electronics

MDPI

Article

Dynamic Artificial Bee Colony Algorithm Based on Permutation Solution

Yongkang Gong [1], Donglin Zhu [2], Chengtian Ouyang [3], Hongjie Guo [2,*] and Changjun Zhou [2]

[1] Institute of Future Technology, Jiangxi University of Applied Science and Technology, Nanchang 330006, China
[2] School of Computer Science and Technology, Zhejiang Normal University, Jinhua 321004, China
[3] School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China
* Correspondence: guohongjie@zjnu.edu.cn

Abstract: The artificial bee colony algorithm (ABC), as a classic swarm intelligence algorithm, has advantages such as fewer parameters and clear logic. However, ABC cannot balance the exploration and development stages well in the iterative process, and is easily affected by local optimal solutions in the final optimization stage, which affects the final optimal solution. To effectively compensate for the shortcomings of the algorithm, a neighbor learning artificial bee colony algorithm based on permutation solutions (CNABC) is proposed. In CNABC, a dynamic neighbor learning strategy is proposed to improve the search ability and optimal selection ability of the algorithm in the exploration phase. To solve the problem of lack of balance between exploration and development, the local optimal solution is used to guide the update of the surrounding difference. After the three stages of the algorithm are completed, a substitution mechanism is introduced, which replaces the worst solution by introducing external candidate solutions as feasible solutions, thereby improving the algorithm's ability to escape from local optima. Finally, comparative algorithm experiments are conducted on the CEC2022 test set, and compared with the TOP algorithm in the CEC competition on the CEC2022 test set. According to the experimental results, CNABC has good competitiveness in the comparative algorithm, which verifies the novelty and optimization ability of CNABC.

Keywords: intelligent optimization; artificial bee colony algorithm; dynamic neighbor learning; exchange mechanism; CEC function test set

check for updates

Citation: Gong, Y.; Zhu, D.; Ouyang, C.; Guo, H.; Zhou, C. Dynamic Artificial Bee Colony Algorithm Based on Permutation Solution. *Electronics* **2024**, *13*, 4934. https://doi.org/10.3390/electronics13244934

Academic Editor: Maciej Ławryńczuk

Received: 14 November 2024
Revised: 10 December 2024
Accepted: 11 December 2024
Published: 13 December 2024

Copyright: © 2024 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 the 21st century, the development of science and technology has made a qualitative leap. The ensuing meta-heuristic algorithms and optimization problems are gradually coming to the public's vision. An optimization problem refers to finding the smallest, best, or most efficient target solution in a complex problem. Optimization is increasingly being seen in biology [1], engineering optimization [2,3], production scheduling [4], feature selection [5,6], image processing [7], and network planning [8]. At present, with the complexity, diversification, and multi-quantification of optimization problems, traditional optimization methods, such as gradient descent method and Newton's method, fail to solve the problem in terms of time efficiency and target accuracy. Therefore, there is an increasing demand for efficient, stable, and robust optimization algorithms. Therefore, the emergence of swarm intelligent optimization algorithms brings good news to this phenomenon. Swarm intelligence optimization algorithms, inspired by cooperation, competition, or symbiosis among natural organisms, have been applied to computational intelligence by many scholars. Eberhart et al. [9] proposed a novel particle swarm optimization (PSO) algorithm in 1995, which attracted the attention of many researchers to swarm intelligence optimization; in 1991, Dorigo M et al. [10] designed and proposed the ant colony optimization (ACO) algorithm, inspired by the pheromones generated by the communication of ant communities when foraging outside; Yang et al. [11] proposed the cuckoo algorithm based on the living

habits of cuckoo birds; Seyedali Mirjalili et al. [12] were inspired by the hunting method of wolves in 2014 and proposed the grey wolf optimizer (GWO) with reasonable logic and simple parameters; Xue et al. [13] proposed the sparrow search algorithm (SSA) in 2020 with simple parameters and population classification based on the cooperative competition relationship during the foraging division of sparrows; Zhu et al. [14] developed a new human memory optimization algorithm (HMO) by simulating human memory behavior. The aforesaid algorithms, such as the particle swarm optimization algorithm, merely employ the information of the global optimum and individual optimum within the swarm, thereby resulting in a relatively lower precision of the outcome. The ant colony algorithm is characterized by complex parameter adjustment, and the inherent positive feedback mechanism of the algorithm itself is prone to being trapped in local optima. The Levy flight strategy of the cuckoo algorithm during the random walk stage makes it arduous to set the step size when updating the candidate solutions. The gray wolf algorithm is highly reliant on parameter settings during optimization and its capacity for global search is unstable. The sparrow search algorithm has a comparatively slow convergence rate during optimization and its parameter settings are rather intricate. The human memory algorithm lacks some dynamic adaptability and is incapable of fully simulating the human memory mechanism. But, compared with traditional optimization methods, swarm intelligence optimization algorithms have the advantages of simple implementation, fewer parameters, strong exploration ability, and strong logical structure, and are widely used in many fields by many researchers [15–18].

Inspired by the division of labor and behavior of the bee population in nectar collection, Karaboga et al. [19] proposed an artificial bee colony algorithm (ABC) in 2005. The algorithm divides the process of honey collection into three stages: the hiring stage, the following stage, and the reconnaissance stage. As part of the swarm intelligence algorithm, the artificial bee colony algorithm has the characteristics of clear logic, complete and easy to understand structure, fast convergence speed, and strong optimization ability. At present, it has certain advantages compared with existing particle swarm optimization algorithms, grey wolf algorithms, etc. ABC has also been applied in many fields, such as parameter optimization supporting SVM models [20], predictive coding based on biological RNA sequences [21], predictive analysis of photovoltaic energy storage models [22], and flight scheduling during adverse weather conditions [23], among other practical problems.

However, as a swarm intelligence algorithm, ABC cannot escape the no free lunch (NFL) law [24], it cannot show the outstanding side in all fields, and will always show the unsatisfactory side in some problems. During the process of search optimization, ABC depends on random population individuals, disregarding the benefits of information exchange. Due to the alternation of information and the mechanism of random search, the convergence efficiency of the algorithm is slowed down. For complex problems, the ABC algorithm may encounter problems such as slow solution updates, slightly insufficient optimization ability, and randomness in target solution search.

Therefore, in order to comprehensively improve the problems of ABC mentioned above, many researchers have improved ABC to make the original ABC more capable of solving complex optimization problems, and finally applied the improved ABC variant to different field projects. Wang et al. [25] proposed a novel binary optimization cross artificial bee colony algorithm applied to the problem of location selection for incompetent facilities (nABC). Based on the fusion of binary ABC, a cross strategy and a new disturbance frequency were proposed to improve the efficiency of information exchange between bees, while eliminating the random disturbance caused by the update strategy in the original ABC. Gürcan Yavuz et al. [26] proposed an artificial bee colony algorithm (SSEABC) based on an adaptive search equation and a competitive local search strategy. This feature eliminates inappropriate search equation groups through adaptive strategies, and then adds new food sources to the optimal food source location through competitive local search strategies and incremental population size strategies to ensure the convergence efficiency of SSEABC. Zhou et al. [27] improved the artificial bee colony algorithm by

combining deep Q-learning strategy and introducing adaptive strategy. The improved ABC more reasonably selects domain search operators, while expanding the search range of the population. At the same time, it minimizes energy allocation and consumption in milk truck scheduling projects and has a good improvement on the inherent Nphard problem. Ye et al. [28] applied the depth-first search strategy to the newly proposed stochastic domain structure strategy, which improved the search ability and solution accuracy of ABC, and to some extent helped the ABC algorithm jump out of local extremum. Zhou et al. [29] proposed a strategy for online fitness value analysis to improve the artificial bee colony algorithm. By identifying the structure of the problem to be solved and selecting the appropriate fitness value problem strategy, the online FLA technology was finally designed to improve the computational power of ABC and improve the efficiency of the algorithm at low cost. Zhou et al. [30] proposed an adaptive domain topology strategy to enhance the population diversity of ABC, enhance the transmission of valuable information, and improve the reconnaissance bee stage, while incorporating a random selection strategy at the end. It is used to reduce computational waste, improve the search efficiency of the algorithm, and jump out of local optima, greatly improving the accuracy of the original ABC. Indadul et al. [31] combined eight different rules with the roulette wheel to improve the accuracy of understanding. At the same time, a fixed number of perturbation techniques are added during the reconnaissance bee stage, so that the points trapped in the local optimum can jump out of the local optimum. Finally, the improved ABC is applied to solve the traveling salesman problem with good results. Zheng et al. [32] proposed an ABC algorithm based on population umbrella layering, which layers the historical information carried by the population and allows information exchange between levels. The above strategy improves the search ability of ABC and delays the convergence efficiency of the algorithm, improving the accuracy of the solution. Finally, the solution can jump out of the local optimum when trapped in a local optimum state. Song et al. [33] adjusted two solution update strategies applied in the exploration and development stages through adaptive mechanisms, and introduced the optimal solution to adjust the exploration and development stages. The algorithm ultimately achieved good results in solution accuracy and success rate. Guan et al. [34] proposed a self-regulating artificial bee colony algorithm (RCABC) based on random grouping, and improved the algorithm by proposing a dynamic random grouping strategy that has the opportunity to allocate the true solution set to subsets for optimization. At the same time, a self-regulating artificial bee colony strategy is used to improve the convergence of the algorithm and detect interaction features in subsets. The improved algorithm has shown good performance on both synthetic and real biological datasets. Wang et al. [35] fully utilized the professional performance advantages of individual bees, combined with the three stages of bee population, to combine behavior-based bee development and achieve individual professional division of labor. Finally, they achieved good results in testing functions and circular packaging problems. Tao et al. [36] combined two-dimensional vectors with adaptive perturbation and local search to generate a multi-strategy combined adaptive artificial bee colony algorithm, which increased the search ability of the algorithm. Finally, the effectiveness of the improved algorithm was verified on a randomly generated dataset combined with an actual workshop scheduling system. Yang et al. [37] proposed a discrete artificial bee colony algorithm (DABC) with a variable domain structure and four domain structures, and selectively applied them to the three stages of ABC. The improved algorithm achieved good results in solving distributed permutation flow shop problems with sequence dependent installation time. All of the above have proposed excellent and constructive enhancements based on ABC; however, certain deficiencies remain. (a) The majority of ABCs employ random or chaotic selection strategies in relation to search factors. Nevertheless, such random or chaotic approaches possess uncertainties and, thus, should not be overly selected or relied upon. (b) Multi-strategy combinations or fusions with other algorithms are implemented without considering whether the improved algorithm after fusion elevates the computational

complexity or time complexity of the algorithm. (c) The aforementioned algorithms give less consideration to the issue of convergence efficiency caused by information exchange.

Numerous scholars have attained remarkable outcomes in the study of ABC, but it is imperative to improve the stability and limitations of the algorithm. Therefore, while conducting a comprehensive analysis and improvement of the ABC literature, this article proposes a dynamic neighbor learning artificial bee colony algorithm based on commutative solutions. Combining the characteristics of ABC solution updates, candidate solutions are introduced from outside, and the combination of introducing candidate competition habits and solution updates is adopted to achieve solutions stopping at different stages due to optimal stagnation. To achieve more comprehensive searches and higher accuracy in different stages to the greatest extent possible, existing uncertainty and instability is eliminated, and ultimately the performance of ABC is further optimized. At the same time, considering the convergence efficiency problem caused by ABC information exchange and addressing the randomness of the hiring bee stage, a dynamic neighbor learning strategy is proposed to improve convergence efficiency while reducing the randomness of selecting bee individuals as hiring bees by learning the characteristics of multiple neighbors. The main novelty and work of CNABC are as follows:

- Based on the ABC solution update model, a novel strategy is proposed to improve the performance of the algorithm by introducing candidate solution schemes externally.
- By using dynamic neighbor learning strategies to stimulate the enthusiasm of local bee populations and create better search efficiency, the optimization performance of the algorithm can be improved.
- The stability and effectiveness of CNABC on the CEC2022 function test set and the TOP competition algorithm is verified.

This paragraph elaborates on the writing style of CNABC, and its paragraph structure is described as follows: In Section 1, we introduce the relevant work of ABC algorithm. In Section 2, we briefly introduce the ideas and related processes of the original ABC. Section 3 provides a detailed introduction to the CNABC proposed in this article. In Section 4, CNABC is compared with different algorithms and improved algorithms to verify its effectiveness and superior performance. In Section 5, we compare it with the TOP algorithm to understand its advantages and limitations in some aspects. Section 6 provides a general summary of the writing CNABC and looks forward to future research directions.

2. Artificial Bee Colony Algorithm

The artificial bee colony algorithm (ABC) is a novel population-based swarm intelligence optimization algorithm proposed by Karaboga [13] based on the foraging behavior of bee colonies in nature. In the algorithm model, three types of bees in nature were emulated, namely hired bees, follower bees, and scout bees, and the algorithm was partitioned into three stages based on this. Based on the corresponding behaviors and division of labor, a positional update formula model for the three types of bees was generated, and, subsequently, the acquisition of the maximum honey source for the three types of bees was accomplished.

Similar to other swarm intelligence optimization algorithms, food sources (SNs) are generated during the population initialization phase. Herein, SN represents the number of populations and is also a candidate solution for the current iteration. The population commences with random sampling, undergoes the stages of hiring bees, following bees, and reconnaissance bees, and exits the cycle when it reaches the predetermined maximal function evaluations (FESmax) for the stopping conditions.

During the honeybee hiring phase, the hired bees use their previous stored information to search for food around the honey source, while recording relevant honey source information (honey source quantity) and transmitting information on the location of food sources within the dance area to recruit onlookers. At this stage, the behavior of hiring bees

is to generate new food sources F_i near the candidate solution x_i. The mathematical model is as follows:

$$F_{i,j} = x_{i,j} + \varphi_{i,j} \cdot (x_{i,j} - x_{r,j}) \quad (1)$$

where φ is a number between $[-1, 1]$ and is the object that hired bees choose to learn from SN honey sources, $i \in \{1, 2, \ldots, SN\}$ and $j \in \{1, 2, \ldots, D\}$ are integers within the defined domain. The last new honey source F_i and x_i choose between i using a greedy strategy, if x_i is poor, then the newly generated F_i will replace.

During the bee-following stage, the following bees lack a certain degree of subjective initiative and tend to follow elite bees to learn the way to find high-quality honey sources. However, the probability P_i of following bees to choose honey sources depends on the fitness value of the amount of nectar, and the elite bees selected by the following bees are determined based on the roulette wheel strategy. The probability formula is as follows:

$$P_i = fit_i / \sum_{i=1}^{SN} fit_i \quad (2)$$

$$F_{i,j} = x_{rand} + \varphi_{i,j} \cdot (x_{i,j} - x_{r,j}) \quad (3)$$

where fit_i is the fitness value of the i-th food source in the current iteration. From Formula (2), it can be seen that the probability P_i is directly proportional to fit_i. The better the fitness value of the food source, the higher the probability of being selected. In Formula (3), x_{rand} is a random bee in the population.

In the final reconnaissance stage, when the food source is limited to the defined range without numerical updates, we determine that the source of the thing does not meet the honey gathering requirements of bees and needs to be discarded. At this point, the reconnaissance bee needs to randomly search for new food sources within the defined global range and the formula is updated as follows:

$$x_{i,j} = lb + rand \cdot (ub - lb) \quad (4)$$

rand is a random number defined as uniformly distributed between [0, 1], while *lb* and *ub* are upper and lower bounds on the source of food.

