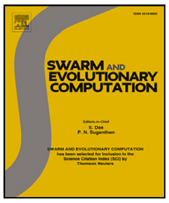
Swarm and Evolutionary Computation 87 (2024) 101568

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo



Differential evolution based on strategy adaptation and deep reinforcement learning for multimodal optimization problems[☆]

Zuowen Liao [a,d], Qishuo Pang [b,*], Qiong Gu [c,*]

[a] Beibu Gulf Ocean Development Research Center, Beibu Gulf University, Qinzhou 535011, China
[b] College of Mechanical, Naval Architecture and Ocean Engineering, Beibu Gulf University, Qinzhou 535011, China
[c] School of Computer Engineering, Hubei University of Arts and Science, Xiangyang 441053, China
[d] Key Laboratory of Beibu Gulf Offshore Engineering Equipment And Technology (Beibu Gulf University), Education Department of Guangxi Zhuang Autonomous Region, Qinzhou 535000, China

ARTICLE INFO

Keywords:
Multimodal optimization problems
Strategy adaptation
Deep reinforcement learning
Differential evolution

ABSTRACT

Multimodal optimization problems (MMOPs) include multiple optima, which are common in practical fields. However, the success of solving MMOPs requires the algorithm to have both exploration and exploitation performance. How to select a reasonable search strategy is a difficult problem facing MMOPs. In this research, a differential evolution based on strategy adaptation and deep reinforcement learning, termed SA-DQN-DE, is proposed to select mutation strategies reasonably and search the optima effectively, which mainly includes three aspects. First, strategy adaptation, which calculates selection probability based on the feedback of different mutation operators in previous evolution, and assigns mutation operations to each individual to provide guidance for the next stage of evolution is developed. Secondly, a new historical individual preservation method is designed to improve search efficiency. Thirdly, deep reinforcement learning is applied to select multiple local search operators to refine the accuracy of potential optimal solutions. The performance of SA-DQN-DE is tested on publicly acknowledged CEC2013 benchmark MMOP set. The experimental results demonstrate that the proposed SA-DQN-DE has competitive performance compared with some of its peer multimodal optimization algorithms.

1. Introduction

Many real-world problems, such as electromagnetic design [1], power system design [2], and truss-structure optimization [3], often need to locate various optima. These problems are named multimodal optimization problems (MMOPs). Multiple optima can provide decision-makers with different feasible solutions for practical application problems. However, MMOPs contain different fitness landscapes, and solving MMOPs remains a challenging task. An algorithm must have exploration and exploitation performance in order to find multiple equally important solutions in a single run [4].

Evolutionary algorithms (EAs) are population-based algorithms that have potential advantages in solving complex optimization problems [5,6]. Many EAs have been used to deal with MMOPs and have achieved great success. To extend the ability of EAs to locate multiple optima of MMOPs, several combinations of niching techniques with EAs have been designed [7]. A niching technique can divide a population into multiple sub-populations. Each sub-population contains one or more global optima. EAs can obtain distinct global optima by searching in several niches. Classical niching techniques include crowding [8], speciation [9], local binary pattern [10] and locality sensitive hashing [11].

Differential evolution (DE), which is a type of EA, combines niching technique to form a variety of effective methods to deal with MMOPs [12]. The crowding [13] and species [14] techniques have been combined with DE, known as CDE and SDE respectively, to solve MMOPs. Because the crowding size and species distance have great effects on performance, Qu et al. proposed neighborhood CDE and SDE respectively, which mitigates the influence of parameters and improves the performance of the algorithm [8]. Gao et al. integrated parameter adaptation with CDE and SDE, abbreviated as Self-CCDE and Self-CSDE [15]. Biswas et al. proposed the PNPCDE method, which uses parent-centric mutation strategies to enhance population diversity [16]. A local information matrix improves search ability, referred to as LoIDE [17]. More recently, Wang et al. has exploited a contour

☆ This work is partly supported by the National Natural Science Foundation of China under (62362001) and the Major Project of Hubei Province Science and Technology (2023BCA006).

* Corresponding authors.
E-mail addresses: liaozuowen@bbgu.edu.cn (Z. Liao), pangqishuo@bbgu.edu.cn (Q. Pang), qionggu@hbuas.edu.cn (Q. Gu).

https://doi.org/10.1016/j.swevo.2024.101568
Received 26 December 2023; Received in revised form 17 February 2024; Accepted 1 April 2024
Available online 25 April 2024
2210-6502/© 2024 Published by Elsevier B.V.

prediction method to determine where the optimal value is likely to exist and guide the evolution of the algorithm [18]. In [19], coarse-grained and fine-grained niching (CFN) strategies were designed to solve MMOPs, referred to as CFNDE. Jiang et al. developed niche center distinguish-based DE (NCD-DE) to locate multiple optima [20]. Zhang et al. proposed proximity ranking-based multimodal differential evolution (PRMDE) [21].

Although DE based multimodal optimization algorithms have improved, they still have several deficiencies. First, existing algorithms generally adopt a single search strategy [13,14,14,15], which is not good for exploring the feasible region. As indicated by the theory of "no free lunch" [22], there is no single mutation strategy can perform well on all the test set. Second, such algorithms [19–21] rarely track individual states in order to select suitable search strategies to improve accuracy, resulting in low convergence efficiency. Therefore, both cases are not conducive to improving the accuracy of solutions in promising regions.

Using an ensemble of search strategies is a promising method to enhance the performance of EAs [23]. Through the coordination of various strategies during evolution, most optimization problems can be solved effectively [24–26]. However, there is still room for improvement in how this method deals with MMOPs. The MMOPs contain terrain with different characteristics, and multiple search strategies are conducive to searching feasible regions. When the algorithm refines multiple potential optimal solutions, different local search strategies may accelerate convergence.

Motivated by these, we propose a multimodal optimization algorithm, called strategy adaptation and deep reinforcement learning based differential evolution (SA-DQN-DE). In SA-DQN-DE, the success rate of different mutation strategies in the past evolution is calculated and their use frequency in the next evolution is assigned to improve search performance. Moreover, the deep Q network (DQN) is used to track the individual state in the exploitation stage, and a suitable local search strategy is selected to refine the fitness. Finally, an improved archive technique is designed to store past individuals, and the historical information is used to assist in evolution.

Experimental results on the CEC2013 demonstrate the superiority of SA-DQN-DE. Compared with peer multimodal algorithms, SA-DQN-DE can be significantly improved through multiple mutation strategies and DQN.

The main contributions of this paper are:

- An ensemble of mutation strategies is applied to solve MMOPs. The search efficiency of the algorithm is improved by calculating the success rate of different strategies in the past evolution and assigning the use frequency of different strategies in the next evolution.
- DQN perceives the state of an individual and selects a suitable local search strategy to refine its fitness.
- An improved archive technique saves past individual information and uses it to assist in the algorithm's evolution.

The remainder of this paper is organized as follows. Section 2 describes DE, EAs for MMOPs, and motivation. Section 3 introduces the proposed approach (SA-DQN-DE) in detail. The experimental analysis and comparison of algorithms are presented in Section 4. In Section 5, parameter sensitivity is investigated. Finally, Section 6 summarizes the paper.

2. Related work

2.1. Differential evolution

DE is a popular heuristic optimization method [27]. It implements three operators:

Mutation: DE adopts the mutation operator to generate a mutant vector $\mathbf{v}_i$. Five classical mutation operations follow:

(1) *DE/rand/1*:

$$\mathbf{v}_i = \mathbf{x}_{r1} + F \cdot (\mathbf{x}_{r2} - \mathbf{x}_{r3}) \tag{1}$$

(2) *DE/best/1*:

$$\mathbf{v}_i = \mathbf{x}_{best} + F \cdot (\mathbf{x}_{r1} - \mathbf{x}_{r2}) \tag{2}$$

(3) *DE/rand-to-best/1*:

$$\mathbf{v}_i = \mathbf{x}_i + F \cdot (\mathbf{x}_{best} - \mathbf{x}_i) + F \cdot (\mathbf{x}_{r1} - \mathbf{x}_{r2}) \tag{3}$$

(4) *DE/best/2*:

$$\mathbf{v}_i = \mathbf{x}_{best} + F \cdot (\mathbf{x}_{r1} - \mathbf{x}_{r2}) + F \cdot (\mathbf{x}_{r3} - \mathbf{x}_{r4}) \tag{4}$$

(5) *DE/rand/2*:

$$\mathbf{v}_i = \mathbf{x}_{r1} + F \cdot (\mathbf{x}_{r2} - \mathbf{x}_{r3}) + F \cdot (\mathbf{x}_{r3} - \mathbf{x}_{r4}) \tag{5}$$

Here, $\mathbf{x}_{r1}, \mathbf{x}_{r2}$, $\mathbf{x}_{r3}$, $\mathbf{x}_{r4}$ and $\mathbf{x}_{r5}$ are randomly selected from the population, and $r1 \neq r2 \neq r3 \neq r4 \neq r5 \neq i$. $\mathbf{x}_{best}$ is the best individual. F is the scaling factor.

Crossover: after mutation, the crossover operator is employed to each pair of $\mathbf{x}_i$ and $\mathbf{v}_i$:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand_j < CR \text{ or } j = j_{rand} \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{6}$$

where CR is the crossover rate; $rand_j$ is a decimal between 0 and 1; $j_{rand} \in \{1, 2, \ldots, D\}$ and D is the dimension.

