Swarm and Evolutionary Computation 75 (2022) 101194

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Differential evolution with hybrid parameters and mutation strategies based on reinforcement learning

Zhiping Tan [a,b,c], Yu Tang [a,b,*], Kangshun Li [d], Huasheng Huang [a,b], Shaoming Luo [a,b]

[a] Academy of Heyuan, Guangdong Polytechnic Normal University, Heyuan, Guangdong, 517001, China
[b] Academy of Interdisciplinary Studies, Guangdong Polytechnic Normal University, Guangzhou, 510665, China
[c] College of Electronics and Information, Guangdong Polytechnic Normal University, Guangzhou, 510665, China
[d] College of Mathematics and Informatics, South China Agricultural University, Guangzhou, 510642, China

ARTICLE INFO

Keywords:
Differential evolution
Hybrid parameters and mutation strategies
Reinforcement learning
Dynamic fitness landscape

ABSTRACT

Differential evolution (DE) has recently attracted a lot of attention as a simple and powerful numerical optimization approach for solving various real-world applications. However, the performance of DE is significantly influenced by the configuration of control parameters and mutation strategy. To address this issue, we proposed reinforcement learning-based hybrid parameters and mutation strategies differential evolution (RL-HPSDE) in this paper. The RL-HPSDE is based on the Q-learning framework, the population individual is regarded as an agent. The optimization problem's dynamic fitness landscape analysis results are utilized to represent the environmental states. The ensemble of parameters and mutation strategy is employed as optional actions for the agent. Furthermore, a reward function is designed to guide the agent to perform the optimal action strategy. Based on its reinforcement learning experience stored by the corresponding Q table, the agent could adaptively select an optimal combination of mutation strategy and parameters to generate offspring individual during each generation. The proposed algorithm is evaluated using the CEC2017 single objective test function set. Several well-known DE variants are also compared with the proposed algorithm. Empirical studies suggest that the proposed RL-HPSDE algorithm is competitive with all other competitors.

1. Introduction

Differential evolution is a famous evolutionary algorithm (EA) proposed by Price and Storn based on the idea of recombination of individual differences, which is suitable for solving various kinds of optimization problems. As an essential part of EA, DE is also a population-based heuristic algorithm and mainly consists of three operating steps, including mutation, crossover, and selection operations [1]. Because of its simple structure and strong convergence ability, DE has been widely utilized to solve engineering optimization problems in practical applications such as feature selection [2], multilevel image thresholding segmentation [3], drug synergy prediction [4], and robot control [5].

The performance of DE largely relies on the combination of mutation strategy and control parameters. Generally, the most suitable parameters and mutation strategy configurations demanded by DE to solve disparate optimization problems are different. The reason is those partial mutation strategies efficiently solve multimodal functions; some others are suitable for composite functions. Likewise, some control parameter settings can effectively improve the convergence speed, and some other settings can enhance the exploration ability. In addition, some related research indicates that even for the same problem, the selection of mutation strategies and parameters is different at each evolutionary stage [6]. Hence, the research on the adjustment of the mutation strategy and control parameters for DE has attracted much attention. However, the traditional empirical and trial-and-error methods for setting up the feasible parameters and strategy are inefficient and time-expensive in most cases, particularly when solving diverse real-world optimization problems [7].

As a result, approaches for the adaptive selection of mutation strategy and tuning parameters received much attention [8]. Consequently, a large amount of enhanced DE variants such as JADE (with new mutation strategy and adaptive parameters) [9], EPSDE (with mutation strategies and parameters combination) [10], MPEDE (with adaptive parameters and multi-population methods) [11], SHADE (with historical memory storage mechanism) [12], LSHADE (with population size linear

* Corresponding author.
E-mail address: yutang@gpnu.edu.cn (Y. Tang).

https://doi.org/10.1016/j.swevo.2022.101194
Received 1 June 2022; Received in revised form 1 September 2022; Accepted 13 October 2022
Available online 14 October 2022

decrease) [13], TVDE (with time-varying strategy) [14], CSDE (with combined mutation strategies) [15], MPPCEDE (with multi-population and multi-strategy) [16], QLDE (with Q-learning based parameter tuning strategy) [17], DEDQN (with mixed mutation strategy) [18], have been proposed. Hence, designing an algorithm framework to select the parameters and mutation strategy during each evolution stage is critical. However, most current approaches lack learning or feedback mechanisms adjustment based on the evolutionary process. Different optimization problem characteristics may require additional parameters and strategies to effectively search for the optimal solution. Therefore, an ideal DE variant is expected to deal with various challenges in solving numerical optimization problems.

Motivated by the self-learning ideas, a reinforcement learning-based hybrid parameters and mutation strategies differential evolution named RL-HPSDE is presented. A reinforcement learning (RL) approach utilizes a Q-learning algorithm as a decision controller, an adaptive parameter, and a mutation strategy technique for DE. In RL-HPSDE, the agent learns the fitness landscape features of the optimization problem and then determines which action to be taken during the evolutionary process. Based on the Q-learning algorithm, the agent can select the optimal action operation to generate offspring individuals using the Q table, according to the optimization problem's dynamic fitness landscape analysis results. After executing each action strategy, a reward function is designed to update the Q table. The agent will efficiently search for the optimal solution through this reinforcement learning method. The CEC2017 function set is applied to evaluate the performance of the RL-HPSDE algorithm. The corresponding experiments are compared and analyzed with the other five well-known DE variants regarding accuracy, stability, and convergence.

The rest of the document is organized as follows: Section 2 presents the classical DE, a few variations of DE, and the Q-learning algorithm. Section 3 introduces two dynamic fitness landscape analysis techniques. The proposed RL-HPSDE algorithm is presented in Section 4. The experimental simulation results and discussion are shown in Section 5. Section 6 draws the conclusions and presents possible future research directions.

2. Preliminaries and related work

Primarily, we describe the framework of the DE algorithm. Next, some improvement measures for DE are briefly presented. Finally, one of the reinforcement learning algorithms, Q-learning is introduced.

2.1. Differential evolution

DE creates new individuals by making differences between individuals in the population. An initial population P is composed of NP individuals. Each population individual X_i is a D-dimensional vector:$X_i = \{x_{1,i}, x_{2,i}, \ldots, x_{D,i}\}$. In the generation times G, the i-th individual of P is marked as:$X_i^G = \{x_{1,i}^G, x_{2,i}^G, \ldots, x_{D,i}^G\}$. The minimum and maximum values of all dimensions for each individual are constrained to $X_L=(x_{1,L}, x_{2,L}, \ldots, x_{D,L})$ and $X_U=(x_{1,U}, x_{2,U}, \ldots, x_{D,U})$, respectively. While initialization, the initial value of the j-th component (j=1,2,...,D) of the i-th individual (i=1,2,...,NP) at generation G=0 is generated based on the following formula:

$$x_{j,i}^0 = x_{j,L} + rand(0,1)\cdot\left(x_{j,U} - x_{j,L}\right) \tag{1}$$

where $rand(0,1)$ is a uniformly distributed random number within the interval [0,1]. After that, DE will perform a mutation, crossover, and selection operations.

Mutation: The mutation operator is to turn each target vector X_i^G into a mutant vector $V_i^G = (v_1^G, v_2^G, \ldots, v_D^G)$. Generally, the most commonly employed mutation strategies are shown as follows [19]:

- "DE/rand/1"

$$V_i^G = X_{r1}^G + F * \left(X_{r2}^G - X_{r3}^G\right) \tag{2}$$

- "DE/best/1"

$$V_i^G = X_{best}^G + F * \left(X_{r2}^G - X_{r3}^G\right) \tag{3}$$

- "DE/rand/2"

$$V_i^G = X_{r1}^G + F * \left(X_{r2}^G - X_{r3}^G\right) + F * \left(X_{r4}^G - X_{r5}^G\right) \tag{4}$$

- "DE/best/2"

$$V_i^G = X_{best}^G + F * \left(X_{r1}^G - X_{r2}^G\right) + F * \left(X_{r3}^G - X_{r4}^G\right) \tag{5}$$

- "DE/current-to-rand/1"

$$V_i^G = X_i^G + F * \left(X_i^G - X_{r1}^G\right) + F * \left(X_{r2}^G - X_{r3}^G\right) \tag{6}$$

- "DE/current-to-best/1"

$$V_i^G = X_i^G + F * \left(X_i^G - X_{best}^G\right) + F * \left(X_{r1}^G - X_{r2}^G\right) \tag{7}$$

where the indices $r_1 \neq r_2, \ldots, \neq r_5 \neq i$, these indices are all different random numbers. F is a positive number with the range [0,1].X_{best}^G is the individual with the best fitness in the current population.

Crossover: The trial vector $U_i^G = (u_{1,i}^G, u_{2,i}^G, \ldots, u_{D,i}^G)$ mainly generated by the binomial crossover operator, using the below scheme:

$$u_{j,i}^G = \begin{cases} v_{j,i}^G, & if\ \left(rand_{j,i}[0,\ 1] \leq Cr\ or\ j = j_{rand}\right) \\ x_{j,i}^G, & otherwise \end{cases} \tag{8}$$

where $rand_{j,i}$ represents a uniformly distributed random number ranging from 0 to 1. $Cr \in [0,1]$ is considered the crossover probability, and j_{rand} is an integer less than D.