3. A Neighbor Learning Artificial Bee Colony Algorithm with Exchange Elements

This paragraph introduces the strategy of substitution and introduces the habit of competitive learning through candidate solutions, with the aim of comparing the advantages and disadvantages of bee populations with candidate solutions and eliminating inferior solutions. Bees are no exception; thus, in artificial bee colony algorithms, individuals (solutions) with poor performance in the bee population are replaced, and their positions are updated through past optimal solutions. Therefore, for this approach, we propose a method to construct candidate solutions by creating new solutions externally. By introducing new and excellent bees to eliminate poorer individuals in the population, the excellent bees are encouraged to seek higher quality honey sources. This idea is evidently in line with the competitive habits of organisms. The strategy of the new solution updating method is based on a candidate solution of the bee population, so there is also a certain interaction between the candidate solutions, which fully utilizes the population-based information through cooperation, competition, and other means. At the same time, due to the cooperative relationship between bee colonies, a dynamic neighbor learning strategy is proposed in the recruitment stage. This strategy, combined with dynamic group discussion, guides poor solutions to a position update through good and excellent solutions, and locks the location of bee sources.

3.1. Exchange Solution Mechanism

Next, before the new candidate solution X_{in} is added to the population, a representative set of solutions $\{X_{best}, X_{avg}, X_{worst}^{rep}, X_{best}^{rep}, X_{rand}\}$ is created externally or selected based

on past bee population information, collectively referred to as candidate solutions. The final substitution solution is solved through the above candidate solutions.

$$\begin{bmatrix} 0 & dist(X_1, X_2) & \cdots & dist(X_1, X_n) \\ dist(X_2, X_1) & 0 & \cdots & dist(X_2, X_n) \\ \cdots & \cdots & \cdots & \cdots \\ dist(X_n, X_1) & dist(X_n, X_2) & \cdots & 0 \end{bmatrix}$$

where $dist()$ represents the Euclidean distance between the bee individual i created in the similarity matrix and the remaining $N-1$ bee individuals. Next, the average Euclidean distance from the i-th bee individual to the remaining bee individuals is represented as follows. The candidate solution with the smallest S_i represents X_{best}^{rep}, the solution with the largest S_i represents X_{worst}^{rep}. $X_{avg} = 1/n\sum_{i=1}^{n} X_i$. X_{avg}, is calculated. Then, X_{in} is obtained by X_{best} and the formula is as follows:

$$S_i = 1/n\sum_{j=1}^{n} Dist_{ij}, i = 1 \cdots n \tag{5}$$

$$X_{in} = X_{best} + N(0, (1 - iter/Maxiter)) * \left(X_{best} - X_{avg}\right) \tag{6}$$

In Formula (6), $iter$ is the current number of iterations. $Maxiter$ is the maximum number of iterations, and N represents the standard deviation of a normal distribution with a mean of 0 and an expected value of $(1 - iter/Maxiter)$. Because the standard deviation is largely influenced by the optimization progress, the worst solution replaced by X_{in} in the early iterations is located further away from the optimal solution in the current iteration. As optimization continues, X_{in} will also become closer to the optimal solution in the current iteration. Then, considering that the positional relationship between the current optimal solution X_{best} and the average X_{avg} cannot represent a more comprehensive search space, there may be insufficient exploration of the region where X_{in} reaches the point. Therefore, the new candidate solutions X_{h1} and X_{h2} will be obtained by X_{best}^{rep}, X_{worst}^{rep}, and X_{rand}. The formula is as follows:

$$X_{h1} = X_{best} + N(0, (1 - iter/Maxiter)) * \left(X_{best} - X_{best}^{rep}\right) \tag{7}$$

$$X_{h2} = X_{best} + N(0, (1 - iter/Maxiter)) * \left(X_{best} - X_{worst}^{rep}\right) \tag{8}$$

Next, the final representative solution X_{rand} is created to avoid the possibility of generating candidate solutions far from the region of interest of the algorithm in future iterations when creating random solutions. Considering that the candidate solutions are externally created, the decision upper and lower bounds of X_{rand} will not use the algorithm's own upper and lower bounds, but instead use X_{min} and X_{max} from the generated similarity matrix as the boundaries of X_{rand}. The formula is as follows:

$$X_{rand} = X_{min} + rand(X_{max} - X_{min}) \tag{9}$$

In Formula (9), $rand$ is a random vector from 0 to 1.

Finally, X_{in} is updated based on the created or selected candidate solutions. The above operation will only be performed when the probability p meets the requirements and $p \leq 0.67$ is generally selected. In order to distinguish new candidate solutions, the obtained candidate solutions are numbered according to their order. The formula for updating candidate solutions is as follows:

$$X_{in}^2 = rand * X_{in} + (1 - rand) * X_{rand} + R, rand \leq 0.5 \tag{10}$$

$$X_{in}^3 = rand * X_{best} + (1 - rand) * X_{rand} + R, rand > 0.5 \tag{11}$$

In the formula, *rand* is a uniform random number in [0, 1]. It combines X_{rand} and X_{h1} as a random weight in X_{best}. The calculation of R is shown in Equation (12).

$$R = N(0, (1 - iter/Maxiter)) * (X_{h1} - X_{h2}) \tag{12}$$

This part of the equation uses the differences between auxiliary solutions to modify the results and moves the solutions to a new area. In the proposed hybrid algorithm, the worst candidate solution is replaced by the solution generated by the replacement solution update strategy.

$$X_{worst} = \begin{cases} X_{in}, rand > 0.67 \\ X_{in}^2, \ rand < 0.5 \\ X_{in}^3, else \end{cases} \tag{13}$$

The model diagram of the exchange element solution mechanism (EESM) is shown in Figure 1, and the pseudocode is shown in Algorithm 1:

Algorithm 1: Pseudocode for ESM

Input
SN: The number of current iteration populations
K: The number of candidate solutions is generally taken as 10% of the population size
X_{best}*: The optimal representative solution in the current iteration number*
X_{avg}*: The average solution of SN population solutions in the current iteration*
X_{worst}^{rep}*: Based on the similarity matrix, determine the average Euclidean distance from the individual to the remaining individuals, with the worst difference*
X_{best}^{rep}*: Obtain the optimal average Euclidean distance from an individual to the remaining individuals based on the similarity matrix*
X_{rand}*: Random solution created*
Output
$\boldsymbol{X_{new}}$***:*** *In the current iteration, the solution that ultimately replaces the difference in value*
1: ***For*** *i = 1 : K*
2: Create new solution () based on and Created, (6)
3: Create solutions, and, (7)–(9)
4: ***If*** *rand < 0.67 then*
5: ***If*** *rand < 0.5*
6: Create X_{in}^2, $X_{new} = X_{in}^2$, *(10)*
7: ***Else***
8: Create X_{in}^3, $X_{new} = X_{in}^3$, *(11)*
9: ***End If***
10: ***End If***
11: Replace the X_{new} *with Preyin*
12: ***End***
13: Return X_{new}

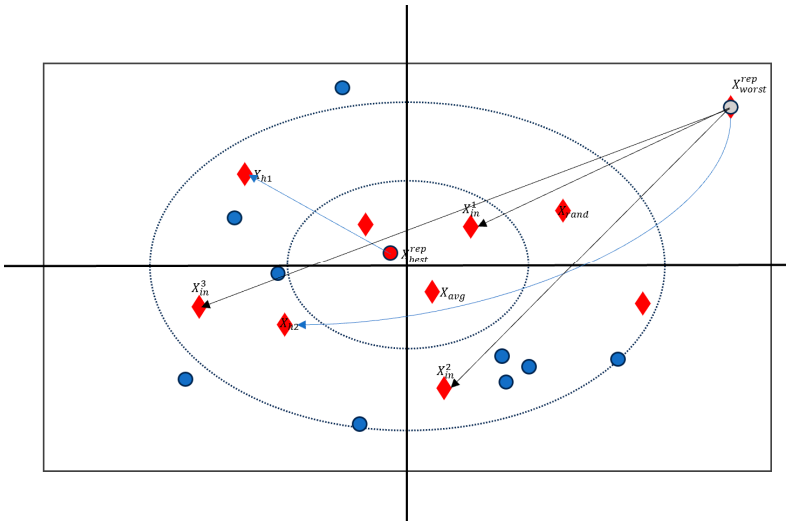


Figure 1. Schematic diagram of the exchange solution mechanism.

3.2. Dynamic Local Optimal Neighbor Learning

The optimization of primitive ABC involves extracting learning information from randomly selected individuals based on food sources and utilizing it as a medium for the optimization process. The original ABC optimization Formula (1) encapsulates this approach, demonstrating commendable performance in optimizing search processes by leveraging random food sources as learning objects, as indicated in [36]. However, the uniform probability associated with the quality of these randomly selected food sources significantly limits their effectiveness in developing and identifying optimal solutions. The current challenge lies in enhancing both the search efficiency and developmental capability of algorithms while achieving a balance between exploration and exploitation. In response to this challenge, this article proposes a dynamic local optimal neighbor learning strategy.

The bee dynamically selects N neighbors from the current iterated population and selects the best learning object among N neighbors for learning, ensuring the algorithm's global and local search capabilities. According to reference [38], the closer the N value is to SN, the closer the candidate solution is to the global optimal solution; on the contrary, it is a local optimal solution. Therefore, the N value in this article is determined to be 5. The updated formula is as follows:

$$F_{i,j} = x_{best}^{n,j} + \varphi_{i,j} \cdot \left(x_{best}^{n,j} - x_{i,j} \right) \tag{14}$$

X is the optimal value among the N neighbors of Xi currently.

The pseudocode for dynamic neighbor learning (DNL) is shown in Algorithm 2:

Algorithm 2: Pseudocode of DNL

Input
SN*: The number of current iteration populations*
$\boldsymbol{x_{best}^{n,j}}$*: The optimal value within N neighbors in the current iteration.*
N: *N bee neighbors selected in the current iteration*
Output
$\boldsymbol{F_{i,j}}$*: Generate new bee positions based on dynamic neighbor strategy in the current iteration*
1: ***For*** *i = 1:SN*
2: Use Formula (14) to update the position of bees
3: ***End for***
4: Return new position $F_{i,j}$

3.3. CNABC Process and Pseudocode

The process of learning artificial bee colony algorithm with swapped neighbors is as follows, with the pseudocode shown in Algorithm 3:

Algorithm 3: Pseudocode of CNABC

Input:
SN*: The number of populations*
N*: Update dynamically selected bee neighbors in iterations*
FESmax*: Maximum number of fitness value evaluations*
Dim: *Dimension of the problem*
K*: Number of candidate solutions*
Output:
$Fitness_{best}$*: The optimal fitness value output at the end of the loop condition*
X_{best}*: At the end of the loop condition, the optimal position corresponding to the optimal value*
1: Generate $x_1,\ x_2,\ x_3,\ \cdots x_n$ *food sources (SN) during population initialization stage*
2: Set the maximum fitness value evaluation frequency FESmax, problem dimension Dim, dynamic bee neighbors N, and K candidate solutions
3: FESmax = Dim $\times$ *10,000*
4: ***While*** *FES < FESmax do*
*5: * Hiring Bee Stage **
6: ***For*** *i = 1 : SN*
7: Execute a dynamic neighbor learning strategy according to Equation (14) to generate a new bee position $X_{i(new)}$
8: ***If*** $f(X_i) > f(X_{i(new)})$ *then*
9: $X_i = X_{i(new)}$
10: trail = 0
11: ***Else***
12: trail = trail + 1
13: ***End if***
14: FES = FES + 1
15: ***End for***
*16: * Reconnaissance Bee Stage **
17: Calculate the probability p of the selected bee using Formula (2)
18: ***For*** *i = 1 : SN do*
19: Calculate the probability p according to Formula (2) to select the food source X_j,
20: Execute a dynamic neighbor guided search strategy according to Equation (14). Generate new candidate solution $X_{j(new)}$
21: ***If*** $f(X_j) > f(X_{j(new)})$ *then*
22: $X_j = X_{j(new)}$
23: trial = 0
24: ***else***
25: trial = trial + 1
26: ***End if***
27: FES = FES + 1
28: ***End For***
*29: *Scout bee phase**
30: ***For*** *i = 1 : SN do*
31: ***If*** *triali > limit then*
32: trial = 0
33: Execute global neighbor search strategy according to Equation (3) to generate a new solution $X_{i,j}$

Algorithm 3: Cont.

34: $X_{i,j} = V_{i,j}$
35: $FES = FES + 1$
36: ***End if***
37: ***End for***
38: ***For*** $i = 1 : K$
39: Create new solution (X_{new}*) based on* X_{best}^{rep} *and the created* X_{avg}*, (5)*
40: Create solutions X_{h1}, X_{h2} *and* X_{rand}*, (7)–(9)*
41: ***If*** *rand < 0.67 then*
42: ***If*** *rand <0.5*
43: Create X_{in}^2, $X_{new} = X_{in}^2$*, (10)*
44: ***Else***
45: Create X_{in}^3, $X_{new} = X_{in}^3$*, (11)*
46: ***End If***
47: ***End if***
48: Replace the X_{new} *with Preyin*
49: ***End For***
50: ***End while***
51: Return X_{best} *&* $Fitness_{best}$

3.4. Analysis of Algorithm Time Complexity

Time complexity is the ability to comprehensively consider the efficiency, performance, and computational cost of an algorithm. Therefore, in order to demonstrate that CNABC does not increase the time complexity of the algorithm, this article conducts macro- and micro-analyses on CNABC and ABC. In the macro analysis, during the operation of ABC, assuming the number of food sources is M, the time required to initialize the consumption of each honey source is O(t1), and the time required to calculate the objective function of the Dim dimension problem is O(D). Therefore, the total time consumed to complete the initialization of the honey source is O(M × (t1)). Entering an iteration, the time consumed to calculate the fitness value in the D dimension is O(M × (D + t)). According to Formula (1), the time required to generate a new position during the hiring phase is O(M × (D + t2)). The time O(M*(D + t3) consumed to generate a new position according to Formulas (2) and (3) is during the following bee phase. In the final reconnaissance bee stage, the time required to generate a new position according to Formula (4) is O(M × (D + t4)). Therefore, assuming ABC is iterated T times, the algorithm time complexity is O(ABC) = O(M × (D + t)) + O(M × (t1)) + T × (O(M × (D + t2)) + O (M × (D + t3)) + O(M × (D + t4)), because O(M × (D + t)) and O(M × (t1)) are single finite, and the t2, t3, and t4 time periods are in a parallel order structure. The final algorithm time complexity is O(T × M(D + t2 + t3 + t4)) = O(T × M(D + t). Regarding the time complexity analysis of CBABC, although dynamic neighbor learning and substitution update mechanisms have been added, dynamic neighbor learning has not changed the algorithm structure and is only an operation for updating the population position. Therefore, the algorithm time complexity is not increased, so this period is still the same as ABC. The time complexity of the added swapping update mechanism is O(K × (t5)) = O (M × (0.1 × t5)), where K is the number of candidate solutions (usually 10% of M). So, the total time complexity O(CNABC) is O(M × (D + t)) + O(M × (t1)) + T × (O(M × (D + t2)) + O(M × (D + t3)) + O(M × (D + t4)) + O(M × (0.1 × t5))). Therefore, the final analysis shows that O (CNABC) = O(T × M(D + t)), which is consistent with the time complexity of ABC.

In micro-analysis, dynamic population learning is introduced in the hiring bee stage, and the time required to select the optimal neighbor from N populations is t1. If the calculation time is t2, the time complexity is O(M × t1 + M × (D × t2)), and the time analysis in the same stage as ABC is O(M × (D × t3)). In the final stage, the substitution update mechanism added is analyzed as O(M × (0.1 × t5)), which is used to adjust the

position of the worst solution. However, the above strategies all increase the number of algorithms for CNABC, so the time complexity of CNABC and ABC is the same, namely O(T × M (D + t)).