Selection: The selection operator can be expressed as follows:

$$\mathbf{x}_i = \begin{cases} \mathbf{u}_i, & \text{if } f(\mathbf{u}_i) \geq f(\mathbf{x}_i) \\ \mathbf{x}_i, & \text{otherwise} \end{cases} \tag{7}$$

where $f(\cdot)$ is the fitness of the individual.

2.2. Multimodal algorithms

(1) Niching techniques with EAs: The algorithms in which crowding and speciation combine with DE are referred to as CDE [28] and SDE [14], respectively. Both algorithms are very sensitive to parameters. To overcome this deficiency, Qu et al. [8] designed a neighborhood-based niching approach and applied it to CDE and SDE, and the resulting algorithms, *i.e.*, NCDE and NSDE, which effectively solved the parameter sensitivity. Gao et al. [15] applied a clustering technique to CDE and SDE and adopted the parameter adaptive mechanism to solve MMOPs. Zhang et al. developed niching techniques based on local sensitive hashing (LSH) [11] and the Voronoi graph [29] to effectively locate multiple global optima of MMOPs. Similarly, Wang et al. adopted affinity propagation clustering to automatically partition population [18] and fitness and distance based local search techniques [30] to handle MMOPs. Nearest-better clustering (NBC) [31] has been used to solve MMOPs. Lin et al. [32] developed NBC-based on speciation DE to solve MMOPs. Luo et al. [33] adopted the NBC technique to construct the raw species, avoiding finding the same optima. Zhao et al. [10] applied local binary pattern (LBP) to search the promising region and locate distinct optima. Additionally, Huang et al. [34] developed a probabilistic niching technique based on binary-space-partition-tree. In order to deal with the problem of niche-parameter sensitivity, Wang et al. proposed a parameter-free niche-method and designed a distributed DE (AED-DDE) [35]. Sun et al. utilized nearest-density clustering (NDC) to partition the population and designed two nbest-based mutation operators to evolve each species (NDC-DE) [36]. Li et al. adopted the Hill-Valley technique to assist DE algorithm in solving MMOPs [37]. A network community-based DE was devised to effectively find multiple optima [38]. Wu et al. proposed a kriging model-based EA with support vector machine to locate multiple optima [39]. Du et al. developed a surrogate-assisted EAs with knowledge transfer for balancing exploration and exploitation [40].

(3) Multi-objective EAs (MOEAs): These methods transform a MMOP into a multi-objective problem (MOP) [41]. Subsequently,

MOEAs can deal with MOPs and find multiple optima of MMOPs. Basak et al. [41] developed a biobjective DE to solve MMOPs. Wang et al. [42] transformed a MMOP into two conflicting objectives and developed MOMMOP to locate multiple optima. However, this algorithm does not perform well on high dimensional problems. Chen et al. [43] used an adaptive diversity indicator as an optimization objective and an adaptive peak detection method (EMO-MMO) to find optima.

3. Our approach

This section explains our motivation and the detailed implementation of the proposed algorithm.

3.1. Motivation

The motivations of this paper are illustrated as follows.

- An ensemble of mutation strategies [23,24,26] can effectively enhance an algorithm's performance to solve optimization problems. However, many efforts focus on finding one optimal solution, so using an ensemble of mutation strategies is rarely done with MMOPs. Specifically, MMOPs contain a more complex functional terrain. Fig. 1 shows the peaks of two different modalities, wherein Modality 1 consists of blue dots (represented as individuals) and a red five-pointed star (represented as optima); Modality 2 consists of a few brown dots (represented by individuals) and a red five-pointed star (represented by the optima). As can be seen from the figure, Modality 1 is relatively steep while Modality 2 is relatively gentle. Obviously, different modalities place higher demands on the search ability of the algorithm. For example, using "DE/rand/1" in Modality 2 may find the optimal solution, but it may have more difficulty finding the optimal solution in Modality 1. Hence, utilizing multiple mutation strategies and employing their different search characteristics may be beneficial to the algorithm.
- In the exploitation phase, local search is an effective method to refine the fitness. Similarly, MMOPs contain multiple global optima, and adopting a single local search strategy may be detrimental to the convergence of the algorithm. However, if multiple local search strategies are used, how can we choose the approximate strategy to help the algorithm evolve? To deal with this problem, DQN is utilized to assist algorithms in selecting local search strategies. DQN can perceive the current state of an individual, discriminate the current behavior through the state, action, and reward function, and guide the next evolution.

3.2. Mutation strategy adaptation

DE with multiple mutation strategies typically behaves differently in dealing with optimization problems. We constructed a candidate strategy pool based on mutation strategies with different characteristics. During the evolution, the algorithm will select a strategy from the candidate pool and execute the mutation operator based on the probability obtained from the previous different mutation strategies producing promising solutions. The better a strategy performed in previous generations, the more likely it is to be selected to generate solutions in the current evolution. In the follows, we select several common strategies from the DE literature to construct the strategy candidate pool.

- "DE/rand/1" has strong exploration capabilities but slow convergence. It is suitable for solving MMOPs because it can conduct a large-scale search throughout the entire region, which is conducive to finding the promising regions.
- "DE/rand-to-best/1" has a fast convergence. It is suitable for solving unimodal problems. For MMOPs, the algorithm can be combined with niching technology to accelerate the convergence to a certain extent.
- The improved strategy [44] uses information from historical individuals to execute the mutation operation and performs well on MMOPs.

"DE/rand/1" has a better exploration ability while "DE/rand-to-best/1"enables the algorithm to accelerate the convergence. Moreover, the improved strategy has advantages in solving MMOPs. Hence, the reason for constructing the candidate pool is that the three strategies have distinct search characteristics, and the use of strategies can show different capabilities in dealing with specific problems at different stages. Therefore, the three mutation strategies included in this work's candidate pool are listed below:

"DE/rand/1":

$$\mathbf{v}_i = \hat{\mathbf{x}}_{r1} + F \cdot (\hat{\mathbf{x}}_{r2} - \hat{\mathbf{x}}_{r3}) \tag{8}$$

"DE/rand-to-best/1":

$$\mathbf{v}_i = \mathbf{x}_i + F \cdot (\hat{\mathbf{x}}_{best} - \mathbf{x}_i) + F \cdot (\hat{\mathbf{x}_{r1}} - \hat{\mathbf{x}}_{r2}) \tag{9}$$

The improved strategy:

$$\mathbf{v}_i = \mathbf{x}_i + F \cdot (\hat{\mathbf{x}}_{r1} - \mathbf{x}_i) + rand \cdot (\hat{\mathbf{x}_{r2}} - \hat{\mathbf{x}}_{r3}) \tag{10}$$

where $\hat{\mathbf{x}}_{r1}, \hat{\mathbf{x}}_{r2}, \hat{\mathbf{x}}_{r3}$ are randomly selected from current population and historical individual archive (refer to Section 3.3 for construction methods), and $r1 \neq r2 \neq r3 \neq i$. $\hat{\mathbf{x}}_{best}$ represents an individual with best fitness in $\mathbf{x}_i$'s neighborhood.

In SA-DQN-DE, each individual chooses a mutation strategy from the candidate pool based on the probability of its success in generating better offspring in previous generations. The probabilities of selecting each strategy in the candidate pool are summed to 1. The probabilities are dynamically adjusted during the evolution based on the performance of different strategies. Suppose the candidate pool has K strategies, the probability of selecting the kth strategy is r_k, $k = 1, 2, \dots, K$. It is worth noting that the selection probability of each strategy at the beginning is $1/K$. The stochastic universal selection method [23] is used to select a mutation strategy for each individual in the population to generate offspring. At the iterations $iter$, after all the generated offspring are evaluated, the number of offspring generated by the kth strategy with better fitness is denoted as $num\_suc_{k,iter}$, while the number of offspring produced by the kth strategy that do not enter the next generation is denoted as $num\_fai_{k,iter}$. Similarly, here we build up success and failure memories, storing this information in a fixed number of previous generations, called the learning generation (LG). Table 1 shows the information contained in the success and failure memories. It is worth noting that if the memories overflow, the earliest recorded information will be discarded so that the new information can be saved in the memories.

According to the information recorded in Table 1, the probability of kth strategy can be calculated as follows:

$$r_{k,iter} = \frac{S_{k,iter}}{\sum_{k=1}^{K} S_{k,iter}} \tag{11}$$

where

$$S_{k,iter} = \frac{\sum_{g=iter-LG}^{iter-1} num\_suc_{k,iter}}{\sum_{g=iter-LG}^{iter-1} num\_suc_{k,iter} + \sum_{g=iter-LG}^{iter-1} num\_fai_{k,iter}} + \epsilon, \quad (k = 1, 2, \dots, K; iter > LG) \tag{12}$$

where $S_{k,iter}$ indicates the success rate within the LG generation. $\epsilon = 0.01$. Obviously, the larger the $r_{k,iter}$, the better the kth strategy performed in the previous evolution, and there should be a larger probability of using it to generate offspring in the next evolution.