Selection: The newly generated trial vector and the original target vector will determine who will be selected to enter the new generation population by comparing their fitness values. For solving the global minimum problem, if the fitness value of the trial vector$f(U_i^G)$is better than the old target vector$f(X_i^G)$, This trial vector will substitute the original target vector and become a new generation of population individuals. In turn, the initial target vector will become a new population individually. The selection process is expressed as the below formula:

$$X_i^{G+1} = \begin{cases} U_i^G, & if\ f\left(U_i^G\right) \leq f\left(X_i^G\right) \\ X_i^G, & otherwise \end{cases} \tag{9}$$

2.2. Related work

DE has attracted a lot of attention due to its simple structure and excellent performance. However, the performance of DE depends on the selection of mutation strategy and relevant control parameters. To settle this problem, the improvement of DE has attracted a lot of attention. Hence, many novel techniques are proposed to enhance its performance. Next, we will give a brief introduction to some DE variants.

Liu and Lampinen presented a fuzzy rules-based adaptive control parameters DE algorithm (FADE). Two different fuzzy logic controllers are used in FADE to adjust the settings [20]. Breast et al. provided a self-adaptive DE (jDE). In jDE, the parameter settings are randomly produced based on preset parameters τ_1 and τ_2, respectively [21]. Zhang

et al. proposed a JADE algorithm. In JADE, the crossover probability *CR* and scale factor *F* are produced by the Normal and Cauchy functions, respectively [9]. Tanabe proposed a SHADE algorithm based on JADE, in which all the parameters corresponding to successful individuals are recorded in the memories [12]. Tanabe proposed an LSHADE algorithm to enhance population diversity, where a linear population size reduction strategy (LPSR) is applied based on the SHADE algorithm. Numerical simulation results demonstrated that this population mechanism dramatically enhances the performance of the LSHADE algorithm [13]. Zhan et al. proposed an adaptive distributed differential evolution (ADDE). In ADDE, a co-enhanced multiple population strategy is used to improve the DE performance, and each population selects an appropriate mutation strategy based on evolutionary information [22]. Ali et al. proposed a sTDE-dR algorithm. This algorithm uses a new adaptive parameter scheme and selects the best adaptive strategy [23]. Deng et al. presented wavelet basis function-based DE, named WMSDE. In WMSDE, the wavelet basis function generates the scaling factor, and the sub-population strategy is also applied to enhance the performance [24]. Li et al. proposed a dual mutation collaboration differential evolution (DMCDE). In DMCDE, an elite guidance mechanism is designed to modify the mutation strategies "DE/rand/2" and "DE/best/2", then a dual mutation collaboration mechanism is used to guide the population evolution [25]. Meng et al. proposed an adaptive learning mechanism differential evolution (PALM-DE), in which a parameter adaptive learning method is used to enhance the former mutation strategy [26].

Motivated by the fact that no single DE strategy is best for all types of problems, a brief review of the considerable approaches to addressing this problem is presented in the following. Wang et al. introduced a composite DE (CoDE). In CoDE, three mutation strategies and three sets of control parameters are stored in strategy pools. For each generation, the control parameters are chosen randomly [27]. Qin et al. suggested a self-adjusted DE (SaDE). In SaDE, the mutation strategy and the associated parameters are selected by previous experience [28]. Mallipeddi et al. proposed a DE (EPSDE) algorithm that combines mutation strategy and control parameters. In EPSDE, strategy pools and parameter pools are designed, and mutation strategy and control parameters are randomly picked up from strategy pools at the beginning of the algorithm. Then the trial vector is generated by mutation operation. The corresponding mutation strategy and control parameter shall be maintained in the strategy pools if the trial vector is higher than the target vector. Instead, mutation strategy and control parameters will be re-selected [10]. Wu et al. proposed multiple sub-populations combining multiple mutation strategies DE (MPEDE). In MPEDE, a population is divided into several sub-populations. In the initial iteration stage, each sub-population will be randomly assigned a mutation strategy, and an indicator will evaluate each mutation strategy's performance. The best-performing mutation strategy based on the indicator will be selected for the candidate pool. The mutation strategy would be chosen from the candidate pool in the next stage of mutation operation [11]. Tan et al. proposed a DE (FLDE) based on the fitness landscape where the mutation strategy is selected based on the landscape characteristics of each optimization problem [29]. Sun et al. proposed combined schemes into DE and named CSDE, in which two mutation operators are used to generate a mutant vector. Two mutation strategies are balanced based on their historical success rate [15]. Sharma et al. proposed double deep Q-Learning based DE (DE-DDQN), in which the selection of the mutation strategy and parameters depends on the agent's decision at each generation [18].

2.3. Q-learning

In an RL algorithm, an agent obtains valuable feedback through interaction with the environment. In other words, the RL only focuses on which actions the agent takes to obtain the maximum cumulative reward. Generally, the training process of an RL agent is a Markov decision process (MDP). The basic framework of RL is shown in Fig. 1. At

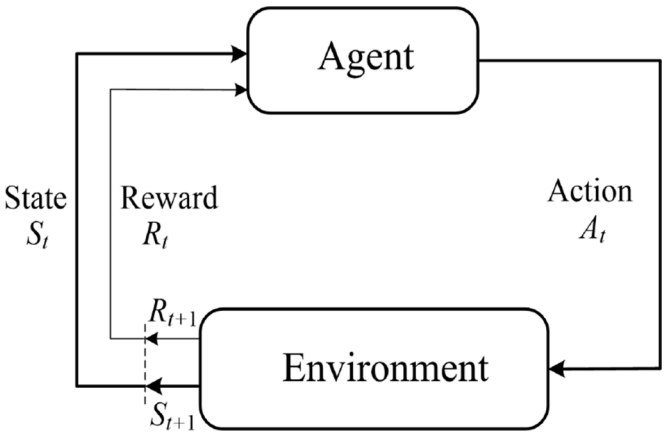

Fig. 1. The agent-environment of RL.

each time step *t*, the agent observes the environment's state S_t=[s_1,s_2,..., s_n], and performs a series of actions A_t=[a_1,a_2,...,a_n]. One time step later, the agent achieves the immediate reward R_{t+1} and executes the next state S_{t+1}. The agent obtains the most appropriate action strategy through this trial-and-error approach [30].

Q-learning is one of the RL algorithms proposed by Watkins in 1989. Known as a model-free RL method, Q-learning is an off-policy temporal-difference (TD) algorithm that will be regarded as the decision controller and applied to DE. Unlike other RL algorithms, Q-learning uses the *Q* table to store different state and action values. An agent uses a selection policy π to select an action for the state s_i. The selection probability $P_r(s_i, a_j)$ of selecting a_j action in state s_i is calculated by Eq. (10) in the *Q*-table. The soft-max function is used as the selection probability, which is identified using a Boltzmann distribution ranking the value-function estimates: where *s* refers to a positive parameter, Q_t (s_i, a_j) is the corresponding value in the *Q* table at time *t*, τ denotes the *maxQ* (s_i; a_j). *K* is the number of candidate actions.

$$\pi(a_j|s_i) = P_r\{a_j = a|s_i = s\} = \frac{e^{Q_t(s_i,a_j)/\tau}}{\sum_{j=1}^{K} e^{Q_t(s_i,a_j)/\tau}} \tag{10}$$

Hence, the agent performs the action according to the *Q* table. The following formula updates the *Q* table:

$$Q(S_t, A_t) \leftarrow (1-\alpha)Q(S_t, A_t) + \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a)\right) \tag{11}$$

where R_{t+1} represents the reward value obtained after A_t action of the agent in the S_t state; α denotes the learning rate within the range from [0,1], and γ is the discount rate between 0 and 1.

The fusion of RL and DE has achieved increasing attention in recent years. RL can enhance the performance of DE due to it can get a series of cumulative rewards by trying different parameters and mutation strategies and helps the algorithm choose the optimal operation.

3. Dynamic fitness landscape analysis

In most cases, measuring the statistical fitness landscape of an optimization problem can be helpful in judging the best solution to the issue, whether easy or difficult to search. Many landscape analysis methods have been proposed, including modality, ruggedness, information content, and dynamic severity. However, the search process for the optimal solution is dynamic. The landscape information of the problem will change dynamically. Therefore, analyzing an optimization problem's dynamic fitness landscape is necessary.

A static fitness landscape Λ_S can be defined by $\Lambda_S = (\Delta S, n, f)$, where *S* is the search space with elements $x \in S$, $n(x)$ is a neighborhood function, which orders for every $x \in S$ a set of direct and possibly also more distant

neighbors, and $f(x)$: $S\rightarrow\Re$is the fitness function giving every search space element a fitness value. Dynamic fitness landscapes differ from static ones in that their topological features changes with time, which can be described as $\Lambda_D = (S\Delta, n, K, F, \varphi)$, where S represents all possible solutions of the optimization problem and the neighborhood function $n(x)$ gives a set of neighbors to every search space point. The time set $K\subseteq Z$ provides a scale for measuring and ordering dynamic changes; F is the set of all fitness functions in time $k \in K$, the transition mapφdefines how the fitness function changes over time [31]. Next, we will introduce two dynamic fitness landscape analysis methods.

