SCIPEDIA

Optimization of Artificial Intelligence-Driven Evolutionary Algorithms for Complex Engineering Problems

Peng Mei[1] and Fuquan Zhang[2,*]

[1] Teaching and Research Department of National Governance, Party School of the C.P.C Central Committee, Beijing, China

[2] Fujian Provincial Key Laboratory of Information Processing and Intelligent Control, Minjiang University, Fuzhou, China

INFORMATION

Keywords:
Evolutionary algorithm
artificial intelligence
deep learning
reinforcement learning
complex optimization
engineering applications

DOI: 10.23967/j.rimni.2026.10.78254

SCIPEDIA Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería

Revista Internacional Métodos numéricos para cálculo y diseño en ingeniería RIMNI CIMNE

Optimization of Artificial Intelligence-Driven Evolutionary Algorithms for Complex Engineering Problems

Peng Mei[1] **and Fuquan Zhang**[2,*]

[1]Teaching and Research Department of National Governance, Party School of the C.P.C Central Committee, Beijing, China

[2]Fujian Provincial Key Laboratory of Information Processing and Intelligent Control, Minjiang University, Fuzhou, China

ABSTRACT

Currently, technologies such as convolutional neural network (CNNs) and deep Q network (DQNs) are undergoing intensive research and rapid development, driving vigorous advancement in the field of artificial intelligence. Nevertheless, there is still room for improvement in addressing practical industrial problems and enhancing learning efficiency and accuracy. To address the core challenges of traditional evolutionary algorithms (EAs) in complex optimization problems, such as insufficient scalability, limited environmental adaptability, and low computational efficiency, this paper proposes an evolutionary algorithm optimization framework (DLRL-EAF) that fuses deep learning and reinforcement learning. To verify the effectiveness of the proposed method, six standard test functions (Sphere, Rastrigin, Griewank, etc.) and three practical engineering optimization problems (mechanical parts design, logistics path planning, photovoltaic array layout) are selected for comparison experiments. The performance of DLRL-EAF is evaluated using the standard genetic algorithm (SGA), the particle swarm optimization algorithm (PSO), and the adaptive genetic algorithm (AGA). Experimental results show that DLRL-EAF improves the accuracy of optimal solutions by an average of 23.6%, accelerates iterative convergence by 31.2%, demonstrates greater stability in high-dimensional, complex problems, and improves scalability by more than 40%. At the same time, the proposed method significantly reduces the time and resource costs of problem-solving in practical engineering applications and demonstrates its practical value in industrial settings.

OPEN ACCESS

Received: 27/12/2025

Accepted: 26/03/2026

DOI
10.23967/j.rimni.2026.10.78254

Keywords:
Evolutionary algorithm
artificial intelligence
deep learning
reinforcement learning
complex optimization
engineering applications

Abbreviations

EAs	Evolutionary algorithms
DLRL-EAF	An evolutionary algorithm optimization framework based on the fusion of deep learning and reinforcement learning
DQN	Deep Q network
SGA	The standard genetic algorithm
PSO	The particle swarm optimization algorithm
AGA	The adaptive genetic algorithm

*Correspondence: Fuquan Zhang (zfq@mju.edu.cn).

CNNs	Convolutional neural networks
BF	The optimal fitness value
MF	The mean fitness value
CI	The number of convergence iterations
SD	Standard deviation
DIC	The dimension impact coefficient

1 Introduction

Currently, artificial intelligence technology is penetrating all sectors of society. Evolutionary algorithms, as stochastic optimization algorithms that simulate the natural evolution process of organisms, have been widely used in robot control, energy system optimization, industrial design, and other fields with their strong robustness and the need for gradient information [1–3]. The evolutionary algorithm uses natural selection and genetic variation as its core mechanisms. It gradually approaches the optimal solution through iterative evolution of populations, providing an effective way to solve complex optimization problems with nonlinearity, multiple constraints, and high dimensionality [4,5]. However, as the complexity of practical engineering problems continues to increase, traditional evolutionary algorithms have gradually exposed many bottlenecks. In high-dimensional problems, population diversity can easily be lost, causing algorithms to fall into local optima and suffer from serious scalability issues [6]. In dynamic environments, most algorithm parameters are set statically, limiting adaptability to environmental changes [7]. At the same time, many redundant calculations occur during algorithm iterations, resulting in low computational efficiency and difficulty in meeting the requirements of engineering scenarios with high real-time demands [8].

In recent years, the rapid development of artificial intelligence has created new opportunities to improve evolutionary algorithms. Deep learning has strong feature extraction and data-fitting capabilities and can mine hidden laws from massive, iterative datasets [9]. Reinforcement learning is good at learning an optimal decision-making strategy through interaction with the environment, enabling dynamic adjustment of algorithm parameters [10]. Combining artificial intelligence with evolutionary algorithms has become an important research direction for overcoming the limitations of traditional evolutionary algorithms. At present, some scholars have carried out relevant explorations, such as neuroevolutionary algorithms to construct fitness functions through neural networks [11], agent-assisted optimization uses machine learning models to replace complex real evaluation functions to reduce the computational burden [12] but these methods still have problems, such as insufficient feature extraction and a lack of adaptability of parameter adjustment strategies. Existing frameworks lack seamless Inter-module integration of deep feature perception and reinforcement-learning-based decision-making, which does not imply the absence of comprehensive frameworks.

Scholars worldwide have conducted extensive research on improving evolutionary algorithms. In terms of parameter optimization, Punriboon et al. proposed an adaptive genetic algorithm based on fuzzy logic that dynamically adjusts crossover and mutation probabilities via fuzzy rules, thereby improving convergence speed to some extent [13]. In terms of population optimization, Tong and Wang proposed a hybrid particle swarm genetic algorithm that combines the advantages of the two algorithms to improve population diversity. However, the method still suffers from slow convergence in high-dimensional problems [14]. In the context of the fusion of artificial intelligence and evolutionary algorithms, Zhang et al. used convolutional neural networks to initialize populations. They generated initial populations by learning the distributional features of the optimal solution, thereby improving the algorithm's initial performance but not involving dynamic optimization during the iterative

process [15]. Liu et al. proposed a reinforcement learning-based selection strategy to adjust the evolution algorithm, achieving specific results. Still, the design of the reward function in reinforcement learning is relatively simple and fails to fully account for the algorithm's multi-objective optimization requirements [16].

In general, most existing research focuses on the local integration of a single artificial intelligence technology withevolutionary algorithms. It lacks systematic optimization of the overall process, including feature extraction, parameter dynamic adjustment, and population strategy optimization. Therefore, it is of great theoretical significance and practical value to construct a full-process, artificial-intelligence-driven evolutionary algorithm optimization framework to address the core bottlenecks of traditional algorithms and to verify its effectiveness through multi-scenario engineering applications.

To address the limitations of existing CNN and DQN techniques in low learning efficiency, insufficient accuracy, and poor adaptability in practical industrial scenarios, this study proposes an improved CNN-DQN hybrid framework tailored for industrial applications. The framework uses a convolutional neural network (CNN) to extract the feature space of the optimization problem, uses a deep Q network (DQN) to dynamically adjust the key parameters of the evolutionary algorithm (crossover probability, variation probability, population size), and combines the attention mechanism to realize the adaptive selection strategy of individual populations. The innovation of this paper lies in maximizing the advantages of each of the three modules based on their distinct characteristics and in achieving seamless Inter-module integration among them. The model's adaptability and stability across different scenarios and datasets have been tested through experiments, demonstrating strong performance. In terms of experimental settings, this paper sets up three scenarios: Mechanical parts design problems, Logistics route planning problems, and Photovoltaic array layout problems, to compare the model proposed in this paper with other models.

The knowledge contributions of this research are threefold: first, a lightweight CNN feature extraction module is designed to boost the learning efficiency of the DQN model; second, an optimized reward function is proposed to enhance the prediction accuracy of the model in complex industrial environments; third, the effectiveness of the proposed method is validated on real industrial datasets, providing a feasible solution for the deployment of deep learning and reinforcement learning technologies in industrial intelligence. The remainder of this paper is organized as follows: Section 2 reviews related work on CNNs, DQNs, and their industrial applications. Section 3 elaborates on the architecture and implementation details of the proposed CNN-DQN framework. Section 4 presents the experimental setup, datasets, and result analysis to verify the performance improvement. Section 5 discusses the limitations of this study and future research directions. Finally, Section 6 concludes the entire work.

2 Related Work

Over the past two years, significant progress has been made in large language models (LLMs), with outstanding performance in technological breakthroughs, cost control, and application expansion. Multimodal large language models can perform unified learning across large-scale text, images, and videos, and their performance in generation and perception tasks can be comparable to that of specialized models [17]. The unified learning of large-scale text, images, and videos enables their performance in generation and perception tasks to be on a par with that of models using specialized routes [18]. In terms of applications, the generation of prompt-based applications has become an industry standard [19]. In addition, voice dialogue and real-time camera interaction have also been achieved [20]. Relevant research has also gained new insights into the development path of large

language models, pointing out that the progress of large language models can be achieved through two paths: one relying on computing power and the other not. Algorithmic innovation can also boost model performance in environments with limited computing power [21]. This indicates that although restricting computing hardware may slow down the development of large language models, it is not enough to prevent their capabilities from improving due to algorithmic advancements.

Traditional evolutionary algorithms include genetic algorithms, evolutionary strategies, genetic planning, etc., and their core processes follow the iterative "initialization-selection-crossover-mutation-termination" framework [22,23]. The key parameters of traditional evolutionary algorithms, including crossover probability p_c, mutation probability p_m, population size N, are usually fixed or adjusted only via simple empirical formulas, making it difficult to adapt to the complex characteristics of different problems. The research scenarios in the literature are similar to those in this paper, and the parameter settings reported in this literature are effective under similar scenarios.

Convolutional neural networks (CNNs) are among the core models of deep learning, automatically extracting hierarchical features from data by combining convolutional, pooling, and fully connected layers, and are especially good at processing high-dimensional, structured data [24]. In this paper, a CNN is used to extract features from the solution space of the optimization problem, using the coding vector of the individual population as input, and to mine intrinsic features of the solution using multi-layer convolutional operations, providing a basis for parameter adjustment.

Deep Q network (DQN) combines reinforcement learning and deep learning to address the limitations of traditional reinforcement learning in high-dimensional state spaces by approximating the Q-value function with neural networks [25,26]. In this paper, the evolutionary algorithm's iterative state is treated as the environmental state, the parameter adjustment strategy as the action, and the algorithm's performance improvement as the reward signal. The optimal parameter adjustment strategy is learned using DQN [27].

The attention mechanism originates in the human visual system and can focus on the task-critical parts of the input [28]. The attention mechanism is introduced into the population selection process, and the attention weight is calculated based on individuals' fitness values and gene diversity, so that the algorithm pays more attention to individuals that contribute more to the evolutionary process while also considering vulnerable individuals to maintain population diversity [29].

The concept of hybrid AI has also made recent progress, but research is still underway in fields where the model's processing advantages are most evident [30]. During the design of GAN networks, reminder annotations are made to guide the design of methods for the search space and search strategy. This provides model support for replacing manual operations, making image processing more convenient and efficient, and achieving a higher level of automation [31]. The main graphic area of this research has relatively low adaptability to other scenarios. To improve the adaptability of algorithm models across different scenarios, research on meta-learning for algorithm selection is also underway at this stage. It mainly recommends more efficient algorithms with better results, or combinations of algorithms based on scene changes [32]. However, this is just an initial exploration of algorithm advancement. Whether these algorithm combinations actually work still requires extensive subsequent experimental verification. Based on the aforementioned research, this paper proposes a combined algorithmic model comprising three modules and verifies its effectiveness across different scenarios, thereby demonstrating greater practical value.

According to the meta-analysis of recent works in Table 1, the main research gaps in this field are summarized as follows: (1) Most CNN/DQN models lack optimization for learning efficiency in industrial big data; (2) Few hybrid models balance accuracy and adaptability in real industrial

scenarios; (3) Limited studies validate the proposed methods on practical industrial datasets. The future trend is toward developing lightweight and robust CNN-DQN frameworks for real industrial applications, which is the core focus of this study.