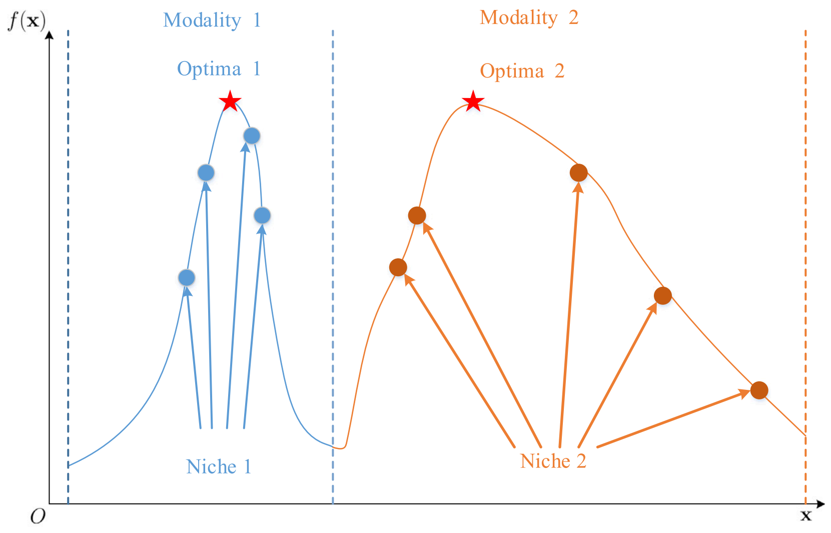


Fig. 1. Different modalities with the same peak.

Table 1
Success and failure memory.

Success memory				Failure memory		
Index	Strategy-1	Strategy-2	Strategy-3	Strategy-1	Strategy-2	Strategy-3
1	$num\_suc_{1,iter-LG}$	$num\_suc_{2,iter-LG}$	$num\_suc_{3,iter-LG}$	$num\_fai_{1,iter-LG}$	$num\_fai_{2,iter-LG}$	$num\_fai_{3,iter-LG}$
2	$num\_suc_{1,iter-LG+1}$	$num\_suc_{2,iter-LG+1}$	$num\_suc_{3,iter-LG+1}$	$num\_fai_{1,iter-LG+1}$	$num\_fai_{2,iter-LG+1}$	$num\_fai_{3,iter-LG+1}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
LG	$num\_suc_{1,iter-1}$	$num\_suc_{2,iter-1}$	$num\_suc_{3,iter-1}$	$num\_fai_{1,iter-1}$	$num\_fai_{2,iter-1}$	$num\_fai_{3,iter-1}$

It is worth noting that SA-DQN-DE also adopts multiple strategies to improve the search efficiency, which has some similarities with SaDE [23], CoDE [25] and EPSDE [26]. The main differences are as follows: (1) SaDE, CoDE and EPSDE mainly seek an optimal solution, while our proposed algorithm finds multiple optimal solutions in one run. (2) The candidate pool constructed by our selected mutation strategies is different from SaDE, CoDE and EPSDE. (3) Historical individuals are saved into archive to help algorithms evolve.

3.3. Construction of history archive

The historical information of individuals can guide the algorithm's search effectively [44]. In general, the individuals discarded during the selection operation are saved to form an individual's historical information. In this section, we screen the individuals stored in the archive, and the main process is as follows:

$$Ac\_rate(i) = val\_ratio(i) \cdot \frac{1}{1+\exp^{-(\beta - T(i))}} \tag{13}$$

$$val\_ratio(i) = 1 - \frac{rank(i)}{NP} \tag{14}$$

where $rank(i)$ represents the ranking of the fitness of the ith individual among the population; $val\_ratio(i)$ is a probability, and the higher the ranking, the greater its value; β is a fixed parameter; $T(i)$ is the number of times that the fitness of the ith individual has not been updated.

The specific construction process is shown in Algorithm 1.

Eq. (13) consists of two parts: one is the probability obtained based on the fitness ranking, and the other is the number of times the fitness has not been updated. Generally, the higher the fitness ranking, the more valuable the individual is. If the number of times $T(i)$ is too large, it indicates that the ith individual may fall into local optima. Eq. (13) indicates that when the fitness ranking is high, if the $T(i)$ is too large, it may be a local optimum, reducing the probability of saving to the archive. If the ranking is high and the $T(i)$ is low, it may be a potential optimal value that can be saved in the archive. In addition, when the ith individual has a low fitness, it indicates that the individual's information is not what the algorithm needs. Regardless of whether the $T(i)$ is high or low, there is a very small chance of it entering the archive. Therefore, the constructed archive can effectively guide the algorithm's search.

Algorithm 1: The construction of history archive

```
Input: poor individual $\mathbf{p}_i$, history archive $\mathcal{A}$, population size $NP$
Output: history archive $\mathcal{A}$
1 if $size(\mathcal{A}) \le 2 * NP$ then
2     save $\mathbf{p}_i$ into $\mathcal{A}$;
3 else if $rand \le Ac\_rate(i)$ then
4     Calculate the distance between $\mathbf{p}_i$ and each individual in $\mathcal{A}$;
5     Find the individual $\mathbf{c}_i$ closest to $\mathbf{p}_i$ from $\mathcal{A}$;
6     if $f(\mathbf{p}_i) \ge f(\mathbf{c}_i)$ then
7         $\mathbf{c}_i = \mathbf{p}_i$;
8         $f(\mathbf{c}_i) = f(\mathbf{p}_i)$;
```

3.4. The setting of DQN

In reinforcement learning (RL), the agent performs an action in an environment and then receives a reward and the next state [45]. The goal is to maximize cumulative rewards for each action performed. RL estimates the value of the action in a given state and returns the policy for the action in a given state. The action function $Q(s,a)$ can be calculated as follows:

$$Q(s,a) = Q(s,a) + \alpha[r(s,a) + \gamma \max Q(s',a') - Q(s,a)] \tag{15}$$

where $r(s,a)$ is the reward. $\max Q(s',a')$ is the maximum long-term cumulative reward $Q(s',a')$ in the next state s' and action a'. γ is the discount factor and α is a constant in [0,1].

Many RL methods employ Q tables to store state and action, and then calculate value function from the information in Q tables. However, if the state set is large, it will take a lot of time to find and store the Q table. Hence, deep RL in which deep neural networks replace value functions is proposed to solve this problem. DQN is a deep RL can use neural networks to obtain nonlinear Q value functions of state features.

In the proposed algorithm, agents can autonomously choose local search strategies in the current environment through training the DQN. The DQN contains state, action, and reward. We will introduce them in detail.

(1) Network structure: The network includes some full connection layers: $fc_1(In, 32)$, $fc_2(32, 16)$, $fc_3(16, Out)$ where In is the length of a transaction and Out is the number of actions. The hidden layers between each fc layer are ±ReLU±. Transaction can be shown as:

$$transaction = (s, a, r_{s,a}, s') \tag{16}$$

(2) State definition: In DQN, the state consists primarily of each individual's information, which can be shown as

$$state = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{num}\} \tag{17}$$

where num indicates the size of the experience pool.

(3) Action definition: "DE/best/1", "DE/best/2", and Eq. (10) are selected to form the action space, and guide the algorithm to conduct local search in the neighborhood. It can be illustrated as follows:

$$action = \{DE/best/1, DE/best/2, \text{Eq. (10)}\} \tag{18}$$

(4) Reward definition: After executing the action, the agent measures whether its behavior is reasonable based on the reward value.

$$reward = \begin{cases} 10, & \text{if } f(\mathbf{u}_i) \geq f(\mathbf{x}_i) \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

The operator selection process for implementing the algorithm using DQL (shown in Fig. 2) is as follows:

- During evolution, the algorithm selects three different mutation strategies (represented by three cuboid of different colors in the figure) to generate offspring.
- In the interaction process, the agent and the environment exchange interactions. In short, the agent gives the environment an action (*i.e.*, mutation strategy), and the environment gives feedback (*i.e.*, reward) on the effect of that action.
- After receiving feedback, the agent learns and improves the strategy in the subsequent process. Firstly, the current record is saved to the experience pool. Secondly, the training data sampled from the experience pool is trained using DQN. The input of DQN is the transaction information and the output is the Q value of the local search strategies adopted in this state.
- According to the $Q-$ value of all actions estimated by DQN, the agent chooses the action with the largest $Q-$ value.

3.5. Proposed SA-DQN-DE framework

In this work, mutation strategy adaptation and DQN are used to assist the algorithm in operation selection to enhance the overall performance of SA-DQN-DE. The pose-code of the algorithm is listed in Algorithm 2. NP is the population size; FES represents the current number of fitness evaluation and Max_FES is the max FES; $\mathcal{P}$ is populations. $\mathcal{A}$ is the individual's historical archive and $\mathcal{A}'$ saves the found roots.

Lines 7–10 calculate the probability of each mutation strategy being selected. In lines 11–14, if the algorithm is in the early stage of evolution, a strategy is assigned to each individual using stochastic universal sampling; otherwise, the mutation operator is selected by DQN to refine the fitness. In lines 17–19, the algorithm uses mutation strategies and the historical information archive to generate offspring. Lines 21–29 compare fitness and update population or historical archive.