3.1. Dynamic fitness distance correlation

The correlation of the distance from individual fitness to the known global optimal solution is usually measured by fitness distance correlation, which is utilized to analyze the characteristics of the optimization problem. The starting point for calculating a measure for fitness distance correlation is a random walk on the fitness landscape. This random walk should have the length T and the step size t_S. So, we can obtain a random walk on the fitness landscape as $x(j+1)=x(j)+t_s \cdot rand$, where *rand* is a uniformly distributed random number. Dynamic fitness distance correlation (DFDC) mainly obtains a series of discrete sampling points at time k by implementing a random walk algorithm [32]. Then calculating the fitness values [33]:

$$f(j,k) = f(x(j),k), \quad j = 1,2,...T \tag{12}$$

The Euclidean distance $d(j,k)$, j=1,2,...,T is used to measure the distance for each individual to the optimal global solution. Thus, the DFDC is defined as the following formula:

$$DFDC(k) = \frac{\frac{1}{T}\sum_{j=1}^{T}(f(j,k)-\overline{f}(j,k))(d(j,k)-\overline{d}(j,k))}{\delta_f \delta_d} \tag{13}$$

where δ_f and δ_d are the variances.$\overline{f}, \overline{d}$are the average values of f and d, respectively.

In general, for the minimization optimization problem, 0.15< DFDC $\leq$ 1 indicates a strong correlation, and the optimal solution can be easily searched. -0.15$\leq$ DFDC $\leq$ 0.15 reveals a moderate correlation, and the global optimum may be found. -1$\leq$ DFDC < -0.15 signifies a weak correlation, the problem is challenging to solve [34].

3.2. Ruggedness of information entropy

The dynamic ruggedness of information entropy (DRIE) is used to measure ruggedness, defined by Richter et al. Based on the information entropy theory [35]. For dynamic ruggedness of information entropy, a discrete-time series of fitness values $f(j, k)$ are obtained by performing a random walk algorithm to search a series of discrete-time points at a set point in time k. A string $S(e, k)=s_1,s_2,...,s_i,s_n$ of symbols$s_i \in \{-1, 0, 1\}$ can be defined by the following rules:

$$S_i(e,k) = \begin{cases} -1, & if\ f(j+1,k)-f(j,k) < -e \\ 0, & if\ |f(j+1,k)-f(j,k)| \leq e \\ 1, & if\ f(j+1,k)-f(j,k) > e \end{cases} \tag{14}$$

where the parameter e represents the scale of the fitness landscape observation.

On account of the calculation rules of the string $S(e,k)$, the information entropy $H(e,k)$ can be measured by the following formula:

$$H(e,k) = -\sum_{p\neq q} P_{[pq]}(e,k)\log_6 P_{[pq]}(e,k) \tag{15}$$

where p and q represent 1, -1, and 0, three different elements besides $p\neq q$.

There are nine combinations of p and q, but it is required that p is not equal to q, so the different combinations [pq] are only six possibilities. The probability $P_{[pq]}$ is evaluated according to the following equation:

$$P_{[pq]} = \frac{n_{[pq]}}{n} \tag{16}$$

where $n_{[pq]}$ is the number of the different combinations [pq] in the string $S(e,k)$.

The string $S(e,k)$ depends on the size of parameter e. When the value of e increases, most of the elements of the $S(e,k)$ are the same, the probability $P_{[pq]}$ is close to zero, and the entropy $H(e,k)$ will close to negative infinity, which proves the landscape structure is relatively flat. As the value of e gets the minimum, the string $S(e,k)$ elements will be 0. This situation is defined as the information entropy stability and is also marked as symbol e^*. Therefore, the value of e^* is the difference between the maximum and minimum values in the fitness value and can be described as:

$$e^* = \max\{f(j+1,k)-f(j,k)\}_{j=1}^{T} \tag{17}$$

To show that the fitness landscape is observed from different scales, some other e^* values are usually taken to represent the roughness of information entropy. In actual application, a single value R_f is adopted to describe the dynamic ruggedness of information entropy:

$$R_f = \max_{\forall e\in[0,\ e^*]} \{H(e,k)\} \tag{18}$$

All the possible e values are:

$$\left\{e | e=0,\ \frac{e^*}{128}, \frac{e^*}{64},\ \frac{e^*}{32},\ \frac{e^*}{16},\ \frac{e^*}{8},\ \frac{e^*}{4},\ \frac{e^*}{2},\ e^*\right\} \tag{19}$$

Therefore, the value of R_f represents the complexity of the landscape. The higher value implies a smooth terrain. Otherwise, it indicates that the landscape is complicated. Generally, 0$\leq R_f <$ 0.5 reveals that the optimization problem is relatively complicated. 0.5$\leq R_f \leq$ 1 shows that the optimization problem is not complicated [36].

4. The proposed algorithm

This section presents an RL-HPSDE algorithm. Firstly, we designed the main innovative component of Q-learning, including state representations, action strategies, and reward function. Then a population renewal mechanism is introduced. Finally, the general framework of RL-HPSDE is outlined.

4.1. State representations

This paper will take the optimization problem's dynamic fitness landscape analysis results as the agent's observation of the environment. The design divides each dynamic fitness landscape analysis result into two states according to the relevant theory. Due to the use of the Q-learning algorithm, the optimization problem will be divided into the following four observation states according to the dynamic fitness landscape analysis results, as shown in Table 1. Hence, the total number of environment states is 4, namely s_1, s_2, s_3, and s_4. The states s_1, s_2, and s_3 imply that the landscape features of the optimization problem are not apparent. On the contrary, the state s_4 denotes the landscape features are easy to observe. According to the dynamic fitness landscape analysis results, the agent can obtain sufficient environmental state information to guide the algorithm to make correct decisions.

Table 1
The state division of environment observation.

State representations		DFDC [-1,0.15)	(0.15,1]
DRIE	[0,0.5)	s_1	s_2
	[0.5,1]	s_3	s_4

4.2. Action options and Q table

The performance of DE largely depends on the control parameters and mutation strategies. In general, the Cauchy and Normal distribution functions were used to produce the scale factor F and crossover rate Cr, respectively. Lévy distribution function is another way to generate random numbers. In our work, the Lévy and Cauchy distribution functions will be selectively applied to produce scale factor F. "DE/current to rand/1" has a specific ability to explore; "DE/current to best/1" can enhance the convergence speed. Therefore, these two mutation strategies will be utilized to perform mutation operations. The action options of the agent will constitute by combining mutation strategies and control parameters, as shown in Table 2.

The Q table is shown in Table 3, which is a 4×4 matrix. The agent's current state picks up a combination of mutation strategy and parameters according to the probabilistic distribution computed by Eq. (10) and generates a mutant vector. The Q-learning algorithm constantly updates this Q-table to determine which action to apply for the given state during each generation.

4.3. Reward of an action

In RL, the reward function guides the agent to perform the following action based on environmental state information. Whether the action performed by the agent is reasonable depends on the return value of the reward value. To a certain extent, the more reward value, the more influential the action strategy executed by the agent. After training, the agent can return the action that maximizes the long-term cumulative reward expectation according to the state value. Evolutionary ability is associated with a population generating excellent offspring. In our work, we will use the population evolution efficiency to evaluate the algorithm's evolutionary ability and use it to act as the reward value of the agent.

Defining individual evolutionary number survivali, which denotes the individual xi survival from the current population to the next population, the survivali can be calculated as follow:

$$survival_i = \begin{cases} survival_i + 1 & if \ \ f(u_i) \le f(x_i) \\ 0 & otherwise \end{cases} \tag{20}$$

Then the population evolutionary efficiency $e(P)$ can be defined as:

$$e(P) = \frac{suvvival_i}{NP_{\text{current}}} \tag{21}$$

where $NP_{current}$ represents the current population size, the population evolutionary efficiency $e(P)$ is regarded as the reward R for performing the action a. From the definition. It can be easily found that a higher evolutionary efficiency indicates a higher value reward and a correct decision made by the agent.

4.4. Population renewal mechanism

To improve the performance of the proposed RL-HPSDE algorithm. In RL-HPSDE, a population size linearly decreases mechanism is used [13]. In the early phases of algorithm evolution, the population size is set to $NP_{\max}$, and as the algorithm optimization process proceeds, the population size will become $NP_{\min}$. The following equation explains this strategy:

Table 2
The action options ensemble of parameters and mutation strategies.

Action options		Mutation strategy DE/current to rand/1	DE/current to best/1
Scale factor F	Cauchy	a_1	a_2
	Lévy	a_3	a_4

Table 3
State-action table (Q-table).

State	Action a_1	a_2	a_3	a_4
s_1	$Q_{11}(s_1, a_1)$	$Q_{12}(s_1, a_2)$	$Q_{13}(s_1, a_3)$	$Q_{14}(s_1, a_4)$
s_2	$Q_{21}(s_2, a_1)$	$Q_{22}(s_2, a_2)$	$Q_{23}(s_2, a_3)$	$Q_{24}(s_2, a_4)$
s_3	$Q_{31}(s_3, a_1)$	$Q_{32}(s_3, a_2)$	$Q_{33}(s_3, a_3)$	$Q_{34}(s_3, a_4)$
s_4	$Q_{41}(s_4, a_1)$	$Q_{42}(s_4, a_2)$	$Q_{43}(s_4, a_3)$	$Q_{44}(s_4, a_4)$

$$NP_{G+1} = \text{round}(NP_{\min} - NP_{\max})/FES_{\max} * FES + NP_{\max} \tag{22}$$

where $NP_{\max}$ is the maximum population size in the initialization phase, $FES_{\max}$ is the maximum number of function evaluations. Besides, all the parameter strategies use a historical memory storage mechanism.