4. Experimental Analysis

4.1. Experimental Setting and Analysis

In order to test the comprehensive performance of the improved algorithm CNABC, this study conducted experiments in both 10-dimensional and 20-dimensional spaces using CNABC alongside 10 other algorithms on the CEC2022 test set. The parameter settings for each algorithm were consistent with those outlined in the original literature. CEC 2022 is also a test set containing unimodal, multimodal, mixed, and combination functions. In the CEC2022 test set, F1 is a unimodal function, F2–F5 is a multimodal function based on F2–F5, F6–F8 are mixed functions, and F9–F12 are combination functions. These two test sets encompass identical function type, thereby facilitating an evaluation of both the search capability and overall robustness of the algorithms employed. The relevant information of CEC 2022 is shown in Table 1.

Table 1. CEC 2022 test function information.

Type	No.	Function Name	*F**
Unimodal	C1	Shifted and Full Rotated Zakharov Function	300
Basic Functions	C2	Shifted and Full Rotated Rosenbrocks Function	400
	C3	Shifted and Full Rotated Expanded Scaffers F6 Function	600
	C4	Shifted and Full Rotated Non-Continuous Rastrigins Function	800
	C5	Shifted and Full Rotated Levy Function	900
Hybrid Functions	C6	Hybrid Function 1 (N = 3)	1800
	C7	Hybrid Function 2 (N = 6)	2000
	C8	Hybrid Function 3 (N = 5)	2200
Composition Functions	C9	Composition Function 1 (N = 5)	2300
	C10	Composition Function 2 (N = 4)	2400
	C11	Composition Function 3 (N = 5)	2600
	C12	Composition Function 4 (N = 6)	2700

To consider the principle of controlling variables and achieve fairness as much as possible should be considered. During the experiment, the population size of all algorithms was set to 100, and the maximum fitness value was calculated Dim × 10,000 times. Each algorithm participating in the test ran independently thirty times, and the average and standard deviations of the data after each algorithm ran independently thirty times were calculated. This was to clarify that the optimal values in the table were subtracted from the theoretical optimal values of the corresponding functions. The optimal value was marked in bold. The experimental results for the 10 and 20 dimensions of the CEC2022 test set are presented in Tables 2 and 3, respectively. The Friedman tests for each algorithm on the test set are at the end of Tables 2 and 3, and the rank sum test results for each algorithm are shown in Tables 4 and 5, respectively. The convergence and box plots indicate the analysis results for the 10 and 20 dimensions of the CEC2022 test set. The 10-dimensional convergence diagram, box plot, and radar plot of CEC2022 are shown in Figures 2–4. The 20-dimensional convergence diagram, box plot, and radar plot of CEC2022 are shown in Figures 5–7.

Table 2. Comparison of results for various functions in CEC2022 test set (dim = 10).

Function	Index	ABC	GWO	AGWO	PSO	VPPSO	WOA	SBWOA	SSA	QMESSA	NABC	CNABC
C01	Mean	8.2546E+03	4.9346E+02	1.9544E+03	3.0289E-02	1.5906E+01	9.9112E+03	8.3029E+03	1.3895E+02	4.9348E+01	1.8785E+03	**1.1973E-03**
	Std	1.8719E+03	1.0539E+03	1.7793E+03	7.0860E-02	2.9497E+01	5.4077E+03	4.8247E+03	3.3065E+02	5.5672E+01	8.7899E+02	**4.3849E-03**
C02	Mean	8.9340E-01	1.5132E+01	4.2507E+01	1.6844E+01	6.7331E+00	1.4244E+01	2.4366E+01	1.3054E+01	6.4125E+00	**5.2521E-01**	7.7435E+00
	Std	**9.2956E-01**	1.3983E+01	2.4314E+01	2.5592E+01	1.2669E+01	2.1196E+01	3.0260E+01	2.3171E+01	1.1854E+01	1.3779E+00	1.7176E+01
C03	Mean	1.6938E-02	6.0941E-01	8.6122E+00	5.8107E+01	4.7492E+00	3.2522E+01	2.7624E+01	6.9257E+00	1.3457E+01	3.6038E-09	**9.8529E-14**
	Std	6.5686E-03	9.1018E-01	5.5624E+00	1.5289E+00	3.7150E+00	1.3220E+01	1.1424E+01	6.7689E+00	9.6438E+00	3.3372E-09	**3.8646E-14**
C04	Mean	2.2731E+01	1.1440E+01	2.0022E+01	3.2436E+01	1.5356E+01	3.4056E+01	3.5387E+01	2.6393E+01	3.4326E+01	**5.9157E+00**	1.0911E+01
	Std	4.7129E+00	6.0415E+00	6.3683E+00	**4.8742E-01**	6.0340E+00	1.3720E+01	1.3062E+01	9.3352E+00	6.7126E+00	1.8201E+00	3.9254E+00
C05	Mean	2.3155E+02	4.0714E+00	5.6884E+01	5.6262E+02	5.0028E+00	4.0786E+02	3.8922E+02	4.3461E+02	4.4084E+02	1.0178E+00	**6.3937E-02**
	Std	1.2641E+02	1.0352E+01	6.3290E+01	2.9178E+01	6.1317E+00	3.1262E+02	2.7645E+02	1.9820E+02	1.7761E+02	6.2615E-01	**1.9356E-01**
C06	Mean	1.8419E+03	3.4077E+03	6.9263E+03	**1.0786E+02**	1.9803E+03	2.6490E+03	2.3237E+03	2.1067E+03	5.0388E+02	2.5580E+02	8.3687E+02
	Std	1.1742E+03	2.2306E+03	4.4139E+03	**9.2902E+01**	1.9824E+03	1.8867E+03	2.0229E+03	2.0572E+03	7.4311E+02	2.3720E+02	1.2552E+03
C07	Mean	5.6068E+00	2.2183E+01	4.0080E+01	3.4922E+02	3.6430E+01	5.7282E+01	5.0860E+01	2.7469E+01	3.3215E+01	**5.5491E-01**	1.8372E+00
	Std	1.9832E+00	1.0368E+01	1.2716E+01	1.2822E+01	1.1143E+01	2.1742E+01	1.8563E+01	1.1009E+01	1.3814E+01	**8.7477E-01**	4.9007E+00
C08	Mean	1.6429E+01	1.9684E+01	2.5943E+01	2.6477E+02	2.3180E+01	3.0538E+01	3.0001E+01	2.1540E+01	2.0810E+01	**1.1323E+01**	1.3731E+01
	Std	2.4709E+03	8.6649E+00	3.9720E+00	9.6430E+01	6.8472E+00	6.6605E+00	4.9137E+00	**2.7324E+00**	3.5512E+00	4.0116E+00	9.3017E+00
C09	Mean	**1.8472E+02**	2.4027E+02	2.7729E+02	2.7338E+02	2.2996E+02	2.4752E+02	2.4913E+02	2.3908E+02	2.2928E+02	2.3432E+02	2.2928E+02
	Std	3.3988E+01	1.9514E+01	2.2971E+01	6.7321E+01	7.5628E-01	3.9621E+01	3.3409E+01	3.6651E+01	1.4211E-13	3.9308E+00	**1.4211E-13**
C10	Mean	**3.9832E+00**	1.4081E+02	1.4821E+02	1.1379E+03	1.0434E+02	1.1843E+02	1.2292E+02	1.4832E+02	1.5707E+02	8.6611E+01	1.0022E+02
	Std	3.2375E+00	5.3259E+01	5.8618E+01	1.7744E+02	2.0022E+01	4.5104E+01	4.9923E+01	6.2871E+01	9.3835E+01	2.9139E+01	**5.5811E-02**
C11	Mean	4.2985E+01	1.0097E+02	1.8766E+02	1.9187E+02	4.3772E+01	2.3244E+02	2.1397E+02	9.8054E+01	2.3013E+02	4.6344E-03	**4.0927E-13**
	Std	2.2436E+01	1.3104E+02	1.4393E+02	1.4834E+02	1.0572E+02	2.0025E+02	1.5753E+02	1.3564E+02	1.0761E+02	2.4957E-02	**1.3642E-13**
C12	Mean	**1.4825E+02**	1.6397E+02	1.7952E+02	3.9439E+02	1.6336E+02	1.7802E+02	1.7983E+02	1.6673E+02	1.6774E+02	1.6598E+02	1.6292E+02
	Std	**6.2750E-01**	4.4185E+00	1.7683E+01	4.4916E+01	1.8335E+00	2.6114E+01	2.1621E+01	5.1859E+00	2.3001E+00	1.0712E+00	1.9242E+00
F-test		3.5833	5.5000	8.1667	8.6667	4.5833	8.8333	8.9167	6.4167	6.5417	2.5833	2.2083

Table 3. Comparison of results for various functions in CEC2022 test set (dim = 20).

Function	Index	ABC	GWO	AGWO	PSO	VPPSO	WOA	SBWOA	SSA	QMESSA	NABC	CNABC
C01	Mean	8.2546E+03	4.9346E+02	1.9544E+03	3.0289E-02	1.5906E+01	9.9112E+03	8.3029E+03	1.3895E+02	4.9348E+01	1.8785E+03	**1.1973E-03**
	Std	1.8719E+03	1.0539E+03	1.7793E+03	7.0860E-02	2.9497E+01	5.4077E+03	4.8247E+03	3.3065E+02	5.5672E+01	8.7899E+02	**4.3849E-03**
C02	Mean	8.9340E-01	1.5132E+01	4.2507E+01	1.6844E+01	6.7331E+00	1.4244E+01	2.4366E+01	1.3054E+01	6.4125E+00	**5.2521E-01**	7.7435E+00
	Std	**9.2956E-01**	1.3983E+01	2.4314E+01	2.5592E+01	1.2669E+01	2.1196E+01	3.0260E+01	2.3171E+01	1.1854E+01	1.3779E+00	1.7176E+01
C03	Mean	1.6938E-02	6.0941E-01	8.6122E+00	5.8107E+01	4.7492E+00	3.2522E+01	2.7624E+01	6.9257E+00	1.3457E+01	3.6038E-09	**9.8529E-14**
	Std	6.5686E-03	9.1018E-01	5.5624E+00	1.5289E+00	3.7150E+00	1.3220E+01	1.1424E+01	6.7689E+00	9.6438E+00	3.3372E-09	**3.8646E-14**
C04	Mean	2.2731E+01	1.1440E+01	2.0022E+01	3.2436E+01	1.5356E+01	3.4056E+01	3.5387E+01	2.6393E+01	3.4326E+01	**5.9157E+00**	1.0911E+01
	Std	4.7129E+00	6.0415E+00	6.3683E+00	**4.8742E-01**	6.0340E+00	1.3720E+01	1.3062E+01	9.3352E+00	6.7126E+00	1.8201E+00	3.9254E+00
C05	Mean	2.3155E+02	4.0714E+00	5.6884E+01	5.6262E+02	5.0028E+00	4.0786E+02	3.8922E+02	4.3461E+02	4.4084E+02	1.0178E+00	**6.3937E-02**
	Std	1.2641E+02	1.0352E+01	6.3290E+01	2.9178E+01	6.1317E+00	3.1262E+02	2.7645E+02	1.9820E+02	1.7761E+02	6.2615E-01	**1.9356E-01**
C06	Mean	1.8419E+03	3.4077E+03	6.9263E+03	**1.0786E+02**	1.9803E+03	2.6490E+03	2.3237E+03	2.1067E+03	5.0388E+02	2.5580E+02	8.3687E+02
	Std	1.1742E+03	2.2306E+03	4.4139E+03	**9.2902E+01**	1.9824E+03	1.8867E+03	2.0229E+03	2.0572E+03	7.4311E+02	2.3720E+02	1.2552E+03
C07	Mean	5.6068E+00	2.2183E+01	4.0080E+01	3.4922E+02	3.6430E+01	5.7282E+01	5.0860E+01	2.7469E+01	3.3215E+01	**5.5491E-01**	1.8372E+00
	Std	1.9832E+00	1.0368E+01	1.2716E+01	1.2822E+01	1.1143E+01	2.1742E+01	1.8563E+01	1.1009E+01	1.3814E+01	**8.7477E-01**	4.9007E+00
C08	Mean	1.6429E+01	1.9684E+01	2.5943E+01	2.6477E+02	2.3180E+01	3.0538E+01	3.0001E+01	2.1540E+01	2.0810E+01	**1.1323E+01**	1.3731E+01
	Std	2.4709E+03	8.6649E+00	3.9720E+00	9.6430E+01	6.8472E+00	6.6605E+00	4.9137E+00	**2.7324E+00**	3.5512E+00	4.0116E+00	9.3017E+00
C09	Mean	**1.8472E+02**	2.4027E+02	2.7729E+02	2.7338E+02	2.2996E+02	2.4752E+02	2.4913E+02	2.3908E+02	2.2928E+02	2.3432E+02	2.2928E+02
	Std	3.3988E+01	1.9514E+01	2.2971E+01	6.7321E+01	7.5628E-01	3.9621E+01	3.3409E+01	3.6651E+01	1.4211E-13	3.9308E+00	**1.4211E-13**
C10	Mean	**3.9832E+00**	1.4081E+02	1.4821E+02	1.1379E+03	1.0434E+02	1.1843E+02	1.2292E+02	1.4832E+02	1.5707E+02	8.6611E+01	1.0022E+02
	Std	3.2375E+00	5.3259E+01	5.8618E+01	1.7744E+02	2.0022E+01	4.5104E+01	4.9923E+01	6.2871E+01	9.3835E+01	2.9139E+01	**5.5811E-02**
C11	Mean	4.2985E+01	1.0097E+02	1.8766E+02	1.9187E+02	4.3772E+01	2.3244E+02	2.1397E+02	9.8054E+01	2.3013E+02	4.6344E-03	**4.0927E-13**
	Std	2.2436E+01	1.3104E+02	1.4393E+02	1.4834E+02	1.0572E+02	2.0025E+02	1.5753E+02	1.3564E+02	1.0761E+02	2.4957E-02	**1.3642E-13**
C12	Mean	**1.4825E+02**	1.6397E+02	1.7952E+02	3.9439E+02	1.6336E+02	1.7802E+02	1.7983E+02	1.6673E+02	1.6774E+02	1.6598E+02	1.6292E+02
	Std	**6.2750E-01**	4.4185E+00	1.7683E+01	4.4916E+01	1.8335E+00	2.6114E+01	2.1621E+01	5.1859E+00	2.3001E+00	1.0712E+00	1.9242E+00
F-test		4.7500	5.6667	8.5833	7.8333	4.7500	8.7500	8.1667	6.4167	5.5000	3.2500	2.3333

Table 4. *p*-values of the Wilcoxon rank sum tests on CEC 2022 functions (Dim = 10).