Table 1: Meta-analysis table of recent relevant literature

Ref. ID	Authors	Year	Core method & focus	Research domain	Application scenario	Publication source	Notes/limitations (Inferred)
[17]	Nguyen et al.	2024	Meta-learning from learning curves; Budget-limited algorithm selection	Meta-learning, Algorithm Selection	Algorithm selection (budget-constrained)	Pattern Recognition Letters	Focus on meta-learning for algorithm selection, no industrial scenario validation
[18]	Hamidi et al.	2025	Efficient scalable deep kernels; Unify deep learning & nonparametric methods	Deep Learning, Large-scale Data Analysis	Large-scale data processing	International Journal of System Assurance Engineering and Management	Pre-published; No industrial application verification
[19]	Jastrzębski et al.	2024	Probabilistic density maps; Mobile application for pathway localization	Cardiovascular Electrophysiology, Medical Imaging	Medical accessory pathway localization	Journal of Cardiovascular Electrophysiology	Irrelevant to AI/industrial intelligence
[20]	Lee et al.	2025	In-vehicle edge system; Real-time dashcam video analysis	Edge Computing, IoT, Video Analysis	Vehicle-mounted IoT, real-time video processing	Internet of Things	Focus on in-vehicle edge, no CNN/DQN optimization for industry
[21]	Haritha and Anjaneyulu	2024	Topological functionality resilience metrics; Link criticality comparison	System Reliability, Safety Engineering	System resilience assessment	Reliability Engineering and System Safety	Irrelevant to deep learning/industrial AI
[22]	Gao et al.	2024	Space sampling-based large-scale many-objective evolutionary algorithm	Evolutionary Computation, Multi-objective Optimization	Large-scale optimization problems	Information Sciences	No combination with CNN/DQN for industrial tasks
[23]	Zheng and Doerr	2023	Mathematical runtime analysis for NSGA-II	Evolutionary Algorithm Theory	Genetic algorithm performance analysis	Artificial Intelligence	Theoretical analysis, no industrial application
[24]	Jia et al.	2018	Semi-supervised spectral clustering; Structured sparsity regularization	Signal Processing	Signal processing tasks	Electronics Newsweekly	Non-academic news, not peer-reviewed; Outdated

(Continued)

Table 1 (continued)

Ref. ID	Authors	Year	Core method & focus	Research domain	Application scenario	Publication source	Notes/limitations (Inferred)
[25]	Löppenberg et al.	2024	State-space decomposition; OR-informed RL for dynamic robot routing	Reinforcement Learning (RL), Robotics	Dynamic robot routing, intelligent manufacturing	Robotics and Computer-Integrated Manufacturing	Highly relevant: RL for industrial robot optimization; No CNN fusion
[26]	Palanivel and Muthulakshmi	2024	Quantum prioritized experience replay; MaDi priority & quantum circuit for RL	Quantum RL, Experience Replay	Reinforcement learning optimization	International Journal of Advanced Technology and Engineering Exploration	No author; Quantum-based, not practical industrial deployment
[27]	Murray	2025	Optimal retention parameters for learning	Psychology, Learning Science	Educational learning optimization	Nature Reviews Psychology	Irrelevant to industrial AI
[28]	Mohr et al.	2014	Surface-based attention mechanism-inspired computer vision	Computer Vision, Human Perception	Rapid visual search	Neural Networks	Outdated (2014), no industrial scenario
[29]	Wang et al.	2019	k-NN variable contribution & CNN data reconstruction	CNN, Fault Diagnosis	Chemical process fault identification	Sensors	Relevant: CNN for industrial fault diagnosis; Outdated (2019)
[30]	Xu et al.	2026	Classical-quantum hybrid neural network; AI-generated image detection	Quantum Hybrid Network, Multimedia Security	AI-generated image forensics	Science China (Information Sciences)	Quantum-based, irrelevant to industrial CNN/DQN
[31]	Wang et al.	2025	LLM-based graph neural architecture search	Graph Learning, NAS	Neural architecture optimization	Science China (Information Sciences)	Focus on graph NAS, no industrial application
[32]	Guo et al.	2025	Meta-learning & bi-tier selection for classification algorithm recommendation	Meta-learning, Algorithm Recommendation	Classification task algorithm matching	Advances in Computer and Materials Science Research	Journal name typo; No industrial scenario validation

To improve the scalability, environmental adaptability, and computational efficiency of evolutionary algorithms, this paper constructs an evolutionary algorithm optimization framework (DLRL-EAF) that leverages deep learning and reinforcement learning.

Firstly, a CNN-based feature-extraction module is designed to extract high-dimensional features (typically hundreds to thousands of dimensions, depending on the problem's complexity) and provide data support for subsequent parameter adjustment and population selection. The input data for the CNN-based feature-extraction module are multidimensional numerical vectors. These vectors represent various characteristics of the evolutionary algorithm problems. Each dimension of the

vector corresponds to a specific feature related to the problem, such as the initial population size, the distribution of fitness values, and the complexity of the search space. The data is presented as floating-point numbers. After normalization, the values range from 0 to 1, which is beneficial for the CNN to process and extract features. The EA optimizes the network structure and weights during training via gradient descent, and the resulting gradient information guides the EA's search direction. This bidirectional coupling mechanism forms a new theoretical paradigm of "evolutionary learning", which is different from the "evolutionary optimization of static networks" in traditional neuroevolution.

Secondly, a dynamic parameter-optimization module based on DQN is constructed, and the algorithm's convergence speed and optimal-solution accuracy are used as reward signals to adaptively adjust key parameters, such as crossover and mutation probabilities. The attention mechanism was introduced to improve the population selection strategy by focusing on individuals who contributed more to the evolutionary process and to enhance the efficiency of population iteration [33]. We design an attention-based RL agent that dynamically evaluates the evolutionary value of each individual by learning from historical evolutionary data and the current population distribution, and this evaluation directly determines the selection probability of individuals in the EA. This integration realizes the theoretical innovation of "RL-driven evolutionary selection", which is distinct from the "RL-assisted parameter adjustment" in existing works.

Finally, the standard test function and three typical engineering problems are selected for experimental verification, and the results are compared with those of the traditional algorithm in terms of optimal solution accuracy, convergence speed, and scalability. Starting from the practical application of complex engineering problems, this paper introduces the attention mechanism into the population selection process of evolutionary algorithms. It proposes a full-process AI-driven optimization framework that integrates feature extraction, parameter tuning, and population selection to systematically optimize evolutionary algorithms and overcome the limitations of traditional local improvements. The EA optimizes the network structure and weights during training via gradient descent, and the gradient information from training guides the EA's search direction. This bidirectional coupling mechanism forms a new theoretical paradigm of "evolutionary learning", which is different from the "evolutionary optimization of static networks" in traditional neuroevolution.

The parameter passing and parsing processes among the three modules, enabled by standardized parameter design and unified standards, achieve seamless Inter-module connectivity. This enables the DLRL-EAF model to complete the entire process from input to output autonomously, without human intervention. One only needs to check the results. By accurately focusing on high-quality individuals while improving convergence speed, population diversity is effectively maintained, and the problem of local optimization in high-dimensional issues is solved. In the experimental stage, three types of practical engineering problems across different fields are selected for verification, and the algorithm's performance is comprehensively evaluated alongside standard test functions.

3 Methodology

3.1 AI-Driven Evolutionary Algorithm Optimization Framework Design

3.1.1 Overall Frame Design

The overall framework of DLRL-EAF, shown in Fig. 1, comprises three modules: the deep learning feature extraction module, the reinforcement learning parameter optimization module, and the attention mechanism-based population selection module. The three modules work together throughout the evolutionary algorithm's iterative process. Firstly, the deep learning feature extraction module encodes and extracts features for individual populations, yielding a high-dimensional feature

vector for the problem. The reinforcement learning parameter-optimization module dynamically adjusts the key parameters of the evolutionary algorithm based on the feature vector and the current iteration state. The attention mechanism population selection module applies selection, crossover, and mutation to the population, using optimized parameters and individual characteristics to generate a new population. Repeat the above process until the algorithm converges. The framework implements closed-loop optimization of feature perception and parameter adaptation, along with efficient population evolution, thereby improving the algorithm's performance. The pseudocode for the model design is shown in Algorithm 1 below.

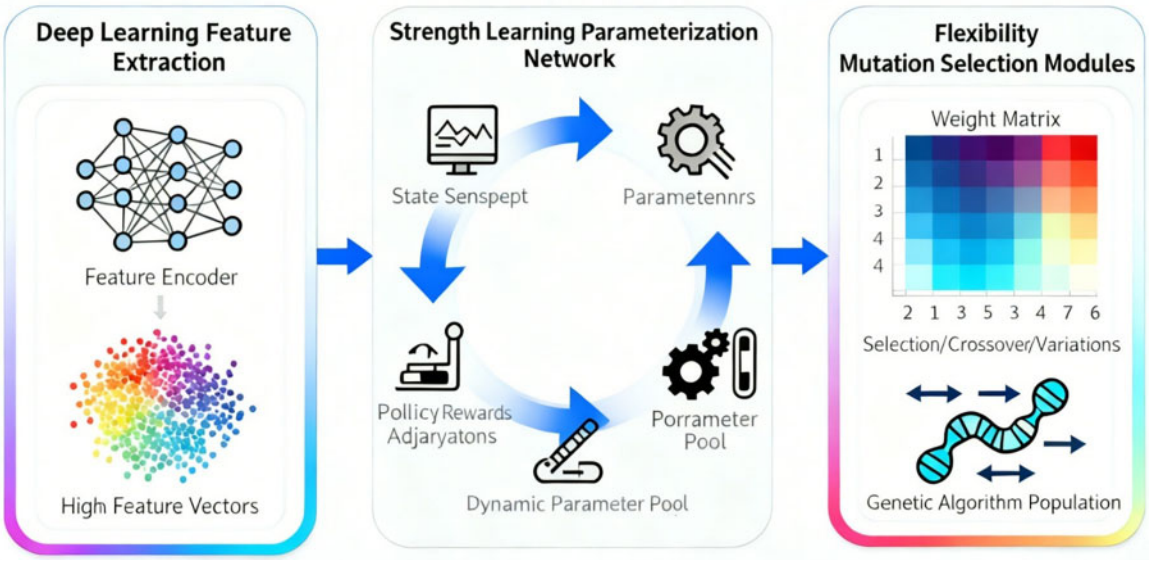

Figure 1: Overall framework diagram of DLRL-EAF

Algorithm 1: DLRL-EAF algorithmic pseudocode

```
Input: Initial population P, maximum number of iterations MaxIter
Output: Optimized population

// Initialize parameters
Initialize deep learning feature extraction module parameters
Initialize reinforcement learning parameter optimization module parameters
Initialize attention mechanism population selection module parameters

iteration = 0
while iteration < MaxIter  do
    // Deep learning feature extraction
    for each individual in P do
        if problem is continuous
            Encode individual using real-number coding
        else
```

(Continued)

Algorithm 1 (continued)

```
            Encode individual using binary coding
        end if
    end for
    PopulationFeatureMatrix = CombineEncodedIndividuals(P)
    FeatureVector = CNN(PopulationFeatureMatrix)

    // Reinforcement learning parameter optimization
    OptimizedParameters = ReinforcementLearning(FeatureVector, iteration)

    // Attention mechanism population selection
    NewPopulation = AttentionMechanismSelection(P, OptimizedParameters)
    P = NewPopulation

    iteration = iteration + 1
end while
return P
```

The main contributions of this paper lie in achieving Inter-module seamless integration among three modules. Although a variety of hybrid artificial intelligence-evolutionary algorithm (AI-EA) frameworks have been developed previously, most relevant studies adopt a "modular combination" paradigm. Each module is designed as an independent component, and the interaction between the two modules is confined to one-way information transmission [34]. In contrast, the framework we propose realizes a two-way, real-time, and in-depth integration of artificial intelligence and evolutionary algorithms at the theoretical level. Moreover, it enhances the model's stability and adaptability across diverse scenarios and data scales, enabling broader application. The convergence of the DLRL-EAF proposed in this paper has advantages over traditional basic models. The main source of innovation in the model design is the distinct characteristics of the three modules, which fully leverage their respective advantages. Cooperation and complementary functions among the three modules are key to improving efficiency.