Algorithm 2: The framework of SA-DQN-DE

```
   Input: Control parameters: $NP$, $FES$, $Max\_FES$
   Output: The final archive $\mathcal{A}'$
1  Set $FES = 0, iter = 1$ and the archive $\mathcal{A}' = \varnothing$;
2  Set $F = 0.9, CR = 0.3, T = \varnothing$;
3  Randomly generate the population $\mathcal{P}$ ;
4  Evaluate the fitness of the population;
5  $FES = FES + NP$;
6  while $FES < Max\_FES$ do
7      if $iter > LG$ then
8          for $k = 1 : K$ do
9              Calculate the probability $p_{k,iter}$ via Eq. (11);
10             Delete $num\_suc_{k,iter-LG}$ and $num\_fai_{k,iter-LG}$ out of Success
                and Failure Memory;
11     if $FES > 0.9 * Max\_FES$ then
12         Apply DQN to choose one strategy $k$ for each individual;
13     else
14         Employ stochastic universal sampling to choose one strategy $k$
            for each individual;
15     for $i = 1 : NP$ do
16         Use Eq. (13) to calculate $Ac\_rate$;
17         Form a new population $\mathcal{P}'$ by combining population $\mathcal{P}$ and
            historical archive $\mathcal{A}$;
18         Find $n$ individuals in $\mathcal{P}'$ that are closest to $\mathbf{x}_i$;
19         Use $k-$th strategy and crowding technique to generate the
            offspring $\mathbf{u}_i$;
20         if $f(\mathbf{u}_i) \geq f(\mathbf{x}_i)$ then
21             $\mathbf{x}_i = \mathbf{u}_i$;
22             $f(\mathbf{x}_i) = f(\mathbf{u}_i)$;
23             $T(i) = 0$;
24             $num\_suc_{k,iter} = num\_suc_{k,iter} + 1$;
25         else
26             if $rand < Ac\_rate(i)$ then
27                 Use Algorithm 1 to update $\mathcal{A}$;
28             $T(i) = T(i) + 1$;
29             $num\_fai_{k,iter} = num\_fai_{k,iter} + 1$;
30     Save $num\_suc_{k,iter}$ and $num\_fai_{k,iter}$ into the Success and Failure
        Memory;
31     Find the convergent individuals in $\mathcal{P}$;
32     Save these individuals in $\mathcal{A}'$;
33     Re-initialize these individuals and evaluate them;
34     Update $FES$;
35     $iter = iter + 1$;
```

The flowchart of SA-DQN-DE is given in Fig. 3. Generally, if the $FES \leq 0.9 * Max\_FES$, it defaults to being in the exploration stage, and adopts strategy adaptation to generate offspring. If $FES > 0.9 * Max\_FES$, the algorithm enters the exploitation phase. DQN assists the algorithm in selecting suitable local search strategies to refine fitness. In addition, after evaluating the fitness, the historical individual archive is updated to assist in the algorithm's evolution and improve its search performance.

3.6. Computational complexity

In this paper, SA-DQN-DE is composed of the operator selection, neural network training and an archive update to generate the offspring. The time complexity of the selected operation part is $\mathcal{O}(NP * log(NP))$. The training data set size is $2 * NP$; the number of neurons per layer is a, and the number of training iterations is $iter\_train$. The training time complexity of the DQN algorithm is $\mathcal{O}(2 * NP * a^2 * iter\_train)$, where a^2 denotes the number of connections between neurons. The complexity of the archive update is $\mathcal{O}(NP * log(NP))$. So the

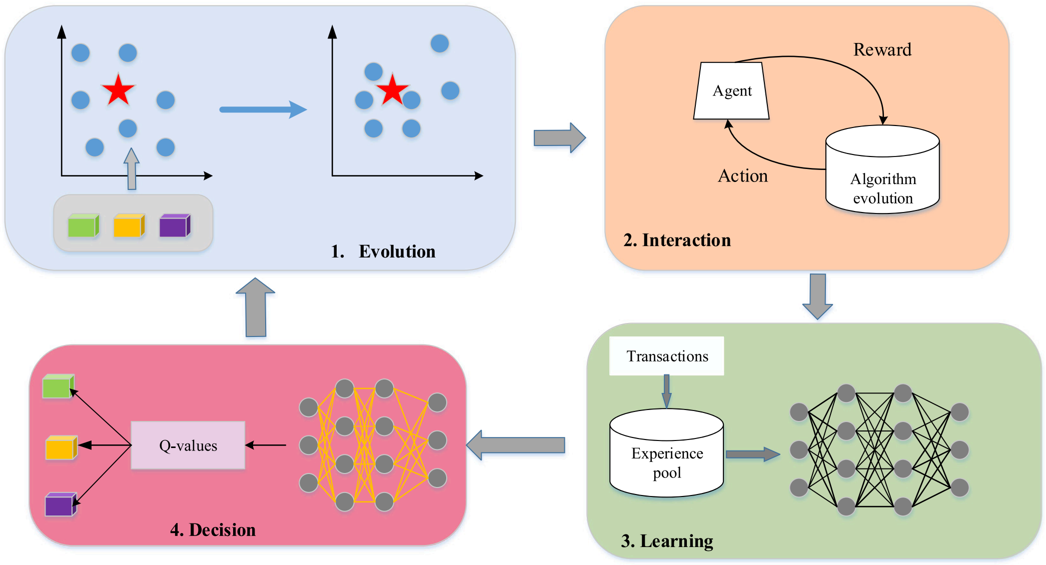


Fig. 2. Training process of DQN.

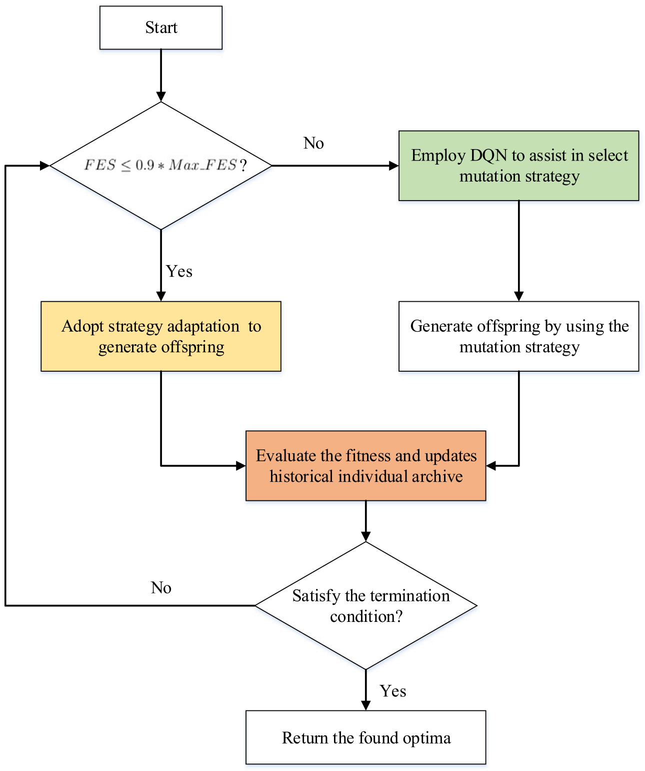

Start
$FES \leq 0.9 * Max\_FES$?
No
Employ DQN to assist in select mutation strategy
Yes
Adopt strategy adaptation to generate offspring
Generate offspring by using the mutation strategy
Evaluate the fitness and updates historical individual archive
Satisfy the termination condition?
No
Yes
Return the found optima

Fig. 3. The flowchart of SA-DQN-DE.

time complexity of the whole algorithm is $\mathcal{O}(NP * log(NP)) + \mathcal{O}(NP * a^2 * iter\_train)$.

4. Experimental results and analysis

In this section, we first introduce benchmark functions and performance criteria; Then, the results of the proposed SA-DQN-DE algorithm are compared with other advanced algorithms. Finally, we analyze the effects of different components and parameters on SA-DQN-DE. Moreover, all of the algorithms are implemented in MATLAB R2022b on a desktop PC with an Intel Core i7-10700U processor @ 2.90 GHz, 16 GB RAM, and the Windows 10 64-bit operating system.

4.1. Benchmark problems and performance criteria

To verify the performance of SA-DQN-DE, numerical experiments were executed on 20 test functions included in CEC2013. Two performance criteria were used to evaluate the effectiveness of multimodal optimization algorithms: peak ratio (PR) and success rate (SR). PR calculates the proportion of algorithms finding multiple global optimal solutions in multiple runs. SR calculates the proportion of all global optima found in multiple runs. PR and SR are expressed as follows:

$$PR = \frac{\sum_{i=1}^{NR} N_{i,o}}{NoG \cdot NR} \tag{20}$$

$$SR = \frac{NS}{NR} \tag{21}$$

where NR represents the total number of runs; $N_{i,o}$ is the number of the found global optima in the ith run; NoG is the known global optima of MMOPs. NS denotes the number of successful runs.

Moreover, the parameters are set as follows:

- In the DE algorithm, $CR = 0.9, F = 0.3$. Neighborhood size is 10, $LG = 50,\ \beta = 20$;
- For DQN, $\gamma = 0.9, \alpha = 0.5$;
- For historical individual archive, the archive size is equal to $2 * NP$.

4.2. Results with different accuracies

Table 2 records the PR and SR achieved by SA-DQN-DE with different accuracies:

- F1-F5, these functions have low dimensions and contain relatively few optima, so SQ-DQN-DE can successfully find all optima with different accuracies (*i.e.*, PR and SR are both equal to 1).
- F6-F10, the dimension of these functions are low but contain many optima. SA-DQN-DE can successfully solve F6 and F10 with five accuracy levels. Although our algorithm could not locate all the optima, it still found most of them. This demonstrates the superiority of SA-DQN-DE.

