mathematics

MDPI

Article

An Artificial Bee Colony Algorithm with Dual Groups and Multiple Strategies Based on Reinforcement Learning

Yang Cao [1,2,3] **and Zilin Li** [1,2,3,*]

[1] School of Computer Science and Engineering, Shenyang Jianzhu University, Shenyang 110168, China; caoyang@sjzu.edu.cn
[2] Liaoning Provincial Key Laboratory of Big Data Management and Analysis of Urban Construction, Shenyang 110168, China
[3] Shenyang Branch of National Special Computer Engineering Technology Research Center, Shenyang 110168, China
* Correspondence: lizilin@stu.sjzu.edu.cn

Abstract

The Artificial Bee Colony (ABC) algorithm is widely used for continuous optimization, but the standard ABC still suffers from insufficient use of neighborhood information, limited adaptability of search behavior, and random restarts in the scout bee phase, which may lead to slow convergence and reduced solution quality on complex problems. To address these limitations, this paper proposes an artificial bee colony algorithm with dual groups and multiple strategies based on reinforcement learning, named RLDMS-ABC. In the employed bee phase, Q-learning is used to adaptively adjust the neighborhood size of each food source according to search feedback, and the best individual selected from the sampled neighborhood guides candidate solution generation. In the onlooker bee phase, selected food sources are divided into elite and ordinary groups according to relative quality, and different search strategies are assigned to balance exploration and exploitation. In the scout bee phase, a guided restart mechanism combining opposition-based learning and the current global best solution is designed to help stagnated individuals escape local optima. Experiments on the CEC 2014 benchmark set show that RLDMS-ABC outperforms several representative ABC variants on most functions in terms of solution quality, convergence speed, and robustness.

Keywords: artificial bee colony algorithm; reinforcement learning; Q-learning; continuous optimization; multiple strategies; neighborhood information

MSC: 90C59; 68W50; 68T05; 68T20

Check for updates

Academic Editor: Jonathan Blackledge

Received: 20 May 2026
Revised: 20 June 2026
Accepted: 21 June 2026
Published: 26 June 2026

Copyright: © 2026 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.

1. Introduction

Continuous optimization problems are widely encountered in engineering design, machine learning, scheduling, path planning, and other scientific and industrial fields. In many practical cases, objective functions are nonlinear, multimodal, high-dimensional, or difficult to describe with accurate mathematical models. Under these conditions, traditional optimization methods that rely on gradient information or strict assumptions may be difficult to apply effectively. Swarm intelligence algorithms provide an alternative solution because of their simple structures, flexible search mechanisms, and strong adaptability. Among them, the Artificial Bee Colony (ABC) algorithm has attracted considerable

attention since it was proposed by Karaboga, owing to its few control parameters, easy implementation, and strong global search ability [1–3].

The standard ABC algorithm simulates the foraging behavior of honey bee colonies through employed bees, onlooker bees, and scout bees [2,3]. Employed bees generate candidate solutions around current food sources, onlooker bees select promising food sources according to shared information, and scout bees replace abandoned food sources to maintain population diversity. Although this framework is simple and effective, the standard ABC algorithm still has several limitations when solving complex continuous optimization problems. First, neighborhood information around food sources is not sufficiently used, and the search range is difficult to adjust according to the search state. Second, the onlooker bee phase usually relies on fixed search patterns, which limits its ability to assign suitable search behaviors to individuals with different qualities. Third, the scout bee phase generally adopts random restarts, which can restore population diversity but do not make full use of useful information accumulated during the search. These limitations may lead to slow convergence and insufficient exploitation.

To improve the performance of ABC, many studies have focused on neighborhood structure design and search strategy enhancement. Wang et al. [4] proposed a neighborhood selection mechanism to strengthen local search capability. Xiao et al. [5] introduced adaptive neighborhood search and Gaussian perturbation to improve exploitation efficiency. Ye et al. [6] developed an ABC algorithm based on a random neighborhood structure, in which each individual is assigned an independent neighborhood size. Zhu and Kwong [7] proposed a global best-guided ABC to enhance exploitation by using the best solution found so far. In addition to neighborhood structure improvement, search strategy organization has also been widely studied. Karaboga and Gorkemli [8] improved the search behavior of onlooker bees in qABC. Wang et al. [9] proposed an ensemble ABC with multiple strategies to improve the balance between exploration and exploitation. Liu et al. [10] combined adaptive search strategies with an elite selection mechanism, and Huang et al. [11] introduced a candidate strategy pool to improve search flexibility. These studies show that neighborhood information and search strategy organization are important for improving the performance of ABC.

Recently, reinforcement learning has been introduced into ABC and other metaheuristic algorithms to provide adaptive search control. Cui et al. [12] used reinforcement learning to regulate the search behavior of ABC. Ni et al. [13] proposed an ABC algorithm with multiple strategies based on Q-learning to realize adaptive strategy selection. These studies indicate that reinforcement learning can help ABC change from fixed search rules to adaptive control based on search feedback. In addition, guided restart mechanisms, such as opposition-based learning [14,15], have been used to improve the scout bee phase and help individuals escape from local optima.

However, most existing studies improve only one part of the ABC framework, such as neighborhood search, strategy selection, or restart operation. The coordinated design of adaptive neighborhood regulation in the employed bee phase, differentiated strategy assignment in the onlooker bee phase, and guided restart in the scout bee phase still needs further investigation.

Motivated by the above analysis, this paper proposes an artificial bee colony algorithm with dual groups and multiple strategies based on reinforcement learning, named RLDMS-ABC. The main contributions of this paper are summarized as follows.

(1) For the employed bee phase, a Q-learning mechanism is designed to adaptively regulate the neighborhood size of each food source according to search feedback. In addition, the best individual selected from the sampled neighborhood is used to guide candidate solution generation, which improves the utilization of neigh-

borhood information. This design helps enhance exploration through adaptive neighborhood adjustment.

(2) For the onlooker bee phase, a search framework with dual groups and multiple strategies is developed. The selected food sources are divided into elite and ordinary groups according to their relative quality, and different search strategies are assigned to the two groups to improve search flexibility and balance exploration and exploitation.

(3) For the scout bee phase, a guided restart mechanism combining opposition-based learning and the current global best solution is introduced. This mechanism helps stagnated individuals escape from local optima while retaining useful information from the current search process.

The remainder of this paper is organized as follows. Section 2 introduces the standard ABC algorithm and related studies. Section 3 describes the proposed RLDMS-ABC in detail. Section 4 presents the experimental results and analysis. Finally, Section 5 concludes this paper and discusses future research directions.

2. Related Work

2.1. The Original Artificial Bee Colony Algorithm

ABC is a swarm intelligence optimization algorithm inspired by the foraging behavior of honey bee colonies [2,3]. In ABC, each food source represents a candidate solution, and the quality of a food source is evaluated by its objective function value. The search process is mainly performed by three types of bees, namely employed bees, onlooker bees, and scout bees. Employed bees explore the neighborhood of their associated food sources, onlooker bees select food sources according to shared fitness information, and scout bees generate new food sources when some existing ones are abandoned.

Assume that the number of food sources is SN and the problem dimension is D. The i-th food source can be represented as $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D})$.

Within the search space, the initial food sources are randomly generated according to Equation (1):

$$x_{i,j} = l_j + \text{rand}(0, 1) \times (u_j - l_j), \tag{1}$$

where l_j and u_j denote the lower and upper bounds of the j-th dimension, respectively, and rand(0, 1) $\in$ [0, 1] is a random number.

In the employed bee phase, each employed bee generates a new candidate solution around the current food source according to Equation (2):

$$v_{i,j} = x_{i,j} + \varnothing_{i,j} \times \left(x_{i,j} - x_{k,j}\right), \tag{2}$$

where k $\neq$ i, k $\in$ {1, 2, …, SN} and $\varnothing_{i,j} \in [-1, 1]$ is a random coefficient. If the newly generated solution is better than the current one, the original food source is replaced; otherwise, the original food source is retained and its trial counter is increased by one. As can be seen, this search equation mainly relies on random differential perturbation and does not explicitly exploit structured neighborhood information, which limits the utilization of neighborhood information in the employed bee phase.

In the onlooker bee phase, onlooker bees select food sources for further exploitation according to their fitness values. For minimization problems, the fitness value is usually calculated by Equation (3):

$$fit(X_i) = \begin{cases} \frac{1}{1+f(X_i)}, f(X_i) \geq 0 \\ 1+|f(X_i)|, f(X_i) < 0 \end{cases}, \tag{3}$$

and the corresponding selection probability is calculated by Equation (4):

$$p_i = \frac{fit(X_i)}{\sum_{n=1}^{SN} fit(X_i)}, \tag{4}$$

After a food source is selected, the onlooker bee generates a new candidate solution using the same search equation as that in the employed bee phase. Greedy selection is then applied to decide whether the new candidate solution replaces the original food source.

In the scout bee phase, if a food source cannot be improved for a predefined number of trials and its trial counter exceeds the parameter limit, it is abandoned and replaced by a newly generated food source using Equation (1). This mechanism helps ABC maintain population diversity during the search process.

The general procedure of the standard ABC algorithm is summarized in Algorithm 1. The algorithm starts with population initialization and then iteratively executes the employed bee, onlooker bee, and scout bee phases until the maximum number of function evaluations is reached.

2.2. Reinforcement Learning

Reinforcement learning (RL) is a machine learning paradigm in which an agent learns decision policies through continuous interaction with an environment [16]. Unlike supervised learning, RL does not rely on explicit labeled data. Instead, the agent gradually improves its decision-making policy according to the reward feedback obtained after taking actions, with the objective of maximizing the expected cumulative return. Because of its feedback-driven and adaptive nature, RL has been widely used in sequential decision-making and dynamic control problems [16].

A key issue in reinforcement learning is how to balance exploration and exploitation. On the one hand, the agent needs to explore different actions in order to discover potentially beneficial strategies. On the other hand, it must exploit the knowledge acquired so far to obtain higher returns. This balance is also closely related to metaheuristic optimization, where search behaviors often need to be adaptively adjusted according to the current search state. In the context of ABC, such adaptive regulation can be reflected in the adjustment of neighborhood size, search strategy, and other search-related behaviors.

Mathematically, reinforcement learning problems are commonly formulated as a Markov Decision Process (MDP), whose basic components include states, actions, state transitions, and reward functions [17]. Through repeated interactions with the environment, the agent learns a policy that maximizes the expected cumulative return. This formulation provides a natural basis for introducing learning mechanisms into optimization algorithms, especially when the search process needs to respond dynamically to different search outcomes.

In this paper, Q-learning is adopted as the reinforcement learning scheme. Q-learning is a value-based reinforcement learning method that assigns a Q-value to each state-action pair in order to estimate the long-term return obtained by taking a specific action under the current state [18]. The general framework of Q-learning is illustrated in Figure 1, and its update rule can be given by Equation (5):

$$Q_{new}(s_t, a_t) = Q(s_t, a_t) + \alpha \times [r_{t+1} + \gamma max Q(s_{t+1}, a_t) - Q(s_t, a_t)], \tag{5}$$

where s_t and s_{t+1} denote the current state and the next state, respectively; a_t denotes the current action; r_{t+1} denotes the immediate reward; α is the learning rate; and γ is the discount factor. By iteratively updating the Q-values, the agent can gradually learn the optimal action selection under different states [19].

Algorithm 1 Pseudocode of the original ABC algorithm

Initialization Phase:

Initialize the number of food sources SN, dimension D, threshold $limit$, trial counters $trail_i = 0$, maximum function evaluations FEs_{max};
Randomly generate solutions X_i (i = 1, 2, . . ., SN) by Equation (1);
Evaluate $f(X_i)$ and compute $fit(X_i)$ by Equation (3);
Memorize the current global best food source;
while $Fes < FEs_{max}$ do

Employed bee phase:

for i = 1 to SN do
    Randomly select $k \in [1, 2, \ldots, SN]$, $k \neq i$;
    Generate candidate solution V_i by Equation (2);
    Evaluate $f(V_i)$ and compute $fit(V_i)$;
    if $f(V_i) < f(X_i)$ **then**
    $X_i \leftarrow V_i$;
    $trail_i \leftarrow 0$;
    else
    $trail_i \leftarrow trail_i + 1$;
    end if
end for

Onlooker Bee Phase:

Calculate selection probability p_i using Equation (4)
for i = 1 to SN do
    Select solution X_i according to p_i (roulette wheel selection);
    Generate V_i using Equation (2);
    Evaluate $f(V_i)$ and compute fitness using Equation (3);
    if $f(V_i) < f(X_i)$ **then**
        $X_i \leftarrow V_i$;
        $trail_i \leftarrow 0$;
     else
        $trail_i \leftarrow trail_i + 1$;
    end if
end for

Scout Bee Phase:

for i = 1 to SN do
    if $trail_i \geq$ limit do
        Replace X_i with a new randomly produced solution by Equation (1);
    $trail_i \leftarrow 0$;
    end if
end for
Memorize the current global best food source;
end while

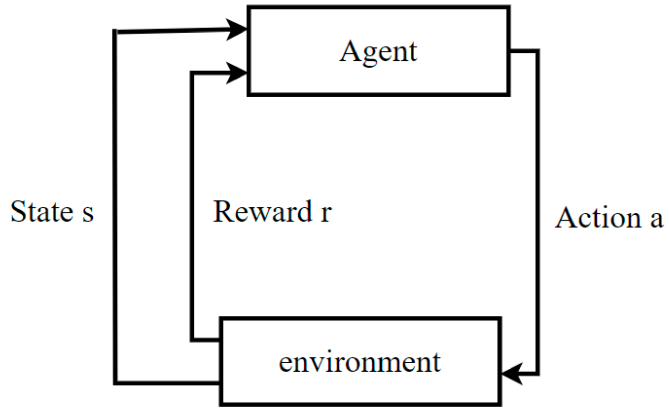


Figure 1. General framework of Q-learning.

In the proposed method, Q-learning is not introduced as an independent decision-making framework, but as a feedback-driven adaptive regulation mechanism for the employed bee phase. Specifically, the search result of the current food source is used to define the state, and the corresponding action is used to regulate the neighborhood size of that food source. In this way, the neighborhood range can be adjusted dynamically according to search feedback, thereby improving the adaptability of the ABC algorithm at different search stages and providing support for a better balance between global exploration and local exploitation.

2.3. Group-Based Search and Multiple Strategies in ABC Algorithms

Group-based search and multiple strategies have become important directions in the development of ABC. The basic idea is to enrich the search behavior of the algorithm and organize individuals according to their search characteristics. In complex optimization problems, different individuals may contribute differently to the search process. Some individuals are more suitable for refining promising regions, while others are more useful for maintaining diversity and expanding the search range. Therefore, assigning suitable search behaviors to different individuals or groups is an effective way to improve the search organization of ABC algorithms.

Multiple strategy mechanisms provide different search patterns within the same algorithm. Strategies with stronger random perturbation are usually helpful for global exploration and diversity maintenance, while strategies guided by high-quality solutions are more suitable for local exploitation and convergence acceleration. By combining different search strategies, ABC variants can avoid relying on a single update rule and can adapt more flexibly to different search conditions. Existing studies have shown that multiple search equations, strategy pools, and adaptive strategy selection mechanisms can improve the search flexibility and robustness of ABC algorithms [19–24].

Group-based search further improves the organization of population information. Instead of treating all individuals in the same way, this type of method divides individuals into different groups or subpopulations according to certain criteria, such as fitness ranking, individual quality, random grouping, or search state information. Different groups can then be assigned different search roles. For example, high-quality groups can be used to strengthen local refinement around promising regions, while other groups can be used to preserve diversity and maintain broader exploration. Related studies have shown that random grouping, elite group guidance, dual subpopulation structures, and adaptive population organization can enhance cooperation among individuals and improve the overall search performance of ABC algorithms [25–28].