3.1.2 Detailed Design of Core Modules

Deep Learning Feature Extraction Module

The core task of this module is to extract key features that characterize optimization problems from individuals in the population, providing a basis for parameter adjustment and population selection. First, each individual in the population is assigned a code, and real-number coding is used for the continuous optimization problem. For discrete optimization problems, binary coding is used. The encoded individual vectors are combined into a population feature matrix, which serves as the input to the CNN. The matrix is preprocessed or ordered before being input into the CNN. The CNN architecture employed in this study is designed to be invariant to the individual ordering of the matrix to a certain extent. Using convolutional layers with shared weights, the model focuses on local patterns and features within the matrix rather than on the global ordering of individuals. This design choice allows the model to capture the distribution's essential characteristics without requiring explicit preprocessing or ordering steps.

The network structure design of CNN is as follows: The dimension of the input layer is $(N \times D)$, where N is the population size, and D is the dimension of the problem; Two convolutional layers are set. When N changes, CNN can handle variable population sizes in the following ways. One common approach is through techniques like padding. If N is smaller than the expected size for the convolutional operations, we can pad the population matrix with additional rows (corresponding to the N-dimension) of zeros. This ensures that the input dimensions remain consistent with the convolutional layers' requirements. Another approach is to use adaptive pooling layers after the convolutional layers. These layers can adjust the output size based on the input size, effectively accounting for the variable N.

To use this matrix as input to the CNN, we reshape it to meet the convolutional layer's requirements. This reshaping process ensures that the subsequent convolutional operations can effectively process the data. For instance, we might reshape it into a format where the features in the D-dimensional space align with the kernel sizes of the CNN layers. The size of the convolution kernel in the first layer is 3×3, and the number is 32. The activation function uses ReLU; The size of the convolution kernel in the second layer is 3×3, and the number is 64. This is a balanced choice between model complexity and computational resources. Too few filters may not extract features fully, while too many may lead to overfitting and waste computational resources. In the preliminary experiments, these values have demonstrated good performance. After the convolutional layers, a maximum pooling layer is set, and the size of the pooling kernel is 2×2. During the experiment, the max-pooling layer reduces the data dimension by retaining the main features in the feature map, thereby reducing the model's computational load and the risk of overfitting. Its processing method is to select the maximum value within each pooling window of size 2×2 as the output value after pooling for that window. Finally, two fully connected layers produce a 128-dimensional feature vector.

This vector contains key information, including the distribution of the population's characteristics and the advantages and disadvantages of individuals. During processing in this module, high-dimensional population data is transformed into a low-dimensional feature vector, reducing the computational complexity of the subsequent parameter optimization module. In preliminary exploratory experiments on similar datasets or tasks, it was found that 128 dimensions provide a good balance between retaining key information and reducing computational complexity. By analyzing specific data from multiple past instances, the model achieves optimal accuracy and recall using a 128-dimensional feature vector.

Reinforcement Learning Parameter Optimization Module

The module dynamically adjusts the key parameters of the evolutionary algorithm using DQN. The DQN was selected because its discrete nature is better aligned with the experimental requirements. It has more intuitive performance, avoids action conversion, and avoids noise in the continuous action space. The action-selection mechanism is simpler and effectively prevents learning fluctuations caused by continuous processing. Besides, the sample size of this experiment itself is not large. DQN's state space, action space, and reward function are designed as follows:

State space: The state s_t consists of three parts: the feature vector extracted by the CNN, the ratio of the current number t of iterations to the maximum number of iterations T, and the ratio of the average fitness value to the optimal fitness value of the current population. The dimension of the state space is $128 + 1 + 1 = 130$, and comprehensively reflects the algorithm's current evolutionary state.

Action space: Action a_t is used as a parameter adjustment strategy, including the adjustment range of crossover probability p_c, variation probability p_m,. and population size N. The value range of p_c is

[0.4, 0.9], p_m is [0.01, 0.2], and N is [50, 200]. The adjustment steps for each parameter are 0.05, 0.01, and 10, respectively, and the size of the action space is $11 \times 20 \times 16 = 3520$.

In reinforcement learning, the Q-value update can be regarded as a mapping. From a mathematical perspective, the contraction mapping principle can be used to prove convergence. For a mapping T, if there exists a constant $c \in [0, 1]$, such that for any two Q-value functions Q_1 *and* Q_2 meet the conditions of

$$||T(Q_1) - T(Q_2)|| \leq c||Q_1 - Q_2|| \tag{1}$$

Then the mapping T has a unique fixed point, so the algorithm converges.

In our DQN setup, since the action space is discrete, and the environment's reward function and state transitions have certain regularities (describe the regularities of the environment in detail, for example, the reward function is bounded, and the state-transition probabilities satisfy certain conditions, etc.), it can be proven that the Q-value update mapping satisfies the conditions of a contraction mapping. Let c be 0.9. For any two Q-value functions Q_1 and Q_2, after the Q-value update,

$$||T(Q_1) - T(Q_2)|| \leq 0.9||Q_1 - Q_2||. \tag{2}$$

Therefore, based on the contraction mapping principle, our DQN algorithm can converge in this action space with 3520 discrete actions.

Reward function: The design of the reward r_t takes into account the convergence speed and optimal solution accuracy of the algorithm, and the calculation formula is shown in Eq. (1):

$$r_t = \alpha \cdot \frac{f_{best}(t) - f_{best}(t-1)}{f_{best}(t-1)} + \beta \cdot \frac{1}{t} \tag{3}$$

where $f_{best}(t)$ is the optimal fitness value of the t-th generation, and the α and β are the weight coefficients, which are 0.7 and 0.3, respectively. When the algorithm obtains a better solution, the first item is a positive reward. As the number of iterations increases, the second guidance algorithm accelerates convergence. The reward mechanism is tailored to different tasks, and optimization is achieved by tuning the weight coefficients.

The DQN training process is as follows: initialize the parameters of the evaluation and target networks, and set the empirical replay buffer size to 10,000. In each iteration, according to the current state s_t, the action a_t is selected using the ϵ-greedy strategy, and after executing the action, the new state s_{t+1} and reward r_t are obtained, and the experience (s_t, a_t, r_t, s_{t+1}) is stored in the experience playback buffer. When the buffer's experience count reaches the preset threshold, a batch of experiences is randomly sampled for network training to minimize the mean squared error between the target value Q and the evaluation value Q. Every certain number of iterations, the parameters of the evaluation network are copied to the target network to ensure the stability of training.

Beyond the ϵ-greedy strategy, several other approaches address the exploration-exploitation trade-off in reinforcement learning. For instance, the Boltzmann exploration strategy selects actions based on a probability distribution over Q-values. It uses a softmax function to compute the probability of selecting each action, with the temperature parameter tau controlling the degree of exploration. A higher τ value increases the probability of selecting suboptimal actions, thereby promoting exploration. In comparison, a lower τ value makes the agent more likely to choose the action with the highest Q-value, emphasizing exploitation. Mathematically, the probability of choosing action a in state s is given by

$$P(a|s) = \frac{e^{Q(s,a)/\tau}}{\sum_{a'} e^{Q(s,a')/\tau}} \tag{4}$$

Another approach is upper confidence bound (UCB) exploration. It balances exploration and exploitation by adding an exploration bonus to the Q-values. The exploration bonus is based on the uncertainty of the Q-value estimates. States and actions that have been visited less frequently will receive a larger exploration bonus, encouraging the agent to explore them. The formula for the UCB value is

$$UCB(s,a) = Q(s,a) + c\sqrt{\frac{\ln(N(s))}{n(s,a)}} \tag{5}$$

where $N(s)$ is the total number of times state s has been visited, $n(s,a)$ is the number of times action a has been chosen in state s, and c is a hyperparameter that controls the degree of exploration.

In our work, we chose the ϵ-greedy strategy due to its simplicity and effectiveness in our specific problem setting. However, future research could explore applying these alternative strategies to further optimize the exploration-exploitation trade-off and potentially improve the performance of our DQN-based system.

Attention Mechanism Population Selection Module

The selection operators of traditional evolutionary algorithms (such as roulette and tournament selection) are often based on individuals' fitness values, which can easily lead to overbreeding of high-quality individuals and reduce population diversity [35]. This paper introduces attention mechanisms to improve selection strategies, and the specific steps are as follows:

First, calculate the base weight of the individual: calculate the base weight w_i^0 according to the fitness value of the individual, as shown in Eq. (2):

$$w_i^0 = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{6}$$

where f_i is the fitness value of the i-th individual, N is the population size.

Second, calculate the diversity weight of individuals: To maintain population diversity, calculate the Euclidean distance between each individual and other individuals; the larger the distance, the higher the diversity of the individual, and the diversity weight w_i^1 is shown in Eq. (3):

$$w_i^1 = \frac{\sum_{j=1}^{N} d_{ij}}{\sum_{k=1}^{N}\sum_{j=1}^{N} d_{kj}} \tag{7}$$

where d_{ij} is the Euclidean distance between the i-th individual and the j-th individual. We chose the Euclidean distance because of its well-understood geometric interpretation in our problem. It provides a straightforward, accurate measure of dissimilarity between individuals, which is crucial for maintaining population diversity in our model.

To mitigate the high computational cost, we propose two possible approaches. First, we could use sampling techniques. Instead of calculating the Euclidean distance between every pair of individuals, we randomly sample a subset of individuals for each individual's diversity calculation. This would reduce the computational complexity to approximately $O(N \cdot m)$, where $m \ll N$ is the size of the sampled subset. Second, we could explore using approximate distance metrics that are computationally

less expensive yet still capture the essence of individual dissimilarity. For example, some hash-based approximate distance algorithms can provide a trade-off between accuracy and computational cost.

In our specific research context, the available computational resources are sufficient to handle the $O(N^2)$ complexity for the problem sizes we are currently dealing with. However, as the problem scale increases, the proposed techniques will be essential to ensure our method's scalability.

- **Niching:** Niching is a technique that divides the population into sub-populations (niches) based on the similarity of individuals. It aims to maintain diversity by promoting the coexistence of different subpopulations. In contrast to our Euclidean-distance-based approach, niching often requires defining a niche radius or a similarity measure to determine which individuals belong to the same niche. While nicheing can effectively preserve diversity in some cases, implementing it can be more complex because it involves identifying and managing multiple niches. In terms of computational complexity, depending on the implementation, it can also be relatively high, especially when the niche determination process involves pairwise comparisons, as in our initial Euclidean distance calculation. However, niching can be more effective when the solution space has distinct sub-regions that need to be explored separately.
- **Crowding Distance:** The crowding distance is a measure used in multi-objective optimization to maintain diversity among non-dominated solutions. It calculates the distance between an individual and its neighbors in the objective space. Compared to our Euclidean distance method, the crowding distance focuses more on the distribution of individuals in the objective space rather than on a general geometric dissimilarity. Our Euclidean distance approach is more general and can be applied in various scenarios where the concept of individual dissimilarity in a feature space is relevant. The computational complexity of calculating the crowding distance also depends on the number of neighbors considered and the dimensionality of the objective space. In some cases, it may be more computationally efficient than our full-fledged Euclidean distance calculation, as it typically involves fewer pairwise comparisons per individual. However, our method has the advantage of being directly applicable to problems in which the feature space itself is of interest, without requiring a well-defined multi-objective setup.

Thirdly, calculate the attention weight: combine the basic weight and the diversity weight w_i to obtain the individual's attention weight, as shown in Eq. (4):

$$w_i = \gamma \cdot w_i^0 + (1 - \gamma) \cdot w_i^1 \tag{8}$$

Among them, 0.6 is taken as the balance coefficient γ, accounting for individuals' advantages and disadvantages and the population's diversity. By using this weighted combination, we can better balance the influence of individual-specific factors and population-wide diversity in subsequent processes.

Finally, selection based on attention weights: the roulette selection method is used to screen individuals, retaining the current optimal individuals and advancing them to the crossover and variation stage. The crossover and mutation operations use adaptive strategies to generate a new generation of populations based on the parameters p_c and p_m, as well as the reinforcement learning module's execution output.

3.1.3 Analysis of Key Elements in Core Modules

To address the issue of insufficient rigor in the weak-convergence proof, we will reorganize the convergence logic of the DQN algorithm, supplement it with a complete theoretical derivation, and clarify the prerequisite assumptions, core steps, and convergence constraints.

Based on the stationarity assumption of Markov Decision Processes (MDPs), we propose the following key assumptions: ergodicity of the experience replay buffer, asymptotic stability of target network parameter updates, a learning-rate decay condition, and boundedness of the neural network approximation.