Table 2
PR and SR results at different accuracies.

Accuracy level	F1		F2		F3		F4		F5	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1.00E−01	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1.00E−02	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1.00E−03	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1.00E−04	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1.00E−05	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Accuracy level	F6		F7		F8		F9		F10	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1.00E−01	1.000	1.000	1.000	1.000	1.000	1.000	0.944	0.000	1.000	1.000
1.00E−02	1.000	1.000	1.000	1.000	1.000	1.000	0.935	0.000	1.000	1.000
1.00E−03	1.000	1.000	1.000	1.000	1.000	1.000	0.931	0.000	1.000	1.000
1.00E−04	1.000	1.000	1.000	1.000	0.998	0.800	0.927	0.000	1.000	1.000
1.00E−05	1.000	1.000	0.989	0.899	0.933	0.667	0.875	0.000	1.000	1.000
Accuracy level	F11		F12		F13		F14		F15	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1.00E−01	1.000	1.000	1.000	1.000	1.000	1.000	0.922	0.750	1.000	1.000
1.00E−02	1.000	1.000	1.000	1.000	1.000	1.000	0.880	0.300	0.880	0.000
1.00E−03	1.000	1.000	1.000	1.000	1.000	1.000	0.880	0.300	0.790	0.000
1.00E−04	1.000	1.000	0.992	0.933	1.000	1.000	0.878	0.267	0.750	0.000
1.00E−05	1.000	1.000	0.985	0.900	1.000	1.000	0.833	0.160	0.675	0.000
Accuracy level	F16		F17		F18		F19		F20	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1.00E−01	0.697	0.000	0.773	0.000	0.667	0.000	0.549	0.000	0.523	0.000
1.00E−02	0.667	0.000	0.708	0.000	0.667	0.000	0.544	0.000	0.497	0.000
1.00E−03	0.667	0.000	0.688	0.000	0.667	0.000	0.544	0.000	0.450	0.000
1.00E−04	0.667	0.000	0.683	0.000	0.667	0.000	0.500	0.000	0.400	0.000
1.00E−05	0.667	0.000	0.633	0.000	0.667	0.000	0.424	0.000	0.345	0.000

- F11-F15, these are low dimensional composite functions. SA-DQN-DE can locate all the optima of F11 and F13 at different accuracy levels. In addition, it can successfully find all the optima solutions of F12 at 1.00E−01, 1.00E−02, and 1.00E−03 accuracy. Moreover, our approach is able to solve F15 at 1.00E−01 accuracy. However, SA-DQN-DE could not successfully find all optimal solutions for F14 with five accuracy levels.
- F16-F20, they are high-dimensional composite functions. It is clear that SA-DQN-DE could not successfully identify all optima for these functions with five accuracy levels.

4.3. Influence of different components in SA-DQN-DE

This section will explain the impact of different parts on the SA-DQN-DE.

- SA-DQN-DE/1, in which strategy adaptation was removed from SA-DQN-DE and replaced by a random selection strategy.
- SA-DQN-DE/2, where DQN was removed from SA-DQN-DE.
- SA-DQN-DE/3, in which the historical archive was removed from SA-DQN-DE.

Table 3 demonstrates the detailed PR and SR results achieved by SA-DQN-DE variants in 20 test functions. Compared to its variants, SA-DQN-DE obtained the best PR value in 18 out of 20 test functions (bprs). SA-DQN-DE/1, SA-DQN-DE/2, and SA-DQN-DE/3 achieved the best results in 12, 11, and 10 functions, respectively. Next, we summarize the performance of the SA-DQN-DE variants:

(1) SA-DQN-DE/1: it successfully identified all the optima on F1-F7, F10-F12. Specially, SA-DQN-DE/1 performed better in F9 and F12 than SA-DQN-DE. This shows that the random selection strategy has certain advantages in solving several functions. But overall, strategy adaptation still achieved better results in many functions, such as F8, F13-F15, F17, F19-F20.

(2) SA-DQN-DE/2: it lacks the DQN auxiliary selection local search operator. In particular, F9 contains more optima, and without DQN, PR value drops significantly. In addition, in solving F8, F12-F15, F17, F19-F20, the obtained PR value has some decrease.

(3) SA-DQN-DE/3: It can be seen that without the assistance of historical information archiving, SA-DQN-DE/3 showed some decrease in PR value when solving the functions of F8-F9 and F12-F20. This also demonstrates the effectiveness of the archive.

Hence, strategy adaptation, DQN, and historical information archive can work together, and it is an alternative way to solve MMOPs.

4.4. Comparison algorithms

In this section, 11 approaches are selected for comparison. These are the PSO-based method (LIPS [46], R3PSO [47], and NMMSO [48]), the DE-based method (Self-CCDE [15], LoICDE [17], PNPCDE [16], AED-DDE [35], LBPADE [10], NBC-DE [36], and PRMDE [49]), and the multi-objective method ([42]). These algorithms are widely used in the research community and have shown promising results in solving MMOPs. All the algorithms are described in Section 2. The results of several algorithms were taken from their original papers, while the results of traditional algorithms are derived from [12]. In order to make a fair comparison, all the comparison algorithms use the same maximum number of fitness evaluations.

From Table 4, SA-DQN-DE has the best PR results for 17 test problems when solving MMOPs. The specific results are analyzed as follows:

- F1-F7: Due to the relatively simple difficulty of testing functions, most algorithms performed well on F1-F7. Specially, SA-DQN-DE and MOMMOP could determine all the optima ($PR = 1, SR = 1$). Other algorithms lost several global optimal solutions in these seven test functions.
- F8-F9: They contain 81 and 216 optimal solutions respectively, which put forward high requirements on the search performance of the algorithm. From the results, MOMMOP obtained all the optimal solutions of these two functions while PRMDE could find all optima of F8. Although SA-DQN-DE lost some optimal

Table 3
The effect of different parts on SA-DQN-DE.

Function	SA-DQN-DE		SA-DQN-DE/1		SA-DQN-DE/2		SA-DQN-DE/3	
Index	PR	SR	PR	SR	PR	SR	PR	SR
F1	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F2	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F3	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F4	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F5	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F6	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F7	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F8	**0.998**	0.800	0.993	0.600	0.988	0.333	0.996	0.733
F9	0.928	0.000	**0.934**	0.000	0.839	0.000	0.927	0.000
F10	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F11	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F12	0.992	0.933	**1.000**	1.000	0.967	0.733	0.992	0.933
F13	**1.000**	1.000	0.967	0.800	0.944	0.733	0.956	0.733
F14	**0.878**	0.267	0.811	0.200	0.811	0.133	0.833	0.167
F15	**0.750**	0.000	0.742	0.000	0.738	0.000	0.720	0.000
F16	**0.667**	0.000	**0.667**	0.000	**0.667**	0.000	**0.667**	0.000
F17	**0.683**	0.000	0.635	0.000	0.630	0.000	0.630	0.000
F18	**0.667**	0.000	**0.667**	0.000	**0.667**	0.000	0.657	0.000
F19	**0.500**	0.000	0.453	0.000	0.431	0.000	0.348	0.000
F20	**0.400**	0.000	0.355	0.000	0.300	0.000	0.325	0.000
bprs	**18**		12		11		10	

Table 4
The detailed results of different algorithms on the CEC2013.