3. Proposed Method

This section presents the proposed RLDMS-ABC algorithm. The algorithm incorporates adaptive neighborhood regulation, a dual group search mechanism with multiple strategies, and a guided restart mechanism into the standard ABC framework.

3.1. Reinforcement Learning-Driven Adaptive Neighborhood Regulation Mechanism

This subsection presents the adaptive neighborhood regulation mechanism used in the employed bee phase. Each food source maintains an individual neighborhood size, which is adjusted by Q-learning according to its search feedback [12,13,29]. After the neighborhood size is updated, a random neighborhood is constructed, and the local reference best individual in this neighborhood is used to guide candidate solution generation. The individual neighborhood-size structure is illustrated in Figure 2.

position	neighborhood size
X_1	K_1
X_2	K_2
…	…
X_i	K_i
…	…
X_{SN}	K_{SN}

Figure 2. Neighborhood-size structure.

Let the current food source be X_i, and let the newly generated candidate solution be V_i. The neighborhood size associated with X_i is denoted by K_i, where $K_i \in [4, 15]$. During initialization, K_i is randomly assigned within the interval [4, 15] for each food source, so that each sampled neighborhood contains enough candidates for local best selection while avoiding an overly large neighborhood that may weaken local search. The state of each food source is determined based on its search feedback. If the candidate solution improves the current food source, the state is regarded as a success state; otherwise, it is regarded as a failure state. The state is defined by Equation (6):

$$s = \begin{cases} s_1, f(V_i) < f(X_i) \\ s_2, f(V_i) \geq f(X_i) \end{cases}, \tag{6}$$

where s_1 denotes the success state, and s_2 denotes the failure state. The action set is used to adjust the neighborhood size of the current food source. It includes three neighborhood regulation actions, namely decreasing the neighborhood size, increasing the neighborhood size, and keeping the neighborhood size unchanged, as shown in Equation (7):

$$A = \{a_1, a_2, a_3\} = \{K_i - 1, K_i + 1, K_i\}, \tag{7}$$

where a_1, a_2, and a_3 denote the actions of decreasing, increasing, and maintaining the neighborhood size, respectively. Through this state-action design, each food source can adaptively regulate its neighborhood scale according to its own search state.

During the learning process, each food source maintains an independent Q-table. Each state-action pair is associated with a Q-value, which evaluates the long-term benefit of selecting an action under the current state. The Q-values are updated according to Equation (5). In this study, the reward is set to 1 if the candidate solution is better than the current food source; otherwise, it is set to 0. The α and γ are set to 0.6 and 0.2, respectively. For each food source, the action with the largest Q-value under the current state is selected to adjust its neighborhood size. If several actions have the same maximum Q-value, one of them is randomly selected.

The computational overhead introduced by Q-learning is limited. Each food source maintains only a small Q-table with two states and three actions. Since the number of states and actions is fixed, both Q-value updating and action selection require only constant-time operations. Compared with candidate solution generation and objective function evaluation, this additional cost is negligible.

The Q-table used for neighborhood-size adjustment is shown in Table 1.

Table 1. Q-table for neighborhood-size adjustment.

State s	**Action** a		
	a_1**:** K_i-1	a_2**:** K_i+1	a_3**:** K_i
S_1: $f(V_i) < f(X_i)$	$Q(S_1, a_1)$	$Q(S_1, a_2)$	$Q(S_1, a_3)$
S_2: $f(V_i) \geq f(X_i)$	$Q(S_2, a_1)$	$Q(S_2, a_2)$	$Q(S_2, a_3)$

After the neighborhood size has been adjusted, a random neighborhood is constructed for the current food source. For X_i, K_i individuals are randomly sampled from the population except itself, and the sampled neighborhood is denoted by N_i. To use the information contained in the neighborhood, the current food source and its sampled neighbors are considered together as $\{X_i\} \cup N_i$.

Then, the local reference best individual x_{irb} is selected from $\{X_i\} \cup N_i$ according to the objective function value. This individual represents the best solution in the current random neighborhood and provides guidance for candidate solution generation.

Subsequently, a neighbor X_{ir} is randomly selected from N_i, and another individual X_k is randomly selected from the population. Then, a dimension j is randomly selected for updating. The candidate solution is generated according to Equation (8):

$$v_{i,j} = x_{ir,j} + \varnothing_{i,j} \times \left(x_{ir,j} - x_{k,j}\right) + \psi_{i,j} \times \left(x_{irb,j} - x_{k,j}\right), \tag{8}$$

where $x_{ir,j}$ denotes the j-th dimension of a randomly selected neighbor, $x_{k,j}$ denotes the j-th dimension of a randomly selected individual from the population, and $x_{irb,j}$ denotes the j-th dimension of the local best individual in the current random neighborhood, $x_{ir} \neq x_{irb}$; $\varnothing_{i,j} \in [-1, 1]$, $\psi_{i,j} \in [0, 1.2]$. The coefficient $\psi_{i,j}$ is sampled from the interval [0, 1.2] to allow a limited extrapolation along the direction of the local reference best individual. Compared with sampling from [0, 1], this setting avoids overly conservative interpolation and helps strengthen the utilization of promising neighborhood information, while the generated candidate solutions are still controlled by boundary handling. This update strategy introduces locally advantageous information from the random neighborhood while retaining random perturbation, which helps improve the search efficiency of the employed bee phase [30–32].

The detailed procedure of the employed bee phase is shown in Algorithm 2.

Algorithm 2 Employed bee phase

for each X_i in the colony do
Randomly construct the sampled neighborhood N_i according to K_i;
select X_{irb} from $\{X_i\} \cup N_i$;
Randomly select another individual X_k from the population, where $X_k \neq X_i$;
Randomly select a dimension j;
Generate V_i by Equation (8);
Evaluate $f(V_i)$ and set FEs++;
    if $f(V_i) < f(X_i)$ **then**
    Replace X_i with V_i;
    $s = s_1$, $reward_i = 1$;
    $trial_i = 0$;
    else
    $trial_i = trial_i + 1$;
    $s = s_2$, $reward_i = 0$;
    end if
Update *Q-table*;
  Obtain the current state s_i of X_i;
  Select an action a_i from the Q-*table*$Q_i(s_i, a)$;
Update the neighborhood size K_i according to a_i;
end for

3.2. Dual Groups and Multiple Strategies

The selected food sources are ranked according to their objective function values [31]. For minimization problems, the elite proportion is denoted by *ei* and set to 0.2. Thus, the top 20% of the selected food sources are assigned to the elite group, while the remaining selected food sources are assigned to the ordinary group. The elite group is mainly used for exploitation around promising regions, whereas the ordinary group contributes to exploration and population diversity. Different search strategies are then applied according to the group identity.

3.2.1. Three-Strategy Collaborative Search Mechanism

After the dual-group partition is established, the onlooker bee phase further adopts a three-strategy collaborative search mechanism to provide differentiated search patterns for individuals of different quality levels. The core idea of this design is to integrate random-neighborhood guidance, local structural exploitation, and role-oriented search behaviors into a unified strategy framework. Existing studies have shown that neighborhood-guided search, multi-strategy collaboration, and adaptive strategy pools can effectively improve the flexibility and robustness of ABC variants [6,9,11,30,32–37]. Motivated by these observations, the proposed mechanism constructs three complementary search strategies for the onlooker bee phase: a global-best-guided exploitative strategy based on a local progressively improving nearest-neighbor sequence, a random-guided exploration strategy, and a local-best-guided exploitation strategy. Through the coordinated use of these strategies, the proposed method aims to improve local refinement in promising regions while maintaining sufficient search diversity in less competitive areas.

(1) **Strategy 1: local modified nearest-neighbor sequence strategy**

To strengthen the exploitation ability in promising regions during the middle and late search stages, a global-best-guided exploitative strategy based on a local modified nearest-neighbor sequence, denoted as LocalMNNS, is developed. In this strategy, the term

"local" indicates that the construction range of the modified nearest-neighbor sequence is limited to the randomly sampled neighborhood of the currently selected food source. This design allows LocalMNNS to extract useful structural information from the local neighborhood around the current solution.

Let the currently selected food source be X_i, and let its random local neighborhood be denoted by N_i. The neighborhood N_i is constructed by randomly sampling K_i individuals from the population excluding X_i, where K_i is the neighborhood size associated with X_i. The construction of LocalMNNS starts from X_i. At each step, only the unused individuals in N_i are considered. Among these individuals, the one that has a better objective function value than the current reference individual and has the shortest distance to it is selected as the next element of the sequence. Then, the newly selected individual becomes the current reference individual, and the same rule is repeatedly applied within the same sampled neighborhood. The construction process stops when no unused individual in N_i is better than the current reference individual.

Accordingly, the LocalMNNS associated with X_i is denoted by M_i and can be expressed as Equation (9):

$$M_i = \left\{ M_i^0, M_i^1, M_i^2, \ldots M_i^{l_i-1} \right\}, \tag{9}$$

where $M_i^0 = X_i$ is the starting element of the sequence, l_i is the length of LocalMNNS, and each subsequent element M_i^q (q = 1, 2, …, $l_i - 1$) denotes the nearest unused individual in N_i whose objective function value is better than that of the previous element M_i^{q-1}. For minimization problems, the objective values of the individuals in LocalMNNS satisfy the progressively improving relationship shown in Equation (10):

$$f(M_i^0) > f(M_i^1) > \ldots f(M_i^{l_i-1}) \tag{10}$$

When $l_i \geq 2$, an adjacent pair (M_i^h, M_i^{h+1}) is randomly selected from LocalMNNS, where h $\in$ {0, 1, …, $l_i - 2$}. Based on this adjacent pair, the candidate solution is generated around the current global best solution according to Equation (11):

$$v_{i,j} = x_{best,j} + F \times \left(m_{i,j}^{h+1} - m_{i,j}^{h} \right), \tag{11}$$

where $v_{i,j}$ is the j-th dimension of the candidate solution, $x_{best,j}$ is the j-th dimension of the current global best solution, $m_{i,j}^h$ and $m_{i,j}^{h+1}$ are the j-th dimensional components of two adjacent individuals in LocalMNNS, and F $\in$ [0, 1] is a random number. In Equation (11), the current global best solution provides the exploitation center, while the displacement between two adjacent elements in LocalMNNS provides a locally meaningful search direction. Therefore, Strategy 1 combines global-best guidance with the progressively improving structure extracted from the random local neighborhood.

Figure 3 illustrates the construction process of LocalMNNS. As shown in Figure 3a, the current solution and its randomly sampled local neighbors are first identified. Figure 3b filters the neighbors whose objective values are better than that of the current solution. Then, as shown in Figure 3c, the modified nearest neighbor is selected according to the dual criterion of being better and nearest. By repeatedly applying this rule within the same random neighborhood, the complete LocalMNNS is obtained, as shown in Figure 3d.

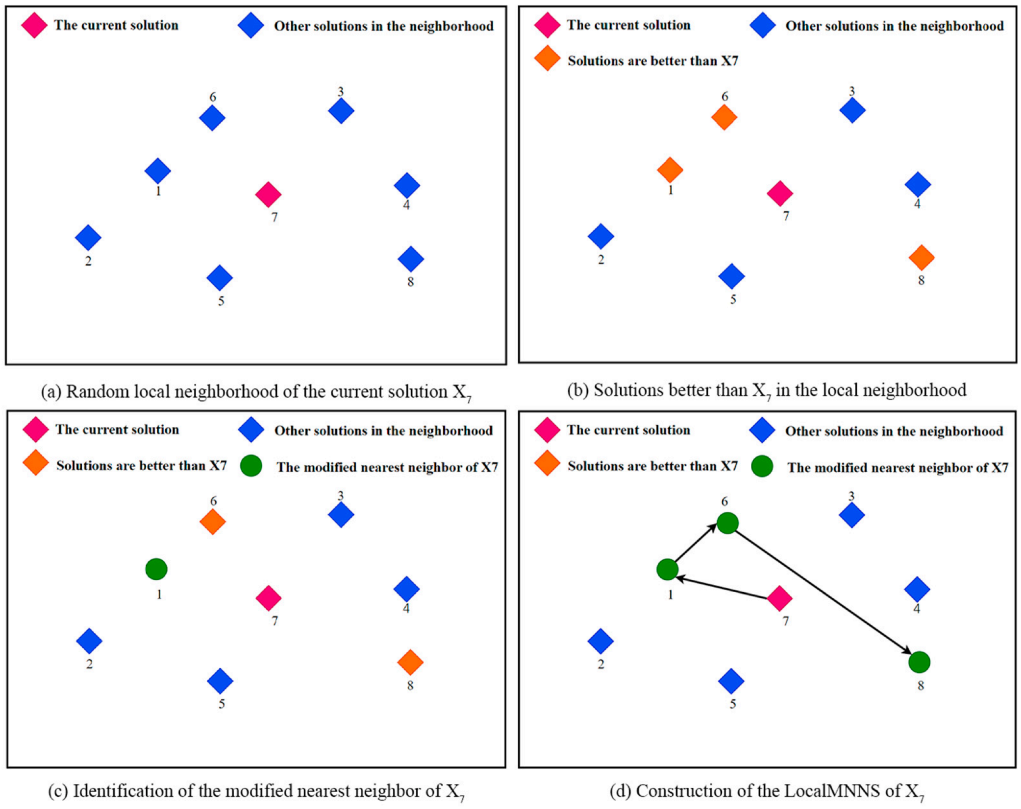


Figure 3. The construction process of LocalMNNS.

It should be noted that LocalMNNS may contain fewer than two elements when the current random neighborhood does not provide enough better individuals. To keep Strategy 1 executable, the auxiliary update shown in Equation (12) is introduced:

$$v_{i,j} = x_{best,j} + F \times \left(x_{best,j} - x_{ai,j}\right), \tag{12}$$

where X_{ai} is an anchor point. It is preferentially selected as the best individual in $\{X_i\} \cup N_i$. If this individual is X_{best}, another individual different from X_{best} is selected from N_i, then X_i, and finally the whole population. This design keeps Strategy 1 executable when the local improving sequence is too short.

(2) **Strategy 2: random-guided exploration strategy**

Strategy 2 is designed to enhance exploration and maintain population diversity. It uses the current food source as the base solution and introduces random guidance and random differential perturbation, as shown in Equation (13):

$$v_{i,j} = x_{i,j} + F \times \left(x_{r1,j} - x_{i,j}\right) + F \times \left(x_{r2,j} - x_{r3,j}\right), \tag{13}$$

where X_{r1}, X_{r2}, X_{r3} are three randomly selected food sources from the population, and $X_{r1} \neq X_{r2} \neq X_{r3} \neq X_i$, $F \in [0, 1]$. In Equation (13), the same random coefficient F is used for the two perturbation terms in each dimension, and F is independently sampled from [0,1] when each dimension is updated. Since the guidance information comes from randomly selected food sources, this strategy has strong exploratory ability and is suitable for expanding the search range of ordinary individuals.

(3) **Strategy 3: local-best-guided exploitation strategy**

Strategy 3 is used to strengthen exploitation around the local best food source. It combines local-best guidance with random differential perturbation, and the candidate solution is generated according to Equation (14):

$$v_{i,j} = x_{i,j} + F \times \left(x_{irb,j} - x_{i,j}\right) + F \times \left(x_{r1,j} - x_{r2,j}\right), \tag{14}$$

where X_{r1}, X_{r2} are two randomly selected food sources from the population, and $X_{r1} \neq X_{r2} \neq X_i$, F $\in$ [0, 1], $x_{irb,j}$ denotes the j-th dimension of the local best individual in the current random neighborhood. This strategy uses the local best food source to provide a stable exploitation direction while retaining random perturbation, making it suitable as the main search operator for the elite group.