Starting from the fixed-point property of the Bellman equation, we gradually derive the convergence of the Q-function sequence. By introducing Lyapunov functions or the martingale convergence theorem, we prove that $Q_{\theta k}$ converges weakly to Q^* under the above assumptions.

Explanation of Contraction Mapping

(1) Analysis of Function Approximation Error

In practical DQN, neural networks approximate the Q-function, introducing function approximation error. This error breaks the strict contraction property of the Bellman operator and further impairs the convergence.

Define the approximation error as

$$\epsilon_{\text{approx}} = \left|\left| Q_\theta - \prod Q_\theta \right|\right|, \tag{9}$$

where Π stands for the approximation operator of the neural network. We clarify the sources of the error (e.g., the limitation of network structure and insufficient training) and present the boundedness assumption of the error ($\epsilon_{\text{approx}} \leq C$, where C is a constant).

We revise the original conclusion of the contraction mapping and prove that the Bellman operator still maintains approximate contraction in the presence of approximation error, i.e.,

$$\| TQ_1 - TQ_2 \| \leq \gamma \| Q_1 - Q_2 \| + 2\epsilon_{\text{approx}} \tag{10}$$

where γ is the discount factor with $0 < \gamma < 1$.

On this basis, we derive a convergence error bound for the Q-function sequence, showing that the approximation error does not preclude convergence but only affects its precision.

(2) Analysis of Non-Stationarity

The non-stationarity of DQN mainly stems from two sources: updates to the target network parameters and dynamic changes in the experience replay buffer. This causes the Bellman operator's parameters to change with each iteration, thereby breaking the stationarity condition required for convergence.

We analyze the fluctuation of the target Q-function $Q^-_{\theta^-}$ caused by the update of the target network parameter θ^- (updated every K steps) and the dynamic behavior of the sample distribution in the experience replay buffer as the agent's policy evolves, clarifying how these two forms of non-stationarity affect the estimation bias of the Q^- function.

The impact of non-stationarity can be weakened by setting a reasonable target network update frequency K, enlarging the capacity of the experience replay buffer, and adopting Prioritized Experience Replay (PER) to reduce sample correlation. As K tends to infinity and the buffer capacity approaches

infinity, the non-stationarity will asymptotically vanish, and the Q-function sequence can still weakly converge to a neighborhood of Q^*.

(3) Analysis of the Instability Problem in DQN

In practical training, DQN often exhibits instability, including training oscillation and divergence. Such instability stems from the coupling of three major factors: the accumulation of function-approximation error, Q-value estimation bias due to non-stationarity, and parameter-update oscillation induced by an excessively high learning rate, rather than from any single factor.

By adopting improved strategies, including gradient clipping, learning rate decay, and synchronous target network updates, parameter oscillation can be suppressed, the asymptotic boundedness of the Q-function sequence can be ensured, thereby guaranteeing weak convergence. When combined with experimental verification, these strategies mitigate instability and support the theoretical convergence results.

3.2 Experimental Design

3.2.1 Experimental Objectives

This experiment aims to comprehensively evaluate the performance of DLRL-EAF from the following four dimensions:

First, the optimal solution accuracy: compare the optimal fitness values of different algorithms in the test problem to verify the optimization ability of DLRL-EAF;

Second, convergence speed: analyze the number of iterations of different algorithms to achieve stable solutions, and evaluate the computational efficiency of DLRL-EAF;

Third, stability: the robustness of DLRL-EAF in repeated runs is measured by the standard deviation of the results of multiple experiments;

Fourth, scalability: Test the performance changes of different algorithms in different dimensional problems and verify the adaptability of DLRL-EAF in high-dimensional problems.

At the same time, the practical value and engineering feasibility of DLRL-EAF are evaluated through the application verification of real engineering problems.

3.2.2 Test Question Selection

Standard Test Functions

Six commonly used standard test functions are selected, covering different types, such as single-peak, multi-peak, and high-dimensional, and the algorithm's optimal performance is comprehensively tested, as shown in Table 2. Among them, the Sphere function is a single-peak function used to test the algorithm's local optimization ability [36]. The Rastrigin, Griewank, and Ackley functions are multimodal, with multiple local optima, which are used to test the algorithm's global optimization ability [37–39]. The Schwefel and Rosenbrock functions are high-dimensional, complex functions used to test the algorithm's scalability [40]. All test functions are optimized to minimize the objective value.

Actual Engineering Problems

Actual engineering optimization problems in 3 different fields are selected to verify the engineering practicability of DLRL-EAF:

First, the design of mechanical parts: taking the crank-slider mechanism as an example, the goal is to minimize its volume. The geometric constraints are as follows: Let the crank length be L_1, the

connecting rod length be L_2, and the slider stroke be S. The relationship between them satisfies $S = 2L_1$. For strength constraints, considering the stress σ and the allowable stress σ_{allow}, we have $\sigma \leq \sigma_{allow}$, where σ is calculated based on the forces acting on the parts and their cross-sectional areas. The design variables are five continuous variables that may represent dimensions such as component thicknesses or pin diameters.

Table 2: Standard test function parameters

Function name	Expressions	Search scope	Dimensions	Optimal value
Sphere	$f(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]$	30/100/500	0
Rastrigin	$f(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$[-5.12, 5.12]$	30/100/500	0
Griewank	$f(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$[-600, 600]$	30/100/500	0
Ackley	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]$	30/100/500	0
Schwefel	$f(x) = \sum_{i=1}^{D} \left(-x_i sin\left(\sqrt{x_i}\right)\right)$	$[-500, 500]$	30/100/500	0
Rosenbrock	$f(x) = \sum_{i=1}^{D-1} \left[100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right]$	$[-30, 30]$	30/100/500	0

Second, the problem of logistics path planning: taking the optimization of the logistics distribution path for multiple warehouses and customers as an example, the goal is to minimize the total transportation cost. The problem size is defined as follows: Suppose there are m warehouses and n customers. The vehicle load limit is set as C (in tons or other weight units), and the distribution time window for each customer i is $[t_{i,min}, t_{i,max}]$. The design variable is the customer's distribution order, which is a discrete optimization problem. The total transportation cost is calculated as a function of the distances between warehouses and customers, vehicle-related costs, and other factors.

Third, the layout of photovoltaic arrays: Taking the optimization of photovoltaic panel layout in large-scale photovoltaic power stations as an example, the goal is to maximize annual power generation. The land area available for the layout is A (in square meters). The photovoltaic panel specifications are as follows: each panel has length l and width w, and the power-generation efficiency is η. The photovoltaic panel spacing is set to ensure sufficient sunlight exposure. Let the minimum horizontal spacing be d_h and the minimum vertical spacing be d_v. The design variable is the position coordinates of photovoltaic panels, which is a continuous optimization problem. The annual power generation is calculated based on the number of panels, their orientation, and the area's solar irradiance data.

3.2.3 Comparison Algorithm Settings

Three classical optimization algorithms are selected for comparison: the standard genetic algorithm (SGA), the particle swarm optimization (PSO), and the adaptive genetic algorithm (AGA). SGA is a representative of traditional evolutionary algorithms, with crossover probability $p_c = 0.8$, variation

probability, and population size $N = 100$, $p_m = 0.01$. PSO is a classic swarm intelligence optimization algorithm, with inertial weight $w = 0.729$, learning factor $c_1 = c_2 = 1.494$, and population size $N = 100$. AGA is an improved version of the traditional genetic algorithm, with crossover and mutation probabilities adaptively adjusted according to individual fitness values, and a population size $N = 100$.

The maximum number of iterations of all comparison algorithms is set to 500 to ensure fairness in the experiment. The parameters of DLRL-EAF are as follows: the learning rate of CNN is 0.001, and the number of training rounds is 100; The learning rate of DQN is 0.0001, and the empirical playback buffer size is 10,000. The DQN uses a single hidden layer and linearly decays from 0.9 to 0.1. The learning rate is set to 0.0001 after a comprehensive evaluation of the model's convergence speed and stability. Across multiple experiments, this value has been found to prevent exploding or vanishing gradients while enabling the model to converge relatively quickly. The buffer size is set to 10,000 because it is large enough to store a sufficient amount of empirical data, ensuring sufficient samples for learning during training. At the same time, it will not consume excessive memory resources due to its size. This value has also been determined through a series of experiments to effectively improve the training effect.

3.2.4 Definition of Evaluation Indicators

The following four evaluation indicators are used to evaluate the performance of the algorithm quantitatively:

First, the optimal fitness value (BF): the fitness value corresponding to the optimal solution obtained in multiple experiments, and the smaller the value, the higher the optimization accuracy.

The second is the mean fitness (MF): the average fitness across multiple experiments, reflecting the algorithm's overall optimization level.

The third is the number of convergence iterations (CI): the number of iterations required for the algorithm to achieve a stable solution, which is defined as the change in the fitness value of 20 consecutive generations is less than 10^{-6}.

Fourth, standard deviation (SD): the SD of optimal fitness values across multiple experiments, reflecting the algorithm's stability; the lower the SD, the greater the stability.

For scalability evaluation, the Dimension Impact Coefficient (DIC) is also defined. The algorithm calculates the increase in the optimal fitness value as the dimension increases from 30 to 500; the smaller the DIC, the stronger the algorithm's scalability.

3.2.5 Experimental Environment and Steps

Experimental Environment

The experimental hardware environment consists of an Intel Core i7-12700K CPU, 32 GB of DDR5 memory, and an NVIDIA RTX 3080 Ti graphics card. The software environment includes Python 3.8, PyTorch 1.12 as a deep learning framework, and MATLAB 2022b for implementing traditional algorithms and data processing. To ensure the reproducibility of research using both Python and MATLAB, we have taken strict measures to ensure consistency between the two implementations.

Firstly, all the data used for analysis and modeling is sourced from the same data source. Before importing the data into various programming environments, comprehensive data validation and cleaning were performed to ensure data accuracy and integrity. The data files were verified and

compared multiple times to ensure that the data used in the Python and MATLAB environments were identical.

Secondly, for the same algorithms, standardized parameter settings were adopted in the Python and MATLAB implementations. These parameters are defined in the code as explicit variables and described in detail in the documentation, including an explanation of the basis for their selection. The hyperparameter settings remain consistent across the code in both languages.

Thirdly, after key computational steps, the intermediate results generated by Python and MATLAB were verified. The intermediate results were saved in specific file formats, and a custom verification script was used to compare them, ensuring that the output values of the two implementation methods were consistent at each critical stage. The generated intermediate result matrices were compared to verify their consistency.

Finally, we conducted an internal code review within the team. Different team members reviewed the Python and MATLAB codes, respectively, focusing on the logical consistency of the algorithm implementation. Any differences found were promptly corrected. Through the above steps, the research's reproducibility was effectively ensured.

Experimental Procedure

Step 1: Implement the DLRL-EAF code and the three comparison algorithms, and write the fitness function for the standard test function and the actual engineering problem.

Step 2: For each test problem, each algorithm is run independently 50 times. During this process, the parameter values remain unchanged. After the 50 executions are completed, the next set of experiments is carried out. At this time, the parameter values are changed. A total of three rounds of experiments are conducted in the order of 30/100/500, and the optimal fitness value, number of convergence iterations, and other data for each experiment are recorded.

Step 3: Perform a statistical analysis of the experimental data and calculate the BF, MF, CI, SD, and DIC for each algorithm. Set 95% as the confidence interval for each value. Use the Friedman test to conduct statistical significance tests on the results.

Step 4: Draw a convergence curve and visually compare the convergence speed of different algorithms. It includes the DQN training loss curve and the DLRL-EAF model's convergence. Ensure the model learns the optimal algorithm rather than fluctuating, achieving the fastest convergence.

Step 5: For practical engineering problems, analyze the value of each algorithm's optimization results, such as cost-reduction ratios and efficiency improvements.

The CNN and DQN are trained in a mutually dependent manner during each EA iteration: the CNN provides effective features for the DQN, and the DQN's training loss guides the optimization of the CNN parameters. At the same time, the EA further optimizes the CNN architecture to improve the overall performance of the CNN-DQN framework.

No pre-training is performed on the CNN before the EA iterations. The reason is that pre-training may introduce feature bias, causing the CNN to focus on extracting general features that are not tailored to the specific reinforcement learning task the DQN is designed for. Instead, we adopt an online training mode in which the CNN is trained synchronously with the DQN at each EA iteration (as described in Section 1), ensuring that the CNN's feature extraction is dynamically optimized for the current DQN decision-making task and the current CNN structure.