Function	SA-DQN-DE		LIPS		R3PSO		Self-CCDE		LoICDE		PNPCDE	
Index	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	**1.000**	1.000	0.892	0.824	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F2	**1.000**	1.000	**1.000**	1.000	0.992	0.961	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F3	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F4	**1.000**	1.000	0.970	0.960	0.971	0.902	**1.000**	1.000	0.940	0.882	**1.000**	1.000
F5	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F6	**1.000**	1.000	0.209	0.000	0.662	0.000	0.928	0.431	**1.000**	1.000	0.559	0.000
F7	**1.000**	1.000	0.537	0.000	0.518	0.000	0.875	0.020	0.691	0.000	0.887	0.000
F8	0.998	0.800	0.081	0.000	0.703	0.000	0.997	0.863	0.000	0.000	0.000	0.000
F9	0.927	0.000	0.196	0.000	0.208	0.000	0.457	0.000	0.115	0.000	0.470	1.000
F10	**1.000**	1.000	0.755	0.000	0.869	0.137	**1.000**	1.000	**1.000**	1.000	**1.000**	**1.000**
F11	**1.000**	1.000	0.944	0.840	0.500	0.000	0.771	0.118	0.660	0.000	0.634	0.000
F12	0.991	0.933	0.586	0.000	0.537	0.000	0.375	0.000	0.529	0.000	0.000	0.000
F13	**1.000**	1.000	0.778	0.118	0.654	0.000	0.663	0.000	0.523	0.000	0.438	0.000
F14	**0.878**	0.267	0.722	0.000	0.667	0.000	0.657	0.000	0.660	0.000	0.330	0.000
F15	**0.758**	0.000	0.348	0.000	0.206	0.000	0.358	0.000	0.314	0.000	0.029	0.000
F16	**0.667**	0.000	0.281	0.000	0.451	0.000	0.657	0.000	0.546	0.000	0.000	0.000
F17	**0.683**	0.000	0.164	0.000	0.120	0.000	0.250	0.000	0.248	0.000	0.000	0.000
F18	**0.667**	0.000	0.111	0.000	0.092	0.000	0.327	0.000	0.216	0.000	0.154	0.000
F19	**0.500**	0.000	0.000	0.000	0.000	0.000	0.103	0.000	0.049	0.000	0.000	0.000
F20	**0.400**	0.000	0.000	0.000	0.000	0.000	0.052	0.000	0.123	0.000	0.000	0.000
bprs	**17**		3		3		6		6		6	
Function	MOMMOP		NMMSO		AED-DDE		LBPADE		NDC-DE		PRMDE	
Index	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F2	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F3	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F4	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F5	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F6	**1.000**	1.000	0.997	0.940	**1.000**	1.000	**1.000**	1.000	0.999	0.960	**1.000**	1.000
F7	**1.000**	1.000	**1.000**	1.000	0.838	0.039	0.889	0.000	0.881	0.000	0.988	0.686
F8	**1.000**	1.000	0.981	0.180	0.747	0.000	0.575	0.000	0.941	0.000	**1.000**	1.000
F9	**1.000**	1.000	0.917	0.000	0.384	0.000	0.470	0.000	0.458	0.000	0.706	0.000
F10	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F11	0.712	0.000	**1.000**	1.000	**1.000**	1.000	0.674	0.000	**1.000**	1.000	0.997	0.980
F12	0.936	0.529	0.998	0.980	**1.000**	1.000	0.750	0.000	0.941	0.520	**1.000**	1.000
F13	0.667	0.000	0.990	0.940	0.686	0.000	0.667	0.000	**1.000**	1.000	0.667	0.000
F14	0.667	0.000	0.710	0.000	0.667	0.000	0.667	0.000	0.833	0.160	0.667	0.000
F15	0.618	0.000	0.670	0.000	0.637	0.000	0.654	0.000	0.753	0.000	0.615	0.000
F16	0.647	0.000	0.660	0.000	**0.667**	0.000	**0.667**	0.000	**0.667**	0.000	**0.667**	0.000
F17	0.502	0.000	0.538	0.000	0.375	0.000	0.532	0.000	0.633	0.000	0.507	0.000
F18	0.493	0.000	0.633	0.000	0.654	0.000	**0.667**	0.000	**0.667**	0.000	**0.667**	0.000
F19	0.221	0.000	0.447	0.000	0.375	0.000	0.475	0.000	0.412	0.000	0.453	0.000
F20	0.125	0.000	0.178	0.000	0.250	0.000	0.275	0.000	0.355	0.000	0.252	0.000
bprs	9		8		10		9		10		11	

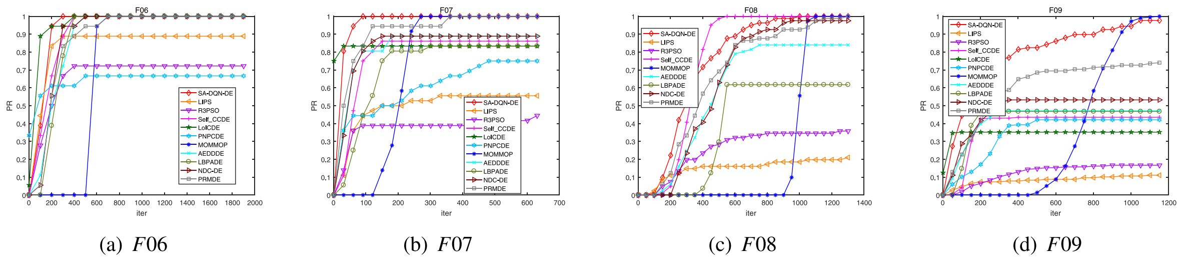

(a) $F06$ (b) $F07$ (c) $F08$ (d) $F09$

Fig. 4. Convergence curves of different algorithms on F6-F9.

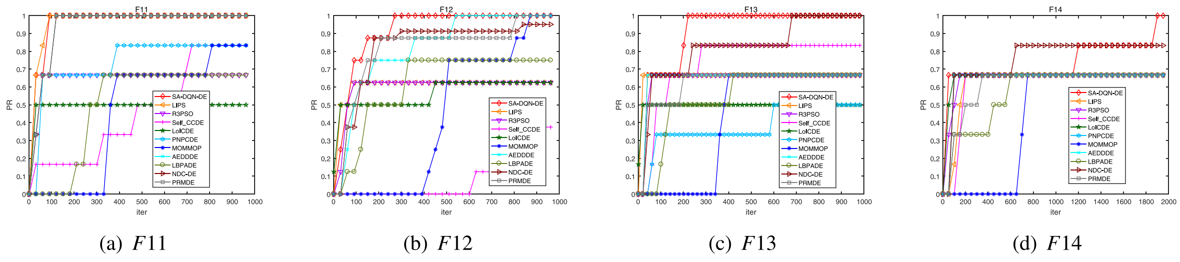

(a) $F11$ (b) $F12$ (c) $F13$ (d) $F14$

Fig. 5. Convergence curves of different algorithms on F11-F14.

solutions, the PR value obtained is relatively high compared with other algorithms. Therefore, it had better search performance.

- F10-F15: For F10, except for LIPS and R3PSO, all algorithms found all optimal solutions. For F11, SA-DQN-DE, NMMSO, AED-DDE, and NDC-DE successfully located all the optima. For F12, AED-DDE and PRMDE obtained the best PR, followed by SA-DQN-DE. For F13-F15, SA-DQN-DE performed well on these three test functions.
- F16-F20: They have high dimensions and are the most complex functions. None of the algorithms successfully obtained all of the optimal solutions for these functions. Even so, compared to other algorithms, SA-DQN-DE obtained the best PR values when solving these five functions. This also indicates that our method has superior search performance.

The Friedman and Wilcoxon test results are shown in Tables 5 and 6. From Table 5, SA-DQN-DE obtained the best ranking in terms of PR and SR. Additionally, Table 6 illustrates that SA-DQN-DE has significant differences in PR since $p-$ values are less than 0.05.

Overall, SA-DQN-DE is an effective algorithm for solving MMOPs. However, it also has several drawbacks, such as performing worse than some algorithms in F8, F9, and F12. Therefore, in the future, we need to further modify the algorithm to enhance its performance.

4.5. Convergence analysis

This section draws conclusions about the convergence of multimodal algorithms during evolutionary. Due to the ease of solving F1-F5 and F10, this section does not draw convergence for multimodal algorithms on these functions. In addition, several algorithms are not shown in some graphs (with PR values equal to 0) to make the presentation more intuitive, such as R3PSO, PNPCDE, and LIPS without drawing convergence in F19 and F20. The convergence analysis is as follows:

- From Figs. 4(a) and 4(b), SA-DQN-DE has found all the optimal solutions and has the fastest convergence. From Fig. 4(c), the convergence of SA-DQN-DE is not as fast as that of Self_CCDE, and a small number of optimal solutions are lost. In Fig. 4(d), although the convergence of SA-DQN-DE is always the fastest in the early stage, it is exceeded by the MOMMOP algorithm in the later stage, and a small number of optimal values are lost.
- From Fig. 5, SA-DQN-DE achieved the best performance and the fastest convergence. Specifically, in Fig. 5(d), there is a significant increase in PR value in the later stages, indicating the effectiveness of DQN strategy selection.
- From Fig. 6, it is clear that SA-DQN-DE achieved the best performance and the fastest convergence on these six test functions.

5. Discussion

5.1. Effect of neighborhood size

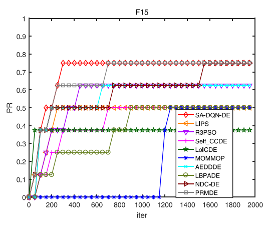
SA-DQN-DE adopts the crowding technique to maintain population diversity, but involves parameters of neighborhood size. In the original algorithm, neighborhood size was set to 10. In order to verify the impact of different neighborhood sizes on the algorithm's performance, we set the neighborhood sizes to 5, 15, and 20to study them. Specially, the neighborhood size of SA-DQN-DE/4 is 5, SA-DQN-DE/5 is 15, and SA-DQN-DE/6 is 20.

Table 7 lists the PR and SR obtained by different neighborhood sizes. It can be seen that as the size of the neighborhood increases, the average PR value obtained first increases and then decreases (0.846 → 0.873 → 0.861 → 0.859). This shows that if the neighborhood size is small, it is not conducive to the algorithm to fully explore the terrain, and it is easy to fall into local optimal. However, if the neighborhood size is large, although it can explore a larger region, the convergence performance of the algorithm is weakened, resulting in the loss of some optimal solutions. Therefore, we believe that it is feasible to set the neighborhood size between [10, 15] when solving MMOPs.

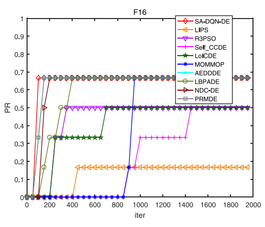
5.2. Effect of LG

SA-DQN-DE includes the parameter LG, and the algorithm can calculate the next selection probability based on the performance of the mutation strategy in the LG generation's evolution. This section will investigate the impact of different LG's on algorithm performance. $LG = 10, LG = 20, LG = 30, LG = 40, LG = 60$ are represented by SA-DQN-DE/7, SA-DQN-DE/8, SA-DQN-DE/9, SA-DQN-DE/10, and SA-DQN-DE/11, respectively.