Function	ABC	GWO	AGWO	PSO	VPPSO	WOA	SBWOA	SSA	QMESSA	NABC
C01	3.02E-11	3.02E-11	3.02E-11	4.08E-05	7.66E-05	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
C02	4.55E-01	4.41E-07	5.43E-06	4.50E-01	3.91E-02	1.24E-04	8.11E-05	2.62E-03	1.05E-01	1.54E-01
C03	4.08E-12	4.08E-12	4.08E-12	4.05E-12	4.08E-12	4.08E-12	4.08E-12	4.08E-12	4.08E-12	4.08E-12
C04	6.06E-10	9.82E-01	5.57E-07	2.99E-11	1.11E-03	4.46E-11	2.35E-10	1.53E-09	2.99E-11	5.83E-06
C05	1.96E-11	4.20E-10	1.96E-11	1.96E-11	5.38E-11	1.96E-11	1.96E-11	1.96E-11	1.96E-11	2.86E-10
C06	2.60E-05	8.20E-07	6.12E-10	1.38E-02	4.23E-03	1.25E-05	7.30E-04	2.62E-03	7.62E-01	2.34E-01
C07	8.48E-09	5.07E-10	3.69E-11	3.02E-11	3.69E-11	3.02E-11	3.02E-11	3.69E-11	6.07E-11	6.00E-01
C08	3.63E-01	9.51E-06	2.67E-09	3.02E-11	7.09E-08	2.15E-10	2.15E-10	1.07E-09	7.09E-08	5.55E-02
C09	1.21E-12	1.21E-12	1.21E-12	6.61E-04	1.21E-12	1.21E-12	1.21E-12	4.57E-12	1.21E-12	1.21E-12
C10	3.02E-11	3.82E-09	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.79E-01	6.35E-02
C11	3.15E-12	3.15E-12	3.15E-12	3.15E-12	3.15E-12	3.15E-12	3.15E-12	3.15E-12	3.15E-12	3.12E-12
C12	2.98E-11	1.33E-01	6.00E-11	2.98E-11	3.79E-01	9.66E-10	1.76E-10	8.11E-05	6.06E-10	5.91E-09

Table 5. *p*-values of the Wilcoxon rank sum tests on CEC 2022 functions (Dim = 20).

Function	ABC	GWO	AGWO	PSO	VPPSO	WOA	SBWOA	SSA	QMESSA	NABC
C01	3.02E-11	1.33E-10	3.02E-11	2.23E-09	7.29E-03	5.87E-04	6.20E-04	4.98E-11	1.09E-10	3.02E-11
C02	6.36E-05	6.74E-06	3.02E-11	2.77E-01	2.61E-02	1.29E-06	6.36E-05	1.91E-01	2.84E-01	2.12E-01
C03	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
C04	3.02E-11	3.26E-01	4.98E-11	3.02E-11	1.73E-06	3.02E-11	3.02E-11	3.02E-11	3.02E-11	5.09E-08
C05	3.02E-11	6.05E-07	7.09E-08	3.02E-11	7.73E-02	3.69E-11	3.69E-11	4.50E-11	4.08E-11	4.84E-02
C06	4.08E-11	2.71E-02	3.02E-11	8.65E-01	5.59E-01	4.42E-06	2.53E-04	1.37E-01	9.35E-01	4.42E-06
C07	3.02E-11	1.25E-07	3.02E-11	3.02E-11	1.33E-10	3.02E-11	3.02E-11	3.02E-11	3.02E-11	4.20E-01
C08	5.57E-10	4.62E-10	3.16E-10	3.02E-11	2.87E-10	5.49E-11	9.92E-11	1.61E-10	2.87E-10	5.57E-10
C09	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
C10	7.77E-09	1.09E-10	6.07E-11	3.02E-11	4.62E-10	8.99E-11	7.39E-11	6.70E-11	6.53E-07	1.46E-10
C11	2.60E-10	1.69E-08	3.01E-11	2.37E-07	2.25E-04	1.99E-05	6.20E-04	9.20E-05	8.66E-05	2.50E-03
C12	3.02E-11	6.35E-02	7.12E-09	3.02E-11	3.92E-02	2.15E-06	5.09E-08	1.70E-02	9.51E-06	2.78E-07

It should be noted that the experimental environment used for this test was provided by MATLAB 2020b, the operating system was Windows 11, the main frequency was 3.20 GHz, and the memory was 16 GB. The comparative algorithms were ABC [19], GWO [12], AGWO [3], PSO [9], VPPSO [39], WOA [40], SBWOA [41], SSA [13], QMESSA [42], and NABC [38].

In the analysis of the 10-dimensional problem of the CEC2022 test set function, CNABC ranked first in the test function with 4. The test data of all four functions were bright spots, far ahead of the comparison algorithms in terms of magnitude. The C03 and C11 test functions were able to find the optimal values. As shown in Figure 2, in terms of the convergence curves of each algorithm in each function, CNABC achieved good results in C01, C03, C05, and C11 under 10 dimensions, indicating that CNABC exhibited good performance in unimodal functions, simple multimodal functions, and combination functions. Although CNABC did not achieve first place in the hybrid function, it was able to maintain the same order of magnitude as the final result of the top ranked algorithm, with C06 in fourth place and C07 and C08 in second place, all on the same order of magnitude as the current best. Overall, CNABC had strong performance in both pre-optimization and post-optimization search. On the combination function, the values of C09, C10, and C12 were slightly worse than ABC, and the overall optimization values were not significantly different.

$$CD = Q\alpha_{0.05}\sqrt{\left(\frac{K(K+1)}{6N}\right)} \tag{15}$$

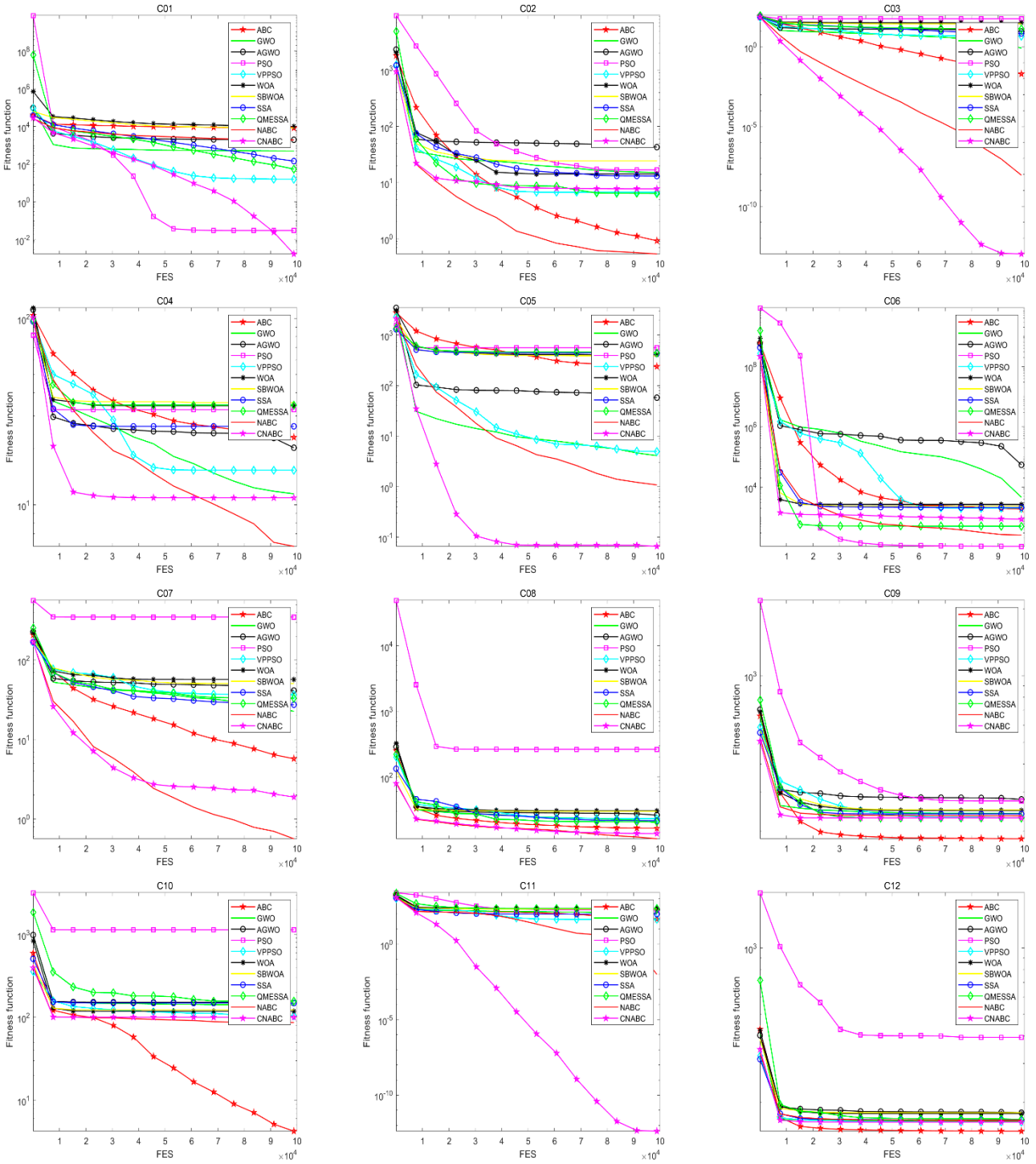


Figure 2. CEC2022 convergence graph (10 dim).

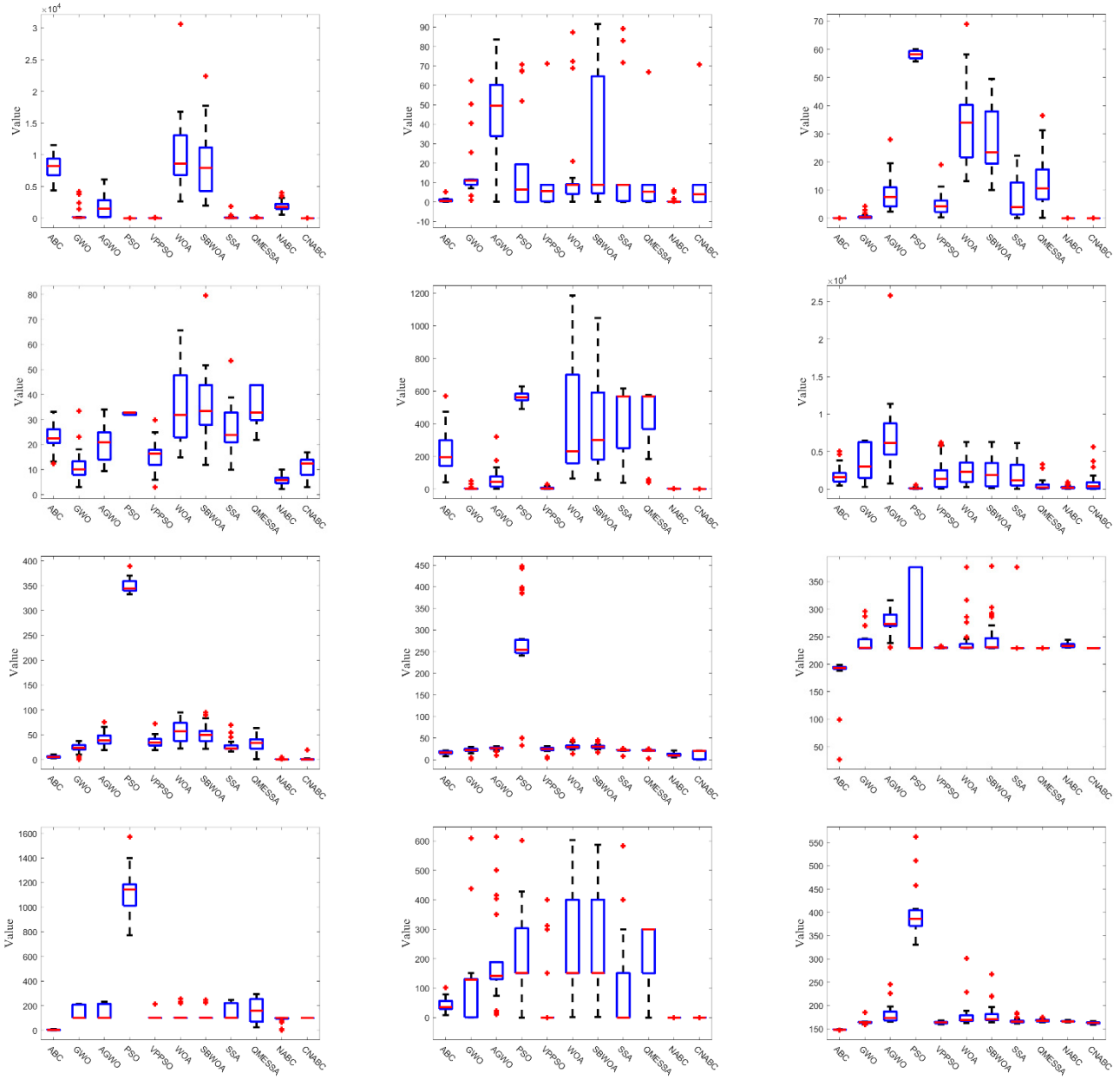


Figure 3. CEC2022 box diagram (10 dim).

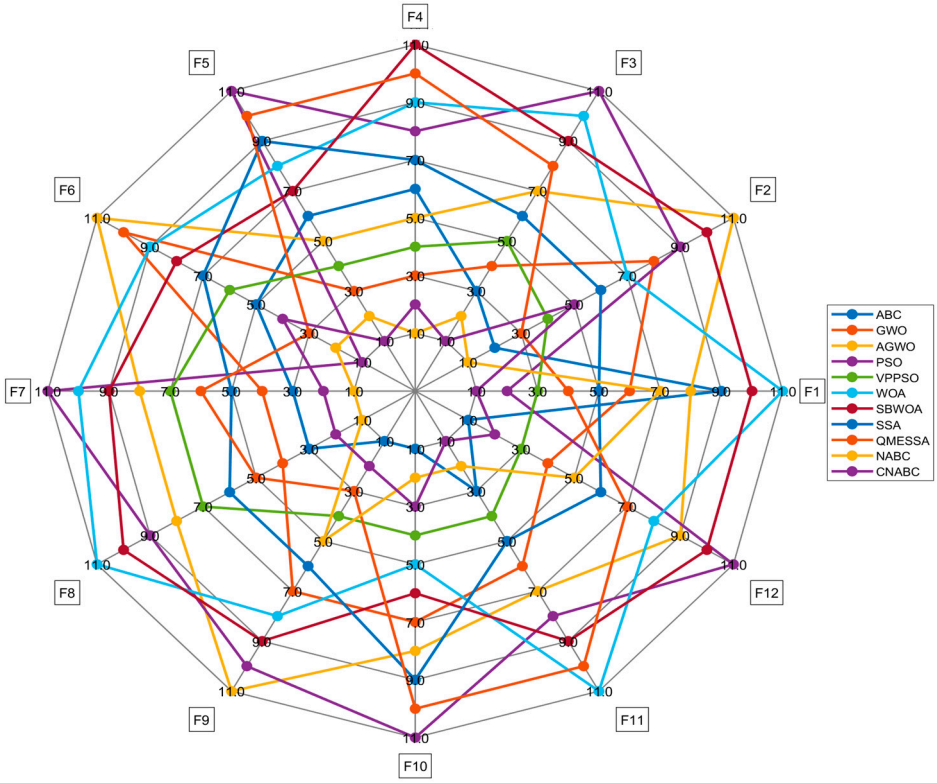


Figure 4. Radar ranking chart of CEC2022 test set for each algorithm (10 dim).

From the results of the Friedman test, the rank ranking of CNABC was 2.2100. According to Formula (15) and by looking up the table [43], the critical CD value of the algorithm was 4.3585. The CD line value obtained by adding the rank ranking to the critical CD value was 6.5685. It can be concluded that there were significant differences between CNABC and AGWO, PSO, WOA, and SBWOA, while the differences between the remaining comparison algorithms were relatively small. Furthermore, as shown in Figure 3, CNABC had smaller optimal, median, and worst values on functions C01, C03, C05, C09, and C11. On the C06, C11, and C12 functions, although the optimal, median, and worst values of CNABC algorithm were not the best, it can be seen that they tended to stabilize in the interval, ranking second or third in the ranking, and were at the forefront of the comparison algorithms. Overall, it can be concluded that CNABC has good comprehensive performance in 10 dimensions.