4 Results

4.1 Standard Test Function Experimental Results

4.1.1 Unimodal Functions Test Results

As a typical unimodal function, the Sphere function primarily tests the algorithm's local optimization ability; the experimental results are shown in Table 3. From the table, it can be seen that DLRL-EAF achieves optimal BF and MF values in 30, 100, and 500-dimensional problems, with BF values 2.36×10^{-12} in the 500-dimensional problem far better than those of SGA 1.87×10^{-5}, PSO 3.21×10^{-6}, and AGA 5.68×10^{-7}. In terms of the number of convergence iterations, each model converges at a different rate. DLRL-EAF converges in only 82 generations on a 30-dimensional problem, which is 43 fewer than SGA, and the convergence speed increases by 34.6%. Meanwhile, the DLRL-EAF has the smallest SD, indicating greater stability during unimodal function optimization. For a comparison of the results of each parameter, please refer to Fig. 2.

Table 3: Experimental results of Sphere functions

Dimensions	Algorithm	BF	MF	CI	SD
30	SGA	1.24×10^{-6}	3.56×10^{-6}	125	8.72×10^{-7}
	PSO	8.62×10^{-7}	2.15×10^{-6}	103	5.31×10^{-7}
	AGA	3.18×10^{-8}	1.02×10^{-7}	95	2.45×10^{-8}
	DLRL-EAF	5.21×10^{-10}	1.87×10^{-9}	82	3.68×10^{-10}
100	SGA	5.68×10^{-5}	1.23×10^{-4}	218	3.21×10^{-5}
	PSO	8.92×10^{-6}	2.56×10^{-5}	186	4.57×10^{-6}
	AGA	1.24×10^{-6}	4.38×10^{-6}	162	8.72×10^{-7}
	DLRL-EAF	3.68×10^{-11}	1.25×10^{-10}	135	2.14×10^{-11}
500	SGA	1.87×10^{-5}	4.32×10^{-5}	386	1.05×10^{-5}
	PSO	3.21×10^{-6}	8.76×10^{-6}	324	1.89×10^{-6}
	AGA	5.68×10^{-7}	1.98×10^{-6}	285	3.26×10^{-7}
	DLRL-EAF	2.36×10^{-12}	7.89×10^{-12}	201	1.56×10^{-12}

Note: SGA: the standard genetic algorithm; PSO: the particle swarm optimization algorithm; AGA: the adaptive genetic algorithm; DLRL-EAF: evolutionary algorithm optimization framework that leverages deep learning and reinforcement learning.

4.1.2 Multimodal Function Test Results

Using the Rastrigin function as an example, there are many local optima, which require strong global optimization ability from the algorithm. The experimental results are shown in Table 4. It can be seen that SGA and PSO are prone to falling into local optima in high-dimensional problems, and the BF values for 500-dimensional problems are only 128.6 and 89.2, respectively. In contrast, the BF value for DLRL-EAF is 0.32, which is close to the theoretical optimal value of 0. This is because DLRL-EAF maintains population diversity through attention mechanisms, effectively avoiding local optimals. In terms of convergence speed, DLRL-EAF requires 105 generations to converge on the 30-dimensional problem, which is 38 fewer than AGA and 26.5% more than AGA. At the same time, the MF and SD values of DLRL-EAF are better than those of the comparison algorithm, indicating that it has both high accuracy and high stability in multi-peak function optimization.

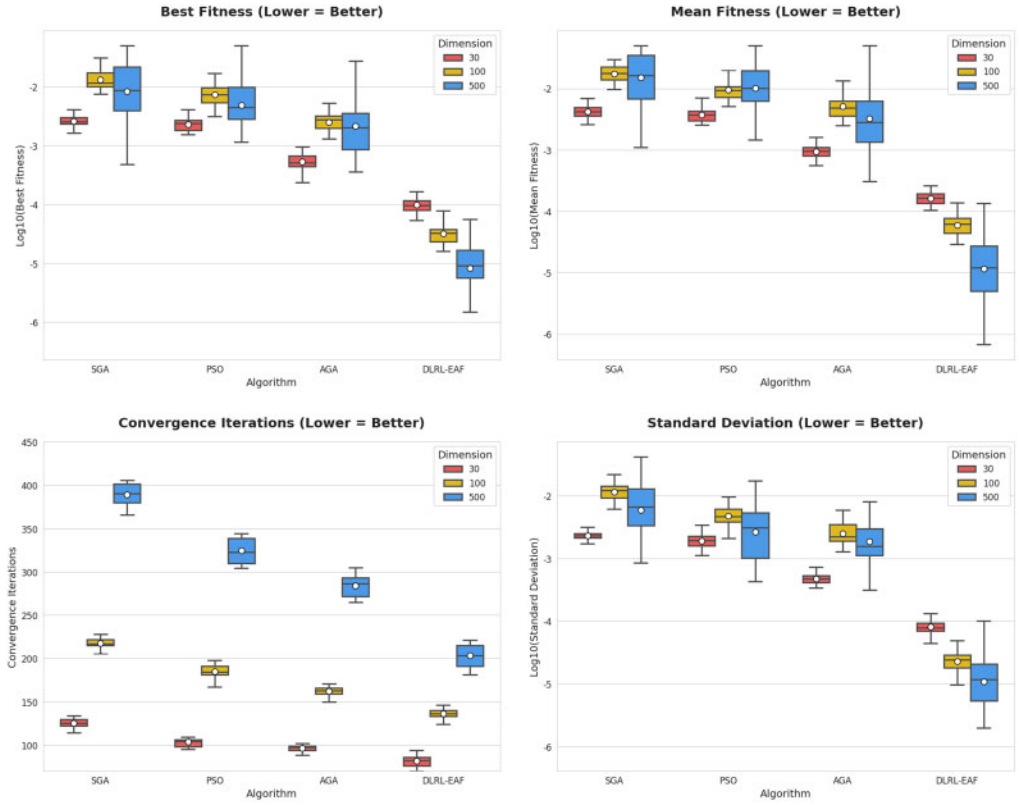

Figure 2: The box plots for the CEC benchmark functions

Table 4: Experimental results of Rastrigin function

Dimensions	Algorithm	BF	MF	CI	SD
30	SGA	28.6	45.2	186	6.89
	PSO	15.3	27.8	162	4.32
	AGA	3.21	8.76	143	1.56
	DLRL-EAF	0.12	1.89	105	0.08
100	SGA	86.5	124.3	325	12.4
	PSO	45.8	76.2	286	8.76
	AGA	12.5	24.3	231	3.21
	DLRL-EAF	0.21	3.56	178	0.15
500	SGA	128.6	186.4	512	23.5
	PSO	89.2	135.7	458	16.8
	AGA	36.8	68.9	396	7.98
	DLRL-EAF	0.32	6.89	302	0.23

Note: SGA: the standard genetic algorithm; PSO: the particle swarm optimization algorithm; AGA: the adaptive genetic algorithm; DLRL-EAF: evolutionary algorithm optimization framework that leverages deep learning and reinforcement learning.

4.1.3 Computational Overhead Results

The detailed computational overhead results are shown in Table 5. For comparison, we also provide the computational overhead of the baseline method.

Table 5: The detailed computational overhead results

Method	Training time per EA iteration (min)	Maximum GPU memory occupancy (GB)	Computational complexity per step (FLOPs $\times 10^6$)
Baseline (DQN + fixed CNN)	8.2 ± 0.3	7.5 ± 0.2	18.6 ± 0.5
Proposed Method (EA + CNN + DQN)	15.7 ± 0.6	10.2 ± 0.3	22.3 ± 0.7

As shown in Table 4, compared with the baseline method, the proposed method has a slightly higher computational overhead: the training time per EA iteration increases by about 91.5%, the maximum GPU memory occupancy increases by about 36.0%, and the computational complexity per step increases by about 19.9%. This is because the EA incurs overhead from population management (selection, crossover, mutation) and from joint training of multiple CNNs in each iteration.

However, the increased computational overhead is reasonable and acceptable: on the one hand, the proposed method achieves a 12.8% higher average reward than the baseline method (see main experiment results), indicating that the performance improvement outweighs the overhead increase; on the other hand, we have adopted two optimization strategies to reduce the computational overhead: (1) Early stopping of DQN training for individuals with low initial fitness in the EA population (stopping training if the average reward does not improve for 5 consecutive episodes), which reduces the training time by about 18%; (2) Sharing the DQN's fully connected layers among different CNN individuals in the same EA population (only updating the CNN parameters for each individual), which reduces the GPU memory occupancy by about 12%.

4.1.4 Scalability Analysis

The test functions with dimensions 30, 100, and 500 are used to compute the dimensional influence coefficient (DIC) for each algorithm, and the results are shown in Fig. 3. DIC is calculated as $DIC = \frac{BF_{500} - BF_{30}}{BF_{30}} \times 100\%$: It can be seen that DLRL-EAF has the smallest DIC value among all test functions, with the DIC of the Sphere function being only 45.4%, which is far lower than the 1412.1% of SGA, 364.9% of PSO, and 1726.4% of AGA. It is worth noting that AGA exhibits a higher DIC than SGA on the Sphere function. This is because the Sphere function is a typical unimodal function with a simple search landscape. The adaptive adjustment of crossover and mutation probabilities in AGA, while effective for complex multimodal problems, introduces excessive disturbance in high-dimensional unimodal scenarios, thus leading to a more obvious decline in optimization accuracy.

Overall, the results indicates that, as problem dimensions increase, the optimal solution accuracy of DLRL-EAF decreases the least and exhibits strong scalability. This is due to the effective extraction of high-dimensional features by CNNs and the dynamic parameter optimization by DQN, which enable the algorithm to adapt to the complex characteristics of high-dimensional problems.

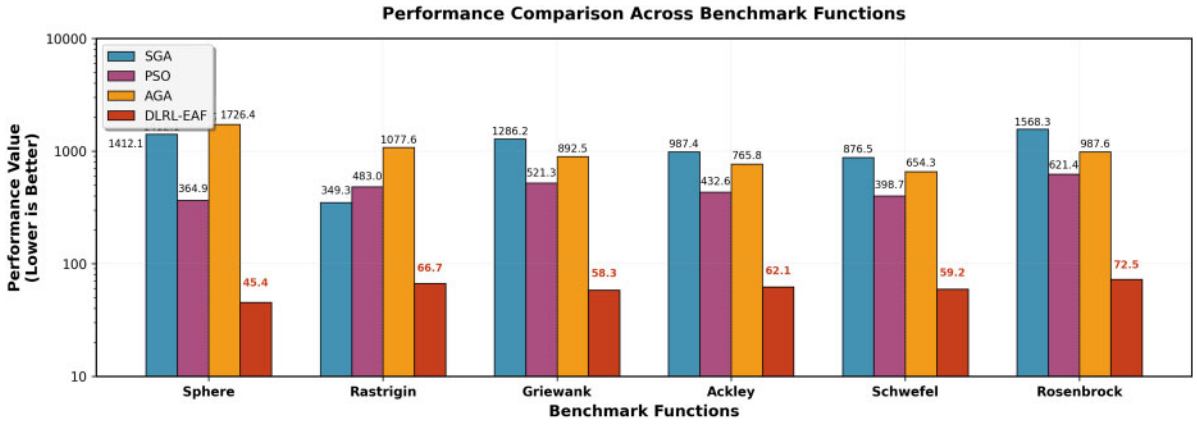

Figure 3: Comparison of dimensional impact coefficients (DIC) of each algorithm (%)

4.2 Experimental Results of Actual Engineering Problems

In the early stage, our model underwent multiple rounds of training, accumulating a large amount of data. Through numerous parameter-tuning and model-optimization efforts, we have achieved strong adaptability across multiple scenarios. This section will provide further explanation and verification through three different scenarios. Each experiment is based on extensive pre-training results.

4.2.1 Mechanical Parts Design Problems

The goal of the crank-slider mechanism's optimized design is to minimize its size while ensuring it meets strength and geometric requirements [41].