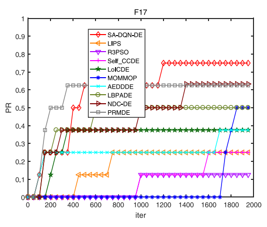
The PR and SR results are provided in Table 8. The mean of PR of SA-DQN-DE with various LG are 0.866, 0.868, 0.868, 0.871, 0.873, and 0.864, respectively. Hence, SA-DQN-DE obtained the best average



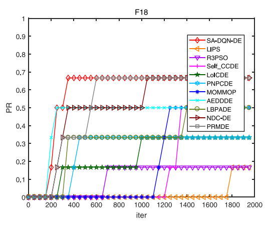
(a) $F15$



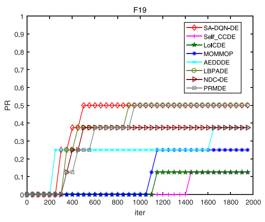
(b) $F16$



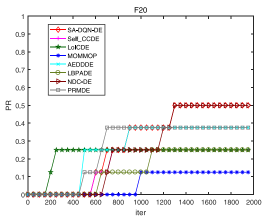
(c) $F17$



(d) $F18$



(e) $F19$



(f) $F20$

Fig. 6. Convergence curves of different algorithms on F11-F14.

Table 5
Average rankings of different algorithms.

Algorithm	Ranking (RR)	Ranking (SR)
SA-DQN-DE	**3.275**	**5.000**
LIPS	9.100	7.775
R3PSO	9.800	8.175
Self-CCDE	7.525	6.600
LoICDE	8.725	7.300
PNPCDE	9.225	7.350
MOMMOP	5.625	5.650
NMMSO	4.700	5.625
AED-DDE	5.550	5.950
LBPADE	5.350	6.950
NDC-DE	4.450	5.950
PRMDE	4.675	5.675

Table 6
Results achieved by the Wilcoxon test.

VS	RR			SR		
	R^+	R^-	p-value	R^+	R^-	p-value
LIPS	198.5	11.5	**1.55E−04**	182.5	27.5	**3.00E−03**
R3PSO	198.5	11.5	**1.44E−04**	182.5	27.5	**2.25E−03**
Self-CCDE	199.5	10.5	**3.90E−04**	148.0	62.0	*8.00E−02*
LoICDE	199.5	10.5	**3.90E−04**	161.0	49.0	**2.03E−02**
PNPCDE	199.5	10.5	**3.90E−04**	156.0	54.0	**1.18E−02**
MOMMOP	171.0	39.0	**1.30E−02**	123.5	86.5	*4.69E−01*
NMMSO	181.5	28.5	**3.71E−03**	132.5	77.5	*2.57E−01*
AED-DDE	173.0	37.0	**5.84E−03**	132.5	77.5	*2.59E−01*
LBPADE	182.0	28.0	**1.82E−03**	157.5	52.5	**4.37E−02**
NDC-DE	170.5	39.5	**1.28E−02**	147.5	62.5	*8.35E−02*
PRMDE	162.0	48.0	**2.06E−02**	124.5	85.5	*4.55E−01*

PR when $LG = 50$. In addition, no matter what LG is, the PR values are unchanged on F1-F7, which demonstrates that F1-F7 are insensitive to LG in SA-DQN-DE. Moreover, SA-DQN-DE obtained the best PR on F8, F12, and F13. SA-DQN-DE/10 has the best PR on F8, F9, F13, F17, and F20. According to the results, we suggest that LG is set to a number in $[40, 50]$.

6. Conclusion

In this work, we have proposed SA-DQN-DE, which utilizes strategy adaptation and DQN to assist in mutation operation selection, for solving MMOPs. First, the strategy adaptation adopts multiple mutation strategies to form a strategy pool and determines the selection probability based on the performance of different mutation strategies in the previous evolution to guide the algorithm in the next operation. Second, during the exploitation phase, DQN is able to perceive individual states and set three local search strategies as actions. Based on the reward values obtained from these actions after local search operations, the algorithm uses appropriate actions to refine the fitness. Moreover, an enhanced historical individual archive is designed to utilizes historical information to effectively improve the search performance of the algorithm. Experimental results on CEC2013 demonstrate the proposed algorithm outperformed eleven peer multimodal EAs.

Nevertheless, there are still some problems to be solved. On the one hand, SA-DQN-DE lost some optimal solutions in both F8 and F9, indicating that the optima-finding ability of the algorithm needs to be further improved. There are two approaches: (1) more mutation strategies with different search characteristics can be integrated and they can

Table 7
The detailed results obtained by different neighborhood size.

Function	SA-DQN-DE		SA-DQN-DE/4		SA-DQN-DE/5		SA-DQN-DE/6	
Index	PR	SR	PR	SR	PR	SR	PR	SR
F1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F3	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F4	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F5	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F6	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F7	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F8	**0.998**	0.800	0.953	0.000	0.963	0.000	0.928	0.000
F9	**0.928**	0.000	0.887	0.000	0.895	0.000	0.902	0.000
F10	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F11	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F12	0.992	0.933	0.896	0.167	0.938	0.500	**1.000**	1.000
F13	**1.000**	1.000	0.917	0.500	0.917	0.500	0.889	0.500
F14	**0.878**	0.267	0.778	0.000	0.861	0.167	0.778	0.000
F15	**0.750**	0.000	0.604	0.000	**0.750**	0.000	**0.750**	0.000
F16	0.667	0.000	0.667	0.000	0.667	0.000	0.667	0.000
F17	0.683	0.000	0.646	0.000	**0.729**	0.000	0.708	0.000
F18	0.667	0.000	0.667	0.000	0.667	0.000	0.667	0.000
F19	0.500	0.000	**0.521**	0.000	0.500	0.000	0.500	0.000
F20	**0.400**	0.000	0.396	0.000	0.333	0.000	0.396	0.000
Avg.	**0.873**		0.846		0.861		0.859	

Table 8
The detailed results obtained by different *LG*s.

Function	SA-DQN-DE		SA-DQN-DE/7		SA-DQN-DE/8		SA-DQN-DE/9		SA-DQN-DE/10		SA-DQN-DE/11	
Index	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F3	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F4	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F5	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F6	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F7	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F8	**0.998**	**0.800**	0.994	0.667	0.993	0.500	0.996	0.667	**0.998**	**0.800**	0.996	0.667
F9	0.927	0.000	0.922	0.000	0.922	0.000	0.901	0.000	**0.934**	0.000	0.933	0.000
F10	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F11	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F12	**0.991**	**0.933**	0.982	0.833	0.979	0.833	0.979	0.833	0.979	0.833	**0.991**	**0.933**
F13	**1.000**	**1.000**	0.972	0.833	**1.000**	**1.000**	0.944	0.667	**1.000**	**1.000**	0.972	0.833
F14	0.878	0.267	0.806	0.167	0.806	0.167	**0.889**	**0.333**	0.806	0.167	0.750	0.167
F15	0.750	0.000	0.750	0.000	0.750	0.000	0.750	0.000	0.750	0.000	0.750	0.000
F16	0.667	0.000	0.667	0.000	0.667	0.000	0.667	0.000	0.667	0.000	0.667	0.000
F17	0.683	0.000	0.667	0.000	0.667	0.000	0.667	0.000	**0.708**	0.000	0.667	0.000
F18	0.667	0.000	0.667	0.000	0.667	0.000	0.667	0.000	0.667	0.000	0.667	0.000
F19	0.500	0.000	0.500	0.000	0.500	0.000	0.500	0.000	0.500	0.000	0.500	0.000
F20	0.400	0.000	0.400	0.000	0.400	0.000	0.396	0.000	**0.417**	0.000	0.396	0.000
Avg.	**0.873**		0.866		0.868		0.868		0.871		0.864	

be used to guide the algorithm operation; (2) multiple EAs can be integrated, including DE, PSO, ACO, and so on. Different algorithms have different search features, and perhaps they can complement each other. The proposed method is applied to deal with real-world problems, such as distributed flexible job shop scheduling [50].

CRediT authorship contribution statement

Zuowen Liao: Writing – original draft, Resources, Formal analysis, Data curation. **Qishuo Pang:** Writing – review & editing, Methodology. **Qiong Gu:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] D.-K. Woo, J. Choi, M. Ali, H.-K. Jung, A novel multimodal optimization algorithm applied to electromagnetic optimization, Magn., IEEE Trans. 47 (2011) 1667–1673.

[2] A.Y. Goharrizi, R. Singh, A.M. Gole, S. Filizadeh, J.C. Muller, R.P. Jayasinghe, A parallel multimodal optimization algorithm for simulation-based design of power systems, IEEE Trans. Power Deliv. 30 (5) (2015) 2128–2137.

[3] K. Deb, S. Gulati, Design of truss-structures for minimum weight using genetic algorithms, Finite Elements Anal. Des. 37 (5) (2001) 447–465.

[4] Z.-J. Wang, Z.-H. Zhan, Y. Li, S. Kwong, S.-W. Jeon, J. Zhang, Fitness and distance based local search with adaptive differential evolution for multimodal optimization problems, IEEE Trans. Emerg. Top. Comput. Intell. 7 (3) (2023) 684–699.

[5] R. Li, W. Gong, L. Wang, C. Lu, C. Dong, Co-evolution with deep reinforcement learning for energy-aware distributed heterogeneous flexible job shop scheduling, IEEE Trans. Syst. Man Cybern.: Syst. (2023) 1–11, http://dx.doi.org/10.1109/TSMC.2023.3305541.

