRef](#)]
8. Rizk-Allah, R.M.; Hassanien, A.E. A hybrid equilibrium algorithm and pattern search technique for wind farm layout optimization problem. *ISA Trans.* **2023**, *132*, 402–418. [[CrossRef](#)]
9. Luo, X.; Du, B.; Gui, P.; Zhang, D.; Hu, W. A Hunger Games Search algorithm with opposition-based learning for solving multimodal medical image registration. *Neurocomputing* **2023**, *540*, 126204. [[CrossRef](#)]
10. Chen, D.; Fang, Z.; Li, S. A Novel BSO Algorithm for Three-Layer Neural Network Optimization Applied to UAV Edge Control. *Neural Process. Lett.* **2023**. [[CrossRef](#)]
11. Savsani, P.; Savsani, V. Passing vehicle search (PVS): A novel metaheuristic algorithm. *Appl. Math. Model.* **2016**, *40*, 3951–3978. [[CrossRef](#)]
12. ALRahhal, H.; Jamous, R. *AFOX: A New Adaptive Nature-Inspired Optimization Algorithm*; no. 123456789; Springer: Dordrecht, The Netherlands, 2023. [[CrossRef](#)]
13. Chen, L.; Hao, C.; Ma, Y. A Multi-Disturbance Marine Predator Algorithm Based on Oppositional Learning and Compound Mutation. *Electronics* **2022**, *11*, 4087. [[CrossRef](#)]
14. Fan, Y.; Zhang, S.; Yang, H.; Xu, D.; Wang, Y. An Improved Future Search Algorithm Based on the Sine Cosine Algorithm for Function Optimization Problems. *IEEE Access* **2023**, *11*, 30171–30187. [[CrossRef](#)]
15. Hussain, K.; Salleh, M.N.M.; Cheng, S.; Shi, Y. Metaheuristic research: A comprehensive survey. *Artif. Intell. Rev.* **2019**, *52*, 2191–2233. [[CrossRef](#)]
16. Zhang, J.; Zhang, G.; Kong, M.; Zhang, T.; Wang, D.; Chen, R. CWOA: A novel complex-valued encoding whale optimization algorithm. *Math. Comput. Simul.* **2023**, *207*, 151–188. [[CrossRef](#)]
17. Xie, L.; Han, T.; Zhou, H.; Zhang, Z.R.; Han, B.; Tang, A. Tuna Swarm Optimization: A Novel Swarm-Based Metaheuristic Algorithm for Global Optimization. *Comput. Intell. Neurosci.* **2021**, *2021*, 9210050. [[CrossRef](#)]
18. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [[CrossRef](#)]
19. Zhang, J. Artificial immune algorithm to function optimization problems. In Proceedings of the 2011 IEEE 3rd International Conference on Communication Software and Networks, ICCSN 2011, Xi'an, China, 27–29 May 2011; pp. 667–670. [[CrossRef](#)]
20. Dorigo, M.; Birattari, M.; Stutzle, T. Ant Colony Optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [[CrossRef](#)]
21. Rao, R.V.; Patel, V. An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems. *Int. J. Ind. Eng. Comput.* **2012**, *3*, 535–560. [[CrossRef](#)]
22. Guilmeau, T.; Chouzenoux, E.; Elvira, V. Simulated Annealing: A Review and a New Scheme. *IEEE Work. Stat. Signal Process. Proc.* **2021**, *2021*, 101–105. [[CrossRef](#)]
23. Sadollah, A.; Sayyaadi, H.; Yadav, A. A dynamic metaheuristic optimization model inspired by biological nervous systems: Neural network algorithm. *Appl. Soft Comput.* **2018**, *71*, 747–782. [[CrossRef](#)]
24. Zhang, Y. Chaotic neural network algorithm with competitive learning for global optimization. *Knowl.-Based Syst.* **2021**, *231*, 107405. [[CrossRef](#)]
25. Zhang, Y.; Jin, Z.; Chen, Y. Hybrid teaching-learning-based optimization and neural network algorithm for engineering design optimization problems. *Knowl.-Based Syst.* **2020**, *187*, 104836. [[CrossRef](#)]
26. Zhang, Y.; Jin, Z.; Chen, Y. Hybridizing grey wolf optimization with neural network algorithm for global numerical optimization problems. *Neural Comput. Appl.* **2020**, *32*, 10451–10470. [[CrossRef](#)]
27. Wang, Y.; Wang, K.; Wang, G. Neural Network Algorithm with Dropout Using Elite Selection. *Mathematics* **2022**, *10*, 1827. [[CrossRef](#)]
28. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [[CrossRef](#)]
29. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M.A. Quasi-oppositional differential evolution. In Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 2229–2236.
30. Jia, D.; Zheng, G.; Khan, M.K. An effective memetic differential evolution algorithm based on chaotic local search. *Inf. Sci.* **2011**, *181*, 3175–3187. [[CrossRef](#)]
31. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [[CrossRef](#)]
32. Xu, Z.; Yang, H.; Li, J.; Zhang, X.; Lu, B.; Gao, S. Comparative Study on Single and Multiple Chaotic Maps Incorporated Grey Wolf Optimization Algorithms. *IEEE Access* **2021**, *9*, 77416–77437. [[CrossRef](#)]
33. Awad, N.H.; Ali, M.Z.; Liang, J.; Qu, B.Y.; Suganthan, P.N. Problem definitions and evaluation criteria for the CEC 2017 special session and competition on real-parameter optimization. *Nanyang Technol. Univ. Singap. Tech. Rep.* **2016**, 1–34.
34. Tizhoosh, H.R. Opposition-based learning: A new scheme for machine intelligence. In Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), Vienna, Austria, 28–30 November 2005; Volume 1, pp. 695–701.

35. Moradi, P.; Imanian, N.; Qader, N.N.; Jalili, M. Improving exploration property of velocity-based artificial bee colony algorithm using chaotic systems. *Inf. Sci.* **2018**, *465*, 130–143. [[CrossRef](#)]
36. Cheng, R.; Jin, Y. A competitive swarm optimizer for large scale optimization. *IEEE Trans. Cybern.* **2015**, *45*, 191–204. [[CrossRef](#)]
37. Heidari, A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Futur. Gener. Comput. Syst.* **2019**, *97*, 849–872. [[CrossRef](#)]
38. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [[CrossRef](#)]
39. Civicioglu, P.; Besdok, E.; Gunen, M.A.; Atasever, U.H. Weighted differential evolution algorithm for numerical function optimization: A comparative study with cuckoo search, artificial bee colony, adaptive differential evolution, and backtracking search optimization algorithms. *Neural Comput. Appl.* **2020**, *32*, 3923–3937. [[CrossRef](#)]
40. Abualigah, L.; Elaziz, M.A.; Sumari, P.; Geem, Z.W.; Gandomi, A.H. Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Syst. Appl.* **2021**, *191*, 116158. [[CrossRef](#)]
41. Bayzidi, H.; Talatahari, S.; Saraee, M.; Lamarche, C.P. Social Network Search for Solving Engineering Optimization Problems. *Comput. Intell. Neurosci.* **2021**, *2021*, 8548639. [[CrossRef](#)] [[PubMed](#)]
42. Wu, F.; Zhang, J.; Li, S.; Lv, D.; Li, M. An Enhanced Differential Evolution Algorithm with Bernstein Operator and Refracted Oppositional-Mutual Learning Strategy. *Entropy* **2022**, *24*, 1205. [[CrossRef](#)] [[PubMed](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.