4.5. The framework of the proposed algorithm

A novel DE algorithm based on RL with hybrid parameters and mutation strategies (RL-HPSDE) is presented, which combines a DE algorithm with an RL mechanism. The framework of the RL-HPSDE is shown in Fig. 2. As described in Fig. 2, the RL-HPSDE realizes adaptive selection of mutation strategy and parameters through Q-learning algorithm. During the execution of the algorithm, the agent observes the dynamic fitness landscape features of the optimization problem to select the policy of the suitable action. Then, DE implements the corresponding operations as guided by the agent. Finally, the reward value of the agent is calculated, and the Q table is updated. By constantly cycling, the agent is able to make the best choice based on previous experience. In this way, DE can search for the optimal solution to the problem.

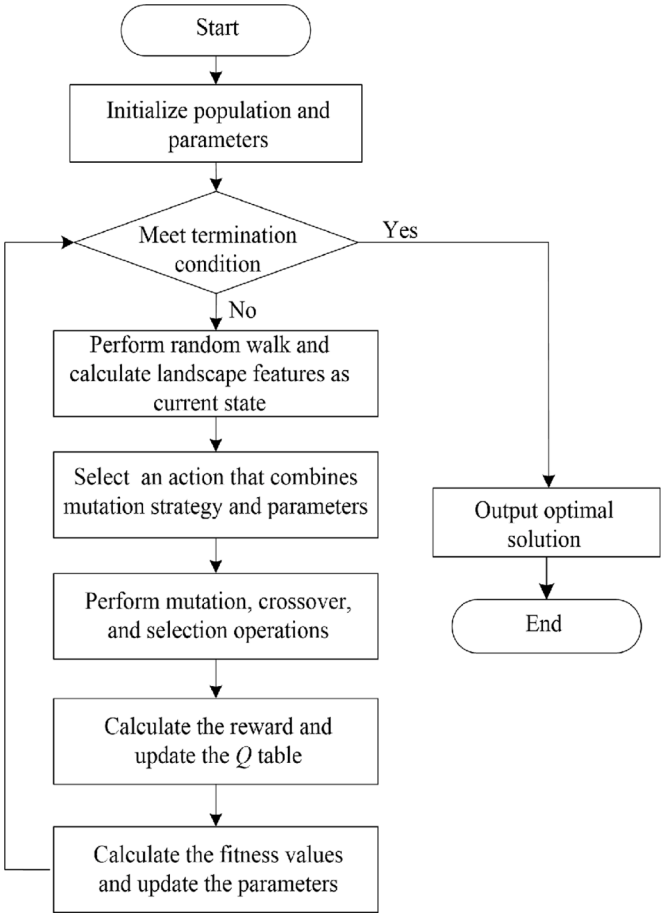

Fig. 2. The framework of RL-HPSDE.

In addition, the pseudo-code of the RL-HPSDE algorithm is shown in Table 4. First, all control parameters are set, the population and Q table are randomly initialized, and the fitness values of the population are calculated. Then, the fitness landscape of the optimization problem is analyzed, and the environment state si based on fitness landscape features is obtained. Next, for each generation, an action strategy is drawn using a Q table by Eq. (10), the trial individuals are generated after mutation and crossover operation, and the new fitness values are calculated. Then, the new Q table is updated by Eq. (11). Finally, the population size and control parameters are updated. When the termination condition is met, the loop is ended, and the founded optimal solution is outputted. Before the proposed algorithm was tested, we trained the proposed algorithm on the CEC2013 and CEC2014 single objective function sets.

5. Experimental study

This section presents the experiment's description, parameter settings of the contrasted algorithm, and time complexity analysis and discusses the proposed algorithm's performance compared to other DE algorithms.

5.1. Experiments description

All the test functions originated from the CEC2017 single-objective test function set [37]. This series of test functions consists of 30 different characteristic functions. For all of the test functions, each dimension is limited to [-100, 100]. The whole function set contains four types of functions: unimodal functions F_1–F_3, basic multimodal functions F4–F10, hybrid functions F11–F20, and composite functions F21–F30.

These benchmark functions are tested in the experiments using a black box, and the solution error f(x)−f(x*) of these functions are collected. If the error values are less than 10−8, the final result is treated as 0, where $f(x)$ is the optimum fitness value searched by each algorithm, and $f(x^*)$ is the foreknown fitness of each test function. Each test function is resolved 51 times independently, and the maximum number of function assessments is set at 10000 D, where D is the dimension of the problem.

5.2. Parameter settings of the contrasted algorithms

To evaluate the performance of the proposed RL-HPSDE algorithm, a total of five well-known DE variants, including JADE, SHADE, LSHADE, iLSHADE [38], and jSO [39], are compared with the proposed algorithm. The reasons for selecting these five DE variants as comparison algorithms are interpreted as follows: First, JADE and SHADE are two well-known DE schemes used as comparison algorithms by numerous related researchers. Second, multiple-strategies mechanisms are applied in LSHADE and iLSHADE. Hence, it is valuable to compare RL-HPSDE with them. Third, jSO is a recently proposed DE variant, an enhanced LSHADE algorithm. For RL-HPSDE algorithm, due to the need for random sampling when performing dynamic fitness landscape analysis, so the number of sampling steps and step size is set to 200 and 10, respectively. These algorithms employ the recommended parameter settings in our executed experiments, summarized in Table 5.

Table 4
Pseudo-code of the RL-HPSDE algorithm.

Procedure of RL-HPSDE
Set M_{CR}=M_F=M_L=0.5, α=0.8, γ=0.5, i=1; NP=$NP^{\max}$, D, $Gmax$;
Initialization: Population-X, Q table $Q(s_i, a_i)$;
Calculate fitness values of the population X;
for G=1,2,...,G_{max} **do**
Analyze the dynamic fitness landscape of the optimization problem;
Obtain the state s_i based on the dynamic fitness landscape features;
Draw an action strategy by $Q(s_i, a_i)$ with Eq. (10);
For j=1 to NP
Perform mutation operation $v_{j,i}$=Mutation ($x_{j,i}$);
Perform crossover operation $u_{j,i}$=Crossover ($x_{j,i}$,$v_{j,i}$);
Calculate fitness $f(u_{j,i})$;
Perform selection operator;
end
Calculate population evolution efficiency $e(P)$;
Update Q table using Eq. (11);
Perform population size renewal mechanism, update population;
Renew M_{CR}, M_F and M_L;
end for
Output the optimal solution

5.3. Time complexity analysis

The time complexity is one of the important metrics to evaluate the performance of an algorithm. The main difference between RL-HPSDE algorithm and other DE variants is the mutation strategy and control parameters selection mechanism. As shown in Table 4, the mutation strategy and control parameters selection of the RL-HPSDE using reinforcement learning. However, the selection of mutation strategy and control parameters will perform dynamic fitness landscape analysis, and a random walk sampling approach needs to be implemented during fitness landscape analysis.

The time complexity of the standard DE is O(Gmax·NPini·D), where Gmax is the maximum number of generations, NPini is the initialized population size, and D is the dimension of the problem. The total complexity of SHADE is O(Gmax·NPini·[D+log(NPini)]). They are considering that SHADE, LSHADE, iLSHADE, and jSO share the same framework of the algorithm. The total complexity of these algorithms is almost the same. Hence, the complexity of RL-HPSDE is $O(G_{max}\cdot NP^{ini}\cdot[D+\log(NP^{ini})]\cdot steps)$, where *steps* denotes the random sampling steps, and smaller sampling steps are generally set in the experiment. Compared with standard DE and the above-mentioned DE variants, the proposed RL-HPSDE increases a little time complexity.

5.4. Comparison against some well-known DE variants

In this part, we mainly compare the performance of the RL-HPSDE with JADE, SHADE, LSHADE, iLSHADE, and jSO. The experiments are conducted under the CEC2017 function test set, which contains 30 different types of functions. The associated simulation results are shown in Tables 6–9 for 10D, 30D, 50D, and 100D, respectively. In these Tables, the average and corresponding standard deviation values of errors are listed for all the contrast algorithms, and the minimum error value for each test function is bold. Additionally, the Wilcoxon rank-sum test results at significance level 0.05 are listed in a separate column. The symbols +, -, and ≈ suggest whether a given algorithm performs significantly better (+), significantly worse (-), or not quite different,

Table 5
Recommended parameter settings and comparison.