3.2.2. Strategy Allocation and Dynamic Switching Mechanism

After the three search strategies are defined, the onlooker bee phase assigns them according to the quality group, search progress, and recent search feedback of each selected food source. This strategy-control scheme is consistent with ABC studies that adjust search operators according to individual quality, search stage, and feedback information [21,22,33].

For ordinary food sources, the search progress is defined as Equation (15):

$$T = \frac{FEs}{maxFEs}, \tag{15}$$

where FEs is the number of consumed function evaluations, and maxFEs is the maximum evaluation budget. Based on T, the activation probability of Strategy 1 is defined by Equation (16):

$$P(T) = \begin{cases} 0, & 0 \leq T < 0.25 \\ \frac{1}{2} \cdot \frac{T-0.25}{0.65}, & 0.25 \leq T < 0.9 \\ \frac{1}{2}, & 0.9 \leq T \leq 1 \end{cases}, \tag{16}$$

For ordinary food sources that do not activate Strategy 1, U_i denotes the retained strategy of the i-th ordinary food source, where $U_i \in \{2, 3\}$. If the candidate solution generated by U_i is accepted, U_i is retained; otherwise, U_i is switched to the other strategy in {2, 3}. The detailed procedure of the onlooker bee phase is shown in Algorithm 3.

3.3. Guided Scout Bee Restart Mechanism with Opposition-Based Learning and Global-Best Guidance

In the standard ABC algorithm, a food source is abandoned and randomly regenerated when its trial counter exceeds the predefined limit. Although this operation helps restore population diversity, purely random regeneration may ignore useful information accumulated during the search. To make the scout bee phase more informative, RLDMS-ABC adopts a guided restart mechanism based on opposition-based learning [14,15] and the current global best solution. Related studies have shown that opposition-based learning and improved scout bee mechanisms can help ABC variants escape from local optima and enhance restart effectiveness [15,38–40].

When a stagnated food source X_i satisfies $trial_i >$ limit, its opposite point is first constructed within the search boundaries as shown in Equation (17):

$$x_{ne,j} = lb_j + ub_j - x_{i,j}, \tag{17}$$

The opposite point provides a relocation direction away from the current stagnated region while remaining consistent with the search boundary.

Algorithm 3 Onlooker bee phase

Initialize t $=$ 0 and $U_i \in \{2, 3\}$ for each ordinary food source;
while $t <$ FoodNumber do
  if $rand(0,\ 1)\ <\ p_i$ **then**
    Determine the group identity of X_i;
    if X_i belongs to the elite group then
      if $rand(0, 1) < 0.9$ **then**
        Generate V_i by Strategy 3 using Equation (14);
      else
        Generate V_i by Strategy 2 using Equation (13);
      end if
    else
      Generate T by Equation (15) and $P(T)$ by Equation (16);
      if rand(0, 1) $< P(T)$ **then**
        Construct N_i and LocalMNNS according to Strategy 1;
        if $l_i \geq 2$ **then**
          Generate V_i by Strategy 1 using Equation (11);
        else
          Generate V_i by auxiliary update using Equation (12);
        end if
      else
        if $U_i = 2$ **then**
          Generate V_i by Strategy 2 using Equation (13);
        else if $U_i = 3$ **then**
          Generate V_i by Strategy 3 using Equation (14);
        end if
      end if
    end if
    Evaluate $f(V_i)$ and set $FEs = FEs + 1$;
    if $f(V_i)\ <\ f(X_i)$ **then**
      Replace X_i with V_i;
      Set $trial_i = 0$;
    else
      Set $trial_i = \ trial_i + 1$;
    end if
    if ordinary food source uses Strategy U_i **then**
      if V_i is accepted **then**
        Keep U_i unchanged;
      else
        $U_i \leftarrow\ 5\ -\ U_i$;
      end if
    end if
    $t = t + 1$;
  end if
end while

After that, the new scout solution is generated by combining the opposite point with the current global best solution, as defined in Equation (18):

$$x_{nw,j} = r_1 \times x_{ne,j} + r_2 \times x_{best,j}, \tag{18}$$

where r_1 is uniformly sampled from [0, 1], and r_2 is defined as $r_2 = 1 - r_1$. Thus, and the new scout solution is generated as a convex combination of the opposite point and the current global best solution.

After X_{nw} is evaluated, $trial_i$ is reset to zero, and the neighborhood size of X_i is reinitialized.

3.4. Overall Framework of RLDMS-ABC

The overall framework of the proposed RLDMS-ABC is summarized in Algorithm 4.

Algorithm 4 Overall framework of RLDMS-ABC

```
Initialize the food source X_i, trail_i, K_i, Q-table, and related parameters;
Evaluate each food source and memorize the current global best solution;
while FEs < maxFEs do
  Execute Algorithm 2 (Employed bee phase);
  Execute Algorithm 3 (Onlooker bee phase);
  // Scout bee phase
  if max{trail_i} > limit then
    Construct the opposite point of X_i by Equation (17);
    Generate the new scout solution by Equation (18);
    Calculate f(x_nw) and set FEs++;
    Replace X_i with x_nw;
    Reset trail_i to 0;
  end if
end while
```

4. Experimental Studies

All experiments were conducted on the same computational platform to ensure fairness and reproducibility. The computer was equipped with a 13th Gen Intel® Core™ i9-13980HX processor @2.20 GHz and 16 GB RAM, running the Windows 11 operating system. The proposed RLDMS-ABC and all comparison algorithms were implemented in C++ and compiled using the GNU g++ compiler based on MinGW-w64 GCC 8.1.0 (MinGW-w64 Project). The experiments were conducted on the Windows 11 operating system (Microsoft Corporation, Redmond, WA, USA). To avoid performance differences caused by inconsistent implementation or hardware conditions, all algorithms were executed under the same computational environment.

For all algorithms, the total colony size NP was set to 100, and the number of food sources $FoodNumber$, equivalent to SN in the standard ABC description, was set to 50; thus, $NP = 2 \times FoodNumber$. To ensure a fair comparison, all compared algorithms adopted the same termination condition, namely maxFEs = 10,000 $\times$ D. In all algorithms, the value of FEs was increased by one whenever a candidate solution was evaluated, and each algorithm was independently executed 30 times on every benchmark function. In all tables in this section, bold values indicate the best result in each row. Additional experimental results are provided in the Supplementary Materials.

4.1. Sensitivity Analysis of the Elite Parameter ei

To analyze the influence of the elite parameter *ei* on the performance of RLDMS-ABC, a parameter sensitivity analysis was conducted on the CEC 2014 benchmark set. Table 2 presents the Friedman ranking results of RLDMS-ABC with different *ei* values when $D = 30$ and $D = 50$. Since a smaller Friedman rank indicates better overall performance, it can be observed that RLDMS-ABC achieves the best ranks when $ei = 0.2$, with ranking values of 2.57 and 2.73 for $D = 30$ and $D = 50$, respectively. This indicates that an appropriate elite proportion can enhance exploitation while maintaining population diversity. Therefore, *ei* is set to 0.2 in the following experiments.

Table 2. Friedman test ranking of RLDMS-ABC at different *ei*.

ei	CEC 2014 (D = 30) Ranking	CEC 2014 (D = 50) Ranking
0.1	2.89	3.27
0.2	**2.57**	**2.73**
0.3	2.93	3.20
0.4	3.25	2.91
0.5	3.36	2.89

4.2. Ablation Study of RLDMS-ABC

To evaluate the contribution of each key component in RLDMS-ABC, four ablation variants were constructed. RLDMS-ABC-I removes the Q-learning-based adaptive neighborhood regulation mechanism and uses a fixed neighborhood size in the employed bee phase. RLDMS-ABC-II removes the dual-group multi-strategy mechanism in the onlooker bee phase and adopts the standard ABC update rule for selected food sources. RLDMS-ABC-III replaces the guided scout bee restart mechanism with random reinitialization. RLDMS-ABC-IV removes the adaptive strategy switching mechanism in the onlooker bee phase; when Strategy 1 is not activated for ordinary food sources, Strategy 2 or Strategy 3 is randomly selected. The comparison results are reported in Tables 3–5.

Table 3. The comparison results of RLDMS-ABC and four ablation variants on the CEC 2014 benchmark set when $D = 30$.

Function	RLDMS-ABC I Average	RLDMS-ABC II Average	RLDMS-ABC III Average	RLDMS-ABC IV Average	RLDMS-ABC Average
$f1$	1.80×10^{5}	4.60×10^{6}	2.43×10^{5}	$\mathbf{8.27 \times 10^{4}}$	1.42×10^{5}
$f2$	5.19×10^{-3}	2.17×10^{2}	2.45×10^{-3}	$\mathbf{1.51 \times 10^{-3}}$	3.94×10^{-3}
$f3$	3.45×10^{0}	9.80×10^{2}	2.50×10^{0}	3.27×10^{0}	$\mathbf{2.13 \times 10^{0}}$
$f4$	2.17×10^{0}	2.99×10^{1}	2.17×10^{0}	6.38×10^{0}	$\mathbf{5.63 \times 10^{-2}}$
$f5$	2.03×10^{1}	$\mathbf{2.02 \times 10^{1}}$	2.03×10^{1}	2.03×10^{1}	2.03×10^{1}
$f6$	1.41×10^{1}	1.34×10^{1}	$\mathbf{1.26 \times 10^{1}}$	$\mathbf{1.26 \times 10^{1}}$	$\mathbf{1.26 \times 10^{1}}$
$f7$	2.14×10^{-3}	3.48×10^{-5}	$\mathbf{0.00 \times 10^{0}}$	4.93×10^{-4}	1.40×10^{-3}
$f8$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	5.92×10^{-17}
$f9$	4.26×10^{1}	6.05×10^{1}	4.05×10^{1}	$\mathbf{3.90 \times 10^{1}}$	$\mathbf{3.90 \times 10^{1}}$
$f10$	2.64×10^{0}	$\mathbf{1.62 \times 10^{-1}}$	1.66×10^{0}	1.79×10^{0}	1.66×10^{0}
$f11$	2.19×10^{3}	$\mathbf{1.80 \times 10^{3}}$	2.38×10^{3}	2.24×10^{3}	2.19×10^{3}
$f12$	3.62×10^{-1}	$\mathbf{1.86 \times 10^{-1}}$	3.93×10^{-1}	3.62×10^{-1}	3.62×10^{-1}
$f13$	2.24×10^{-1}	2.36×10^{-1}	2.05×10^{-1}	$\mathbf{8.16 \times 10^{-2}}$	2.14×10^{-1}
$f14$	2.06×10^{-1}	$\mathbf{1.85 \times 10^{-1}}$	2.06×10^{-1}	2.06×10^{-1}	2.06×10^{-1}
$f15$	$\mathbf{5.06 \times 10^{0}}$	5.65×10^{0}	5.43×10^{0}	5.22×10^{0}	5.18×10^{0}

Table 3. *Cont.*

Function	RLDMS-ABC I Average	RLDMS-ABC II Average	RLDMS-ABC III Average	RLDMS-ABC IV Average	RLDMS-ABC Average
$f16$	$\mathbf{9.70 \times 10^{0}}$	9.72×10^{0}	9.72×10^{0}	9.74×10^{0}	$\mathbf{9.70 \times 10^{0}}$
$f17$	1.64×10^{4}	1.76×10^{6}	1.58×10^{4}	$\mathbf{8.21 \times 10^{3}}$	1.39×10^{4}
$f18$	6.90×10^{3}	$\mathbf{5.77 \times 10^{2}}$	7.04×10^{3}	5.09×10^{3}	6.90×10^{3}
$f19$	4.63×10^{0}	7.01×10^{0}	4.56×10^{0}	4.91×10^{0}	$\mathbf{4.29 \times 10^{0}}$
$f20$	1.36×10^{2}	8.39×10^{3}	1.71×10^{2}	$\mathbf{1.04 \times 10^{2}}$	1.50×10^{2}
$f21$	1.27×10^{4}	1.93×10^{5}	$\mathbf{1.26 \times 10^{4}}$	1.38×10^{4}	$\mathbf{1.26 \times 10^{4}}$
$f22$	1.27×10^{2}	2.35×10^{2}	1.59×10^{2}	1.08×10^{2}	$\mathbf{9.99 \times 10^{1}}$
$f23$	$\mathbf{2.90 \times 10^{2}}$	$\mathbf{2.90 \times 10^{2}}$	$\mathbf{2.90 \times 10^{2}}$	$\mathbf{2.90 \times 10^{2}}$	$\mathbf{2.90 \times 10^{2}}$
$f24$	2.01×10^{2}	$\mathbf{1.97 \times 10^{2}}$	2.01×10^{2}	2.01×10^{2}	2.01×10^{2}
$f25$	$\mathbf{2.07 \times 10^{2}}$	2.13×10^{2}	2.08×10^{2}	2.08×10^{2}	$\mathbf{2.07 \times 10^{2}}$
$f26$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$
$f27$	4.05×10^{2}	4.12×10^{2}	4.01×10^{2}	4.13×10^{2}	$\mathbf{3.89 \times 10^{2}}$
$f28$	$\mathbf{4.21 \times 10^{2}}$	4.60×10^{2}	4.24×10^{2}	4.24×10^{2}	$\mathbf{4.21 \times 10^{2}}$
$f29$	1.29×10^{7}	$\mathbf{7.10 \times 10^{3}}$	1.34×10^{7}	1.29×10^{7}	1.11×10^{7}
$f30$	1.01×10^{3}	1.81×10^{3}	1.05×10^{3}	1.55×10^{3}	$\mathbf{9.97 \times 10^{2}}$
$w/t/l$	16/11/3	18/2/10	18/8/4	14/8/8	

Table 4. The comparison results of RLDMS-ABC and four ablation variants on the CEC 2014 benchmark set when $D = 50$.