As shown in Figure 7, a specific analysis was conducted on the 20-dimensional problem of the CEC2022 test set function. CNABC ranked first in the test function with four numbers, namely C03, C04, C07, and C08. From this perspective, CNABC had a good effect on higher dimensional unimodal functions, simple multimodal functions, and mixed functions. In the analysis of the combination function, although it did not achieve the first ranking, all rankings in the combination function were in the top three positions, still ranking among the top in the comparison algorithm. From the convergence curve graph, it can be seen that CNABC had good search performance in both the early and late stages of iteration, and when dealing with local optima, it can adjust the worst solution through EESM. There was also a downward trend in exploration on C01, C03, C04, and C07.

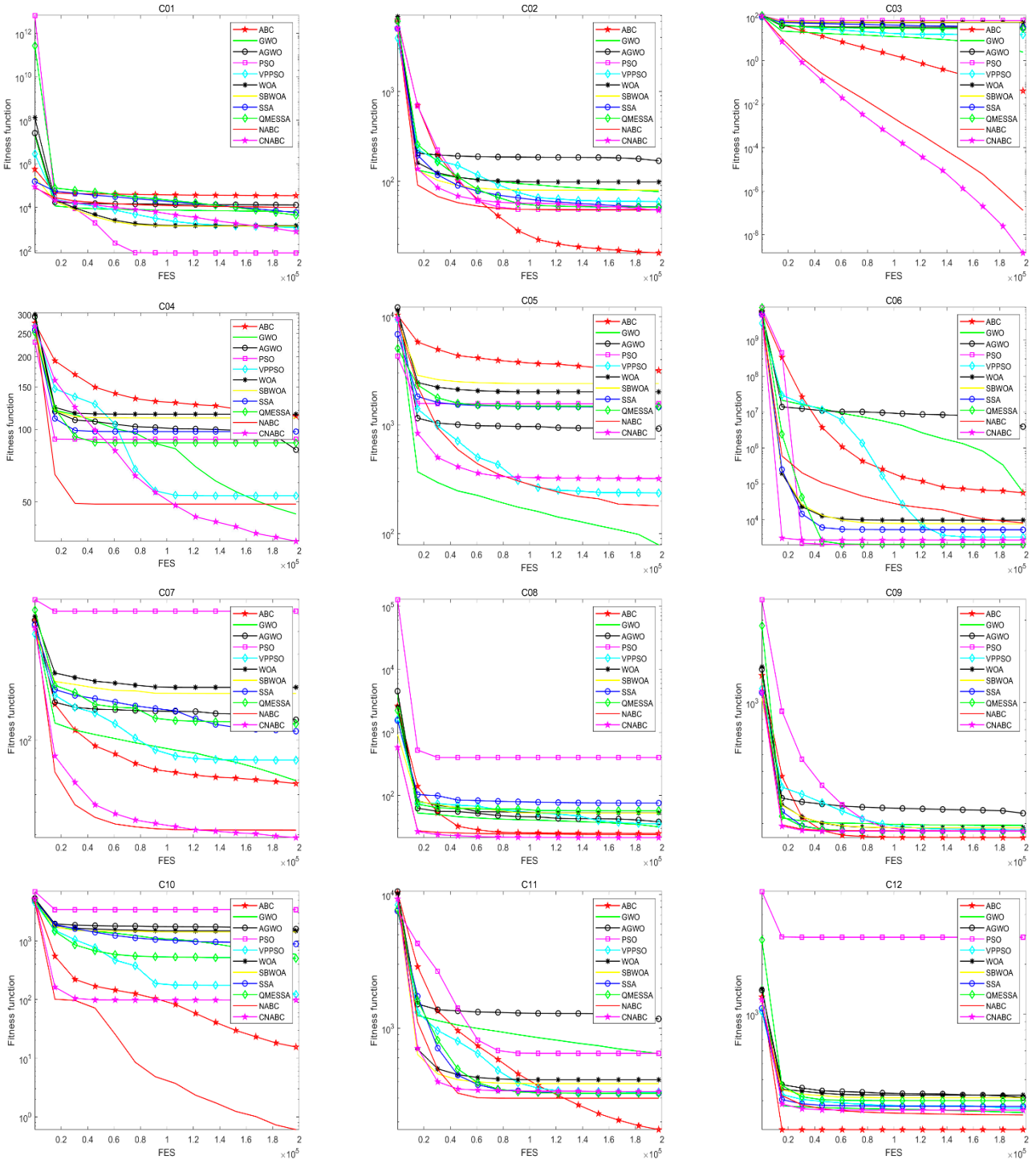


Figure 5. CEC2022 convergence graph (20 dim).

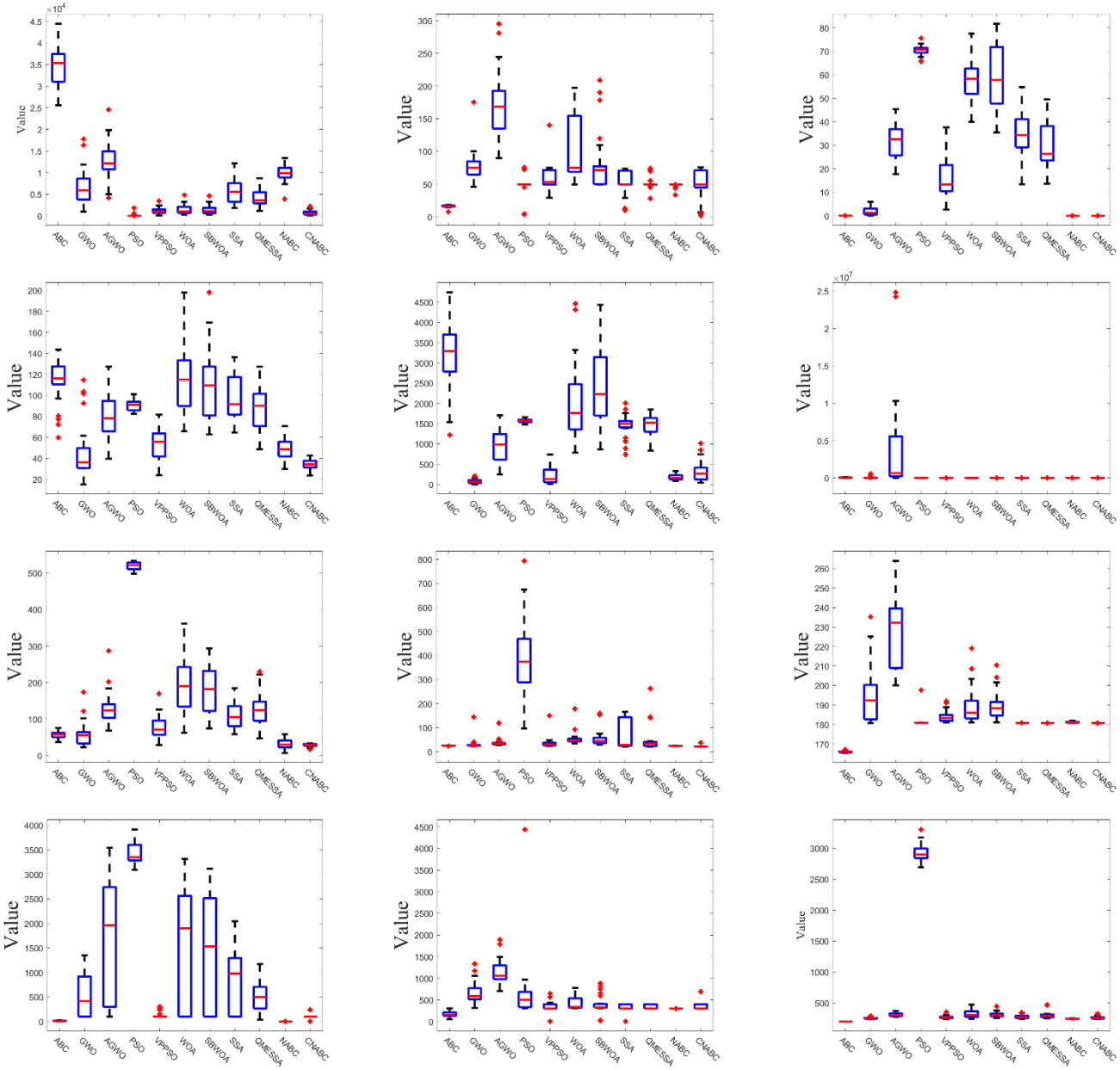


Figure 6. CEC2022 box diagram (20 dim).

From the Friedman test results, it can be seen that CNABC had some advantages in comparison algorithms. Its average rank value was 2.333, which was a good performance compared with the second highest ranking. Although there were only four ranked as first in the 20 dimensions, it can reflect the fact that it can still maintain its position before the exam without ranking first in other functions compared with the algorithm, and the optimal value, variance, and other factors were similar. For example, the above situation can be reflected on functions C02, C06, and C09. Next, according to the 10-dimensional analysis, the critical value CD was 4.3585, so the CD line value can be calculated as 6.6918. Therefore, it can be concluded that there was a significant difference between CNABC and AGWO, PSO, WOA, and SBWOA algorithms, and there was no significant difference compared with the remaining algorithms. Finally, from Figure 6, the box plot analysis of the 20 dimensions shows that the top ranked functions C03, C04, C07, and C08 were superior to other comparison algorithms in terms of optimal value, median, and worst value. Although the performance on C05, C09, and C10 was somewhat unsatisfactory, it can still be accepted that CNABC ranked around third place without the free lunch rule.

Based on the above two points, analyzing the data from the 20 dimensions can reflect some advantages of CNABC in handling this test set function, and also demonstrates that the algorithm is at an average leading level among the selected comparative algorithms.

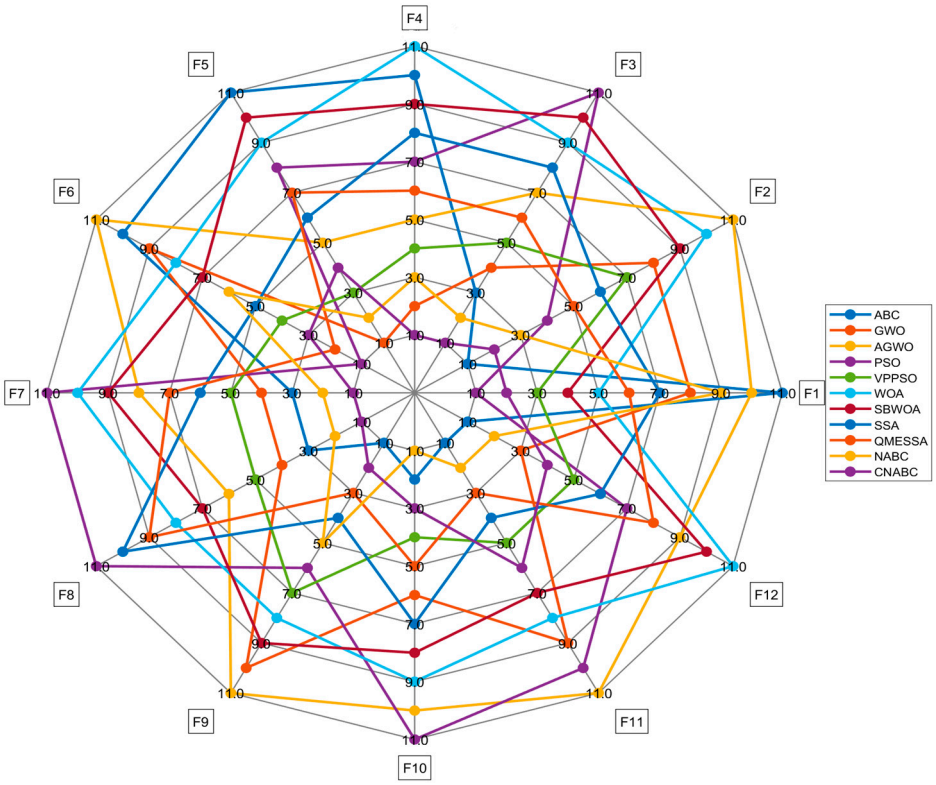


Figure 7. Radar ranking chart of CEC2022 test set for each algorithm (20 dim).

The data analysis of CNABC were compared with other comparative algorithms in different dimensions on two different test sets. Overall, the improved algorithm in this article had good optimization performance and good competitiveness. However, we can see that on both test sets, as the problem dimension increased, the comprehensive advantage of CNABC also slightly decreased. Therefore, further research is needed to enhance the search performance of CNABC in high-dimensional problems.

In the above analysis, the convergence graphs of each algorithm in 10 dimensions are recorded, as shown in Figure 2; the box plot is shown in Figure 3; and the radar chart of algorithm ranking is shown in Figure 4. The convergence graph of the 20 dimensional algorithm is recorded here, as shown in Figure 5; the box plot, as shown in Figure 6; and the radar chart of the algorithm ranking, as shown in Figure 7. From the convergence chart of 10 dimensions, C01, C03, and C011 demonstrated excellent performance; however, it can also be observed that C06 and C10 performed inadequately. From the convergence chart of 20 dimensions, C01, C03, C04, and C07 exhibited outstanding performance, but it was evident that C05, C11, and C12 showed insufficient performance. This further substantiates that CNABC is in line with the NFL theorem, which stipulates that an algorithm cannot have exceptional performance on all issues. From the data of algorithm testing, the average value, variance, box plot, and convergence plot of the algorithm demonstrated the performance of CNABC in algorithm iteration, the stability of algorithm optimization, and the search ability of the algorithm during iteration. The Friedman rank sum test analysis of algorithm differences reflected the difference between CNABC and comparison algorithms, but it was still difficult to judge the poor habits of some comparison algorithms. Therefore, this article also added the Wilcoxon rank sum test to further analyze and compare the significant differences in the algorithm. The population size was 100, and each algorithm ran independently 30 times under D $\times$ 10,000 maximum evaluation times. Tables 4 and 5

show the test results of each algorithm at the p = 5% significance level under different dimensions of the two test set functions. When p < 5%, it is considered that there is a significant difference in algorithm time. When p > 5%, it is considered that the difference between the two algorithms is not significant, NaN indicates that the difference in results between the two algorithms cannot be compared, and the underlined data indicate that p > 5%.

As can be seen from Tables 4 and 5, the number of occurrences of NaN in both tables is 0, and there is less underlined data in the functions of the two test sets. This implies that CNABC and the comparison algorithm had lower similarity in search results in different dimensions of the CEC2022 test set, and there was a significant difference in optimization results. Therefore, it can be seen that there was still a significant difference in optimization performance between CNABC and the comparison algorithm. Therefore, based on the comprehensive analysis of the test set function results, rank ranking, convergence function graph, and box plot in Tables 2 and 3, we can see that the comprehensive performance of CNABC was the most outstanding among many algorithms. Therefore, it can be verified that CNABC has certain core competitiveness and better optimization performance compared with other algorithms.

In addition, to evaluate the effectiveness of the proposed strategy for problem solving, this paper designed four algorithms: CNABC, ABC, and ABC-1, which utilize only a commutative solution mechanism in their search processes; and ABC-2, which employs solely dynamic neighbor learning during the search. During the experiment, variables were rigorously controlled with a population size set at 100 and a maximum evaluation budget of Dim × 10,000. Each algorithm was executed independently for 30 iterations, from which the average and variance of the results were computed. The test outcomes for dimensions 10 and 20 are presented in Tables 6 and 7.

Table 6. Contribution experiments of each strategy in CEC2022 (10 dim).