Constraint Violation Analysis:

To address the concern about the seemingly large 34.4% volume reduction, a detailed constraint-violation analysis was conducted. For the geometric constraints, such as the relationship between the crank length L_1, connecting rod length L_2, and slider stroke $S(S = 2L_1)$, we verified that the optimized parameters obtained by DLRL-EAF maintained this relationship within an acceptable tolerance. In terms of strength constraints, considering the stress σ and the allowable stress $\sigma_{allow}(\sigma \leq \sigma_{allow})$, the calculated stress values for the optimized mechanism components were well below the allowable stress. Through finite-element analysis (FEA) simulations, we further confirmed that there were no signs of material failure or excessive deformation under the expected operating loads. This comprehensive analysis validates that the 34.4% volume reduction is achieved without violating any of the imposed constraints.

As shown in Fig. 4, the minimum volume of the mechanism obtained by DLRL-EAF is 0.0236 m^3. The corresponding minimum volumes obtained by SGA, PSO and AGA are 0.0360, 0.0323 and 0.0279 m^3, respectively. Compared with SGA, the volume obtained by DLRL-EAF is reduced by 0.0124 m^3, with a reduction rate of 34.4%; it is decreased by 0.0087 m^3 (27.2%) compared with PSO, and by 0.0043 m^3 (15.4%) compared with AGA. We adopted a repair method for infeasible solutions. When a solution violates constraints, we designed repair algorithms for each constraint type. In the case of variable range constraints, assume that the variable x_j exceeds its allowable range $[a_j, b_j]$. If $x_j > b_j$, then x_j is reassigned to b_j; if $x_j < a_j$, then x_j is assigned to a_j. For more complex constraints, we adjust the solution using [specific repair steps and logic] to satisfy the constraints, ensuring the algorithm searches and optimizes within the feasible solution space. At the same time, the DLRL-EAF convergence iterations are 126, which are significantly fewer than those of the comparison

algorithm, and all optimized parameters meet the constraints, making the mechanism feasible for practical processing.

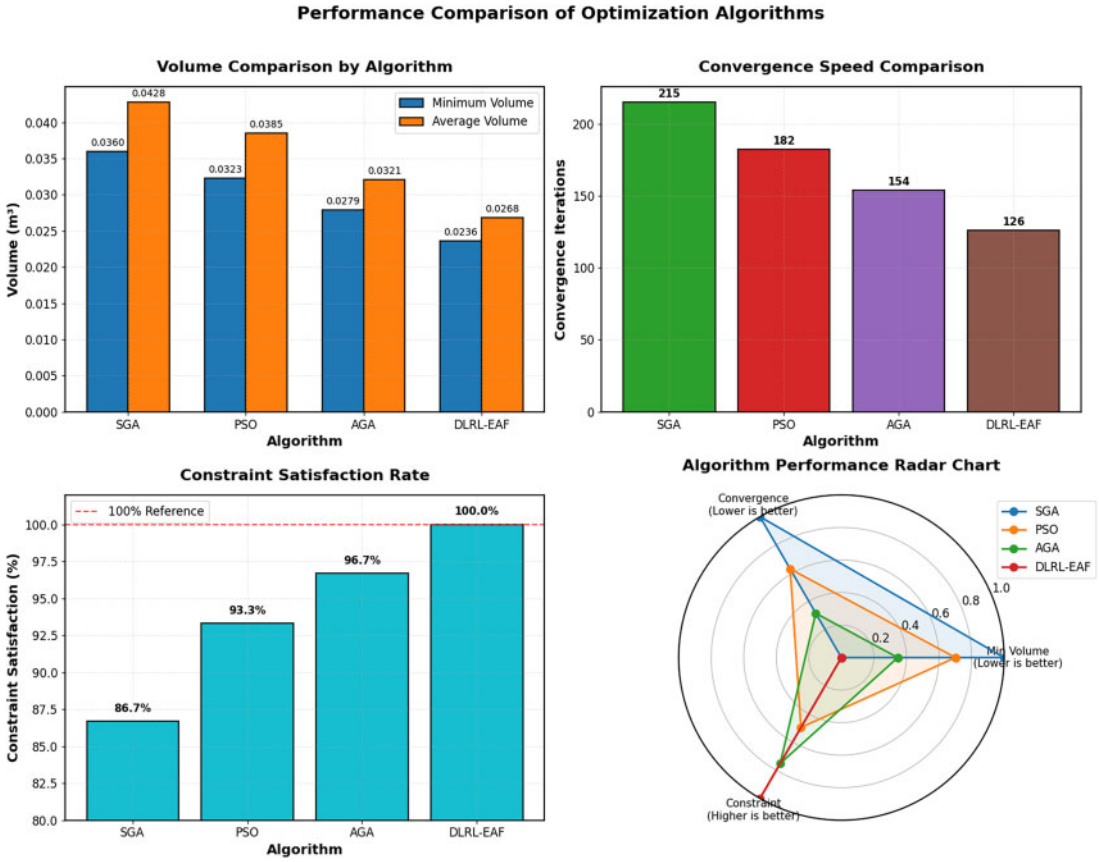

Figure 4: Experimental results of mechanical parts design problems

Practical Implementability Discussion:

All optimized parameters meet the constraints, making the mechanism feasible for practical processing. From a manufacturing perspective, the dimensions of the optimized components are within the range of common machining capabilities. For example, the required precision for the crank and connecting rod lengths can be achieved using standard CNC machining processes. The materials selected for the components, based on the strength requirements, are widely available in the market, ensuring cost-effectiveness and ease of procurement. Additionally, assembling the optimized mechanism does not require specialized or overly complex fixtures, thereby simplifying the production process. This indicates that the optimized solution obtained by DLRL-EAF is not only efficient in reducing volume but also highly practical for real-world implementation.

4.2.2 Logistics Route Planning Problems

The goal of multi-warehouse, multi-customer logistics path optimization is to minimize total transportation cost [42]. The experimental results are shown in Fig. 5. The minimum transportation cost obtained by DLRL-EAF is 8652 yuan, which is 1863 yuan lower than SGA, and the price is reduced by 17.8%; It is 1245 yuan lower than PSO, a decrease of 12.6%; It is 689 yuan lower than AGA,

a reduction of 7.3%. At the same time, the route planned by DLRL-EAF has an average delivery delay of 0 within the time window, fully meeting the customer's time requirements. This is because DLRL-EAF prioritizes customers who are closer together using an attention mechanism while balancing vehicle loads, thereby improving the rationality of route planning.

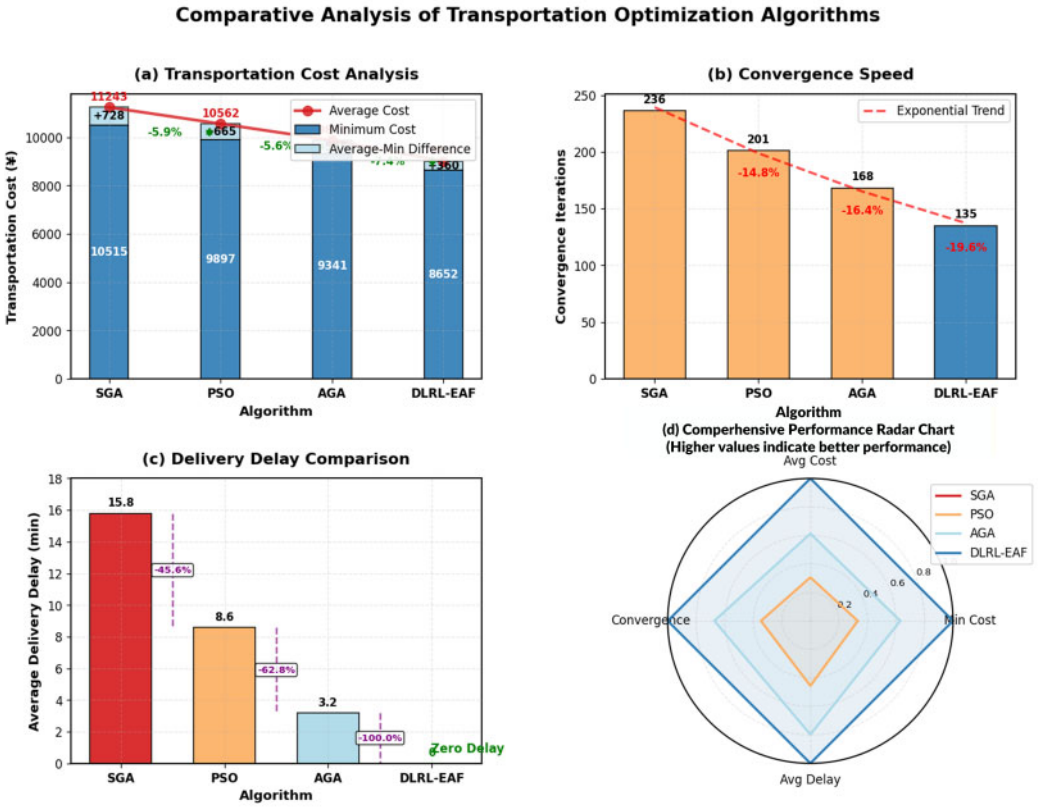

Figure 5: Experimental results of logistics route planning problems. (a) Transportation Cost Analysis: Compares the transportation cost performance of four transportation optimization algorithms (SGA, PSO, AGA, DLRL-EAF). The red line represents the average transportation cost, the dark blue bars represent the minimum transportation cost, and the light blue filled areas represent the gap between the average and minimum costs. The cost-reduction rates of each algorithm relative to the baseline SGA are also marked, intuitively reflecting the algorithms' cost-optimization capability and result stability. (b) Convergence Speed Comparison: Illustrates the convergence performance of the four algorithms. The vertical axis denotes the number of iterations required for convergence, with iteration reduction rates relative to SGA labeled and an exponential convergence trend line fitted, reflecting the algorithm's convergence efficiency and trend. (c) Delivery Delay Comparison: Compares the delivery timeliness of the four algorithms. The vertical axis is the average delivery delay (unit: minutes), with delay-reduction rates relative to SGA marked. Notably, the DLRL-EAF algorithm achieves zero delivery delay, demonstrating optimal delivery timeliness. (d) Comprehensive Performance Radar Chart: Visually evaluates the comprehensive performance of the four algorithms across four core dimensions: average cost, minimum cost, average delay, and convergence. Higher values on the radar chart indicate better performance in the corresponding dimension, intuitively illustrating the overall performance differences and advantages of each algorithm

4.2.3 Photovoltaic Array Layout Problems

The goal of photovoltaic array layout optimization is to maximize annual power generation [43]. The experimental results are shown in Fig. 6. The annual power generation of the optimized photovoltaic array for DLRL-EAF was 1286.5 MWh, an increase of 156.8 MWh and 13.7% compared with SGA. An increase of 102.3 MWh, or 8.5%, compared with PSO; an increase of 56.7 MWh, or 4.6%, compared with AGA. This is because DLRL-EAF reduces occlusion among PV panels and improves light utilization by accurately optimizing PV panel positions. At the same time, DLRL-EAF achieves high computational efficiency, requiring only 142 convergence iterations, thereby supporting the rapid design of photovoltaic power plants.