Function	RLDMS-ABC I Average	RLDMS-ABC II Average	RLDMS-ABC III Average	RLDMS-ABC IV Average	RLDMS-ABC Average
$f1$	5.70×10^{5}	1.30×10^{7}	5.42×10^{5}	5.14×10^{5}	$\mathbf{4.97 \times 10^{5}}$
$f2$	1.68×10^{4}	$\mathbf{1.32 \times 10^{3}}$	2.01×10^{4}	1.35×10^{4}	6.59×10^{3}
$f3$	$\mathbf{9.52 \times 10^{0}}$	4.51×10^{3}	1.46×10^{1}	1.41×10^{1}	3.53×10^{1}
$f4$	9.81×10^{1}	7.38×10^{1}	7.57×10^{1}	8.11×10^{1}	$\mathbf{7.22 \times 10^{1}}$
$f5$	2.04×10^{1}	$\mathbf{2.02 \times 10^{1}}$	2.04×10^{1}	2.04×10^{1}	2.04×10^{1}
$f6$	2.82×10^{1}	2.85×10^{1}	2.84×10^{1}	2.83×10^{1}	$\mathbf{2.81 \times 10^{1}}$
$f7$	8.22×10^{-4}	1.37×10^{-3}	1.40×10^{-3}	1.64×10^{-3}	$\mathbf{4.93 \times 10^{-4}}$
$f8$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	3.32×10^{-2}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$
$f9$	9.92×10^{1}	1.37×10^{2}	9.32×10^{1}	9.46×10^{1}	$\mathbf{9.30 \times 10^{1}}$
$f10$	5.83×10^{0}	$\mathbf{2.89 \times 10^{-1}}$	3.98×10^{0}	4.11×10^{0}	3.92×10^{0}
$f11$	4.72×10^{3}	$\mathbf{3.91 \times 10^{3}}$	4.67×10^{3}	4.78×10^{3}	4.59×10^{3}
$f12$	3.81×10^{-1}	$\mathbf{2.00 \times 10^{-1}}$	3.99×10^{-1}	4.01×10^{-1}	3.76×10^{-1}
$f13$	3.10×10^{-1}	3.18×10^{-1}	3.15×10^{-1}	3.18×10^{-1}	$\mathbf{3.05 \times 10^{-1}}$
$f14$	$\mathbf{2.44 \times 10^{-1}}$	2.64×10^{-1}	2.66×10^{-1}	2.67×10^{-1}	2.62×10^{-1}
$f15$	1.23×10^{1}	1.45×10^{1}	1.25×10^{1}	$\mathbf{1.14 \times 10^{1}}$	$\mathbf{1.14 \times 10^{1}}$
$f16$	1.81×10^{1}	$\mathbf{1.78 \times 10^{1}}$	1.81×10^{1}	1.81×10^{1}	1.80×10^{1}
$f17$	$\mathbf{2.71 \times 10^{4}}$	3.40×10^{6}	3.14×10^{4}	4.67×10^{4}	4.30×10^{4}
$f18$	1.80×10^{3}	1.29×10^{3}	1.91×10^{3}	2.17×10^{3}	$\mathbf{1.13 \times 10^{3}}$
$f19$	1.83×10^{1}	1.85×10^{1}	1.31×10^{1}	$\mathbf{1.18 \times 10^{1}}$	$\mathbf{1.18 \times 10^{1}}$
$f20$	$\mathbf{1.52 \times 10^{2}}$	2.91×10^{4}	2.08×10^{2}	1.72×10^{2}	1.72×10^{2}
$f21$	4.84×10^{4}	1.92×10^{6}	$\mathbf{4.82 \times 10^{4}}$	4.95×10^{4}	4.97×10^{4}
$f22$	4.96×10^{2}	6.97×10^{2}	5.33×10^{2}	4.96×10^{2}	$\mathbf{4.92 \times 10^{2}}$
$f23$	$\mathbf{3.27 \times 10^{2}}$	$\mathbf{3.27 \times 10^{2}}$	$\mathbf{3.27 \times 10^{2}}$	$\mathbf{3.27 \times 10^{2}}$	$\mathbf{3.27 \times 10^{2}}$
$f24$	$\mathbf{2.03 \times 10^{2}}$	2.04×10^{2}	$\mathbf{2.03 \times 10^{2}}$	$\mathbf{2.03 \times 10^{2}}$	$\mathbf{2.03 \times 10^{2}}$
$f25$	$\mathbf{2.06 \times 10^{2}}$	2.17×10^{2}	$\mathbf{2.06 \times 10^{2}}$	$\mathbf{2.06 \times 10^{2}}$	$\mathbf{2.06 \times 10^{2}}$
$f26$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$
$f27$	$\mathbf{4.37 \times 10^{2}}$	$\mathbf{4.37 \times 10^{2}}$	4.54×10^{2}	$\mathbf{4.37 \times 10^{2}}$	$\mathbf{4.37 \times 10^{2}}$
$f28$	5.07×10^{2}	5.61×10^{2}	5.30×10^{2}	$\mathbf{4.96 \times 10^{2}}$	5.00×10^{2}
$f29$	5.81×10^{7}	$\mathbf{4.29 \times 10^{3}}$	5.65×10^{7}	5.74×10^{7}	5.65×10^{7}
$f30$	8.74×10^{3}	1.15×10^{4}	9.01×10^{3}	$\mathbf{7.90 \times 10^{3}}$	$\mathbf{7.90 \times 10^{3}}$
$w/t/l$	18/7/5	19/4/7	21/6/3	16/11/3	

Table 5. The comparison results of RLDMS-ABC and four ablation variants on the CEC 2014 benchmark set when $D = 100$.

Function	RLDMS-ABC I Average	RLDMS-ABC II Average	RLDMS-ABC III Average	RLDMS-ABC IV Average	RLDMS-ABC Average
$f1$	$\mathbf{2.13 \times 10^{6}}$	6.79×10^{7}	2.70×10^{6}	6.53×10^{6}	2.69×10^{6}
$f2$	1.97×10^{4}	$\mathbf{9.17 \times 10^{3}}$	1.97×10^{4}	1.62×10^{4}	1.39×10^{4}
$f3$	$\mathbf{2.60 \times 10^{0}}$	1.59×10^{4}	6.45×10^{0}	3.46×10^{0}	6.30×10^{0}
$f4$	1.79×10^{2}	1.73×10^{2}	$\mathbf{1.65 \times 10^{2}}$	1.84×10^{2}	1.70×10^{2}
$f5$	2.06×10^{1}	$\mathbf{2.04 \times 10^{1}}$	2.06×10^{1}	2.06×10^{1}	2.06×10^{1}
$f6$	7.56×10^{1}	7.72×10^{1}	7.49×10^{1}	7.57×10^{1}	$\mathbf{7.44 \times 10^{1}}$
$f7$	1.40×10^{-3}	3.16×10^{-3}	$\mathbf{1.61 \times 10^{-15}}$	1.23×10^{-3}	$\mathbf{1.56 \times 10^{-15}}$
$f8$	$\mathbf{9.47 \times 10^{-16}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{8.29 \times 10^{-16}}$	3.32×10^{-2}	$\mathbf{8.29 \times 10^{-16}}$
$f9$	3.54×10^{2}	5.03×10^{2}	3.10×10^{2}	3.18×10^{2}	$\mathbf{3.07 \times 10^{2}}$
$f10$	1.08×10^{1}	$\mathbf{5.80 \times 10^{-1}}$	8.39×10^{0}	8.64×10^{0}	8.09×10^{0}
$f11$	1.24×10^{4}	$\mathbf{1.10 \times 10^{4}}$	1.24×10^{4}	1.23×10^{4}	1.23×10^{4}
$f12$	5.45×10^{-1}	$\mathbf{2.77 \times 10^{-1}}$	5.47×10^{-1}	5.46×10^{-1}	5.45×10^{-1}
$f13$	4.16×10^{-1}	$\mathbf{3.58 \times 10^{-1}}$	4.11×10^{-1}	4.20×10^{-1}	4.08×10^{-1}
$f14$	2.75×10^{-1}	2.89×10^{-1}	2.95×10^{-1}	2.93×10^{-1}	$\mathbf{2.66 \times 10^{-1}}$
$f15$	3.59×10^{1}	4.82×10^{1}	$\mathbf{3.56 \times 10^{1}}$	3.60×10^{1}	$\mathbf{3.56 \times 10^{1}}$
$f16$	4.03×10^{1}	4.05×10^{1}	4.03×10^{1}	$\mathbf{4.01 \times 10^{1}}$	4.03×10^{1}
$f17$	3.36×10^{5}	1.28×10^{7}	2.17×10^{5}	2.36×10^{5}	$\mathbf{2.15 \times 10^{5}}$
$f18$	1.84×10^{3}	8.67×10^{2}	2.71×10^{3}	2.73×10^{3}	$\mathbf{8.38 \times 10^{2}}$
$f19$	9.40×10^{1}	$\mathbf{7.37 \times 10^{1}}$	9.48×10^{1}	9.39×10^{1}	9.39×10^{1}
$f20$	4.11×10^{2}	1.05×10^{5}	4.37×10^{2}	$\mathbf{3.69 \times 10^{2}}$	4.02×10^{2}
$f21$	1.02×10^{5}	8.38×10^{6}	7.96×10^{4}	1.19×10^{5}	$\mathbf{7.11 \times 10^{4}}$
$f22$	1.81×10^{3}	1.88×10^{3}	1.85×10^{3}	1.81×10^{3}	$\mathbf{1.80 \times 10^{3}}$
$f23$	3.48×10^{2}	3.53×10^{2}	3.48×10^{2}	3.98×10^{2}	$\mathbf{3.19 \times 10^{2}}$
$f24$	3.10×10^{2}	3.61×10^{2}	3.73×10^{2}	$\mathbf{2.71 \times 10^{2}}$	2.87×10^{2}
$f25$	2.39×10^{2}	2.50×10^{2}	2.36×10^{2}	2.36×10^{2}	$\mathbf{2.35 \times 10^{2}}$
$f26$	1.97×10^{2}	2.05×10^{2}	1.91×10^{2}	2.00×10^{2}	$\mathbf{1.87 \times 10^{2}}$
$f27$	$\mathbf{2.14 \times 10^{3}}$	2.28×10^{3}	2.18×10^{3}	2.18×10^{3}	2.18×10^{3}
$f28$	3.30×10^{3}	3.55×10^{3}	3.27×10^{3}	$\mathbf{3.18 \times 10^{3}}$	3.30×10^{3}
$f29$	8.95×10^{7}	$\mathbf{5.64 \times 10^{3}}$	6.73×10^{7}	7.25×10^{7}	8.76×10^{7}
$f30$	9.71×10^{3}	3.77×10^{4}	$\mathbf{9.07 \times 10^{3}}$	9.42×10^{3}	9.38×10^{3}
$w/t/l$	22/5/3	21/1/8	20/6/4	20/4/6	

As shown in Tables 3–5, the complete RLDMS-ABC generally achieves more competitive performance than its ablation variants. Here, $w/t/l$ denotes the number of functions on which RLDMS-ABC performs better than, similarly to, or worse than the corresponding ablation variant. The results indicate that removing each component leads to performance degradation on different functions, demonstrating that adaptive neighborhood regulation, dual-group multi-strategy search, guided restart, and adaptive strategy switching all contribute to the overall effectiveness of RLDMS-ABC.

4.3. Comparison with Representative ABC Variants

To further evaluate the performance of the proposed RLDMS-ABC on more complex continuous optimization problems, comparative experiments were conducted on the CEC 2014 benchmark set [41]. This benchmark set contains 30 test functions, including unimodal, multimodal, hybrid, and composition functions. Specifically, unimodal functions mainly evaluate the exploitation ability and convergence accuracy of an optimizer, multimodal functions examine exploration ability and the capacity to escape local optima, and hybrid and composition functions provide more complex search landscapes for assessing robustness and scalability. Therefore, the characteristics of these functions provide a basis for

the subsequent function-level interpretation of the experimental results. All algorithms were tested under the same experimental conditions to ensure a fair comparison, and the parameter settings of the comparison algorithms followed their original references or commonly used settings. Tables 6–8 report the experimental results of RLDMS-ABC and five comparison algorithms, namely ABC, ABCG [42], ARABC [33], APABC [37], and DPGABC [26], for the cases of $D = 30$, $D = 50$, and $D = 100$, respectively. For clarity, the best results in each row are highlighted in boldface, and w/t/l is used to summarize the comparison results between each competing algorithm and RLDMS-ABC.

Table 6. Results on the CEC 2014 benchmark set with $D = 30$.

Function	ABC Average	ABCG Average	ARABC Average	APABC Average	DPGABC Average	RLDMS-ABC Average
$f1$	4.82×10^{6}	8.49×10^{6}	2.10×10^{7}	2.74×10^{7}	5.67×10^{6}	$\mathbf{1.42 \times 10^{5}}$
$f2$	1.43×10^{2}	5.05×10^{3}	6.04×10^{2}	8.86×10^{2}	4.81×10^{2}	$\mathbf{3.94 \times 10^{-3}}$
$f3$	8.98×10^{2}	7.41×10^{2}	1.88×10^{3}	2.64×10^{3}	5.03×10^{2}	$\mathbf{2.13 \times 10^{0}}$
$f4$	2.62×10^{1}	7.94×10^{1}	6.00×10^{1}	6.78×10^{1}	8.87×10^{1}	$\mathbf{5.63 \times 10^{-2}}$
$f5$	$\mathbf{2.01 \times 10^{1}}$	2.03×10^{1}	2.02×10^{1}	2.02×10^{1}	2.04×10^{1}	2.03×10^{1}
$f6$	1.44×10^{1}	1.52×10^{1}	1.41×10^{1}	1.39×10^{1}	$\mathbf{1.19 \times 10^{1}}$	1.26×10^{1}
$f7$	1.59×10^{-5}	$\mathbf{2.33 \times 10^{-16}}$	1.91×10^{-4}	1.53×10^{-3}	2.41×10^{-16}	1.40×10^{-3}
$f8$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	3.32×10^{-1}	5.92×10^{-17}
$f9$	8.75×10^{1}	6.34×10^{1}	4.64×10^{1}	5.40×10^{1}	4.04×10^{1}	$\mathbf{3.90 \times 10^{1}}$
$f10$	2.17×10^{-1}	3.40×10^{-1}	5.06×10^{-1}	$\mathbf{2.46 \times 10^{-2}}$	1.41×10^{1}	1.66×10^{0}
$f11$	2.05×10^{3}	2.91×10^{3}	1.93×10^{3}	$\mathbf{1.88 \times 10^{3}}$	2.63×10^{3}	2.19×10^{3}
$f12$	$\mathbf{1.67 \times 10^{-1}}$	4.75×10^{-1}	3.11×10^{-1}	2.75×10^{-1}	5.40×10^{-1}	3.62×10^{-1}
$f13$	2.33×10^{-1}	3.09×10^{-1}	2.76×10^{-1}	2.97×10^{-1}	3.07×10^{-1}	$\mathbf{2.14 \times 10^{-1}}$
$f14$	$\mathbf{2.03 \times 10^{-1}}$	2.53×10^{-1}	2.72×10^{-1}	2.50×10^{-1}	2.61×10^{-1}	2.06×10^{-1}
$f15$	8.07×10^{0}	7.90×10^{0}	6.02×10^{0}	7.26×10^{0}	6.37×10^{0}	$\mathbf{5.18 \times 10^{0}}$
$f16$	9.91×10^{0}	1.03×10^{1}	9.52×10^{0}	$\mathbf{9.36 \times 10^{0}}$	1.01×10^{1}	9.70×10^{0}
$f17$	1.50×10^{6}	1.20×10^{6}	3.83×10^{6}	5.68×10^{6}	1.22×10^{6}	$\mathbf{1.39 \times 10^{4}}$
$f18$	$\mathbf{4.75 \times 10^{2}}$	8.14×10^{2}	2.45×10^{4}	3.21×10^{4}	1.57×10^{3}	6.90×10^{3}
$f19$	7.07×10^{0}	6.15×10^{0}	7.74×10^{0}	7.94×10^{0}	7.16×10^{0}	$\mathbf{4.29 \times 10^{0}}$
$f20$	8.10×10^{3}	5.03×10^{3}	8.85×10^{3}	9.53×10^{3}	4.44×10^{3}	$\mathbf{1.50 \times 10^{2}}$
$f21$	1.48×10^{5}	1.25×10^{5}	8.03×10^{5}	9.36×10^{5}	1.47×10^{5}	$\mathbf{1.26 \times 10^{4}}$
$f22$	2.91×10^{2}	1.80×10^{2}	2.91×10^{2}	3.19×10^{2}	1.57×10^{2}	$\mathbf{9.99 \times 10^{1}}$
$f23$	$\mathbf{2.90 \times 10^{2}}$	$\mathbf{2.90 \times 10^{2}}$	$\mathbf{2.90 \times 10^{2}}$	$\mathbf{2.90 \times 10^{2}}$	$\mathbf{2.90 \times 10^{2}}$	$\mathbf{2.90 \times 10^{2}}$
$f24$	2.01×10^{2}	2.00×10^{2}	2.01×10^{2}	$\mathbf{1.98 \times 10^{2}}$	2.01×10^{2}	2.01×10^{2}
$f25$	2.13×10^{2}	2.15×10^{2}	2.14×10^{2}	2.15×10^{2}	2.10×10^{2}	$\mathbf{2.07 \times 10^{2}}$
$f26$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$
$f27$	4.11×10^{2}	4.13×10^{2}	4.18×10^{2}	4.17×10^{2}	4.11×10^{2}	$\mathbf{3.89 \times 10^{2}}$
$f28$	4.69×10^{2}	5.00×10^{2}	4.56×10^{2}	4.82×10^{2}	4.30×10^{2}	$\mathbf{4.21 \times 10^{2}}$
$f29$	$\mathbf{6.10 \times 10^{3}}$	4.51×10^{5}	6.80×10^{5}	6.24×10^{4}	5.51×10^{6}	1.11×10^{7}
$f30$	1.51×10^{3}	1.12×10^{3}	3.14×10^{3}	3.48×10^{3}	1.42×10^{3}	$\mathbf{9.97 \times 10^{2}}$
$w/t/l$	18/3/9	21/3/6	19/3/8	20/2/8	23/3/4	

Tables 6 and 7 present the comparison results on the CEC 2014 benchmark set for D = 30 and D = 50, respectively. From the tables, it can be observed that RLDMS-ABC outperforms most comparison algorithms on the majority of the test functions. Specifically, at D = 30, RLDMS-ABC performs better than ABC, ABCG, ARABC, APABC, and DPGABC on 18, 21, 19, 20, and 23 functions, respectively. When the dimension increases to D = 50, RLDMS-ABC still performs better on 17, 22, 22, 20, and 23 functions, respectively. These results indicate that RLDMS-ABC has strong overall optimization performance on the CEC 2014 benchmark set under both low and medium dimensional settings.