Algorithm	Parameters initial setting and comparison
JADE	$u_F = 0.5$, $F \sim C(u_F, 0.1)$, $u_{Cr} = 0.5$, $Cr \sim N(u_F, 0.1)$, $NP = 100$,$p = 0.05$, $c = 0.1$.JADE utilizes a new mutation strategy "DE/current-to-pbest/1" with an optional external archive.
SHADE	$u_F = 0.5$, $F \sim C(u_F, 0.1)$, $u_{Cr} = 0.5$, $Cr \sim N(u_F, 0.1)$, $NP = 100$,$p = 0.2$, $r^{arc} = 2$, $H = 100$.SHADE uses a success history parameter adaptation mechanism and "current-to-pbest/1" mutation strategy.
LSHADE	On the basis of SHADE, a linear population size reduction mechanism is used. NP_{max}=18·NP, NP_{min}=4·NP, H=5, p=0.1, r^{arc}=2.
iLSHADE	$u_F = 0.8$, $u_{Cr} = 0.5$, F, Cr ,r^{arc} same as LSHADE, H=6, F_H=Cr_H=0.9, NP=12D∼4, p=0.2∼0.1.
jSO	A "current-to-pbest-w/1" mutation strategy is used, p-value for mutation strategy linearly decreases from p_{max} to p_{min}, p_{max} =0.25, p_{min} =0.125, H=5, u_F=0.3, NP^{init}=25log(D)$\sqrt{D}$.
RL-HPSDE	A reinforcement learning algorithm is utilized. NP=18D∼4, H=5, α=0.8, γ=0.5, *steps*=200, *step size*=10.

Table 6
Comparison for 10*D*.

	JADEMean (Std Dev)	SHADEMean (Std Dev)	LSHADEMean (Std Dev)	iLSHADEMean (Std Dev)	jSOMean (Std Dev)	RL-HPSDEMean (Std Dev)
F_1	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_2	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_3	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	4.2667E+00(2.33E+01) -	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_4	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_5	3.2464E+00(8.02E-01) -	3.6475E+00(7.64E-01) -	3.3515E+00(1.15E+00) -	**1.5621E+00(8.49E-01)** +	1.7558E+00(7.51E-01) ≈	2.0113E+00(8.73E-01)
F_6	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	1.5158E-14(3.93E-14) -	1.1145E-14(3.41E-14) -	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_7	1.37074E+01(8.54E-01) -	1.3404E+01(9.75E-01) -	1.3140E+01(7.65E-01) -	1.1857E+01(6.22E-01) ≈	**1.1791E+01(5.92E-01)** ≈	1.1872E+01(5.36E-01)
F_8	3.6065E+00(8.56E-01) -	3.6676E+00(1.06E+00) -	3.5838E+00(1.26E+00) -	**1.7771E+00(7.78E-01)** +	1.9509E+00(7.32E-01) ≈	2.0523E+00(8.99E-01)
F_9	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_{10}	7.7236E+01(5.08E+01) -	6.9502E+01(6.16E+01) -	3.4633E+01(5.02E+01) -	2.8241E+01(4.74E+01) ≈	3.5897E+01(5.51E+01) -	**2.6694E+01 (4.69E+01)**
F_{11}	2.3775E+00(7.81E-01) -	1.8893E+00(7.95E-01) -	1.6998E+00(3.83E-01) -	1.4987E+00(3.92E-01) -	**0.0000E+00 (0.00E+00)** +	1.5731E-01(4.79E-01)
F_{12}	3.5886E+02(1.82E+03) -	7.1533E+01(7.42E+01) -	2.4512E+01(4.84E+01) ≈	6.9509E+01(6.93E+01) -	2.7231E+01(1.72E+01) -	**2.2256E+01 (4.74E+01)**
F_{13}	3.6680E+00(2.67E+00) ≈	3.5322E+00(2.46E+00) +	4.0857E+00(1.98E+00) -	4.2390E+00(2.25E+00) -	**3.3868E+00 (2.37E+00)** ≈	3.7172E+00 (2.12E+00)
F_{14}	1.6027E+00(2.82E+00) -	2.6651E-01(4.42E-01) -	2.8221E-01(3.38E-01) -	8.4753E-01(3.91E+00) -	5.9516E-02(2.53E-01) ≈	**4.9365E-02(4.03E-01)**
F_{15}	4.3617E-01(1.92E-01) -	1.5543E-01(2.11E-01) ≈	1.8319E-01(1.95E-01) ≈	2.6139E-01(2.12E-01) -	2.2208E-01(1.98E-01) ≈	**1.2871E-01(1.96E-01)**
F_{16}	1.3245E+00(8.09E-01) -	7.7125E-01(2.91E-01) -	3.8473E-01(1.92E-01) ≈	4.1514E-01(1.88E-01) ≈	5.6671E-01(2.36E-01) -	**3.2852E-01(1.77E-01)**
F_{17}	4.9891E-01(2.63E-01) -	4.6351E-01(2.53E-01) -	1.1734E-01(1.25E-01) ≈	**9.6231E-02(1.21E-01)** +	5.1236E-01(3.62E-01) -	1.3078E-01(1.96E-01)
F_{18}	5.4553E-01(5.84E-01) -	**1.8077E-01(1.95E-01)** ≈	1.8535E-01(2.05E-01) ≈	1.0533E+00(3.87E+00) -	3.1245E-01(1.27E-01) -	1.9344E-01(2.04E-01)
F_{19}	4.5829E-02(2.31E-02) -	2.5726E-02(2.22E-02) -	2.3317E-02(1.49E-02) -	4.2245E-02(2.11E-01) -	1.0713E-02(2.24E-02) -	**7.0911E-03(9.83E-03)**
F_{20}	8.2539E-09(4.48E-08) -	1.2242E-02(6.11E-02) -	**0.0000E+00 (0.00E+00)** ≈	6.1210E-02(1.25E-01) -	3.4289E-01(1.35E-01) -	**0.0000E+00 (0.00E+00)**
F_{21}	1.4250E+02(4.77E+01) +	1.6142E+02(4.59E+01) -	1.5003E+02(4.92E+01) +	1.6549E+02(4.80E+01) -	**1.3352E+02 (5.01E+01)** +	1.5427E+02 (4.91E+01)
F_{22}	**9.6107E+01(1.95E+01)** +	9.8556E+01(1.03E+01) +	1.0000E+02(7.50E-13) ≈	9.8039E+01(1.40E+01) +	1.0000E+02(0.00E+00) ≈	1.0000E+02 (0.00E+00)
F_{23}	3.0498E+02(1.25E+00) -	3.0514E+02(1.03E+00) -	3.0451E+02(1.21E+00) -	3.0283E+02(1.35E+00) ≈	**3.0121E+02 (1.68E+00)** ≈	3.0252E+02 (1.47E+00)
F_{24}	2.8396E+02(8.83E+01) ≈	2.9658E+02(7.46E+01) -	3.0172E+02(7.90E+01) -	3.1908E+02(4.53E+01) -	2.9659E+02(7.72E+01) -	**2.8214E+02 (8.80E+01)**
F_{25}	4.1889E+02(2.33E+01) -	4.1599E+02(2.26E+01) -	4.0858E+02(1.95E+01) +	4.0966E+02(2.01E+01) +	**4.0825E+02 (1.73E+01)** +	4.1421E+02 (2.21E+01)
F_{26}	**3.0000E+02(0.00E+00)** ≈	**3.0000E+02 (0.00E+00)** ≈	**3.0000E+02 (0.00E+00)** ≈	**3.0000E+02(0.00E+00)** ≈	**3.0000E+02 (0.00E+00)** ≈	**3.0000E+02 (0.00E+00)**
F_{27}	**3.8905E+02(4.37E-01)** ≈	3.8929E+02(2.53E-01) ≈	3.8939E+02(2.20E-01) ≈	3.8943E+02(1.88E-01) ≈	3.8938E+02(2.36E-01) ≈	3.8942E+02(2.25E-01)
F_{28}	3.9397E+02(1.40E+02) -	3.9953E+02(1.42E+02) -	3.5955E+02(1.21E+02) -	3.7233E+02(1.27E+02) -	**3.3908E+02 (9.59E+01)** +	3.6647E+02 (1.28E+02)
F_{29}	2.4640E+02(5.39E+00) -	2.4339E+02(6.25E+00) -	2.3742E+02(3.18E+00) -	2.3524E+02(4.19E+00) -	2.3419E+02(2.92E+00) -	**2.3120E+02 (2.71E+00)**
F_{30}	6.1488E+02(5.60E+02) -	4.8564E+04(1.94E+05) -	4.9536E+02(2.88E+02) -	7.1160E+04(2.24E+05) -	**3.9452E+02(4.45E-02)** +	4.1042E+02 (2.39E+01)
+	2	2	2	5	5	- -
≈	10	10	13	11	16	- -
-	18	18	15	14	9	- -

better or worse (≈) than the RL-HPSDE algorithm. All the statistics are summarized at the bottom of the Tables.

For 10D optimization, our proposed RL-HPSDE algorithm obtains an overall better performance in terms of optimization accuracy. The proposed RL-HPSDE can find the global optimum on 7 out of 30 functions during the total 51 runs, and JADE can find 6, SHADE 6, LSHADE 5, iLSHADE 5, and jSO 7 out of 30 functions. All the algorithms can find the global optimum on F_1-F_4, F_6, and F_9, except LSHADE on F_3 and F_6, iLSHADE on F_6. Besides, RL-HPSDE is superior to other DE algorithms on hybrid functions F_{11}-F_{20}. From Wilcoxon rank-sum test results, RL-HPSDE is significantly better than JADE, SHADE, LSHADE, iLSHADE, and jSO on 18, 18, 15, 14, and 9 functions, similar to them on 10, 10, 13, 11, and 16 functions, and worse than them on 2, 2, 2, 5, and 5, respectively.