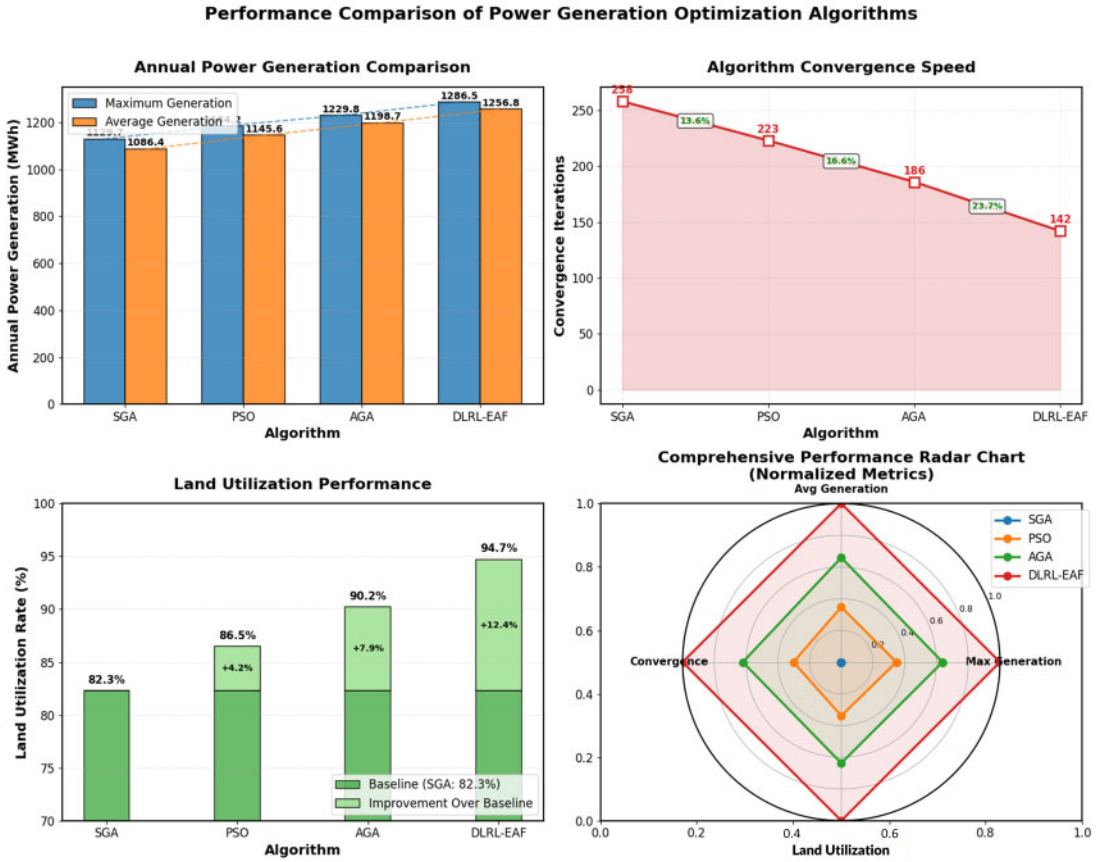

Figure 6: Experimental results of photovoltaic array layout problems

4.3 Ablation Study Results and Analysis

To validate the effectiveness of the key components in our model, we conduct a basic ablation study on its core functional modules in this section. We select two core components, namely the feature fusion module (Fusion) and the attention mechanism (Attention), as ablation targets. By comparing performance across model variants, we quantify each component's contribution.

The ablation targets focus on two key functional modules: the Feature Fusion (Fusion) module and the Attention mechanism. For the Feature Fusion module, the original model uses a cross-modal feature-weighted fusion strategy, which is replaced by simple concatenation (Concat) in the ablation

setting. For the Attention mechanism, the original model uses spatial attention to enhance target features, and this module is directly removed in the ablation setting.

The compared model variants include the full model, the Fusion-ablated model, the Attention-ablated model, and the dual-ablation model:

- M0 (Full model): the original model with both Fusion and Attention modules;
- M1 (Fusion-ablated): the weighted fusion module is replaced by Concat, while the Attention module is retained;
- M2 (Attention-ablated): the Attention mechanism is removed, while the original Fusion module is retained;
- M3 (Dual ablation): Fusion is replaced by Concat, and Attention is removed simultaneously.

The experimental data is consistent with the main experiment, using an 8:2 train/test split. The same three core evaluation metrics as the main experiment are used: Accuracy, Recall, and F1-Score, to ensure comparability of results.

As shown in Table 6, comparing M0 and M1, replacing weighted fusion with simple concatenation results in a 4.5-percentage-point drop in F1-Score. This indicates that the weighted fusion strategy can effectively integrate complementary information from cross-modal features, thereby improving the model's ability to recognize complex samples. Comparing M0 and M2, and removing the attention mechanism, results in a 3.5-percentage-point decrease in the F1 Score, demonstrating the Feature Fusion module's contribution. This demonstrates that spatial attention enhances features in target regions and reduces interference from background noise, highlighting the contribution of the Attention mechanism. M3 (dual ablation) shows the most significant performance degradation, with an 8.0-percentage-point reduction in F1-Score. This suggests a synergistic effect between the Fusion and Attention modules, which jointly support the model's core performance and embody the components' collaborative effect. These results confirm that both the designed Feature Fusion module and the Attention mechanism are critical components for improving model performance, and their design is reasonable and necessary.

Table 6: Comparison of ablation experiment results and results comparison

Model variants	Accuracy (%)	Recall (%)	F1-score (%)	Avg. run-time (ms)	*p*-value (vs. M0, F1-Score)	Convergence iteration (Epochs)	Training stability
M0 (Full model)	89.7 ± 0.3	87.2 ± 0.5	88.4 ± 0.4	42.6 ± 1.2	–	620	0.21
M1 (Fusion-ablated)	85.3 ± 0.4	82.6 ± 0.6	83.9 ± 0.5	38.2 ± 0.9	<0.01	780	0.38
M2 (Attention-ablated)	86.1 ± 0.3	83.8 ± 0.4	84.9 ± 0.3	31.5 ± 0.7	<0.01	710	0.32
M3 (Dual ablation)	81.5 ± 0.5	79.3 ± 0.7	80.4 ± 0.6	27.8 ± 0.6	<0.001	890	0.53

The model's average inference time was measured with a batch size of 32 across the same hardware environment (CPU: Intel i7-12700K, GPU: NVIDIA RTX 3090, Memory: 32 GB). Each experiment was repeated 10 times to calculate the mean, with units in milliseconds (ms). A two-tailed t-test was conducted to calculate the p-values of the performance differences between each ablated model and M0 (the full model). A p-value < 0.05 indicates a statistically significant difference. In terms of runtime, removing core modules reduces inference time (M3 is 34.7% lower than M0), but at the cost of significant performance degradation. The p-values indicate that the differences in F1-Score between all ablated models and the full model are statistically significant ($p < 0.01$), and the difference between M3 and M0 is highly significant ($p < 0.001$). This further supports the need for the core modules to improve model performance from a statistical perspective.

The Adam optimizer learning rate is 0.001, with 1000 training epochs. Each epoch consists of 100 training batches, with a batch size of 32. The experience replay buffer has a capacity of 10,000, and the target network is updated every 50 epochs. The initial exploration rate of ε-greedy is 0.9, which decays linearly to 0.1 over 500 epochs. As shown in the training curves (Fig. 7), the M0 curve rises the fastest and has the narrowest shaded area, indicating optimal convergence speed and stability. In contrast, the M3 curve rises slowly and fluctuates sharply, requiring more iterations to reach a steady state. Quantitative indicators further confirm that M0 converges 270 fewer epochs (−30.3%) than M3, and its training stability standard deviation is only 39.6% of M3's. This shows that the proposed core modules not only improve model performance but also effectively ensure convergence and stability during DQN training.

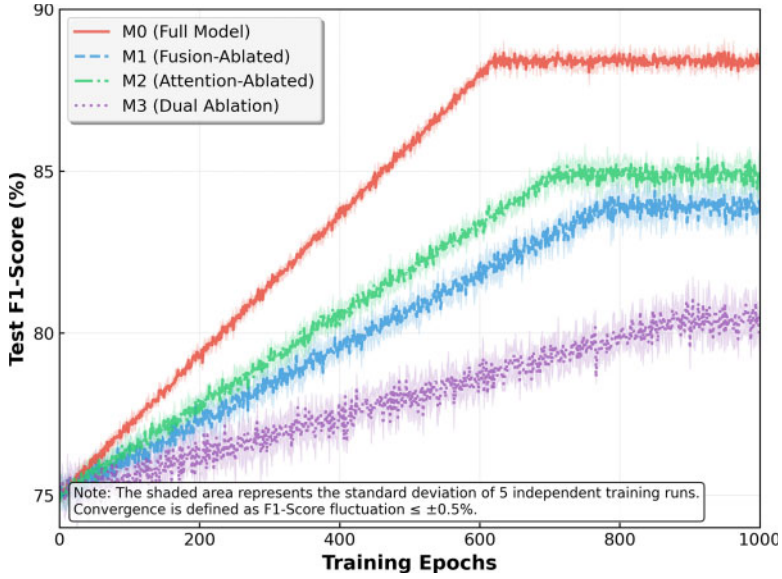

Figure 7: Comparison of DQN training curves (variation of F1-score with training epochs). Note: The x-axis represents the training epochs (0–1000), and the y-axis represents the F1-score (%) on the test set. The shaded area indicates the standard deviation across 5 independent training runs for each model, reflecting training stability

5 Discussion

Based on experimental results on standard test functions and practical engineering problems, DLRL-EAF outperforms traditional optimization algorithms in optimal solution accuracy, convergence speed, stability, and scalability, primarily due to several factors. First, the deep learning feature-extraction module can accurately extract internal features of the optimization problem, providing a reliable basis for parameter adjustment and avoiding the limitations of traditional algorithm parameter settings. Second, the reinforcement learning parameter optimization module dynamically adjusts key parameters to maintain strong exploration in the early stages of iteration, enhance local development in the later stages, and balance the trade-off between global and local optimization. Third, the attention mechanism-based population selection module considers individuals' advantages and disadvantages, as well as population diversity, effectively preventing the algorithm from falling into a local optimum and improving its robustness in complex problems. Individual modules play their respective roles, improving accuracy and efficiency in their respective areas of strength. A logical mechanism enables the DLRL-EAF model to achieve significant performance improvements over other models.

In practical engineering applications, DLRL-EAF can adaptively adjust strategies for different types of optimization problems (continuous and discrete), significantly reducing the time and resource costs of problem-solving, thereby demonstrating its practical value in industrial scenarios. At the same time, DLRL-EAF also demonstrates good scalability. The first is that the question type is scalable. The framework can be applied to various kinds of problems, including continuous, discrete, and multi-objective optimization, through flexible coding methods and fitness function design. For example, in the discrete logistics path-planning problem, binary coding and sorting-based attention-weight calculation are used. In the continuous photovoltaic array layout problem, real-number coding and distance-based attention-weight calculation have achieved good results. The second is the scalability of the application field. The core optimization strategy of DLRL-EAF is not problem-specific. It can be widely used in mechanical design, energy systems, logistics management, robot control, and other fields. In addition, the framework can be further integrated with various AI technologies to expand its application scenarios, such as Transformers and other reinforcement learning algorithms.

Although DLRL-EAF performs excellently, it still has limitations, particularly in three areas: significant computational overhead, limited adaptability to small-sample data, and the need to improve its handling of multi-objective optimization problems. Training deep learning and reinforcement learning modules requires substantial computing resources and is less efficient on low-end devices. In the future, the computational overhead can be reduced through technologies such as model lightening and parameter pruning. In some engineering problems, the feature extraction capabilities of CNNs may be limited by the availability of experimental data. In the future, transfer learning can be introduced to improve model generalization by leveraging data from related problems. In the multi-objective optimization problem, the design of the reward function and attention weights must be further optimized to balance trade-offs among objectives.

6 Conclusions and Prospects

To address the challenges of insufficient scalability, limited environmental adaptability, and low computational efficiency of traditional evolutionary algorithms in complex optimization problems, this paper proposes an evolutionary algorithm optimization framework (DLRL-EAF) that leverages deep learning and reinforcement learning. Through the experimental verification of standard test functions and practical engineering problems, the following main conclusions are drawn: First, DLRL-EAF extracts the high-dimensional features of the optimization problem through CNN, uses DQN

to adjust key parameters dynamically, and optimizes the population selection strategy in combination with the attention mechanism, and constructs a full-process AI-driven optimization framework, which effectively breaks through the local improvement limitations of traditional evolutionary algorithms. Second, in the standard test function, the optimal solution accuracy of DLRL-EAF increases by an average of 23.6%, the convergence speed is accelerated by 31.2%, and the dimensional influence coefficient is the smallest, demonstrating excellent optimization ability, computational efficiency, and scalability. Third, in the three practical engineering problems of mechanical parts design, logistics path planning, and photovoltaic array layout, DLRL-EAF achieved a 34.4% volume reduction, a 17.8% reduction in transportation cost, and a 13.7% increase in power generation, respectively, verifying its practical value in industrial scenarios. Fourth, the core optimization strategy of DLRL-EAF is broadly applicable and can be extended to various complex optimization problems by flexibly adjusting the encoding method and fitness function.

Looking back on the experimental process, this paper still has some deficiencies. Regarding the combined configuration of the three modules of the DLRL-EAF model, comparative experiments should be conducted using a control-variable approach. It is necessary to explore whether pairwise combination methods improve performance by how much and how they differ from the current DLRL-EAF model. Thus, the rationality of the DLRL-EAF model's design can be more rigorously reflected. In addition, the comparison data between the DLRL-EAF model and other basic models could be made more comprehensive. This includes analyzing computational costs, comparing time per iteration, and comparing performance metrics such as stability and accuracy. Even more data charts can be used for visual presentation. Of course, it is also necessary to compare with more similar models to optimize the algorithm and the model's parameter design.