Table 7. Results on the CEC 2014 benchmark set with $D = 50$.

Function	ABC Average	ABCG Average	ARABC Average	APABC Average	DPGABC Average	RLDMS-ABC Average
$f1$	1.08×10^{7}	2.48×10^{7}	2.08×10^{7}	2.25×10^{7}	1.62×10^{7}	$\mathbf{4.97 \times 10^{5}}$
$f2$	$\mathbf{1.19 \times 10^{3}}$	1.99×10^{3}	3.16×10^{4}	7.62×10^{4}	4.35×10^{3}	6.59×10^{3}
$f3$	7.86×10^{3}	2.99×10^{3}	7.91×10^{3}	8.21×10^{3}	4.90×10^{3}	$\mathbf{3.53 \times 10^{1}}$
$f4$	$\mathbf{6.56 \times 10^{1}}$	1.01×10^{2}	8.69×10^{1}	9.41×10^{1}	9.75×10^{1}	7.22×10^{1}
$f5$	$\mathbf{2.02 \times 10^{1}}$	2.04×10^{1}	2.03×10^{1}	2.03×10^{1}	2.06×10^{1}	2.04×10^{1}
$f6$	3.09×10^{1}	3.69×10^{1}	2.84×10^{1}	2.80×10^{1}	$\mathbf{2.77 \times 10^{1}}$	2.81×10^{1}
$f7$	1.80×10^{-3}	1.73×10^{-9}	3.87×10^{-3}	1.31×10^{-2}	$\mathbf{6.74 \times 10^{-11}}$	4.93×10^{-4}
$f8$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	3.32×10^{-2}	$\mathbf{0.00 \times 10^{0}}$	5.31×10^{-1}	$\mathbf{0.00 \times 10^{0}}$
$f9$	1.93×10^{2}	1.76×10^{2}	1.08×10^{2}	1.15×10^{2}	1.23×10^{2}	$\mathbf{9.30 \times 10^{1}}$
$f10$	6.88×10^{-1}	8.05×10^{0}	7.27×10^{-1}	$\mathbf{6.89 \times 10^{-2}}$	3.94×10^{1}	3.92×10^{0}
$f11$	4.24×10^{3}	6.76×10^{3}	4.19×10^{3}	$\mathbf{4.14 \times 10^{3}}$	7.13×10^{3}	4.59×10^{3}
$f12$	$\mathbf{1.66 \times 10^{-1}}$	5.23×10^{-1}	3.09×10^{-1}	2.73×10^{-1}	7.29×10^{-1}	3.76×10^{-1}
$f13$	3.16×10^{-1}	4.22×10^{-1}	3.08×10^{-1}	3.60×10^{-1}	4.12×10^{-1}	$\mathbf{3.05 \times 10^{-1}}$
$f14$	$\mathbf{2.59 \times 10^{-1}}$	3.11×10^{-1}	2.96×10^{-1}	3.16×10^{-1}	3.05×10^{-1}	2.62×10^{-1}
$f15$	1.85×10^{1}	1.80×10^{1}	1.38×10^{1}	1.61×10^{1}	1.72×10^{1}	$\mathbf{1.14 \times 10^{1}}$
$f16$	1.81×10^{1}	1.89×10^{1}	$\mathbf{1.77 \times 10^{1}}$	$\mathbf{1.77 \times 10^{1}}$	1.94×10^{1}	1.80×10^{1}
$f17$	2.76×10^{6}	1.85×10^{6}	8.01×10^{6}	8.22×10^{6}	3.27×10^{6}	$\mathbf{4.30 \times 10^{4}}$
$f18$	$\mathbf{1.07 \times 10^{3}}$	1.25×10^{3}	8.75×10^{3}	1.38×10^{4}	1.51×10^{3}	1.13×10^{3}
$f19$	1.57×10^{1}	3.76×10^{1}	2.47×10^{1}	2.24×10^{1}	2.88×10^{1}	$\mathbf{1.18 \times 10^{1}}$
$f20$	3.05×10^{4}	8.60×10^{3}	3.01×10^{4}	2.96×10^{4}	8.46×10^{3}	$\mathbf{1.72 \times 10^{2}}$
$f21$	1.73×10^{6}	1.53×10^{6}	4.99×10^{6}	6.05×10^{6}	1.86×10^{6}	$\mathbf{4.97 \times 10^{4}}$
$f22$	8.03×10^{2}	4.98×10^{2}	8.15×10^{2}	8.65×10^{2}	5.71×10^{2}	$\mathbf{4.92 \times 10^{2}}$
$f23$	$\mathbf{3.27 \times 10^{2}}$	$\mathbf{3.27 \times 10^{2}}$	$\mathbf{3.27 \times 10^{2}}$	$\mathbf{3.27 \times 10^{2}}$	$\mathbf{3.27 \times 10^{2}}$	$\mathbf{3.27 \times 10^{2}}$
$f24$	2.05×10^{2}	$\mathbf{2.03 \times 10^{2}}$	2.04×10^{2}	2.04×10^{2}	$\mathbf{2.03 \times 10^{2}}$	$\mathbf{2.03 \times 10^{2}}$
$f25$	2.17×10^{2}	2.16×10^{2}	2.18×10^{2}	2.21×10^{2}	2.12×10^{2}	$\mathbf{2.06 \times 10^{2}}$
$f26$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$
$f27$	$\mathbf{4.31 \times 10^{2}}$	4.55×10^{2}	4.54×10^{2}	4.54×10^{2}	4.50×10^{2}	4.37×10^{2}
$f28$	5.73×10^{2}	6.37×10^{2}	5.68×10^{2}	5.90×10^{2}	5.19×10^{2}	$\mathbf{5.00 \times 10^{2}}$
$f29$	$\mathbf{2.52 \times 10^{3}}$	6.74×10^{3}	5.59×10^{3}	5.56×10^{3}	5.31×10^{3}	5.65×10^{7}
$f30$	1.05×10^{4}	8.54×10^{3}	1.60×10^{4}	1.58×10^{4}	1.02×10^{4}	$\mathbf{7.90 \times 10^{3}}$
$w/t/l$	17/3/10	22/5/3	22/2/6	20/3/7	23/3/4	

For the unimodal functions ($f_1 \sim f_3$), RLDMS-ABC obtains the best results on f_1, f_2, and f_3, indicating that the proposed algorithm has good convergence ability. For the multimodal functions ($f_4 \sim f_{16}$), RLDMS-ABC achieves superior results to all comparison algorithms on f_4, f_9, f_{13}, and f_{15}. For the remaining multimodal functions, RLDMS-ABC also shows good convergence performance and outperforms most of the compared algorithms on f_6, f_{14} and f_{16}. This indicates that RLDMS-ABC is able to avoid local optima and search for better solutions on multimodal functions. For the hybrid and composition functions ($f_{17} \sim f_{30}$), RLDMS-ABC obtains superior results to all comparison algorithms on f_{17}, f_{19}, f_{20}, f_{21}, f_{22}, f_{25}, f_{27}, f_{28}, and f_{30}, and achieves comparable results on f_{23} and f_{26}. This shows that RLDMS-ABC can obtain better results on complex optimization problems. In summary, the proposed strategies effectively improve the search performance of RLDMS-ABC under both $D = 30$ and $D = 50$.

Table 8 reports the comparison results on the CEC 2014 benchmark set when D = 100. As the problem dimension increases, the optimization difficulty becomes higher for all algorithms. Nevertheless, RLDMS-ABC still obtains competitive results on most test functions. According to the w/t/l results, RLDMS-ABC performs better than ABC, ABCG, ARABC, APABC, and DPGABC on 19, 22, 19, 20, and 26 functions, respectively. These results indicate that the proposed algorithm maintains good robustness and scalability in

high-dimensional optimization. However, RLDMS-ABC does not achieve the best result on all functions, especially on some complex composition functions such as f_{29}, suggesting that there is still room for further improvement in high-dimensional complex landscapes.

Table 8. Results on the CEC 2014 benchmark set with $D = 100$.

Function	ABC Average	ABCG Average	ARABC Average	APABC Average	DPGABC Average	RLDMS-ABC Average
$f1$	6.33×10^{7}	1.97×10^{8}	1.14×10^{8}	1.62×10^{8}	2.04×10^{8}	$\mathbf{2.69 \times 10^{6}}$
$f2$	$\mathbf{6.57 \times 10^{3}}$	7.89×10^{3}	6.58×10^{3}	1.27×10^{4}	1.71×10^{4}	1.39×10^{4}
$f3$	2.28×10^{4}	2.49×10^{4}	1.92×10^{4}	1.94×10^{4}	1.79×10^{3}	$\mathbf{6.30 \times 10^{0}}$
$f4$	1.78×10^{2}	1.95×10^{2}	1.85×10^{2}	1.77×10^{2}	1.90×10^{2}	$\mathbf{1.70 \times 10^{2}}$
$f5$	$\mathbf{2.03 \times 10^{1}}$	2.04×10^{1}	2.04×10^{1}	2.04×10^{1}	2.09×10^{1}	2.06×10^{1}
$f6$	8.24×10^{1}	8.18×10^{1}	7.65×10^{1}	7.67×10^{1}	9.05×10^{1}	$\mathbf{7.44 \times 10^{1}}$
$f7$	1.22×10^{-3}	1.59×10^{-10}	3.62×10^{-3}	1.71×10^{-3}	1.23×10^{-7}	$\mathbf{1.56 \times 10^{-15}}$
$f8$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	6.66×10^{-2}	$\mathbf{0.00 \times 10^{0}}$	3.00×10^{1}	8.29×10^{-16}
$f9$	6.97×10^{2}	6.17×10^{2}	5.31×10^{2}	5.59×10^{2}	5.14×10^{2}	$\mathbf{3.07 \times 10^{2}}$
$f10$	1.93×10^{0}	4.66×10^{2}	$\mathbf{1.69 \times 10^{0}}$	5.77×10^{0}	5.36×10^{2}	8.09×10^{0}
$f11$	1.15×10^{4}	1.37×10^{4}	1.23×10^{4}	$\mathbf{1.04 \times 10^{4}}$	2.05×10^{4}	1.23×10^{4}
$f12$	$\mathbf{2.40 \times 10^{-1}}$	1.05×10^{0}	2.69×10^{-1}	2.87×10^{-1}	1.39×10^{0}	5.45×10^{-1}
$f13$	3.44×10^{-1}	3.72×10^{-1}	$\mathbf{3.14 \times 10^{-1}}$	3.61×10^{-1}	5.95×10^{-1}	4.08×10^{-1}
$f14$	2.84×10^{-1}	2.70×10^{-1}	2.73×10^{-1}	2.99×10^{-1}	3.42×10^{-1}	$\mathbf{2.66 \times 10^{-1}}$
$f15$	6.19×10^{1}	5.62×10^{1}	5.31×10^{1}	5.81×10^{1}	5.51×10^{1}	$\mathbf{3.56 \times 10^{1}}$
$f16$	4.06×10^{1}	4.11×10^{1}	$\mathbf{3.99 \times 10^{1}}$	4.09×10^{1}	4.35×10^{1}	4.03×10^{1}
$f17$	1.14×10^{7}	1.38×10^{7}	1.12×10^{7}	1.43×10^{7}	2.58×10^{7}	$\mathbf{2.15 \times 10^{5}}$
$f18$	8.88×10^{2}	1.08×10^{3}	1.03×10^{3}	9.84×10^{2}	1.31×10^{3}	$\mathbf{8.38 \times 10^{2}}$
$f19$	$\mathbf{6.59 \times 10^{1}}$	1.13×10^{2}	9.39×10^{1}	1.01×10^{2}	1.05×10^{2}	9.39×10^{1}
$f20$	1.09×10^{5}	1.01×10^{5}	1.09×10^{5}	1.12×10^{5}	1.03×10^{4}	$\mathbf{4.02 \times 10^{2}}$
$f21$	7.39×10^{6}	7.85×10^{6}	7.52×10^{6}	7.79×10^{6}	1.10×10^{7}	$\mathbf{7.11 \times 10^{4}}$
$f22$	1.94×10^{3}	$\mathbf{1.71 \times 10^{3}}$	1.89×10^{3}	2.05×10^{3}	1.82×10^{3}	1.80×10^{3}
$f23$	3.49×10^{2}	3.57×10^{2}	3.51×10^{2}	3.52×10^{2}	$\mathbf{3.18 \times 10^{2}}$	3.19×10^{2}
$f24$	3.58×10^{2}	3.59×10^{2}	3.58×10^{2}	3.59×10^{2}	3.64×10^{2}	$\mathbf{2.87 \times 10^{2}}$
$f25$	2.49×10^{2}	2.52×10^{2}	2.50×10^{2}	2.48×10^{2}	2.66×10^{2}	$\mathbf{2.35 \times 10^{2}}$
$f26$	$\mathbf{1.59 \times 10^{2}}$	1.62×10^{2}	1.63×10^{2}	1.63×10^{2}	1.64×10^{2}	1.87×10^{2}
$f27$	$\mathbf{1.97 \times 10^{3}}$	2.08×10^{3}	2.02×10^{3}	2.01×10^{3}	2.25×10^{3}	2.18×10^{3}
$f28$	5.53×10^{3}	7.75×10^{3}	5.55×10^{3}	7.72×10^{3}	$\mathbf{2.21 \times 10^{3}}$	3.30×10^{3}
$f29$	3.18×10^{3}	$\mathbf{3.13 \times 10^{3}}$	4.21×10^{3}	5.95×10^{3}	3.18×10^{3}	8.76×10^{7}
$f30$	2.93×10^{4}	3.83×10^{4}	3.60×10^{4}	3.62×10^{4}	2.74×10^{4}	$\mathbf{9.38 \times 10^{3}}$
$w/t/l$	19/0/11	22/0/8	19/2/9	20/0/10	26/0/4	

Table 9 presents the Friedman ranking results of all experimental algorithms on the CEC 2014 benchmark set. As shown in the table, the average ranks of RLDMS-ABC are 2.40, 2.27, and 2.43 for $D = 30$, $D = 50$, and $D = 100$, respectively, which are the smallest among all compared algorithms. Since a smaller Friedman rank indicates better overall performance, these results further confirm that RLDMS-ABC achieves superior comprehensive optimization performance under different dimensional settings. Overall, the proposed method not only obtains good results on most test functions, but also shows strong stability and competitiveness in terms of overall statistical performance.