Function	Index	ABC	CNABC-1	CNABC-2	CNABC
C01	Mean	8.2546E+03	8.6449E+03	6.6300E+03	**1.1973E-03**
	Std	1.8719E+03	2.4694E+03	1.7684E+03	**4.3849E-03**
C02	Mean	8.9340E-01	**6.1016E-01**	1.0583E+00	7.7435E+00
	Std	9.2956E-01	**4.0496E-01**	1.6864E+00	1.7176E+01
C03	Mean	1.6938E-02	1.7948E-02	2.3692E-09	**9.8529E-14**
	Std	6.5686E-03	7.1088E-03	2.0757E-09	**3.8646E-14**
C04	Mean	2.2731E+01	2.3656E+01	1.3585E+01	**1.0911E+01**
	Std	4.7129E+00	5.2725E+00	**3.2801E+00**	3.9254E+00
C05	Mean	2.3155E+02	2.0214E+02	2.9538E+01	**6.3937E-02**
	Std	1.2641E+02	1.1178E+02	1.8018E+01	**1.9356E-01**
C06	Mean	1.8419E+03	1.2539E+03	**4.1614E+02**	8.3687E+02
	Std	1.1742E+03	7.5952E+02	**3.2035E+02**	1.2552E+03
C07	Mean	5.6068E+00	5.4867E+00	**5.2867E-01**	1.8372E+00
	Std	1.9832E+00	2.2070E+00	**5.1254E-01**	4.9007E+00
C08	Mean	1.6429E+01	1.5061E+01	1.5199E+01	**1.3731E+01**
	Std	2.4709E+03	4.9502E+00	**3.9161E+00**	9.3017E+00
C09	Mean	**1.8472E+02**	1.8788E+02	2.3500E+02	2.2928E+02
	Std	3.3988E+01	2.0674E+01	3.0032E+00	**1.4211E-13**
C10	Mean	**3.9832E+00**	3.2194E+00	1.6798E+01	1.0022E+02
	Std	3.2375E+00	3.2337E+00	2.6190E+01	**5.5811E-02**
C11	Mean	4.2985E+01	4.1253E+01	4.8308E+00	**4.0927E-13**
	Std	2.2436E+01	2.3979E+01	2.4598E+01	**1.3642E-13**
C12	Mean	1.4825E+02	**1.4802E+02**	1.6575E+02	1.6292E+02
	Std	**6.2750E-01**	6.7384E-01	7.2086E-01	1.9242E+00

Table 7. Contribution experiments of each strategy in CEC2022 (20 dim).

Function	Index	ABC	CNABC-1	CNABC-2	CNABC
C01	Mean	3.4642E+04	3.4690E+04	2.6389E+04	**6.9833E+02**
	Std	4.2200E+03	5.4458E+03	3.6725E+03	**5.6114E+02**
C02	Mean	**1.5929E+01**	1.6771E+01	4.9083E+01	4.7313E+01
	Std	**1.8484E+00**	8.9264E-01	9.5493E-02	2.4233E+01
C03	Mean	3.4459E-02	3.3505E-02	7.7464E-08	**5.5911E-10**
	Std	1.1165E-02	8.8402E-03	3.4143E-08	**7.2191E-10**
C04	Mean	1.1296E+02	1.1210E+02	8.1940E+01	**3.4144E+01**
	Std	1.9159E+01	2.1172E+01	1.2208E+01	**4.8546E+00**
C05	Mean	3.1665E+03	3.2314E+03	1.5164E+03	**3.2107E+02**
	Std	7.5259E+02	6.8987E+02	4.4258E+02	**2.4450E+02**
C06	Mean	5.0428E+04	5.1701E+04	2.3538E+04	**2.7414E+03**
	Std	2.9475E+04	2.4541E+04	1.5881E+04	**2.9384E+03**
C07	Mean	5.6819E+01	5.4296E+01	3.2103E+01	**2.8564E+01**
	Std	8.5524E+00	8.5808E+00	4.2517E+00	**3.5577E+00**
C08	Mean	2.5000E+01	2.5078E+01	2.4429E+01	**2.1493E+01**
	Std	7.6912E-01	**7.1254E-01**	5.6114E-01	3.0215E+00
C09	Mean	1.6597E+02	**1.6589E+02**	1.8112E+02	1.8078E+02
	Std	3.8039E-01	3.2523E-01	2.1197E-01	**2.8422E-14**
C10	Mean	1.4781E+01	1.4277E+01	**3.5878E-01**	9.8298E+01
	Std	5.8370E+00	6.9522E+00	**4.4507E-01**	3.5593E+01
C11	Mean	**1.7419E+02**	1.6484E+02	3.0000E+02	3.3640E+02
	Std	5.6744E+01	5.3780E+01	**0.0000E+00**	7.8303E+01
C12	Mean	**2.0000E+02**	**2.0000E+02**	2.4599E+02	2.6224E+02
	Std	**7.9512E-05**	9.2856E-05	3.7484E+00	2.0616E+01

From Table 6, it can be seen that CNABC had the most bold individuals, ranking first in the number of optimal values. It can be seen that CNABC exhibited excellent optimization performance on multiple types of test functions. Looking at the comprehensive data of ABC-1 and ABC-2, it can be seen that both improved compared with ABC. However, the overall improvement of ABC-1 was relatively small compared with ABC-2. However, the comprehensive data of ABC-2 was worse than CNABC, indicating that the substitution mechanism can compensate for the shortcomings of dynamic neighbor learning when trapped in local optima. Therefore, it can be concluded that the strategy proposed in the 10 dimensions has good performance.

From Table 7, it can be seen that CNABC had the best overall results with the most bold font, and ABC-1 and ABC-2 were not significantly different from the 10 dimensional analysis compared with ABC. At 20 dimensions, ABC-1 can better maintain the advantages of ABC and had an improvement compared with ABC, while ABC-2 had a good effect on C03, C08, and C10 compared with ABC. Although ABC-1 and ABC-2 showed good improvements in other functions, overall, CNABC, which combined the advantages of both strategies, still outperformed. Overall, CNABC performed better than single strategy ABC in both the 10 and 20 dimensions.

4.2. Analysis of the Running Time of Each Algorithm

The time complexity of the algorithm was analyzed to prove that the time complexity of CNABC remained unchanged. In this section, the time complexity analysis of CNABC and the comparison algorithm is included. Compared with the original ABC, CNABC is of the same order of magnitude in terms of time consumption, with a minor difference. Compared with the time consumption of other comparison algorithms, the time consumed

by CNABC is in the upper-middle position, which takes less time and is more efficient. The time consumption chart is presented in Table 8.

Table 8. Time consumption table of each algorithm.

Function	dim	ABC	GWO	AGWO	PSO	VPPSO	WOA	SBWOA	SSA	QMESSA	NABC	CNABC
C01	10	6.66E-01	7.67E-01	1.30E+00	1.09E+00	6.40E-01	6.86E-01	6.57E-01	6.00E-01	5.69E-01	8.63E-01	7.88E-01
	20	1.38E+00	1.76E+00	3.92E+00	1.98E+00	1.53E+00	1.37E+00	1.40E+00	5.79E-01	8.30E-01	1.75E+00	1.70E+00
C02	10	6.61E-01	7.46E-01	1.30E+00	9.48E-01	6.46E-01	6.60E-01	6.58E-01	5.64E-01	4.27E-01	8.67E-01	7.79E-01
	20	1.38E+00	1.75E+00	3.82E+00	1.95E+00	1.42E+00	1.37E+00	1.42E+00	5.79E-01	8.28E-01	1.75E+00	1.64E+00
C03	10	8.40E-01	9.19E-01	1.47E+00	1.13E+00	7.40E-01	8.32E-01	8.25E-01	6.71E-01	4.81E-01	9.84E-01	9.54E-01
	20	2.10E+00	2.46E+00	4.52E+00	2.67E+00	1.81E+00	2.07E+00	2.09E+00	8.01E-01	9.38E-01	2.22E+00	2.37E+00
C04	10	7.04E-01	7.86E-01	1.33E+00	9.86E-01	6.47E-01	7.02E-01	6.92E-01	5.84E-01	4.34E-01	8.88E-01	8.18E-01
	20	1.53E+00	1.95E+00	3.99E+00	2.09E+00	1.45E+00	1.52E+00	1.58E+00	6.34E-01	8.59E-01	1.86E+00	1.80E+00
C05	10	7.21E-01	8.06E-01	1.35E+00	1.08E+00	6.63E-01	7.26E-01	7.08E-01	6.00E-01	4.53E-01	8.96E-01	8.33E-01
	20	1.56E+00	2.00E+00	4.04E+00	2.18E+00	1.48E+00	1.57E+00	1.58E+00	6.45E-01	8.75E-01	1.89E+00	1.85E+00
C06	10	6.77E-01	7.61E-01	1.32E+00	9.64E-01	6.49E-01	6.74E-01	6.69E-01	5.73E-01	4.23E-01	8.71E-01	8.07E-01
	20	1.43E+00	1.82E+00	3.87E+00	2.03E+00	1.38E+00	1.42E+00	1.43E+00	6.00E-01	8.41E-01	1.80E+00	1.68E+00
C07	10	8.81E-01	9.63E-01	1.52E+00	1.16E+00	7.70E-01	8.79E-01	8.76E-01	6.98E-01	4.93E-01	1.00E+00	1.01E+00
	20	2.22E+00	2.59E+00	4.63E+00	2.78E+00	1.85E+00	2.20E+00	2.21E+00	8.33E-01	9.68E-01	2.35E+00	2.49E+00
C08	10	9.36E-01	1.01E+00	1.57E+00	1.22E+00	7.97E-01	9.37E-01	9.19E-01	7.26E-01	4.98E-01	1.04E+00	1.07E+00
	20	2.36E+00	2.74E+00	4.80E+00	2.93E+00	1.96E+00	2.35E+00	2.32E+00	8.77E-01	9.92E-01	2.44E+00	2.64E+00
C09	10	8.51E-01	9.47E-01	1.50E+00	1.14E+00	7.87E-01	8.56E-01	8.54E-01	6.98E-01	4.78E-01	9.75E-01	9.75E-01
	20	2.25E+00	2.65E+00	4.80E+00	2.84E+00	2.00E+00	2.25E+00	2.25E+00	8.78E-01	9.65E-01	2.36E+00	2.52E+00
C10	10	8.34E-01	9.17E-01	1.47E+00	1.11E+00	7.44E-01	8.23E-01	8.08E-01	6.60E-01	4.69E-01	9.63E-01	9.54E-01
	20	2.00E+00	2.37E+00	4.45E+00	2.54E+00	1.75E+00	2.00E+00	1.96E+00	7.87E-01	9.42E-01	2.20E+00	2.27E+00
C11	10	9.44E-01	1.03E+00	1.59E+00	1.27E+00	8.04E-01	9.40E-01	9.29E-01	7.38E-01	5.03E-01	1.04E+00	1.07E+00
	20	2.49E+00	2.87E+00	4.89E+00	3.02E+00	2.00E+00	2.47E+00	2.48E+00	9.30E-01	1.01E+00	2.50E+00	2.77E+00
C12	10	9.87E-01	1.07E+00	1.63E+00	1.26E+00	9.72E-01	9.85E-01	9.72E-01	7.84E-01	5.21E-01	1.07E+00	1.17E+00
	20	2.73E+00	3.06E+00	5.15E+00	3.21E+00	2.48E+00	2.66E+00	2.66E+00	9.97E-01	1.03E+00	2.67E+00	3.02E+00

4.3. Effectiveness Analysis of Neighbor Population

A new parameter, namely the size of the neighbor population N, was incorporated into the DNL strategy in CNABC. To comprehend what the most suitable size of N was in the new algorithm, we added parameter analysis experiments for parameter N in this section to better determine the value of N for the optimal algorithm performance, where N is selected from the set {2, 3,. . .15}. Finally, we set the identical parameters as per Section 4.1; the test function was CEC2022; the dimension was 10; and a Friedman test was conducted to select the most appropriate N from a statistical perspective.

It can be observed from the results that, for CNABC, the value of N influenced the algorithm's outcomes. We analyzed that, when the value of N was small, the small learning population reduced the diversity of the population, leading to the decline of its search ability. When the value of N was large, the value was more similar to the global search, and the development ability of CNABC was diminished. The results of Friedman's test are similar. It indicates that the value of N had a negligible impact on the overall optimization result. Therefore, we chose N with a higher ranking as the most appropriate value, that is, when N = 5. In conclusion, when N = 5, the development ability and population diversity of the algorithm will not be affected by an overly large or small N. The experimental results of different N values under CEC2022 are shown in Table 9.

Table 9. Experimental result data with different N values (dim = 10).

Function	Index	2	3	4	5	6	7	8
C01	Mean	4.89E-04	8.89E-04	**3.41E-05**	1.20E-03	3.12E-03	1.24E-03	2.14E-04
	Std	3.58E-04	2.22E-03	5.40E-05	4.38E-03	8.83E-03	3.86E-03	6.62E-04
C02	Mean	8.71E+00	6.52E+00	2.89E+00	7.74E+00	**2.62E+00**	3.42E+00	1.75E+00
	Std	2.03E+01	3.92E+00	1.97E+00	1.72E+01	3.01E+00	3.47E+00	3.03E+00
C03	Mean	1.14E-13	1.14E-13	1.42E-07	**9.85E-14**	1.14E-13	1.14E-13	1.20E-06
	Std	0.00E+00	0.00E+00	4.49E-07	3.86E-14	0.00E+00	0.00E+00	3.79E-06
C04	Mean	**7.96E+00**	1.13E+01	8.26E+00	1.09E+01	9.35E+00	1.28E+01	1.09E+01
	Std	2.77E+00	4.35E+00	2.85E+00	3.93E+00	5.27E+00	4.44E+00	4.40E+00
C05	Mean	1.36E-01	**5.68E-14**	9.98E-02	6.39E-02	9.09E-02	4.54E-02	1.36E-01
	Std	8.00E+00	5.99E-14	2.11E-01	1.94E-01	1.92E-01	1.44E-01	3.07E-01
C06	Mean	1.17E+03	1.33E+03	1.41E+03	**8.37E+02**	1.30E+03	1.35E+03	1.26E+03
	Std	4.95E+02	1.06E+03	1.54E+03	1.26E+03	1.48E+03	1.95E+03	1.56E+03
C07	Mean	2.30E+00	3.62E-01	4.24E-01	1.84E+00	4.61E-01	**2.62E-01**	2.46E+00
	Std	6.23E+00	6.05E-01	4.66E-01	4.90E+00	4.98E-01	4.33E-01	6.19E+00
C08	Mean	1.16E+01	1.04E+01	1.06E+01	1.37E+01	**1.29E+01**	1.23E+01	1.66E+01
	Std	9.63E+00	1.04E+01	1.02E+01	9.30E+00	9.56E+00	1.03E+01	7.70E+00
C09	Mean	**2.29E+02**	**2.29E+02**	**2.29E+02**	**2.29E+02**	**2.29E+02**	**2.29E+02**	**2.29E+02**
	Std	0.00E+00	0.00E+00	0.00E+00	1.42E-13	0.00E+00	0.00E+00	0.00E+00
C10	Mean	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**	1.11E+02	**1.00E+02**
	Std	4.17E-02	4.89E-02	6.67E-02	5.58E-02	3.95E-02	3.44E+01	7.21E-02
C11	Mean	5.46E-13	4.55E-13	4.55E-13	**4.09E-13**	1.50E+01	5.00E-13	**4.09E-13**
	Std	1.92E-13	0.00E+00	2.14E-13	1.36E-13	4.76E+01	1.44E-13	1.44E-13
C12	Mean	1.62E+02	1.63E+02	1.63E+02	1.63E+02	1.64E+02	1.63E+02	1.63E+02
	Std	1.75E+00	1.51E+00	1.88E+00	1.92E+00	1.43E+00	1.69E+00	2.02E+00
Optimal number		3	3	3	5	4	2	3
Function	**Index**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
C01	Mean	1.17E-04	2.71E-04	1.67E-04	3.25E-04	4.40E-03	1.41E-03	2.93E-04
	Std	2.68E-04	4.59E-04	4.90E-04	4.74E-04	4.61E-03	1.04E-03	3.01E-04
C02	Mean	3.74E+00	3.90E+00	9.72E+00	3.35E+00	1.09E+01	2.66E+00	5.52E+00
	Std	2.47E+00	3.26E+00	2.18E+01	2.69E+00	2.12E+01	3.72E+00	2.47E+00
C03	Mean	1.02E-13	1.14E-13	1.13E-06	1.14E-13	2.02E-05	2.83E-07	2.83E-07
	Std	3.60E-14	0.00E+00	3.58E-06	0.00E+00	6.38E-05	8.96E-07	8.96E-07
C04	Mean	9.95E+00	1.22E+01	1.11E+01	1.36E+01	1.01E+01	1.17E+01	1.02E+01
	Std	3.35E+00	6.28E+00	4.14E+00	6.68E+00	3.53E+00	4.11E+00	5.12E+00
C05	Mean	9.09E-14	1.18E-01	4.48E-02	9.09E-02	6.33E-02	1.79E-02	4.38E-01
	Std	4.79E-14	2.26E-01	4.62E-02	1.92E-01	1.48E-01	5.66E-02	1.23E+00
C06	Mean	1.26E+03	1.47E+03	1.11E+03	8.68E+02	1.12E+03	1.27E+03	1.03E+03
	Std	1.50E+03	1.86E+03	1.15E+03	1.10E+03	1.15E+03	1.71E+03	1.34E+03
C07	Mean	2.86E+00	6.22E-01	7.47E-01	1.23E+00	5.97E-01	4.52E+00	6.30E+00
	Std	6.20E+00	5.87E-01	6.15E-01	5.67E-01	5.14E-01	8.17E+00	9.46E+00
C08	Mean	1.23E+01	**9.55E+00**	1.42E+01	1.63E+01	1.73E+01	1.22E+01	1.76E+01
	Std	1.03E+01	9.96E+00	9.50E+00	8.19E+00	6.44E+00	9.48E+00	6.97E+00
C09	Mean	**2.29E+02**	**2.29E+02**	**2.29E+02**	**2.29E+02**	**2.29E+02**	**2.29E+02**	**2.29E+02**
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
C10	Mean	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**	1.12E+02	**1.00E+02**
	Std	4.75E-02	6.00E-02	4.77E-02	3.85E-02	7.49E-02	3.58E+01	6.36E-02
C11	Mean	1.50E+01	4.55E-13	1.50E+01	1.50E+01	**4.09E-13**	1.50E+01	1.50E+01
	Std	4.76E+01	0.00E+00	4.76E+01	4.76E+01	1.44E-13	4.76E+01	4.76E+01
C12	Mean	**1.62E+02**	1.63E+02	1.63E+02	1.63E+02	1.64E+02	1.63E+02	1.63E+02
	Std	1.50E+00	2.00E+00	1.06E+00	2.18E+00	1.39E+00	2.16E+00	1.74E+00
Optimal number		3	2	2	2	3	1	2