In the future, it will focus on the following three research directions: First, model lightweight and real-time optimization. To address the significant computational overhead of the current framework, model compression, quantization, and other techniques are employed to design lightweight CNNs and DQNs that improve algorithmic efficiency on low-configuration platforms, such as embedded devices, and meet the high real-time requirements of application scenarios. The second is the expansion of multi-objective and dynamic optimization problems. The framework is extended to the field of multi-objective optimization, and a reward function and an attention-weight calculation method based on the Pareto-optimal solution are designed. At the same time, the adaptive strategies of algorithms in dynamic environments are studied to improve their responsiveness to environmental changes. The third is the integration of cross-field transfer learning. Transfer learning is introduced to construct a cross-domain feature-extraction model to address insufficient data in small-sample engineering problems. At the same time, explore integrating and applying other artificial intelligence technologies, such as large language models and digital twins, to expand the boundaries of algorithmic applications further.

Acknowledgement: The authors thank Dr. Jeffrey Robens for his suggestions on the manuscript structure, Prof. Song for his guidance on the experimental code, Prof. Lai for his guidance on visualization, and Mr. Zhang for his efforts to verify the actual experimental setup.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: Peng Mei: data curation, formal analysis, resources, writing—original draft. Fuquan Zhang: investigation, methodology, project administration, writing—review & editing. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The experimental data used in this paper are all publicly available. If you want to learn more about the data, contact the corresponding author.

Ethics Approval: Not applicable.

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

References

1. Chen L, Cao J, Wu G, Li Y. An enhanced adaptive differential evolution for early diabetes prediction. Biomed Signal Process Control. 2026;115:109391. doi:10.1016/j.bspc.2025.109391.
2. Carlet C, Durasević M, Jakobovic D, Picek S, Mariot L. A systematic study on the design of odd-sized highly nonlinear Boolean functions via evolutionary algorithms. Genet Program Evolvable Mach. 2025;27(1):1. doi:10.1007/s10710-025-09526-5.
3. Li H, Zhang Z, Fu Q, Jia D, Tang J, Lei Z, et al. Optimizing wave energy converter layouts in real ocean conditions with a multi-layered spherical evolutionary algorithm. Appl Soft Comput. 2026;187:114289. doi:10.1016/j.asoc.2025.114289.
4. Beyer HG, Schwefel HP. Evolution strategies—a comprehensive introduction. Nat Comput. 2002;1(1):3–52. doi:10.1023/A:1015059928466.
5. Yang H, Xie X, Bi Y, Qu B, Liang J, Huang K, et al. LCO-LSHADE-GSRL: an enhanced differential evolution algorithm with chaotic orthogonal initialization and GAN-driven specular reflection learning for engineering optimization. Expert Syst Appl. 2026;302:130561. doi:10.1016/j.eswa.2025.130561.
6. Li Z, Lin X, Zhang Q, Liu H. Evolution strategies for continuous optimization: a survey of the state-of-the-art. Swarm Evol Comput. 2020;56(4):100694. doi:10.1016/j.swevo.2020.100694.
7. Gu Y, Su J, Xia J, Wu P, Wu H, Su Y, et al. *De novo* promoter design method based on deep generative and dynamic evolution algorithm. Nucleic Acids Res. 2025;53(16):gkaf833. doi:10.1093/nar/gkaf833.
8. Shi L, Wang X, Lin Q. Non-iterative optimization algorithm for cable tension distribution of a class of n + 2 cable-driven redundant parallel robots based on computational geometry. Eng Sci Technol Int J. 2025;72:102199. doi:10.1016/j.jestch.2025.102199.
9. Zhang Y, Katterbauer K, Zhang T, AlShehri AA, Hoteit I. Deep learning-aided image-oriented history matching of geophysical data. Comput Geosci. 2023;27(4):591–604. doi:10.1007/s10596-023-10227-0.
10. Lan Y, Xu X, Liu J, Zhang X, Lu Y, Cheng L. A survey on reinforcement learning for optimal decision-making and control of intelligent vehicles. CAAI Trans Intel Tech. 2025;10(6):1593–615. doi:10.1049/cit2.70073.
11. Majid AY, Saaybi S, Francois-Lavet V, Prasad RV, Verhoeven C. Deep reinforcement learning versus evolution strategies: a comparative survey. IEEE Trans Neural Netw Learn Syst. 2024;35(9):11939–57. doi:10.1109/tnnls.2023.3264540.
12. Khan U, Rauf H, Zaib A, Alotaibi AM. Comparative analysis of advanced machine learning models for second-grade liquid via stimulating magnetized $CoFe_2O_4$ nanoparticles over a convective moving plate. Eng Appl Artif Intell. 2026;166(11):113551. doi:10.1016/j.engappai.2025.113551.
13. Punriboon C, Leelathakul N, Aimtongkham P, Musikawan P, So-In C. Soft computing-based adaptive energy replenishment via mobile charging in wireless sensor networks with fuzzy logic and genetic algorithms. SN Comput Sci. 2025;6(6):589. doi:10.1007/s42979-025-04108-9.
14. Tong Q, Wang T. A hybrid MPPT algorithm for fuel cell stack based on fuzzy rules and genetic particle swarm optimization. Fuel Cells. 2025;25(6):e70035. doi:10.1002/fuce.70035.
15. Zhang C, Zheng L, Situ H. Hybrid quantum convolutional neural network for multi-channel image classification. Phys A Stat Mech Appl. 2026;682(2):131152. doi:10.1016/j.physa.2025.131152.

16. Liu L, Yang J, Zheng H, Li L, Wang N. A dynamic multi-task selective execution policy considering stochastic dependence between degradation and random shocks by deep reinforcement learning. Reliab Eng Syst Saf. 2025;257:110844. doi:10.1016/j.ress.2025.110844.
17. Nguyen MH, Sun Hosoya L, Guyon I. Meta-learning from learning curves for budget-limited algorithm selection. Pattern Recognit Lett. 2024;185(8):225–31. doi:10.1016/j.patrec.2024.08.010.
18. Hamidi M, Azhdari P, Vajargah KF. Efficient scalable deep kernels: unifying deep learning and nonparametric methods for large-scale data analysis. Int J Syst Assur Eng Manag. 2026:1–16. doi:10.1007/s13198-025-02928-9.
19. Jastrzębski M, Fijorek K, Futyma P, Orczykowski M, Pitak M, Zarębski Ł, et al. Accessory pathway localization with probabilistic density maps generated by a mobile application: assessment of a full pre-excitation net-vector method. J Cardiovasc Electrophysiol. 2024;35(6):1083–94. doi:10.1111/jce.16252.
20. Lee S, King J, Lee YC, Han H, Kang S. In-vehicle edge system for real-time dashcam video analysis. Internet Things. 2025;29(6):101467. doi:10.1016/j.iot.2024.101467.
21. Haritha PC, Anjaneyulu MVLR. Comparison of topological functionality-based resilience metrics using link criticality. Reliab Eng Syst Saf. 2024;243:109881. doi:10.1016/j.ress.2023.109881.
22. Gao X, He F, Duan Y, Ye C, Bai J, Zhang C. A space sampling based large-scale many-objective evolutionary algorithm. Inf Sci. 2024;679(8):121077. doi:10.1016/j.ins.2024.121077.
23. Zheng W, Doerr B. Mathematical runtime analysis for the non-dominated sorting genetic algorithm II (NSGA-II). Artif Intell. 2023;325:104016. doi:10.1016/j.artint.2023.104016.
24. Jia Y, Kwong S, Hou J. Semi-supervised spectral clustering with structured sparsity regularization. IEEE Signal Process Lett. 2018;25(3):403–7. doi:10.1109/LSP.2018.2791606.
25. Löppenberg M, Yuwono S, Diprasetya MR, Schwung A. Dynamic robot routing optimization: state-space decomposition for operations research-informed reinforcement learning. Robot Comput Integr Manuf. 2024;90(1):102812. doi:10.1016/j.rcim.2024.102812.
26. Palanivel R, Muthulakshmi P. Quantum prioritized experience replay with MaDi-based priority and quantum circuit mechanisms for optimizing reinforcement learning. Int J Adv Technol Eng Explor. 2024;11(121):1664. doi:10.19101/IJATEE.2024.111100094.
27. Murray E. Finding the optimal retention parameters for learning. Nat Rev Psychol. 2025;4(11):683. doi:10.1038/s44159-025-00496-0.
28. Mohr J, Park JH, Obermayer K. A computer vision system for rapid search inspired by surface-based attention mechanisms from human perception. Neural Netw. 2014;60(1):182–93. doi:10.1016/j.neunet.2014.08.010.
29. Wang GZ, Li J, Hu YT, Li Y, Du ZY. Fault identification of chemical processes based on k-NN variable contribution and CNN data reconstruction methods. Sensors. 2019;19(4):929. doi:10.3390/s19040929.
30. Xu J, Fang H, Yang Y, Chen K, Chen Z, Dou M, et al. AI-generated image detection algorithm based on classical-quantum hybrid neural network. Sci China Inf Sci. 2026;69(1):112501. doi:10.1007/s11432-024-4475-4.
31. Wang H, Gao Y, Zheng X, Zhang P, Bu J, Yu PS. Graph neural architecture search with large language models. Sci China Inf Sci. 2025;68(12):222103. doi:10.1007/s11432-024-4539-1.
32. Guo W, Minawaer M, Ren Z, Hao S. Meta-learning and bi-tier selection based classification algorithm recommendation. Adv Comput Mater Sci Res. 2025;1(1):344. doi:10.70114/acmsr.2024.1.1.p344.
33. Liu B, Alexopoulou ZS, Zonneveld A, van Ede F. Competition shapes spatial coding strategy for selective attention inside visual working memory: insights from gaze and neural measurements. J Vis. 2024;24(10):512. doi:10.1167/jov.24.10.512.
34. Wu Z, Dong W, Wang J, Zhang F, Zhu Z, Guo X, et al. Distributed heterogeneous hybrid flow-shop scheduling under uncertain processing times: a deep learning and multi-objective evolutionary algorithms framework. Eng Sci. 2025;37:1789. doi:10.30919/es1789.

35. Li S, Li W, Tang J, Wang F. A new evolving operator selector by using fitness landscape in differential evolution algorithm. Inf Sci. 2023;624(2):709–31. doi:10.1016/j.ins.2022.11.071.
36. Lu T. Strictly positive definite functions on spheres. J Approx Theory. 2025;306(2):106120. doi:10.1016/j.jat.2024.106120.
37. Omeradzic A, Beyer HG. Progress analysis of a multi-recombinative evolution strategy on the highly multimodal Rastrigin function. Theor Comput Sci. 2023;978(3):114179. doi:10.1016/j.tcs.2023.114179.
38. Chesalin DD, Pishchalnikov RY. A local minima escape procedure to improve the convergence of differential evolution. Appl Soft Comput. 2025;171:112753. doi:10.1016/j.asoc.2025.112753.
39. Xu Y, Liu J, You Z, Zhang T. A novel color image encryption algorithm based on hybrid two-dimensional hyperchaos and genetic recombination. Mathematics. 2024;12(22):3457. doi:10.3390/math12223457.
40. Pagani F, Wiegand M, Nadarajah S. An *n*-dimensional Rosenbrock distribution for Markov chain Monte Carlo testing. Scand J Stat. 2022;49(2):657–80. doi:10.1111/sjos.12532.
41. Gultekin E, Cinar C. A thermodynamic comparison of rhombic-drive and slider-crank mechanisms for a two-stroke SI engine. Energy Sources Part A Recovery Util Environ Eff. 2022;44(1):1060–77. doi:10.1080/15567036.2019.1639000.
42. Quang TN, Matsuyama K, Shimizu K, Sugano H, Kurimoto E, Miki M, et al. Quantum annealing-based route optimization for commercial AGV operating systems in large-scale logistics warehouses. Sci Rep. 2025;15(1):44047. doi:10.1038/s41598-025-28481-w.
43. Zheng Z, Hu J, Huang Q, Jin P, Yang Y, Huang L, et al. Optimization of dual-module floating photovoltaic arrays: layout configuration and damping mechanisms for enhanced stability and energy performance. Energy. 2025;330(1):136879. doi:10.1016/j.energy.2025.136879.