Figures 4–6 show the convergence curves of RLDMS-ABC and other ABC variants on the CEC 2014 benchmark set at 30, 50, and 100 dimensions, respectively. As can be seen from the Figures, RLDMS-ABC exhibits faster convergence speed and better convergence accuracy on most test functions. Compared with the other algorithms, RLDMS-ABC can decrease rapidly in the early stage of the search and still maintain good optimization capability in the later stage, thus obtaining better final results. This indicates that the

proposed method can effectively balance global exploration and local exploitation, thereby improving the overall search efficiency of the algorithm.

Table 9. Friedman test ranking of experimental algorithms.

Algorithms	CEC 2014 (D = 30) Ranking	CEC 2014 (D = 50) Ranking	CEC 2014 (D = 100) Ranking
ABC	3.10	3.08	2.85
ABCG	3.85	4.03	4.15
ARABC	3.93	3.83	3.12
APABC	4.08	4.05	3.83
DPGABC	3.63	3.73	4.62
RLDMS-ABC	**2.40**	**2.27**	**2.43**

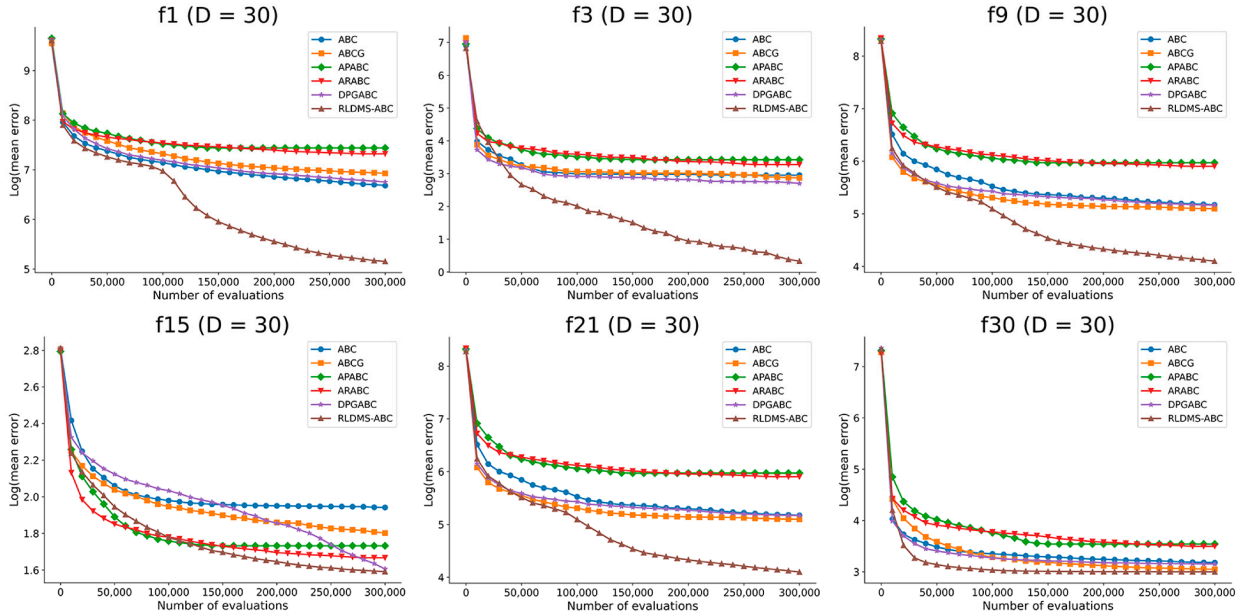


Figure 4. Convergence curves of RLDMS-ABC and the existing ABC variants when $D = 30$.

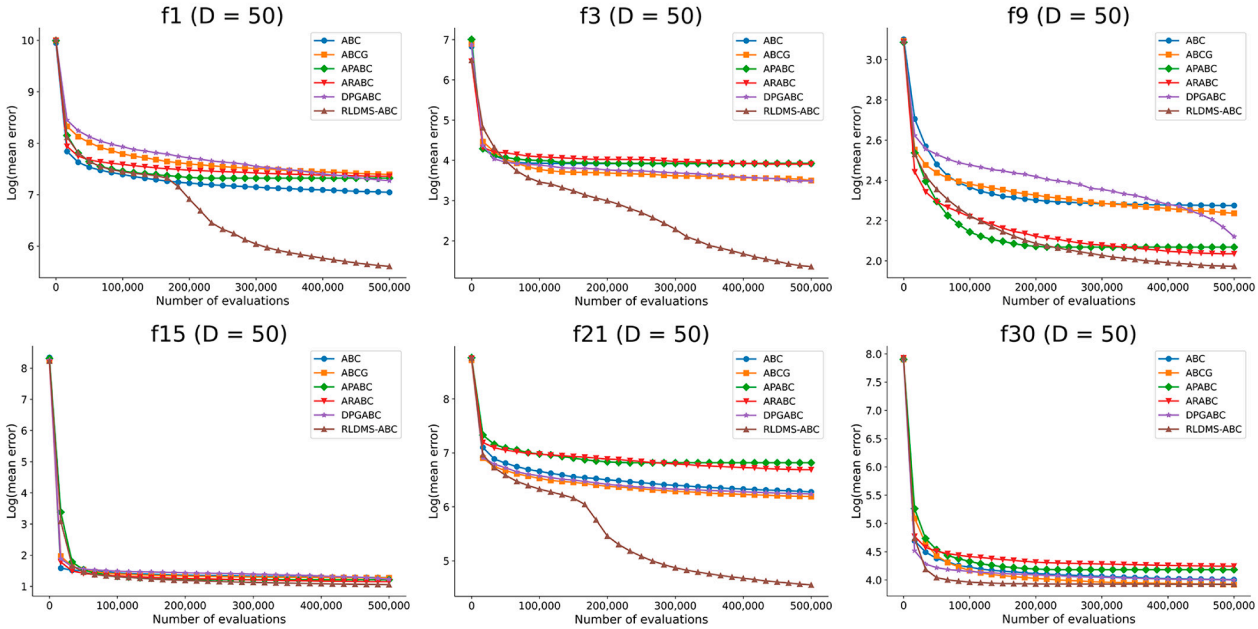


Figure 5. Convergence curves of RLDMS-ABC and the existing ABC variants when $D = 50$.

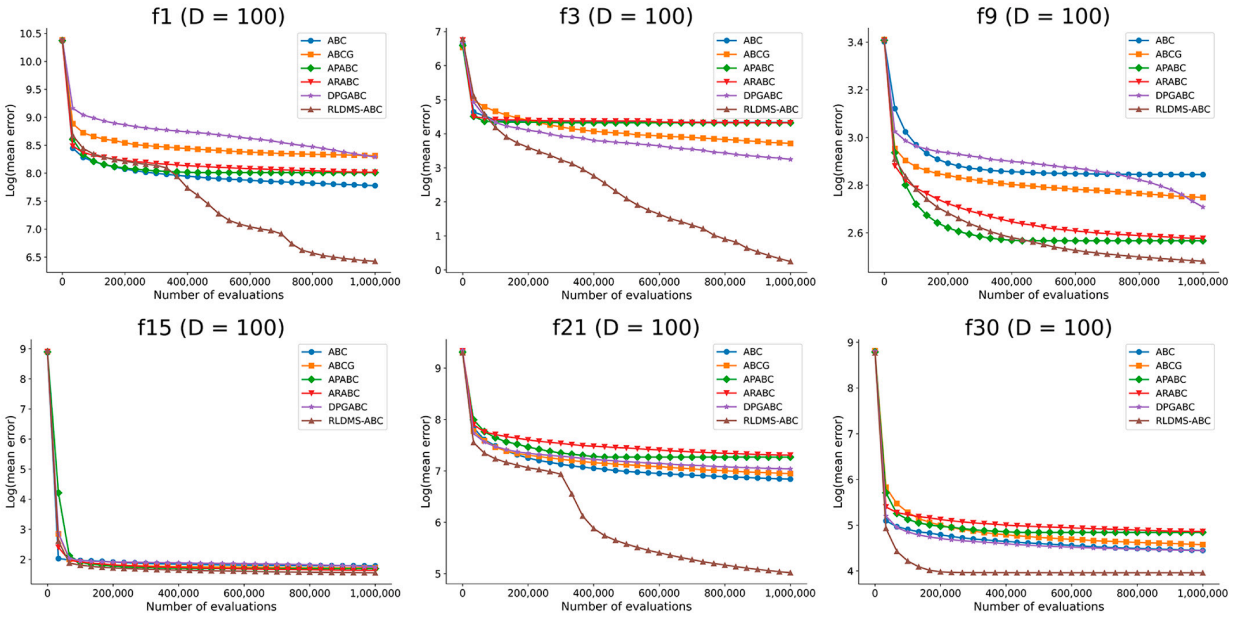


Figure 6. Convergence curves of RLDMS-ABC and the existing ABC variants when $D = 100$.

To provide a more balanced convergence analysis, Figure 7 shows the convergence curves on f_{11}, f_{18}, and f_{29} under D = 30, D = 50, and D = 100. These functions are selected because RLDMS-ABC does not always show superior performance on them, and they represent simple multimodal, hybrid, and composition functions, respectively.

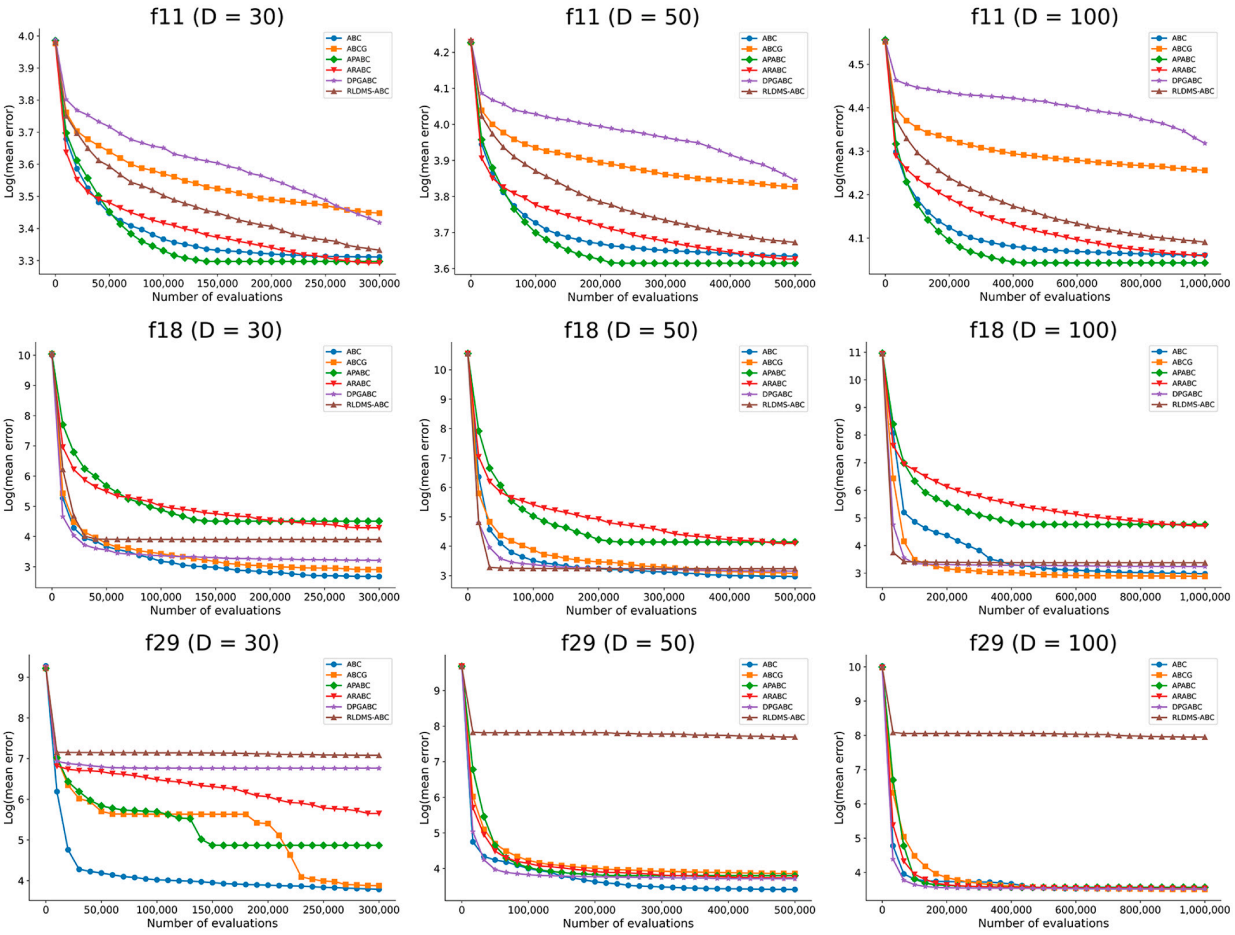


Figure 7. Additional convergence curves under $D = 30$, $D = 50$, and D =100.

The results in Figure 7 show that RLDMS-ABC still has convergence ability on f_{11} and f_{18}, but its advantage is not stable across all dimensions. The limitation is more obvious on f_{29}. On this function, RLDMS-ABC reduces the error rapidly in the early stage, but then stays at a relatively high error level. This indicates that the algorithm can quickly locate a promising region, but further improvement becomes difficult after the population concentrates in that region.

This behavior is related to the difference between unimodal and composition functions. For unimodal functions, the search landscape has a dominant optimum region and fewer misleading local regions. Therefore, the adaptive neighborhood regulation, local reference best guidance, and LocalMNNS-based exploitation can effectively strengthen local search and accelerate convergence. In contrast, f_{29} is a composition function with mixed landscape characteristics and multiple competing local regions. In this case, the local guidance and exploitation-oriented search of RLDMS-ABC tend to reinforce the current promising region. If this region is not close to the global optimum, the population concentration limits further exploration, leading to stagnation at a higher error level.

To further examine the statistical significance of the performance differences, the Wilcoxon signed-rank test is conducted at the significance level of $\alpha = 0.05$. Unlike the Friedman test, which mainly provides an overall ranking among algorithms, the Wilcoxon signed-rank test can be used for pairwise comparison between RLDMS-ABC and each competing algorithm. The test is performed based on the mean error values obtained on the 30 benchmark functions. Tables 10–12 report the Wilcoxon test results under D = 30, D = 50, and D = 100, respectively. In these tables, R^+ denotes the sum of ranks for the functions where RLDMS-ABC performs better, while R^- denotes the sum of ranks for the functions where the competing algorithm performs better.

Table 10. Wilcoxon signed-rank test results for $D = 30$.

Algorithm	R+	R−	*p*-Value
ABC	287.0	91.0	0.018
ABCG	310.0	68.0	0.003
ARABC	306.0	72.0	0.004
APABC	321.0	85.0	0.007
DPGABC	319.0	59.0	0.001

Table 11. Wilcoxon signed-rank test results for $D = 50$.

Algorithm	R+	R−	*p*-Value
ABC	251.0	127.0	0.141
ABCG	279.0	46.0	0.001
ARABC	329.0	77.0	0.004
APABC	302.0	76.0	0.006
DPGABC	325.0	53.0	0.001

Table 12. Wilcoxon signed-rank test results for $D = 100$.