For 30*D* optimization, the RL-HPSDE shows more apparent advantages than other DE algorithms. Only jSO and RL-HPSDE can find the global optimum on 5 out of 30 functions during the total 51 runs, and JADE can find 2, SHADE 3, LSHADE 3, and iLSHADE 3 out of 30 functions. Among them, only F_1-F_3, F_6, and F_9 can be searched for the optimum solution by jSO and RL-HPSDE. Besides, the optimal solutions of functions F_{10}, F_{12}, and F_{30} seem challenging to search for all algorithms. For composite functions, the errors are more significant than 100. Furthermore, From Wilcoxon rank-sum test results, RL-HPSDE shows better performance than JADE, SHADE, LSHADE, iLSHADE, and jSO on 22, 20, 20, 19, and 14 functions, similar to them on 7, 8, 9, 9, and 13 functions, and worse than them on 1, 2, 1, 2, and 3, respectively.

Table 7
Comparison for 30*D*.

	JADEMean (Std Dev)	SHADEMean (Std Dev)	LSHADEMean (Std Dev)	iLSHADEMean (Std Dev)	jSOMean (Std Dev)	RL-HPSDEMean (Std Dev)
F_1	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_2	1.9607E-02(1.40E-01) -	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_3	5.3675E-14(5.31E-13) -	3.5146E-14(9.15E-14) -	2.5727E-14(2.34E-13) -	1.1145E-15(7.95E-15) -	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_4	**5.7631E+01 (8.30E+00)** ≈	5.8561E+01(1.96E-14) ≈	5.8561E+01(2.68E-14) ≈	5.8562E+01(2.89E-14) ≈	5.8671E+01(7.71E-01) ≈	5.8562E+01(2.41E-14)
F_5	5.5369E+01(4.91E+00) -	4.4466E+01(4.61E+00) -	1.9479E+01(3.19E+00) -	7.5368E+00(1.83E+00) -	8.5568E+00(2.07E+00) -	**5.5214E+00 (1.38E+00)**
F_6	7.3722E-04(4.64E-04) -	4.5755E-08(9.75E-08) -	1.1368E-13(2.98E-14) -	1.1368E-13(0.00E+00) -	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_7	9.0077E+01(5.49E+00) -	7.6923E+01(5.11E+00) -	4.6369E+01(2.68E+00) -	3.9113E+01(1.38E+00) -	3.8926E+01(1.44E+00) -	**3.6421E+01 (1.19E+00)**
F_8	5.4927E+01(5.03E+00) -	4.4953E+01(4.77E+00) -	2.2099E+01(3.13E+00) -	8.2221E+00(1.28E+00) -	9.0918E+00(1.82E+00) -	**5.2167E+00 (1.53E+00)**
F_9	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_{10}	3.3804E+03(1.97E+02) -	2.9998E+03(2.49E+02) -	1.4252E+03(2.02E+02) ≈	1.4198E+03(2.15E+02) ≈	1.5298E+03(2.81E+02) -	**1.4093E+03 (2.12E+02)**
F_{11}	4.1280E+01(3.09E+01) -	3.9550E+01(2.96E+01) -	2.3059E+01(2.70E+01) -	2.1911E+01(2.58E+01) -	2.0857E+01(2.75E+00) -	**1.8021E+01 (2.17E+01)**
F_{12}	1.1258E+03(3.86E+02) -	**8.7621E+02 (3.65E+02)** +	1.1077E+03(4.48E+02) -	9.0226E+02(3.38E+02) +	9.4025E+02(1.02E+02) +	9.7661E+02 (4.30E+02)
F_{13}	5.2598E+01(3.03E+01) -	4.2813E+01(2.04E+01) -	1.6963E+01(6.17E+00) -	1.5276E+01(7.11E+00)	1.4985E+01(5.12E+00) ≈	**1.4838E+01 (6.20E+00)**
F_{14}	4.0767E+01(4.37E+00) -	3.1319E+01(2.02E+00) -	2.1206E+01(3.94E+00) ≈	2.1445E+01(1.48E+00) -	2.1825E+01(1.22E+00) -	**2.0844E+01 (2.80E+00)**
F_{15}	6.7693E+01(4.10E+00) -	1.0000E+01(3.83E+00) -	3.7812E+00(2.33E+00) -	3.0948E+00(1.82E+00) -	3.0879E+00(6.85E-01) -	**2.7597E+00 (1.62E+00)**
F_{16}	5.4911E+02(1.59E+02) -	4.2052E+02(1.18E+02) -	2.1529E+02(1.17E+02) -	4.1784E+01(3.70E+01) -	7.8923E+01(8.39E+01) -	**2.8678E+01 (2.89E+01)**
F_{17}	1.3663E+02(2.52E+01) -	1.1220E+02(1.89E+01) -	3.9252E+01(6.22E+00) -	**2.9926E+01(6.57E+00)** +	3.2948E+01(8.23E+00) -	3.1466E+01 (5.83E+00)
F_{18}	8.2388E+01(3.51E+00) -	2.4517E+01(2.87E+00) -	2.2815E+01(3.86E+00) ≈	2.1917E+01(1.47E+00) ≈	**2.0411E+01 (2.84E+00)** +	2.1855E+01 (1.24E+00)
F_{19}	3.7766E+01(3.85E+00) ≈	1.5087E+01(1.99E+00) ≈	**4.8538E+00(1.50E+00)** +	5.112E+00(1.83E+00) ≈	5.1536E+00(1.82E+00) -	4.9698E+00 (1.62E+00)
F_{20}	1.7185E+02(5.83E+01) -	1.4106E+02(4.17E+01) -	4.0483E+01(7.85E+00) -	2.9898E+01(6.38E+00) -	2.9345E+01(6.12E+00) ≈	**2.8139E+01 (4.99E+00)**
F_{21}	2.5449E+02(4.97E+00) -	2.4520E+02(5.24E+00) -	2.2276E+02(4.03E+00) -	2.0883E+02(1.44E+00) -	2.0953E+02(1.86E+00) -	**2.0617E+02 (1.50E+00)**
F_{22}	1.0000E+02(1.00E-13) ≈	1.0000E+02(1.00E-13) ≈	**1.0000E+02(0.00E+00)** ≈	1.0000E+02(1.00E-13) ≈	**1.0000E+02 (0.00E+00)** ≈	1.0000E+02(1.00E-13)
F_{23}	3.9969E+02(5.40E+00) -	3.9012E+02(5.54E+00) -	3.6827E+02(5.04E+00) -	3.5277E+02(3.23E+00) -	3.5171E+02(3.26E+00) ≈	**3.4960E+02 (2.21E+00)**
F_{24}	4.6165E+02(6.95E+00) -	4.5483E+02(4.71E+00) -	4.3663E+02(5.30E+00) -	4.2848E+02(2.26E+00) -	4.2645E+02(2.44E+00) ≈	**4.2545E+02 (1.59E+00)**
F_{25}	3.8690E+02(8.45E-02) ≈	3.8678E+02(3.13E-02) ≈	3.8673E+02(2.63E-02) ≈	3.8673E+02(2.10E-02) ≈	**3.8669E+02(7.62E-03)** ≈	3.8674E+02(2.56E-02)
F_{26}	1.3973E+03(6.11E+01) -	1.3393E+03(5.05E+01) -	1.1133E+03(6.46E+01) -	9.5590E+02(4.08E+01) -	9.2032E+02(4.36E+01) -	**9.1265E+02 (3.77E+01)**
F_{27}	5.0249E+02(4.03E+00) ≈	5.0397E+02(4.05E+00) ≈	5.0443E+02(3.53E+00) -	5.0523E+02(5.93E+00) -	5.1738E+02(7.21E+00) -	**5.0162E+02 (5.18E+00)**
F_{28}	3.2005E+02(4.42E+01) +	3.2001E+02(4.75E+01) +	3.3565E+02(5.23E+01) -	3.3430E+02(5.37E+01) -	**3.0872E+02 (3.12E+01)** +	3.2592E+02 (4.77E+01)
F_{29}	5.3885E+02(2.10E+01) -	5.3474E+02(1.79E+01) -	4.4186E+02(1.26E+01) -	4.3372E+02(1.23E+01) -	4.3167E+02(1.36E+01) ≈	**4.3058E+02 (6.67E+00)**
F_{30}	2.0007E+03(6.24E+01) -	2.0000E+03(6.88E+01) -	2.0196E+03(7.71E+01) -	1.9861E+03(5.41E+01) -	1.9791E+03(1.92E+01) -	**1.8915E+03 (5.83E+01)**
+	1	2	1	2	3	- -
≈	7	8	9	9	13	- -
-	22	20	20	19	14	- -

For 50*D* optimization, RL-HPSDE is also competitive on unimodal functions, basic multimodal functions, and composite functions and is capable of finding the optimal solution on functions F_1-F_3, F_6, and F_9. Compared with the other five DE variants, only jSO obtains identical results and outperforms RL-HPSDE on functions F_{11}-F_{15}, F_{18}-F_{19}, and F_{28}-F_{29}. Besides, algorithms have difficulties in solving functions F_{10}, F_{12}, F_{16}-F_{17}, F_{22}, F_{26}, and F_{30}. From Wilcoxon rank-sum test results, RL-HPSDE also outperforms JADE, SHADE, LSHADE, iLSHADE, and jSO on 27, 23, 27, 20, and 13 functions, similar to them on 1, 3, 2, 4, and 8 functions, and worse than them on 2, 4, 1, 6, and 9, respectively. These results show RL-HPSDE is more competitive on high-dimensional optimization.