5. Comparative Experiment with TOP Algorithm

Based on the experimental analysis in the previous section, CNABC has certain core competitiveness and better optimization performance compared with other algorithms. In order to further demonstrate the competitiveness of CNABC, this section will conduct experimental comparisons between CNABC and the TOP algorithm proposed in recent years in the CEC competition, in order to demonstrate CNABC's optimization ability and look forward to future work directions that need to be supplemented. The algorithms include IMODE [44], AGSK [45], AGSKIMODE [46], and EA4eig [47]. IMODE is the top

ranked algorithm in CEC2020, AGSK is the second ranked algorithm, AGSK-IMODE is the fourth ranked algorithm in CEC 2021, and EA4eig is the first ranked algorithm in CEC 2022. The parameters of each algorithm are shown in the original text, and while the experimental parameters remain unchanged, the maximum evaluation frequency FES is still Dim × 10,000. The specific optimization results are shown in Tables 10 and 11, and the radar charts for the 10 and 20 dimensions are shown in Figures 8 and 9. The rank sum test value is p, and data results less than E-10 are considered 0.

Table 10. Comparison of experimental results with TOP algorithm (10 dim).

Function	Index	AGSK	APGSK-IMODE	EA4eig	IMODE	CNABC
C01	Mean	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	1.1973E-03
	Std	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	4.3849E-03
	Rank	1	1	1	1	2
	P	7.57E-12	1.77E-11	1.21E-12	1.21E-12	
C02	Mean	4.3152E+00	1.6766E-06	6.6443E-01	0.0000E+00	7.7435E+00
	Std	1.2296E+00	6.4403E-06	1.4857E+00	0.0000E+00	1.7176E+01
	Rank	4	2	3	1	5
	P	5.29E-01	2.92E-11	3.52E-09	1.19E-12	
C03	Mean	2.1819E-04	0.0000E+00	0.0000E+00	2.2624E-06	0.0000E+00
	Std	2.4169E-04	0.0000E+00	0.0000E+00	2.2078E-06	0.0000E+00
	Rank	3	1	1	2	1
	P	4.08E-12	2.25E-04	1.97E-11	4.08E-12	
C04	Mean	9.4244E+00	4.8590E+00	1.0281E+00	8.8883E+00	1.0911E+01
	Std	1.9325E+00	1.4671E+00	9.4332E-01	2.1483E+00	3.9254E+00
	Rank	4	2	1	3	5
	P	5.55E-02	3.48E-07	2.28E-11	3.14E-02	
C05	Mean	0.0000E+00	0.0000E+00	0.0000E+00	1.8424E+00	6.3937E-02
	Std	0.0000E+00	0.0000E+00	0.0000E+00	3.0732E+00	1.9356E-01
	Rank	1	1	1	3	2
	P	8.39E-03	1.20E-05	1.20E-05	8.41E-09	
C06	Mean	3.2980E-01	4.0670E-01	4.4834E-02	1.6979E+00	8.3687E+02
	Std	1.1129E-01	1.1401E-01	9.0058E-02	1.4846E+00	1.2552E+03
	Rank	2	3	1	4	5
	P	3.02E-11	3.01E-11	3.02E-11	3.02E-11	
C07	Mean	3.8735E-01	4.3631E-04	2.0934E-10	5.4350E-04	1.8372E+00
	Std	5.5332E-01	9.4390E-04	6.7468E-10	5.7188E-04	4.9007E+00
	Rank	4	2	1	3	5
	P	7.84E-01	4.45E-04	2.11E-11	4.51E-02	
C08	Mean	3.8053E+00	1.1567E+00	3.3965E-01	2.3765E+00	1.3731E+01
	Std	2.4133E+00	1.1865E+00	2.1644E-01	1.9064E+00	9.3017E+00
	Rank	4	2	1	3	5
	P	1.44E-02	1.86E-03	1.87E-05	4.23E-03	
C09	Mean	2.2928E+02	2.2928E+02	1.8550E+02	2.1400E+02	2.2928E+02
	Std	8.5265E-14	8.5265E-14	3.4654E-13	5.7194E+01	1.4211E-13
	Rank	3	3	1	2	3
	P	1.69E-14	1.69E-14	8.64E-14	6.91E-10	
C10	Mean	1.0029E+02	1.0026E+02	1.0019E+02	8.2928E+01	1.0022E+02
	Std	3.1802E-02	4.6911E-02	4.0734E-02	3.4986E+01	5.5811E-02
	Rank	5	4	2	1	3
	P	7.22E-06	6.10E-03	7.01E-02	3.85E-03	
C11	Mean	0.0000E+00	0.0000E+00	0.0000E+00	3.7651E-08	0.0000E+00
	Std	0.0000E+00	0.0000E+00	0.0000E+00	2.3221E-08	0.0000E+00
	Rank	1	1	1	2	1
	P	2.20E-09	5.35E-07	3.94E-12	1.14E-09	
C12	Mean	1.6101E+02	1.6246E+02	1.4719E+02	1.6384E+02	1.6292E+02
	Std	9.2072E-01	1.0661E+00	3.2503E+00	7.0591E-01	1.9242E+00
	Rank	2	3	1	5	4
	P	1.47E-06	2.40E-01	3.64E-11	3.14E-02	

Table 11. Comparison of experimental results with TOP algorithm (20 dim).

Function	Index	AGSK	APGSK-IMODE	EA4eig	IMODE	CNABC
C01	Mean	2.4887E-09	5.3677E+02	0.0000E+00	0.0000E+00	6.9833E+02
	Std	1.1681E-08	2.6484E+02	0.0000E+00	0.0000E+00	5.6114E+02
	Rank	2	3	1	1	4
	P	2.99E-11	5.59E-01	1.21E-12	1.21E-12	
C02	Mean	4.8945E+01	4.6162E+01	5.2945E+00	3.4493E+01	4.7313E+01
	Std	7.5195E-01	9.2179E+00	2.9091E+00	2.2299E+01	2.4233E+01
	Rank	5	3	1	2	4
	P	9.53E-04	1.59E-03	3.01E-07	3.52E-05	
C03	Mean	6.1868E-03	7.6936E-07	0.0000E+00	1.2810E+00	0.0000E+00
	Std	4.2229E-03	5.8359E-07	0.0000E+00	8.1892E-01	0.0000E+00
	Rank	4	3	1	5	1
	P	3.02E-11	3.01E-11	1.21E-12	3.02E-11	
C04	Mean	3.7751E+01	2.9132E+01	1.1144E+01	6.0063E+01	3.4144E+01
	Std	5.2045E+00	5.5429E+00	4.4228E+00	1.1524E+01	4.8546E+00
	Rank	4	2	1	5	3
	P	2.07E-02	9.02E-04	2.98E-11	5.49E-11	
C05	Mean	4.9963E-01	4.8880E+00	0.0000E+00	7.1216E+02	3.2107E+02
	Std	3.4276E-01	4.5295E+00	0.0000E+00	2.7596E+02	2.4450E+02
	Rank	2	3	1	5	4
	P	3.02E-11	3.01E-11	1.21E-12	2.00E-06	
C06	Mean	3.8645E+01	1.1731E+03	6.7614E-01	2.2384E+01	2.7414E+03
	Std	1.4041E+01	8.9562E+02	5.5963E-01	6.6968E+00	2.9384E+03
	Rank	3	4	1	2	5
	P	3.02E-11	1.26E-01	3.02E-11	3.02E-11	
C07	Mean	2.9975E+01	2.4774E+01	1.2557E+01	3.4001E+01	2.8564E+01
	Std	2.0967E+00	3.1719E+00	8.9337E+00	4.1091E+00	3.5577E+00
	Rank	4	2	1	5	3
	P	1.15E-01	3.59E-05	3.47E-10	5.46E-06	
C08	Mean	2.4295E+01	2.2690E+01	1.9660E+01	2.1835E+01	2.1493E+01
	Std	3.3946E-01	5.9443E-01	4.2501E+00	6.4596E-01	3.0215E+00
	Rank	5	4	1	2	3
	P	5.57E-10	6.11E-10	2.07E-02	2.68E-06	
C09	Mean	1.8078E+02	1.8078E+02	1.6534E+02	1.8078E+02	1.8078E+02
	Std	8.5265E-14	5.0618E-07	3.9794E-09	1.0012E-07	2.8422E-14
	Rank	2	2	1	2	2
	P	1.21E-12	9.75E-10	3.01E-11	6.74E-06	
C10	Mean	1.0040E+02	1.0053E+02	1.0412E+02	1.0062E+02	9.8298E+01
	Std	4.7991E-02	6.8048E-02	2.0291E+01	6.8914E-02	3.5593E+01
	Rank	2	3	5	4	1
	P	3.65E-08	6.12E-10	3.77E-04	5.57E-10	
C11	Mean	3.5000E+02	3.0000E+02	3.3000E+02	2.8046E+02	3.3640E+02
	Std	5.0000E+01	6.4438E-10	4.5826E+01	7.4998E+01	7.8303E+01
	Rank	5	2	3	1	4
	P	7.39E-02	2.05E-03	1.40E-04	1.44E-02	
C12	Mean	2.3987E+02	2.3896E+02	1.9971E+02	2.5429E+02	2.6224E+02
	Std	2.6782E+00	2.6503E+00	1.5617E+00	6.5203E+00	2.0616E+01
	Rank	3	2	1	4	5
	P	4.87E-11	4.97E-11	3.02E-11	3.63E-01	

From the data results in Tables 10 and 11, it can be seen that the competition algorithm in CEC performs significantly better in the same experimental environment, while, in terms of the comprehensive ranking of CNABC, the 20-dimensional algorithm is lower than the 10-dimensional algorithm and is relatively backward.

From Figures 8 and 9, it can be seen that the top three functions of CNABC in the 10-dimensional rank are C01, C03, C05, C09, C10, and C11, while the top three functions in the 20-dimensional rank are C03, CO4, C07, C08, C09, and C11. Among these functions, it can be seen that CNABC's search ability is commendable. The TOP algorithm is not significantly different from C03 and C11 in terms of optimization, but we acknowledge that the search ability of the TOP algorithm is indeed better than that of CNABC. However,

CNABC also has good optimization capabilities in some functions, which is also what we need to see.

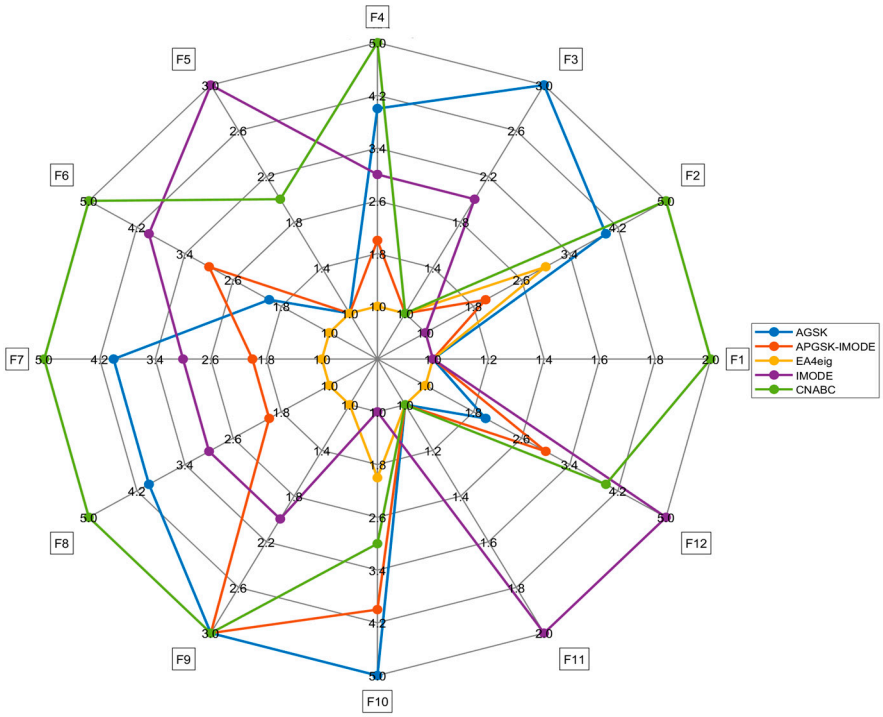


Figure 8. Radar ranking chart of the competition algorithm CEC2022 test set (10 dim).