Algorithm	R+	R−	*p*-Value
ABC	320.0	145.0	0.071
ABCG	357.5	107.5	0.010
ARABC	297.0	109.0	0.032
APABC	336.0	129.0	0.032
DPGABC	396.0	69.0	0.000

As shown in Tables 10–12, R^+ is larger than R^- in all pairwise comparisons, indicating that RLDMS-ABC obtains better overall performance tendency than the compared algorithms. For D = 30, all p-values are smaller than 0.05, showing that RLDMS-ABC significantly outperforms all compared algorithms. For D = 50 and D = 100, RLDMS-ABC significantly outperforms ABCG, ARABC, APABC, and DPGABC, while its differences from ABC are not statistically significant. Overall, the Wilcoxon test results further confirm the competitive performance of RLDMS-ABC.

4.4. Comparison with Representative ABC Variants Using Different Neighborhood Structures

Existing studies have shown that neighborhood structure plays an important role in determining the search behavior, exploitation efficiency, and convergence performance of ABC variants. Representative examples include neighborhood-radius-based selection, adaptive neighborhood search, random neighborhood structure design, and neighborhood-search-enhanced modified ABC variants [4–6,32,43]. To further examine whether the proposed RLDMS-ABC can utilize neighborhood information more effectively, an additional comparative study was conducted against several representative ABC variants with different neighborhood structures on the CEC 2014 benchmark set. Specifically, five neighborhood-related ABC variants were selected for comparison, namely NABC [43], NSABC [4], MABC-NS [32], ABCNG [5], and RNSABC [6].

Tables 13 and 14 give the comparison results of RLDMS-ABC and the above five ABC variants on the CEC 2014 benchmark set at $D = 30$ and $D = 50$, respectively. From the tables, it can be observed that RLDMS-ABC outperforms most comparison algorithms on most test functions. At $D = 30$, compared with NABC, NSABC, MABC-NS, ABCNG, and RNSABC, RLDMS-ABC achieves better results on 19, 17, 30, 20, and 16 functions, respectively. At D = 50, RLDMS-ABC achieves better results on 18, 18, 28, 18, and 19 functions, respectively. These results indicate that RLDMS-ABC has strong competitiveness against the existing ABC variants based on neighborhood structures.

Table 13. The comparison results of RLDMS-ABC with another five ABC variants on the CEC 2014 benchmark set when D = 30.

Function	NABC Average	NSABC Average	MABC-NS Average	ABCNG Average	RNSABC Average	RLDMS-ABC Average
$f1$	5.50×10^{6}	4.99×10^{6}	1.90×10^{8}	1.78×10^{7}	6.71×10^{6}	$\mathbf{1.42 \times 10^{5}}$
$f2$	1.56×10^{2}	5.80×10^{2}	1.09×10^{10}	1.18×10^{2}	4.66×10^{2}	$\mathbf{3.94 \times 10^{-3}}$
$f3$	5.17×10^{2}	6.91×10^{2}	1.92×10^{4}	7.65×10^{2}	1.06×10^{3}	$\mathbf{2.13 \times 10^{0}}$
$f4$	7.07×10^{1}	4.57×10^{1}	8.55×10^{2}	6.69×10^{1}	4.54×10^{1}	$\mathbf{5.63 \times 10^{-2}}$
$f5$	2.02×10^{1}	$\mathbf{2.00 \times 10^{1}}$	2.09×10^{1}	2.01×10^{1}	2.02×10^{1}	2.03×10^{1}
$f6$	1.34×10^{1}	$\mathbf{1.19 \times 10^{1}}$	2.80×10^{1}	1.44×10^{1}	1.29×10^{1}	1.26×10^{1}
$f7$	2.02×10^{-6}	$\mathbf{1.11 \times 10^{-6}}$	1.10×10^{2}	3.48×10^{-5}	2.21×10^{-5}	1.40×10^{-3}
$f8$	$\mathbf{0.00 \times 10^{0}}$	3.32×10^{-2}	1.43×10^{2}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	5.92×10^{-17}
$f9$	4.71×10^{1}	4.41×10^{1}	1.74×10^{2}	5.69×10^{1}	4.30×10^{1}	$\mathbf{3.90 \times 10^{1}}$
$f10$	1.71×10^{0}	1.81×10^{0}	3.70×10^{3}	2.17×10^{0}	$\mathbf{3.49 \times 10^{-1}}$	1.66×10^{0}
$f11$	1.83×10^{3}	$\mathbf{1.63 \times 10^{3}}$	4.43×10^{3}	2.03×10^{3}	1.76×10^{3}	2.19×10^{3}
$f12$	2.64×10^{-1}	$\mathbf{1.13 \times 10^{-1}}$	8.90×10^{-1}	1.86×10^{-1}	2.22×10^{-1}	3.62×10^{-1}
$f13$	2.69×10^{-1}	2.28×10^{-1}	2.30×10^{0}	2.54×10^{-1}	$\mathbf{2.08 \times 10^{-1}}$	2.14×10^{-1}
$f14$	2.51×10^{-1}	$\mathbf{1.90 \times 10^{-1}}$	2.95×10^{1}	2.55×10^{-1}	2.12×10^{-1}	2.06×10^{-1}
$f15$	5.59×10^{0}	$\mathbf{4.13 \times 10^{0}}$	2.70×10^{3}	5.81×10^{0}	4.97×10^{0}	5.18×10^{0}
$f16$	9.70×10^{0}	$\mathbf{9.07 \times 10^{0}}$	1.22×10^{1}	9.75×10^{0}	9.19×10^{0}	9.70×10^{0}
$f17$	1.69×10^{6}	1.25×10^{6}	1.13×10^{7}	4.20×10^{6}	1.86×10^{6}	$\mathbf{1.39 \times 10^{4}}$
$f18$	3.49×10^{2}	2.11×10^{3}	8.34×10^{5}	$\mathbf{3.39 \times 10^{2}}$	4.95×10^{3}	6.90×10^{3}
$f19$	7.27×10^{0}	6.47×10^{0}	1.06×10^{2}	8.23×10^{0}	6.94×10^{0}	$\mathbf{4.29 \times 10^{0}}$

Table 13. *Cont.*

Function	NABC Average	NSABC Average	MABC-NS Average	ABCNG Average	RNSABC Average	RLDMS-ABC Average
$f20$	5.76×10^{3}	6.87×10^{3}	4.55×10^{4}	8.47×10^{3}	7.14×10^{3}	$\mathbf{1.50 \times 10^{2}}$
$f21$	2.28×10^{5}	2.13×10^{5}	1.32×10^{6}	4.85×10^{5}	3.91×10^{5}	$\mathbf{1.26 \times 10^{4}}$
$f22$	2.29×10^{2}	2.30×10^{2}	7.11×10^{2}	2.92×10^{2}	2.97×10^{2}	$\mathbf{9.99 \times 10^{1}}$
$f23$	$\mathbf{2.90 \times 10^{2}}$	$\mathbf{2.90 \times 10^{2}}$	3.96×10^{2}	$\mathbf{2.90 \times 10^{2}}$	$\mathbf{2.90 \times 10^{2}}$	$\mathbf{2.90 \times 10^{2}}$
$f24$	$\mathbf{2.01 \times 10^{2}}$	$\mathbf{2.01 \times 10^{2}}$	2.18×10^{2}	$\mathbf{2.01 \times 10^{2}}$	$\mathbf{2.01 \times 10^{2}}$	$\mathbf{2.01 \times 10^{2}}$
$f25$	2.13×10^{2}	2.11×10^{2}	2.50×10^{2}	2.13×10^{2}	2.13×10^{2}	$\mathbf{2.07 \times 10^{2}}$
$f26$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	1.01×10^{2}	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$
$f27$	4.12×10^{2}	4.06×10^{2}	6.85×10^{2}	4.18×10^{2}	4.13×10^{2}	$\mathbf{3.89 \times 10^{2}}$
$f28$	4.55×10^{2}	4.53×10^{2}	7.62×10^{2}	4.58×10^{2}	4.57×10^{2}	$\mathbf{4.21 \times 10^{2}}$
$f29$	1.77×10^{6}	5.04×10^{6}	5.27×10^{7}	3.43×10^{6}	$\mathbf{1.09 \times 10^{6}}$	1.11×10^{7}
$f30$	1.58×10^{3}	1.54×10^{3}	2.68×10^{5}	3.81×10^{3}	1.95×10^{3}	$\mathbf{9.97 \times 10^{2}}$
$w/t/l$	19/4/7	17/3/10	30/0/0	20/3/7	16/3/11	

Table 14. The comparison results of RLDMS-ABC with another five ABC variants on the CEC 2014 benchmark set when $D = 50$.

Function	NABC Average	NSABC Average	MABC-NS Average	ABCNG Average	RNSABC Average	RLDMS-ABC Average
$f1$	1.19×10^{7}	7.45×10^{6}	3.48×10^{8}	2.09×10^{7}	1.28×10^{7}	$\mathbf{4.97 \times 10^{5}}$
$f2$	2.19×10^{3}	3.78×10^{3}	4.28×10^{10}	$\mathbf{7.40 \times 10^{2}}$	5.84×10^{3}	6.59×10^{3}
$f3$	6.60×10^{3}	6.44×10^{3}	7.85×10^{4}	9.57×10^{3}	6.62×10^{3}	$\mathbf{3.53 \times 10^{1}}$
$f4$	9.77×10^{1}	7.24×10^{1}	6.62×10^{3}	9.16×10^{1}	8.01×10^{1}	$\mathbf{7.22 \times 10^{1}}$
$f5$	2.03×10^{1}	$\mathbf{2.00 \times 10^{1}}$	2.11×10^{1}	2.02×10^{1}	2.03×10^{1}	2.04×10^{1}
$f6$	2.96×10^{1}	$\mathbf{2.53 \times 10^{1}}$	5.44×10^{1}	3.01×10^{1}	2.81×10^{1}	2.81×10^{1}
$f7$	$\mathbf{3.25 \times 10^{-4}}$	5.99×10^{-3}	4.21×10^{2}	4.86×10^{-4}	3.28×10^{-3}	4.93×10^{-4}
$f8$	$\mathbf{0.00 \times 10^{0}}$	6.63×10^{-2}	3.44×10^{2}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$
$f9$	1.21×10^{2}	1.07×10^{2}	3.95×10^{2}	1.31×10^{2}	1.09×10^{2}	$\mathbf{9.30 \times 10^{1}}$
$f10$	3.22×10^{0}	5.44×10^{0}	8.28×10^{3}	2.95×10^{0}	$\mathbf{9.12 \times 10^{-1}}$	3.92×10^{0}
$f11$	4.14×10^{3}	$\mathbf{3.76 \times 10^{3}}$	9.18×10^{3}	4.44×10^{3}	3.95×10^{3}	4.59×10^{3}
$f12$	2.65×10^{-1}	$\mathbf{1.11 \times 10^{-1}}$	1.19×10^{0}	2.25×10^{-1}	2.30×10^{-1}	3.76×10^{-1}
$f13$	3.28×10^{-1}	3.06×10^{-1}	4.38×10^{0}	3.09×10^{-1}	$\mathbf{2.47 \times 10^{-1}}$	3.05×10^{-1}
$f14$	2.78×10^{-1}	$\mathbf{1.81 \times 10^{-1}}$	1.04×10^{2}	2.96×10^{-1}	2.66×10^{-1}	2.62×10^{-1}
$f15$	1.39×10^{1}	$\mathbf{1.14 \times 10^{1}}$	7.96×10^{4}	1.35×10^{1}	1.21×10^{1}	$\mathbf{1.14 \times 10^{1}}$
$f16$	1.78×10^{1}	1.80×10^{1}	2.15×10^{1}	1.80×10^{1}	$\mathbf{1.74 \times 10^{1}}$	1.80×10^{1}
$f17$	3.61×10^{6}	2.43×10^{6}	5.12×10^{7}	1.03×10^{7}	4.85×10^{6}	$\mathbf{4.30 \times 10^{4}}$
$f18$	1.46×10^{3}	1.95×10^{3}	2.30×10^{8}	$\mathbf{6.53 \times 10^{2}}$	1.70×10^{3}	1.13×10^{3}
$f19$	2.89×10^{1}	1.83×10^{1}	2.20×10^{2}	2.57×10^{1}	2.10×10^{1}	$\mathbf{1.18 \times 10^{1}}$
$f20$	2.33×10^{4}	2.61×10^{4}	8.42×10^{4}	3.21×10^{4}	3.26×10^{4}	$\mathbf{1.72 \times 10^{2}}$
$f21$	2.23×10^{6}	2.02×10^{6}	1.10×10^{7}	4.63×10^{6}	2.36×10^{6}	$\mathbf{4.97 \times 10^{4}}$
$f22$	7.07×10^{2}	7.82×10^{2}	1.85×10^{3}	7.60×10^{2}	7.59×10^{2}	$\mathbf{4.92 \times 10^{2}}$
$f23$	$\mathbf{3.27 \times 10^{2}}$	$\mathbf{3.27 \times 10^{2}}$	4.10×10^{2}	$\mathbf{3.27 \times 10^{2}}$	3.28×10^{2}	$\mathbf{3.27 \times 10^{2}}$
$f24$	2.04×10^{2}	2.04×10^{2}	2.97×10^{2}	2.04×10^{2}	2.04×10^{2}	$\mathbf{2.03 \times 10^{2}}$
$f25$	2.18×10^{2}	2.12×10^{2}	3.55×10^{2}	2.20×10^{2}	2.18×10^{2}	$\mathbf{2.06 \times 10^{2}}$
$f26$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	1.02×10^{2}	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$
$f27$	4.37×10^{2}	4.27×10^{2}	$\mathbf{4.00 \times 10^{2}}$	4.54×10^{2}	4.40×10^{2}	4.37×10^{2}
$f28$	5.53×10^{2}	5.49×10^{2}	$\mathbf{4.00 \times 10^{2}}$	5.83×10^{2}	5.55×10^{2}	5.00×10^{2}
$f29$	$\mathbf{4.38 \times 10^{3}}$	5.79×10^{3}	3.12×10^{8}	1.77×10^{4}	1.28×10^{4}	5.65×10^{7}
$f30$	1.02×10^{4}	9.57×10^{3}	2.16×10^{6}	2.36×10^{4}	1.27×10^{4}	$\mathbf{7.90 \times 10^{3}}$
$w/t/l$	18/4/8	18/4/8	28/0/2	18/4/8	19/3/8	

Compared with these neighborhood-based ABC variants, the advantage of RLDMS-ABC lies in that it not only utilizes neighborhood information, but also adaptively regulates neighborhood size through reinforcement learning in the employed bee phase and further coordinates neighborhood-guided search with multi-strategy collaboration and guided restart in the onlooker bee and scout bee phases [4–6,12,13,30–32,38–40]. Therefore, the proposed method can make more effective use of local structural information while preserving sufficient search flexibility and population diversity.

Table 15 presents the Friedman ranking results of ABC algorithms with different neighborhood structures on the CEC 2014 benchmark set. As shown in the table, the average ranks of RLDMS-ABC are 2.38 and 2.38 at $D = 30$ and $D = 50$, respectively, which are the smallest among all compared algorithms. Since a smaller average rank in the Friedman test indicates better overall performance, these results demonstrate that RLDMS-ABC achieves the best comprehensive optimization performance under different dimensional settings. Overall, Table 10 further confirms that the proposed method can utilize neighborhood information more effectively and is statistically superior to the other ABC variants based on neighborhood structures.

Table 15. Friedman test ranking of algorithms with different neighborhood structures.