For 100*D* optimization, the proposed RL-HPSDE algorithm performs better than other peer DE algorithms, finding the global optimum on 2 out of 30 functions during the total 51 runs. However, jSO is slightly better than RL-HPSDE in searching optimum. In addition, F_{10}, F_{12}, F_{16}-

Table 8
Comparison for 50*D*.

	JADEMean (Std Dev)	SHADEMean (Std Dev)	LSHADEMean (Std Dev)	iLSHADEMean (Std Dev)	jSOMean (Std Dev)	RL-HPSDEMean (Std Dev)
F_1	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_2	2.1568E-01(4.61E-01) -	1.5686E-01(3.67E-01) -	8.3333E-01(8.74E-01) -	3.5291E-01(5.94E-01) -	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_3	4.0895E+04(6.50E+04) -	2.1771E+04(5.14E+04) -	1.501E+05(8.43E+04) -	1.1814E-13(3.56E-14) -	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_4	6.7916E+01(4.26E+01) -	7.2552E+01(4.62E+01) -	7.3337E+01(4.76E+01) -	6.0002E+01(5.04E+01) -	5.6812E+01(4.96E+01) ≈	**5.5014E+01 (5.02E+01)**
F_5	1.4419E+02(8.89E+00) -	1.3324E+02(9.37E+00) -	3.4514E+01(7.15E+00) -	1.5144E+01(2.54E+00) -	1.7245E+01(3.56E+00) -	**9.3257E+00 (2.07E+00)**
F_6	8.2455E-04(7.28E-04) -	**0.0000E+00 (0.00E+00)** ≈	2.8558E-08(7.14E-08) -	5.2997E-08(2.38E-07) -	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_7	1.9798E+02(1.10E+01) -	1.8720E+02(8.10E+00) -	7.5687E+01(4.37E+00) -	6.6982E+01(2.04E+00) -	6.6472E+01(3.52E+00) -	**6.1008E+01 (1.58E+00)**
F_8	1.4080E+02(8.89E+00) -	1.3105E+02(7.38E+00) -	3.4843E+01(5.59E+00) -	1.5137E+01(2.11E+00) -	1.7035E+01(3.32E+00) -	**9.1591E+00 (2.18E+00)**
F_9	3.5109E-03(1.75E-02) -	**0.0000E+00 (0.00E+00)** ≈	3.7896E-14(5.45E-14) -	5.1270E-14(5.71E-14) -	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_{10}	7.1857E+03(3.53E+02) -	6.8281E+03(3.37E+02) -	3.2568E+03(2.65E+02) -	**3.0360E+03(3.06E+02)** +	3.1392E+03(3.63E+02) -	3.0826E+03 (2.40E+02)
F_{11}	1.4881E+02(1.83E+02) -	3.7604E+01(6.16E+00) +	4.8328E+01(9.90E+00) -	3.9919E+01(5.90E+00) -	**2.7938E+01 (3.46E+00)** +	4.6047E+01 (1.05E+01)
F_{12}	2.1752E+03(8.23E+02) -	2.6090E+03(9.87E+02) -	2.2637E+03(5.83E+02) -	2.0793E+03(5.26E+02) -	**1.6805E+03 (5.31E+02)** +	2.0303E+03 (5.40E+02)
F_{13}	1.8324E+02(5.36E+01) -	1.9610E+02(3.58E+01) -	7.0595E+01(3.54E+01) -	5.5433E+01(3.26E+01) +	**3.0521E+01 (2.01E+01)** +	5.7518E+01 (2.64E+01)
F_{14}	8.7348E+03(3.91E+04) -	6.6218E+01(8.78E+00) -	3.0593E+01(3.44E+00) -	2.7477E+01(2.41E+00) +	**2.4963E+01 (1.85E+00)** +	2.8648E+01 (2.98E+00)
F_{15}	4.1542E+01(1.20E+01) -	4.1876E+01(1.02E+01) -	4.2134E+01(1.23E+01) -	3.3378E+01(6.36E+00) +	**2.4125E+01 (2.96E+00)** +	3.8508E+01 (1.14E+01)
F_{16}	1.4352E+03(1.34E+02) -	1.2141E+03(1.66E+02) -	5.9667E+02(1.25E+02) -	3.9361E+02(1.14E+02) -	4.5125E+02(1.36E+02) -	**3.7110E+02 (1.04E+02)**
F_{17}	1.0124E+03(1.24E+02) -	8.8053E+02(1.42E+02) -	3.4703E+02(9.19E+01) -	2.4509E+02(7.24E+01) -	2.8351E+02(8.59E+01) -	**2.1528E+02 (8.04E+01)**
F_{18}	7.8876E+01(4.40E+01) -	6.7642E+01(2.36E+01) -	5.1759E+01(2.14E+01) -	3.2875E+01(6.66E+00) +	**2.4125E+01 (2.01E+00)** +	3.6394E+01 (8.69E+00)
F_{19}	3.8032E+01(1.17E+01) -	3.3209E+01(8.35E+00) -	2.7794E+01(7.70E+00) -	2.0329E+01(3.50E+00) +	**1.4253E+01 (2.36E+00)** +	2.0845E+01 (7.59E+00)
F_{20}	8.4084E+02(1.31E+02) -	7.9010E+02(1.13E+02) -	2.3734E+02(7.87E+01) -	1.4409E+02(6.05E+01) ≈	**1.4021E+02 (7.76E+01)** ≈	1.4223E+02 (4.01E+01)
F_{21}	3.4488E+02(1.02E+01) -	3.3350E+02(1.21E+01) -	2.3374E+02(6.12E+00) -	2.1708E+02(1.70E+00) -	2.1919E+02(3.95E+00) -	**2.1138E+02 (2.18E+00)**
F_{22}	3.7468E+03(3.14E+03) -	3.1234E+03(2.91E+03) -	1.3540E+03(1.75E+03) -	2.7249E+03(1.31E+03) -	1.4872E+03(1.73E+03) -	**1.2771E+03 (1.68E+03)**
F_{23}	5.6067E+02(1.14E+01) -	5.4980E+02(9.77E+00) -	4.5155E+02(6.35E+00) -	4.3367E+02(4.35E+00) -	4.3015E+02(6.23E+00) -	**4.2745E+02 (4.96E+00)**
F_{24}	6.0921E+02(9.42E+00) -	6.0884E+02(1.05E+01) -	5.1898E+02(5.96E+00) -	5.1332E+02(2.87E+00) -	5.0926E+02(4.12E+00) -	**5.0617E+02 (2.77E+00)**
F_{25}	5.0062E+02(3.36E+01) -	4.8604E+02(1.95E+01) -	4.8136E+02(3.60E+00) ≈	4.8210E+02(4.26E+00) ≈	4.8125E+02(2.81E+00) ≈	**4.8058E+02 (3.63E+00)**
F_{26}	2.2738E+03(1.06E+02) -	2.1813E+03(7.78E+01) -	1.3551E+03(6.49E+01) -	1.1945E+03(5.45E+01) -	1.1991E+03(5.56E+01) -	**1.1342E+03 (5.27E+01)**
F_{27}	5.2492E+02(8.74E+00) +	5.2260E+02(9.19E+00) +	5.2727E+02(1.22E+01) +	5.3416E+02(1.86E+01) -	**5.1128E+02 (1.12E+01)** +	5.2932E+02 (1.13E+01)
F_{28}	4.7896E+02(2.42E+01) -	4.6363E+02(1.46E+01) +	4.7240E+02(2.11E+01) -	4.6922E+02(2.00E+01) ≈	**4.5982E+02 (6.67E+00)** +	4.6816E+02 (1.93E+01)
F_{29}	6.1424E+02(5.24E+01) -	5.6911E+02(4.80E+01) -	3.5590E+02(1.16E+01) -	3.6057E+02(1.14E+01) -	3.6293E+02(1.32E+01) -	**3.5159E+02 (1.13E+01)**
F_{30}	6.2600E+05(4.80E+04) +	**6.1911E+05 (4.49E+04)** +	6.4129E+05(5.51E+04) -	6.6737E+05(8.49E+04) -	6.3705E+05(2.95E+04) -	6.2845E+05 (6.89E+04)
+	2	4	1	6	9	- -
≈	1	3	2	4	8	- -
-	27	23	27	20	13	- -

F_{17}, F_{20}, F_{26}, F_{29}, and F_{30} seem challenging to solve. From Wilcoxon rank-sum test results, RL-HPSDE is significantly better than JADE, SHADE, LSHADE, iLSHADE, and jSO on 28, 26, 23, 21, and 13 functions, similar to them on 0, 1, 4, 1, and 4 functions, and worse than them on 2, 3, 3, 8, and 13, respectively. The jSO exhibits the same performance compared with RL-HPSDE.