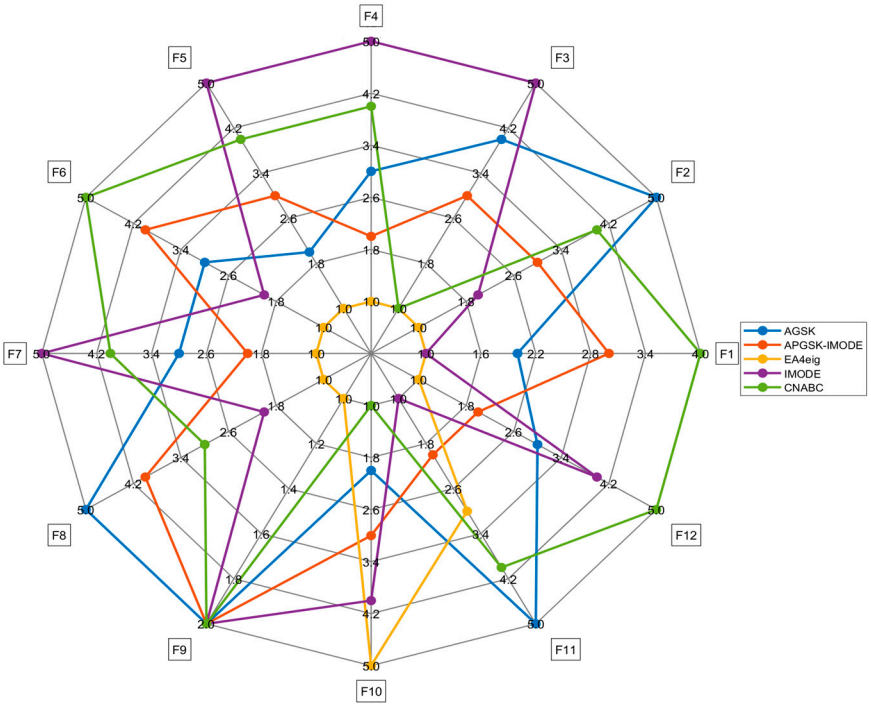


Figure 9. Radar ranking chart of the competition algorithm CEC2022 test set (10dim).

6. Conclusions

To assess the comprehensive performance of the enhanced algorithm CNABC, this study undertook experiments in both 10-dimensional and 20-dimensional spaces using the CEC2022 test set. The CNABC was contrasted with 10 different algorithms, with

parameters for each algorithm configured according to their respective original literature. Within the CEC2022 test set, F1 is a unimodal function, F2–F5 is a multimodal function based on F2–F5, F6–F8 is a hybrid function, and F9–F12 is a combination function. The two test sets contain the same function types and can evaluate the algorithm's search ability and overall robustness. This study presents a dynamic neighbor artificial bee colony algorithm (CNABC) based on commutative solutions. In CNABC, bee leaders are updated in the three stages of the original bee population, and the randomness of randomly selected hired bees as leaders is reduced by selecting the optimal individuals in a small population. The best leader in a small range is maintained as a highlight to improve ABC. Secondly, upon the completion of the three stages, a novel approach is introduced to reduce the risk of the worst solution falling into local or global optima by introducing candidate solutions through the exchange solution mechanism. Next, to verify the search ability of the improved CNABC, experiments were conducted on the CEC2022 test sets. Based on the experimental results, it was verified that CNABC possesses excellent search performance and certain competitiveness in an algorithm through mean and standard deviation. The box plot can reflect that CNABC's stability performance in the comparison algorithm is good, and there is no significant unreasonable existence, The Wilcoxon rank sum test and Friedman test show that there is a difference in the optimization results between CNABC and the comparison algorithm, that is, the optimization method has novelty, which comprehensively verifies the effectiveness, novelty, and competitiveness of the improved CNABC.

Although CNABC has shown good performance in the optimization search aspect in the comparison algorithm, it is still possible to know the limitations of CNABC. It satisfies the NFL theorem and cannot show good performance in many problems. This shows the obvious shortcomings of CNABC. First, there is still room for improvement in the convergence speed and optimization accuracy in the 10-dimensional C06 and C10, and the 20-dimensional C05, C11, and C12 in the two test sets. Secondly, there is still a gap compared with the TOP algorithm in the CEC competition, especially in terms of combinatorial functions. Therefore, the focus of future work will be on the following two aspects:

- On the basis of CNABC, let it shine on composite functions, no longer limited to the other three functions, but more on problem functions;
- Explore and design more transformation search mechanisms to further improve the algorithm's optimization ability and the ability to jump out of local optima.

Author Contributions: Conceptualization, C.O. and Y.G.; methodology, C.O. and Y.G.; software, Y.G.; validation, Y.G. and D.Z.; formal analysis, Y.G. and D.Z.; investigation, Y.G. and D.Z.; resources, H.G. and D.Z.; data curation, Y.G. and D.Z.; writing—original draft preparation, Y.G.; writing—review and editing, H.G. and D.Z.; visualization, Y.G. and D.Z.; supervision, H.G. and C.Z.; project administration, C.Z.; funding acquisition, C.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Nos. 62272418, 62102058), and the basic public welfare research program of Zhejiang Province (No. LGG18E050011).

Data Availability Statement: All data for this study are available from the corresponding author.

Acknowledgments: The authors thank the reviewers and editors for their valuable suggestions, which substantially improve this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhu, D.; Huang, Z.; Liao, S.; Zhou, C.; Yan, S.; Chen, G. Improved Bare Bones Particle Swarm Optimization for DNA Sequence Design. *IEEE Trans. Nanobiosci.* **2022**, *22*, 603–613. [CrossRef] [PubMed]
2. Sun, G.; Han, R.; Deng, L.; Li, C.; Yang, G. Hierarchical structure-based joint operations algorithm for global optimization. *Swarm Evol. Comput.* **2023**, *79*, 101311. [CrossRef]

3. Ma, C.; Huang, H.; Fan, Q.; Wei, J.; Du, Y.; Gao, W. Grey wolf optimizer based on Aquila exploration method. *Expert Syst. Appl.* **2022**, *205*, 117625. [CrossRef]
4. Hassan, M.H.; Houssein, E.H.; Mahdy, M.A.; Kamel, S. An improved Manta ray foraging optimizer for cost-effective emission dispatch problems. *Eng. Appl. Artif. Intell.* **2021**, *100*, 104155. [CrossRef]
5. Wang, J.; Zhang, Y.; Hong, M.; He, H.; Huang, S. A self-adaptive level-based learning artificial bee colony algorithm for feature selection on high-dimensional classification. *Soft Comput.* **2022**, *26*, 9665–9687. [CrossRef]
6. Yu, W.; Kang, H.; Sun, G.; Liang, S.; Li, J. Bio-Inspired Feature Selection in Brain Disease Detection via an Improved Sparrow Search Algorithm. *IEEE Trans. Instrum. Meas.* **2022**, *72*, 2500515. [CrossRef]
7. Jena, B.; Naik, M.K.; Panda, R.; Abraham, A. Maximum 3D Tsallis entropy based multilevel thresholding of brain MR image using attacking Manta Ray foraging optimization. *Eng. Appl. Artif. Intell.* **2021**, *103*, 104293. [CrossRef]
8. Zhang, J.; Zheng, J.; Xie, X.; Lin, Z.; Li, H. Mayfly sparrow search hybrid algorithm for RFID Network Planning. *IEEE Sens. J.* **2022**, *22*, 16673–16686. [CrossRef]
9. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the Mhs95 Sixth International Symposium on Micro Machine & Human Science, Nagoya, Japan, 4–6 October 1995.
10. Dorigo, M. The ant system: An autocatalytic optimizing process. In Proceedings of the First European Conference on Artificial Life, Paris, France, 11–13 December 1991.
11. Yang, X.-S.; Deb, S. Cuckoo Search via Lvy flights. In Proceedings of the 2009 World Congress Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214. [CrossRef]
12. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
13. Xue, J.; Shen, B. A novel swarm intelligence optimization approach: Sparrow search algorithm. *Syst. Sci. Control. Eng.* **2020**, *8*, 22–34. [CrossRef]
14. Zhu, D.; Wang, S.; Zhou, C.; Yan, S.; Xue, J. Human Memory Optimization Algorithm: A memory-inspired optimizer for Global Optimization Problems. *Expert Syst. Appl.* **2024**, *237*, 121597. [CrossRef]
15. Adhikary, J.; Acharyya, S. Randomized Balanced Grey Wolf Optimizer (RBGWO) for solving real life optimization problems. *Appl. Soft Comput.* **2022**, *117*, 108429. [CrossRef]
16. Precup, R.-E.; David, R.-C.; Roman, R.-C.; Szedlak-Stinean, A.-I.; Petriu, E.M. Optimal tuning of interval type-2 fuzzy controllers for nonlinear servo systems using Slime Mould Algorithm. *Int. J. Syst. Sci.* **2021**, *54*, 2941–2956. [CrossRef]
17. Meng, Z.; Zhong, Y.; Mao, G.; Liang, Y. PSO-sono: A novel PSO variant forsingle-objective numerical optimization. *Inf. Sci.* **2022**, *586*, 176–191. [CrossRef]
18. Zhu, D.; Wang, S.; Shen, J.; Zhou, C.; Li, T.; Yan, S. A multi-strategy particle swarm algorithm with exponential noise and fitness-distance balance method for low-altitude penetration in secure space. *J. Comput. Sci.* **2023**, *74*, 102149. [CrossRef]
19. Karaboga, N.; Latifoglu, F. Adaptive filtering noisy transcranial Doppler signal by using artificial bee colony algorithm. *Eng. Appl. Artif. Intell.* **2013**, *26*, 677–684. [CrossRef]
20. Chiang, H.-S.; Sangaiah, A.K.; Chen, M.-Y.; Liu, J.-Y. A novel artificial bee colony optimization algorithm with SVM for bio-inspired software-defined networking. *Int. J. Parallel Program.* **2020**, *48*, 310–328. [CrossRef]
21. Lalwani, S.; Kumar, R. An efficient three-Level parallel ABC algorithm for secondary structure prediction of complex RNA sequences. *Appl. Soft Comput.* **2021**, *99*, 106848. [CrossRef]
22. Naware, D.; Badigenchala, R.; Mitra, A.; Das, D. Impact of demand response on battery energy storage degradation using gbest-guided artificial bee colony algorithm with forecasted solar insolation. *J. Energy Storage* **2022**, *52*, 104915. [CrossRef]
23. Erdem, E.; Aydın, T.; Erkayman, B. Flight scheduling incorporating bad weather conditions through big data analytics: A comparison of metaheuristics. *Expert Syst.* **2021**, *38*, e12752. [CrossRef]
24. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
25. Xiang, W.L.; Li, Y.Z.; He, R.C.; An, M.Q. Artificial bee colony algorithm with a pure crossover operation for binary optimization. *Comput. Ind. Eng.* **2021**, *152*, 107011. [CrossRef]
26. Yavuz, G.; Aydın, D. Improved Self-adaptive Search Equation-based Artificial Bee Colony Algorithm with competitive local search strategy. *Swarm Evol. Comput.* **2019**, *51*, 100582. [CrossRef]
27. Zhou, B.; Zhao, Z. An adaptive artificial bee colony algorithm enhanced by Deep Q-Learning for milk-run vehicle scheduling problem based on supply hub. *Knowl.-Based Syst.* **2023**, *264*, 110367. [CrossRef]
28. Ye, T.; Wang, W.; Wang, H.; Cui, Z.; Wang, Y.; Zhao, J.; Hu, M. Artificial bee colony algorithm with efficient search strategy based on random neighborhood structure. *Knowl.-Based Syst.* **2022**, *241*, 108306. [CrossRef]
29. Zhou, X.; Song, J.; Wu, S.; Wang, M. Artificial bee colony algorithm based on online fitness landscape analysis. *Inf. Sci.* **2023**, *619*, 603–629. [CrossRef]
30. Zhou, X.; Wu, Y.; Zhong, M.; Wang, M. Artificial bee colony algorithm based on adaptive neighborhood topologies. *Inf. Sci.* **2022**, *610*, 1078–1101. [CrossRef]
31. Khan, I.; Maiti, M.K. A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem. *Swarm Evol. Comput.* **2019**, *44*, 428–438. [CrossRef]
32. Zheng, T.; Zhang, H.; Zhang, B.; Cai, Z.; Wang, K.; Todo, Y.; Gao, S. Umbrellalike Hierarchical Artificial Bee Colony Algorithm. *IEICE Trans. Inf. Syst.* **2023**, *106*, 410–418. [CrossRef]

33. Song, X.; Zhao, M.; Yan, Q.; Xing, S. A high-efficiency adaptive artificial bee colony algorithm using two strategies for continuous optimization. *Swarm Evol. Comput.* **2019**, *50*, 100549. [CrossRef]
34. Guan, B.; Xu, T.; Zhao, Y.; Li, Y.; Dong, X. A random grouping-based self-regulating artificial bee colony algorithm for interactive feature detection. *Knowl.-Based Syst.* **2022**, *243*, 108434. [CrossRef]
35. Wang, Y.; Jiao, J.; Liu, J.; Xiao, R. A labor division artificial bee colony algorithm based on behavioral development. *Inf. Sci.* **2022**, *606*, 152–172. [CrossRef]
36. Tao, X.R.; Pan, Q.K.; Gao, L. An efficient self-adaptive artificial bee colony algorithm for the distributed resource-constrained hybrid flowshop problem. *Comput. Ind. Eng.* **2022**, *169*, 108200. [CrossRef]
37. Bilal, M.E. COVID-19 detection based on pre-trained deep networks and LSTM model using X-ray images enhanced contrast with artificial bee colony algorithm. *Expert Syst.* **2022**, *40*, e13185.
38. Peng, H.; Deng, C.; Wu, Z. Best neighbor-guided artificial bee colony algorithm for continuous optimization problems. *Soft Comput.* **2019**, *23*, 8723–8740. [CrossRef]
39. Shami, T.M.; Mirjalili, S.; Al-Eryani, Y.; Daoudi, K.; Izadi, S.; Abualigah, L. Velocity pausing particle swarm optimization: A novel variant for global optimization. *Neural Comput. Appl.* **2023**, *35*, 9193–9223. [CrossRef]
40. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
41. Too, J.; Mafarja, M.; Mirjalili, S. Spatial bound whale optimization algorithm: An efficient high-dimensional feature selection approach. *Neural Comput. Appl.* **2021**, *33*, 16229–16250. [CrossRef]
42. Wu, R.; Huang, H.; Wei, J.; Ma, C.; Zhu, Y.; Chen, Y.; Fan, Q. An improved sparrow search algorithm based on quantum computations and multi-strategy enhancement. *Expert Syst. Appl.* **2023**, *215*, 119421. [CrossRef]
43. Liu, S.-H.; Mernik, M.; Hrnčič, D.; Črepinšek, M. A parameter control method of evolutionary algorithms using exploration and exploitation measures with a practical application for fitting Sovova's mass transfer model. *Appl. Soft Comput.* **2013**, *13*, 3792–3805. [CrossRef]
44. Sallam, K.M.; Elsayed, S.M.; Chakrabortty, R.K.; Ryan, M.J. Improved multi-operator differential evolution algorithm for solving unconstrained problems. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 9–24 July 2020; pp. 1–8. [CrossRef]
45. Mohamed, A.W.; Hadi, A.A.; Awad, N.H. Evaluating the performance of adaptive gainingsharing knowledge based algorithm on CEC 2020 benchmark problems. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 9–24 July 2020; pp. 1–8. [CrossRef]
46. Mohamed, A.W.; Hadi, A.A.; Agrawal, P.; Sallam, K.M.; Mohamed, A.K. Gaining-sharing knowledge based algorithm with adaptive parameters hybrid with imode algorithm for solving cec 2021 benchmark problems. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation, Kraków, Poland, 28 June–1 July 2021; pp. 841–848. [CrossRef]
47. Bujok, P.; Kolenovsky, P. Eigen crossover in cooperative model of evolutionary algorithms applied to CEC 2022 single objective numerical optimization. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation, Padua, Italy, 18–23 July 2022; pp. 1–8. [CrossRef]

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.