[6] R. Li, W. Gong, L. Wang, C. Lu, X. Zhuang, Surprisingly popular-based adaptive memetic algorithm for energy-efficient distributed flexible job shop scheduling, IEEE Trans. Cybern. (2023) 1–11, http://dx.doi.org/10.1109/TCYB.2023.3280175.

[7] X. Li, M.G. Epitropakis, K. Deb, A. Engelbrecht, Seeking multiple solutions: An updated survey on niching methods and their applications, IEEE Trans. Evol. Comput. 21 (4) (2017) 518–538.

[8] B. Qu, P. Suganthan, J. Liang, Differential evolution with neighborhood mutation for multimodal optimization, IEEE Trans. Evol. Comput. 16 (5) (2012) 601–614.

[9] Z. Liao, W. Gong, L. Wang, X. Yan, C. Hu, A decomposition-based differential evolution with reinitialization for nonlinear equations systems, Knowl.-Based Syst. 191 (2020) 105312.

[10] H. Zhao, Z.-H. Zhan, Y. Lin, X. Chen, X.-N. Luo, J. Zhang, S. Kwong, J. Zhang, Local binary pattern-based adaptive differential evolution for multimodal optimization problems, IEEE Trans. Cybern. 50 (7) (2020) 3343–3357.

[11] Y.-H. Zhang, Y.-J. Gong, H.-X. Zhang, T.-L. Gu, J. Zhang, Toward fast niching evolutionary algorithms: A locality sensitive hashing-based approach, IEEE Trans. Evol. Comput. 21 (3) (2017) 347–362.

[12] Z. Liao, X. Mi, Q. Pang, Y. Sun, History archive assisted niching differential evolution with variable neighborhood for multimodal optimization, Swarm Evol. Comput. 76 (2023) 101206.

[13] R. Thomsen, Multimodal optimization using crowding-based differential evolution, in: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753), Vol. 2, 2004, pp. 1382–1389.

[14] X. Li, Efficient differential evolution using speciation for multimodal function optimization, in: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, Association for Computing Machinery, 2005, pp. 873–880.

[15] W. Gao, G.G. Yen, S. Liu, A cluster-based differential evolution with self-adaptive strategy for multimodal optimization, IEEE Trans. Cybern. 44 (8) (2014) 1314–1327.

[16] S. Biswas, S. Kundu, S. Das, An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution, IEEE Trans. Cybern. 44 (10) (2014) 1726–1737.

[17] S. Biswas, S. Kundu, S. Das, Inducing niching behavior in differential evolution through local information sharing, IEEE Trans. Evol. Comput. 19 (2) (2015) 246–263.

[18] Z.-J. Wang, Z.-H. Zhan, Y. Lin, W.-J. Yu, H. Wang, S. Kwong, J. Zhang, Automatic niching differential evolution with contour prediction approach for multimodal optimization problems, IEEE Trans. Evol. Comput. 24 (1) (2020) 114–128.

[19] T. Ma, H. Zhao, X. Li, F. Yang, C.S. Liu, J. Liu, A coarse- and fine-grained niching-based differential evolution for multimodal optimization problems and its application in multirobot task allocation, Swarm Evol. Comput. 83 (2023) 101412.

[20] Y. Jiang, Z.-H. Zhan, K.C. Tan, J. Zhang, Optimizing niche center for multimodal optimization problems, IEEE Trans. Cybern. 53 (4) (2023) 2544–2557.

[21] J. Zhang, D. Chen, Q. Yang, Y. Wang, D. Liu, S.-W. Jeon, J. Zhang, Proximity ranking-based multimodal differential evolution, Swarm Evol. Comput. 78 (2023) 101277.

[22] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.

[23] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417.

[24] G. Wu, X. Shen, H. Li, H. Chen, A. Lin, P.N. Suganthan, Ensemble of differential evolution variants, Inform. Sci. 423 (2018) 172–186.

[25] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (1) (2011) 55–66.

[26] R. Mallipeddi, P. Suganthan, Q. Pan, M. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2) (2011) 1679–1696.

[27] F. Ming, W. Gong, L. Gao, Adaptive auxiliary task selection for multitasking-assisted constrained multi-objective optimization [feature], IEEE Comput. Intell. Mag. 18 (2) (2023) 18–30.

[28] R. Thomsen, Multimodal optimization using crowding-based differential evolution, in: IEEE Congress on Evolutionary Computation, 2004. CEC2004, Vol. 2, 2004, pp. 1382 – 1389.

[29] Y.-H. Zhang, Y.-J. Gong, Y. Gao, H. Wang, J. Zhang, Parameter-free voronoi neighborhood for evolutionary multimodal optimization, IEEE Trans. Evol. Comput. 24 (2) (2020) 335–349.

[30] Z.-J. Wang, Z.-H. Zhan, Y. Li, S. Kwong, S.-W. Jeon, J. Zhang, Fitness and distance based local search with adaptive differential evolution for multimodal optimization problems, IEEE Trans. Emerg. Top. Comput. Intell. 7 (3) (2023) 684–699.

[31] M. Preuss, Niching the CMA-ES via Nearest-Better Clustering, GECCO ’10, Association for Computing Machinery, New York, NY, USA, 2010, pp. 1711–1718.

[32] X. Lin, W. Luo, P. Xu, Differential evolution for multimodal optimization with species by nearest-better clustering, IEEE Trans. Cybern. 51 (2) (2021) 970–983.

[33] W. Luo, Y. Qiao, X. Lin, P. Xu, M. Preuss, Hybridizing niching, particle swarm optimization, and evolution strategy for multimodal optimization, IEEE Trans. Cybern. 52 (7) (2022) 6707–6720.

[34] T. Huang, Y.-J. Gong, W.-N. Chen, H. Wang, J. Zhang, A probabilistic niching evolutionary computation framework based on binary space partitioning, IEEE Trans. Cybern. 52 (1) (2022) 51–64.

[35] Z.-J. Wang, Y.-R. Zhou, J. Zhang, Adaptive estimation distribution distributed differential evolution for multimodal optimization problems, IEEE Trans. Cybern. 52 (7) (2022) 6059–6070.

[36] Y. Sun, G. Pan, Y. Li, Y. Yang, Differential evolution with nearest density clustering for multimodal optimization problems, Inform. Sci. 637 (2023) 118957.

[37] Y. Li, L. Huang, W. Gao, Z. Wei, T. Huang, J. Xu, M. Gong, History information-based Hill-Valley technique for multimodal optimization problems, Inform. Sci. 631 (2023) 15–30.

[38] X.-Y. Chen, H. Zhao, J. Liu, A network community-based differential evolution for multimodal optimization problems, Inform. Sci. 645 (2023) 119359.

[39] X. Wu, Q. Lin, W. Lin, Y. Ye, Q. Zhu, V.C. Leung, A kriging model-based evolutionary algorithm with support vector machine for dynamic multimodal optimization, Eng. Appl. Artif. Intell. 122 (2023) 106039.

[40] W. Du, Z. Ren, J. Wang, A. Chen, A surrogate-assisted evolutionary algorithm with knowledge transfer for expensive multimodal optimization problems, Inform. Sci. 652 (2024) 119745.

[41] K. Deb, A. Saha, Multimodal optimization using a bi-objective evolutionary algorithm, Evol. Comput. 20 (1) (2012) 27–62.

[42] Y. Wang, H.-X. Li, G. Yen, W. Song, MOMMOP: Multiobjective optimization for locating multiple optimal solutions of multimodal optimization problems, IEEE Trans. Cybern. 45 (4) (2015) 830–843.

[43] R. Cheng, M. Li, K. Li, X. Yao, Evolutionary multiobjective optimization-based multimodal optimization: Fitness landscape approximation and peak detection, IEEE Trans. Evol. Comput. 22 (5) (2018) 692–706.

[44] Z. Liao, X. Mi, Q. Pang, Y. Sun, History archive assisted niching differential evolution with variable neighborhood for multimodal optimization, Swarm Evol. Comput. 76 (2023) 101206.

[45] F. Ming, W. Gong, L. Wang, L. Gao, A constrained many-objective optimization evolutionary algorithm with enhanced mating and environmental selections, IEEE Trans. Cybern. 53 (8) (2023) 4934–4946.

[46] B.Y. Qu, P.N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization, IEEE Trans. Evol. Comput. 17 (3) (2013) 387–402.

[47] X. Li, Niching without niching parameters: Particle swarm optimization using a ring topology, IEEE Trans. Evol. Comput. 14 (1) (2010) 150–169.

[48] J.E. Fieldsend, Running up those hills: Multi-modal search with the niching migratory multi-swarm optimiser, in: 2014 IEEE Congress on Evolutionary Computation, CEC, 2014, pp. 2593–2600.

[49] J. Zhang, D. Chen, Q. Yang, Y. Wang, D. Liu, S.-W. Jeon, J. Zhang, Proximity ranking-based multimodal differential evolution, Swarm Evol. Comput. 78 (2023) 101277.

[50] R. Li, W. Gong, C. Lu, L. Wang, A learning-based memetic algorithm for energy-efficient flexible job-shop scheduling with type-2 fuzzy processing time, IEEE Trans. Evol. Comput. 27 (3) (2023) 610–620.