Algorithms	CEC 2014 (D = 30) Ranking	CEC 2014 (D = 50) Ranking
NABC	3.17	3.20
NSABC	2.40	2.62
MABC-NS	6.00	5.67
ABCNG	3.97	3.82
RNSABC	3.08	3.32
RLDMS-ABC	**2.38**	**2.38**

4.5. Complexity Analysis

To evaluate the additional computational overhead, a timing-based complexity measure is adopted. Here, T0 is the machine-dependent time for fixed basic operations, T1 is the time consumed only by objective function evaluations, and $\overline{T2}$ is the average running time of the complete algorithm until the maximum number of function evaluations is reached. The index $(\overline{T2} - T1)/T0$ measures the extra overhead introduced by the algorithm itself.

As shown in Table 16, RLDMS-ABC has higher complexity than ABC, APABC, and ARABC because reinforcement learning-based neighborhood adjustment, dynamic multi-neighborhood search, and multiple onlooker bee strategies are introduced. However, its complexity is lower than that of DPGABC and ABCG, indicating that the proposed mechanisms keep the computational cost within an acceptable range while improving optimization performance.

Table 16. Computational complexity comparison on CEC2014 when $D = 30$.

Algorithm	$\overline{T2}$	$(\overline{T2} - T1)/T0$
ABC	4.6127	1.5291
APABC	3.7658	3.6399
ARABC	4.2379	4.7088
DPGABC	3.9187	10.1658
ABCG	5.4249	16.8762
RLDMS-ABC	4.7565	7.1740

4.6. Engineering Application: Tension/Compression Spring Design Problem

To further verify the practical applicability of the proposed RLDMS-ABC algorithm, a classical constrained engineering design problem, namely the tension/compression spring design problem, is used in this subsection. The objective of this problem is to minimize the weight of a spring while satisfying several engineering constraints.

The design variables are x_1, x_2, and x_3, which denote the wire diameter, mean coil diameter, and number of active coils, respectively. The mathematical model of this problem is formulated in Equations (19) and (20).

$$minf(x) = (\ x_3 + 2\ )x_2x_1^2, \tag{19}$$

subject to

$$\begin{gathered} g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0, \\ g_2(x) = \frac{4x_2^2 - x_1x_2}{12566\left(x_2x_1^3 - x_1^4\right)} + \frac{1}{5108x_1^2} - 1 \leq 0, \\ g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0, \\ g_4(x) = \frac{x_1+x_2}{1.5} - 1 \leq 0, \\ 0.05 \leq x_1 \leq 2.00,\ 0.25 \leq x_2 \leq 1.30,\ 2.00 \leq x_3 \leq 15.00, \end{gathered} \tag{20}$$

Since the spring design problem is a constrained optimization problem, a penalty function is used to handle the constraints. The constrained optimization problem is transformed into an unconstrained optimization problem by Equation (21).

$$F(x) = f(x) + \lambda \sum_{k=1}^{4} \max(0, g_k(x))^2, \tag{21}$$

where λ is set to 10^8. If a candidate solution satisfies all constraints, the penalty term is zero. Otherwise, the solution is penalized according to its constraint violation. The experimental results are shown in Table 17.

Table 17. Results on the tension/compression spring design problem.

Algorithm	Best	Mean	Std	Worst	Max Violation
ABC	1.2679×10^{-2}	1.2773×10^{-2}	7.1134×10^{-5}	1.2971×10^{-2}	$\mathbf{0.0000 \times 10^{0}}$
ARABC	1.2704×10^{-2}	1.3040×10^{-2}	2.3202×10^{-4}	1.3493×10^{-2}	$\mathbf{0.0000 \times 10^{0}}$
ABCG	1.2992×10^{-2}	1.4210×10^{-2}	6.1745×10^{-4}	1.5249×10^{-2}	$\mathbf{0.0000 \times 10^{0}}$
APABC	1.2733×10^{-2}	1.3617×10^{-2}	3.7062×10^{-4}	1.4312×10^{-2}	$\mathbf{0.0000 \times 10^{0}}$
RLDMS-ABC	$\mathbf{1.2665 \times 10^{-2}}$	$\mathbf{1.2681 \times 10^{-2}}$	$\mathbf{1.9048 \times 10^{-5}}$	$\mathbf{1.2742 \times 10^{-2}}$	$\mathbf{0.0000 \times 10^{0}}$

As shown in Table 17, all compared algorithms obtain feasible solutions on the tension/compression spring design problem, since the maximum constraint violation is zero within the numerical tolerance. Compared with ABC, ARABC, ABCG, and APABC, RLDMS-ABC achieves the smallest best, mean, standard deviation, and worst values. These results indicate that RLDMS-ABC has better solution quality and stability on this engineering design problem. Therefore, the proposed algorithm can be applied not only to benchmark functions but also to practical engineering optimization problems.

5. Conclusions

In this paper, an artificial bee colony algorithm with dual groups and multiple strategies based on reinforcement learning, named RLDMS-ABC, is proposed for continuous optimization problems. The algorithm improves the standard ABC framework by introducing adaptive neighborhood regulation, dual-group multi-strategy search, and guided scout bee restart. These mechanisms enhance the utilization of neighborhood information,

improve search flexibility, and help balance exploration and exploitation. Experiments on the CEC 2014 benchmark set under different dimensional settings demonstrate that RLDMS-ABC achieves competitive performance in terms of solution quality, convergence speed, and robustness. Compared with representative ABC variants and neighborhood-structure-based ABC algorithms, the proposed algorithm obtains better or comparable results on most test functions. The ablation study further confirms the effectiveness of the proposed components. In addition, a constrained engineering design problem, namely the tension/compression spring design problem, is introduced to verify the practical applicability of RLDMS-ABC. The experimental results show that RLDMS-ABC obtains a feasible solution and achieves better solution quality and stability than the compared algorithms on this engineering application. This further indicates that the proposed algorithm can be applied not only to benchmark function optimization but also to practical engineering optimization problems. Future work will focus on improving its performance on high-dimensional and constrained optimization problems, as well as further investigating its application to practical engineering optimization scenarios.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/math14132276/s1, Table S1: Comparison results between RLDMS-ABC and five ABC variants on the DFS benchmark set for $D = 30$; Table S2: Comparison results between RLDMS-ABC and five ABC variants on the DFS benchmark set for $D = 50$.

Author Contributions: Conceptualization, Z.L.; Methodology, Z.L.; Software, Y.C.; Investigation, Y.C.; Data curation, Y.C.; Writing—original draft, Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the General Project of the Education Department of Liaoning Province, grant number LJ212410153017.

Data Availability Statement: The data presented in this study are openly available in CEC 2014 set described set in Ref. [40]. Additional information is available online: https://arxiv.org/html/2603.24140v1, (accessed on 19 May 2026).

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

References

1. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization*; Technical Report TR06; Erciyes University, Engineering Faculty, Computer Engineering Department: Kayseri, Turkey, 2005.
2. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
3. Karaboga, D.; Gorkemli, B.; Ozturk, C.; Karaboga, N. A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. *Artif. Intell. Rev.* **2014**, *42*, 21–57. [CrossRef]
4. Wang, H.; Wang, W.; Xiao, S.; Cui, Z.; Xu, M.; Zhou, X. Improving artificial bee colony algorithm using a new neighborhood selection mechanism. *Inf. Sci.* **2020**, *527*, 227–240. [CrossRef]
5. Xiao, S.; Wang, H.; Wang, W.; Huang, Z.; Zhou, X.; Xu, M. Artificial bee colony algorithm based on adaptive neighborhood search and Gaussian perturbation. *Appl. Soft Comput.* **2021**, *100*, 106955. [CrossRef]
6. 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]
7. Zhu, G.; Kwong, S. Gbest-guided artificial bee colony algorithm for numerical function optimization. *Appl. Math. Comput.* **2010**, *217*, 3166–3173. [CrossRef]
8. Karaboga, D.; Gorkemli, B. A quick artificial bee colony (qABC) algorithm and its performance on optimization problems. *Appl. Soft Comput.* **2014**, *23*, 227–238. [CrossRef]
9. Wang, H.; Wu, Z.; Rahnamayan, S.; Sun, H.; Liu, Y.; Pan, J.S. Multi-strategy ensemble artificial bee colony algorithm. *Inf. Sci.* **2014**, *279*, 587–603. [CrossRef]

10. Liu, J.; Wang, W.; Wu, J.; Wang, H.; Zhang, H.; Hu, M. Artificial bee colony based on adaptive search strategies and elite selection mechanism. In *International Conference on Neural Computing for Advanced Applications (NCAA 2023)*; Zhang, H., Ed.; Communications in Computer and Information Science; Springer: Singapore, 2023; Volume 1869, pp. 302–314. [CrossRef]
11. Huang, Y.; Yu, Y.; Guo, J.; Wu, Y. Self-adaptive artificial bee colony with a candidate strategy pool. *Appl. Sci.* **2023**, *13*, 10445. [CrossRef]
12. Cui, Y.; Hu, W.; Rahmani, A. A reinforcement learning based artificial bee colony algorithm with application in robot path planning. *Expert Syst. Appl.* **2022**, *203*, 117389. [CrossRef]
13. Ni, X.; Hu, W.; Fan, Q.; Cui, Y.; Qi, C. A Q-learning-based multi-strategy integrated artificial bee colony algorithm with application in unmanned vehicle path planning. *Expert Syst. Appl.* **2024**, *236*, 121303. [CrossRef]
14. Tizhoosh, H.R. Opposition-based learning: A new scheme for machine intelligence. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation, Vienna, Austria, 28–30 November 2005*; IEEE: Piscataway, NJ, USA, 2005; Volume 1, pp. 695–701. [CrossRef]
15. Zhao, J.; Lv, L.; Sun, H. Artificial bee colony using opposition-based learning. In *Genetic and Evolutionary Computing*; Sun, H., Yang, C.Y., Lin, C.W., Pan, J.S., Snasel, V., Abraham, A., Eds.; Advances in Intelligent Systems and Computing; Springer: Cham, Switzerland, 2015; Volume 329, pp. 3–10. [CrossRef]
16. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018.
17. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: New York, NY, USA, 1994.
18. Watkins, C.J.C.H.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
19. Gao, W.F.; Huang, L.L.; Liu, S.Y.; Chan, F.T.S.; Dai, C.; Shan, X. Artificial bee colony algorithm with multiple search strategies. *Appl. Math. Comput.* **2015**, *271*, 269–287. [CrossRef]
20. Yang, J.; Cui, J.; Xia, X.; Gao, X.; Yang, B.; Zhang, Y.D. An artificial bee colony algorithm with an adaptive search strategy selection mechanism and its application on workload prediction. *Comput. Ind. Eng.* **2024**, *189*, 109982. [CrossRef]
21. Zeng, T.; Wang, W.; Wang, H.; Cui, Z.; Wang, F.; Wang, Y.; Zhao, J. Artificial bee colony based on adaptive search strategy and random grouping mechanism. *Expert Syst. Appl.* **2022**, *192*, 116332. [CrossRef]
22. Kong, D.; Chang, T.; Dai, W.; Wang, Q.; Sun, H. An improved artificial bee colony algorithm based on elite group guidance and combined breadth-depth search strategy. *Inf. Sci.* **2018**, *442–443*, 54–71. [CrossRef]
23. Gao, W.F.; Liu, S.Y.; Huang, L.L. A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. *IEEE Trans. Cybern.* **2013**, *43*, 1011–1024. [CrossRef] [PubMed]
24. Li, G.; Niu, P.; Xiao, X. Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. *Appl. Soft Comput.* **2012**, *12*, 320–332. [CrossRef]
25. Zhou, X.; Lu, J.; Huang, J.; Zhong, M.; Wang, M. Enhancing artificial bee colony algorithm with multi-elite guidance. *Inf. Sci.* **2021**, *543*, 242–258. [CrossRef]
26. Guo, Z.; Li, H.; Li, K. Dual subpopulation artificial bee colony algorithm based on individual gradation. *Egypt. Inform. J.* **2024**, *25*, 100452. [CrossRef]
27. Zhou, X.; Zhang, X.; Gao, W.; Wang, H.; Ma, Y. Adaptive multi-population artificial bee colony algorithm based on fitness landscape analysis. *Appl. Soft Comput.* **2024**, *164*, 111952. [CrossRef]
28. Lin, R.; Xu, Z.; Yu, L.; Wei, T. EABC-AS: Elite-driven artificial bee colony algorithm with adaptive population scaling. *Swarm Evol. Comput.* **2025**, *94*, 101893. [CrossRef]
29. Wang, J.; Tang, H.; Lei, D. A Q-learning artificial bee colony for distributed assembly flow shop scheduling with factory eligibility, transportation capacity and setup time. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106230. [CrossRef]
30. Singh, A.; Deep, K. Exploration-exploitation balance in artificial bee colony algorithm: A critical analysis. *Soft Comput.* **2019**, *23*, 9525–9536. [CrossRef]
31. Gao, W.F.; Chan, F.T.S.; Huang, L.L.; Liu, S.Y. Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood. *Inf. Sci.* **2015**, *316*, 180–200. [CrossRef]
32. Zhou, X.; Wang, H.; Wang, M.; Wan, J. Enhancing the modified artificial bee colony algorithm with neighborhood search. *Soft Comput.* **2017**, *21*, 2733–2743. [CrossRef]
33. Cui, L.; Li, G.; Wang, X.; Lin, Q.; Chen, J.; Lu, N.; Lu, J. A ranking-based adaptive artificial bee colony algorithm for global numerical optimization. *Inf. Sci.* **2017**, *417*, 169–185. [CrossRef]
34. Guo, Z.; Li, H.; Zhang, W. Artificial bee colony algorithm with multi-strategy adaptation. *Int. J. Bio-Inspired Comput.* **2024**, *23*, 135–147. [CrossRef]
35. Wang, J.; Liu, Y.; Rao, S.; Zhou, X.; Hu, J. A novel self-adaptive multi-strategy artificial bee colony algorithm for coverage optimization in wireless sensor networks. *Ad Hoc Netw.* **2023**, *150*, 103284. [CrossRef]
36. Chen, X.; Tianfield, H.; Li, K. Self-adaptive differential artificial bee colony algorithm for global optimization problems. *Swarm Evol. Comput.* **2019**, *45*, 70–91. [CrossRef]

37. Cui, L.; Li, G.; Zhu, Z.; Lin, Q.; Wen, Z.; Lu, N.; Wong, K.C.; Chen, J. A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization. *Inf. Sci.* **2017**, *414*, 53–67. [CrossRef]
38. Anuar, S.; Selamat, A.; Sallehuddin, R. A modified scout bee for artificial bee colony algorithm and its performance on optimization problems. *J. King Saud Univ.—Comput. Inf. Sci.* **2016**, *28*, 395–406. [CrossRef]
39. Zhang, X.; Wu, Z. An artificial bee colony algorithm with history-driven scout bees phase. In *Advances in Swarm and Computational Intelligence, ICSI 2015*; Tan, Y., Shi, Y., Buarque, F., Gelbukh, A., Das, S., Engelbrecht, A., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2015; Volume 9140, pp. 239–246. [CrossRef]
40. Kaya, E.; Kaya, C.B. A novel neural network training algorithm for the identification of nonlinear static systems: Artificial bee colony algorithm based on effective scout bee stage. *Symmetry* **2021**, *13*, 419. [CrossRef]
41. Liang, J.J.; Qu, B.Y.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*; Technical Report 201311; Zhengzhou University: Zhengzhou, China; Nanyang Technological University: Singapore, 2013.
42. Xiang, W.L.; Meng, X.L.; Li, Y.Z.; He, R.C.; An, M.Q. An improved artificial bee colony algorithm based on the gravity model. *Inf. Sci.* **2018**, *429*, 49–71. [CrossRef]
43. Peng, H.; Deng, C.; Wu, Z. Best neighbor-guided artificial bee colony algorithm for continuous optimization problems. *Soft Comput.* **2019**, *23*, 8723–8740. [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.