For all the dimensions, the proposed RL-HPSDE algorithm achieves an overall better performance from the optimization accuracy and has obvious advantages compared with the other five DE algorithms. All the statistical results show that RL-HPSDE has remarkable improvement. Furthermore, a Friedman test is conducted based on the average results, which shows the average ranking of all algorithms. The test results are summarized in Table 10. It can be summarized that there is a significant difference between the performances of the algorithms. The best ranks are shown in bold. Besides, we can clearly see that RL-HPSDE obtains the best ranking in 10, 30, and 50 dimensions, jSO ranked best in 100

Table 9
Comparison for 100*D*.

	JADEMean (Std Dev)	SHADEMean (Std Dev)	LSHADEMean (Std Dev)	iLSHADEMean (Std Dev)	jSOMean (Std Dev)	RL-HPSDEMean (Std Dev)
F_1	2.2577E-08(2.50E-08) -	5.3312E-06(3.79E-06) -	6.7312E-13(7.59E-13) -	7.1419E-12(1.98E-11) -	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_2	5.1139E+14(3.56E+15) -	9.4992E+12(6.78E+13) -	8.3048E+12(3.27E+13) -	2.8708E+06(1.88E+07) -	3.1243E+11(2.41E+13) -	**3.9558E+10 (1.86E+11)**
F_3	2.6467E+05(2.12E+05) -	1.3489E+05(1.84E+05) -	4.0359E+05(1.64E+05) -	3.0203E-07(3.92E-07) -	**0.0000E+00(0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
F_4	2.0750E+02(2.46E+01) -	2.0215E+02(9.78E+00) -	1.9652E+02(4.73E+00) -	1.9444E+02(1.86E+01) -	1.9062E+02(2.93E+01) -	**1.8813E+02 (2.44E+01)**
F_5	4.1080E+02(1.58E+01) -	4.2974E+02(1.42E+01) -	5.9195E+01(6.41E+00) -	4.3184E+01(4.51E+00) -	4.4107E+01(5.76E+00) -	**2.7479E+01 (5.66E+00)**
F_6	5.7162E-09(1.96E-08) +	5.4052E-09(1.60E-08) +	1.2151E-03(8.98E-04) +	3.0433E-03(2.63E-03) ≈	**0.0000E+00(0.00E+00)** +	3.7227E-03(4.91E-03)
F_7	5.2029E+02(1.34E+01) -	5.5902E+02(1.48E+01) -	1.5798E+02(6.24E+00) -	1.4815E+02(4.45E+00) -	1.4528E+02(6.23E+00) -	**1.3411E+02 (3.96E+00)**
F_8	4.0105E+02(1.77E+01) -	4.2017E+02(1.81E+01) -	5.9443E+01(8.52E+00) -	4.3012E+01(4.34E+00) -	4.3261E+01(5.63E+00) -	**2.7057E+01 (5.90E+00)**
F_9	3.4559E-01(3.78E-01) -	**5.2663E-03(2.12E-02)** +	9.2958E-02(1.60E-01) ≈	1.9331E-01(3.54E-01) -	4.6243E-02(7.01E-02) ≈	4.5932E-02(4.54E-01)
F_{10}	2.0157E+04(4.22E+02) -	2.0554E+04(3.83E+02) -	1.0426E+04(3.40E+02) ≈	**9.6811E+03(5.74E+02)** +	9.7062E+03(6.86E+02) +	1.0122E+04 (5.72E+02)
F_{11}	2.4873E+04(1.38E+04) -	3.9454E+04(1.14E+04) -	5.9642E+03(9.84E+03) -	2.6929E+02(6.66E+01) +	**1.1425E+02(4.32E+01)** +	4.4562E+02 (1.05E+02)
F_{12}	1.8633E+04(7.37E+03) +	2.8848E+04(1.14E+04) -	2.4871E+04(9.48E+03) -	1.9470E+04(8.61E+03) -	**1.8452E+04(8.62E+03)** +	1.9373E+04 (6.08E+03)
F_{13}	2.0374E+03(1.15E+03) -	1.9987E+03(8.56E+02) -	8.7797E+02(5.22E+02) -	3.4391E+02(.28E+02) +	**1.4525E+02(3.82E+01)** +	5.6764E+02 (4.93E+02)
F_{14}	2.9384E+02(4.17E+01) -	2.6514E+02(3.72E+01) -	8.7797E+02(5.22E+02) -	2.3272E+02(3.21E+01) +	**6.4352E+01(1.12E+01)** +	2.5838E+02 (3.17E+01)
F_{15}	2.4676E+02(5.63E+01) -	2.4359E+02(4.13E+01) -	2.4029E+02(4.69E+01) ≈	2.5212E+02(4.38E+01) -	**1.6232E+02(3.84E+01)** +	2.4081E+02 (4.81E+01)
F_{16}	4.6590E+03(2.86E+02) -	4.2593E+03(3.07E+02) -	2.0222E+03(3.27E+02) -	1.7438E+03(2.70E+02) -	1.8569E+03(3.67E+02) -	**1.3638E+03 (3.08E+02)**
F_{17}	3.2718E+03(2.36E+02) -	2.9576E+03(2.21E+02) -	1.4645E+03(1.91E+02) -	1.2102E+03(1.96E+02) -	1.2876E+03(2.41E+01) -	**1.0118E+03 (1.93E+02)**
F_{18}	2.7693E+02(6.69E+01) -	2.3436E+02(4.91E+01) -	2.2450E+02(3.73E+01) -	2.1964E+02(5.07E+01) -	**1.6727E+01(3.75E+01)** +	2.0985E+02 (4.76E+01)
F_{19}	1.8080E+02(3.46E+01) -	1.6306E+02(2.04E+01) +	1.7156E+02(2.14E+01) ≈	1.7341E+02(2.25E+01) -	**1.0455E+02(2.12E+01)** +	1.7131E+02 (2.37E+01)
F_{20}	3.4084E+03(2.30E+02) -	3.4231E+03(1.90E+02) -	1.6533E+03(1.90E+02) -	1.4268E+03(1.65E+02) -	1.3868E+03(2.43E+02) ≈	**1.3812E+03 (2.04E+02)**
F_{21}	6.2825E+02(1.95E+01) -	6.5098E+02(1.85E+01) -	2.8262E+02(5.77E+00) -	2.6831E+02(4.10E+00) -	2.6532E+02(6.12E+00) -	**2.4901E+02 (5.87E+00)**
F_{22}	2.1074E+04(1.32E+03) -	2.1459E+04(1.60E+03) -	1.1482E+04(4.58E+02) -	1.0442E+04(5.29E+02) +	**1.0235E+04(2.23E+03)** +	1.1073E+04 (6.24E+02)
F_{23}	8.8236E+02(1.48E+01) -	8.7513E+02(1.20E+01) -	5.9936E+02(8.69E+00) -	**5.6540E+02(5.80E+00)** +	5.7265E+02(1.05E+01) -	5.6841E+02 (9.26E+00)
F_{24}	1.2338E+03(1.83E+01) -	1.2645E+03(1.18E+01) -	9.3349E+02(7.88E+00) -	9.3223E+02(7.90E+00) -	9.1542E+02(8.25E+00) -	**9.0536E+02 (8.31E+00)**
F_{25}	7.4409E+02(3.46E+01) -	7.5357E+02(2.46E+01) -	7.4000E+02(3.60E+01) +	7.3821E+02(3.84E+01) +	**7.3525E+02(3.64E+01)** +	7.4325E+02 (3.38E+01)
F_{26}	6.2726E+03(1.70E+02) -	6.7217E+03(2.01E+02) -	3.5413E+03(8.79E+01) -	3.4945E+03(8.15E+01) -	3.3788E+03(7.94E+01) -	**3.2812E+03 (7.30E+01)**
F_{27}	6.3610E+02(1.39E+01) -	6.1608E+02(1.24E+01) ≈	6.3295E+02(1.82E+01) -	6.2933E+02(1.68E+01) -	**5.8547E+02(2.31E+01)** +	6.1962E+02 (1.86E+01)
F_{28}	5.7558E+02(3.12E+01) -	5.3353E+02(2.41E+01) -	**5.1946E+02(2.35E+01)** +	5.2655E+02(2.26E+01) -	5.2751E+02(2.84E+01) -	5.2263E+02 (2.53E+01)
F_{29}	3.0411E+03(1.76E+02) -	2.9481E+03(2.00E+02) -	1.3184E+03(1.14E+02) -	1.3068E+03(1.63E+02) -	1.3563E+03(1.91E+02) -	**1.1771E+03 (1.62E+02)**
F_{30}	2.3921E+03(2.77E+02) -	3.8599E+03(7.77E+02) -	2.3935E+03(1.71E+02) -	2.3740E+03(1.39E+02) +	**2.3546E+03(1.24E+02)** +	2.3961E+03 (1.48E+02)
+	2	3	3	8	13	- -
≈	0	1	4	1	4	- -
-	28	26	23	21	13	- -

dimensions. Regarding mean ranking, RL-HPSDE gets the first ranking, followed by jSO, iLSHADE, and LSAHDE obtain a similar ranking, JADE and SHADE achieve the last ranking. All the experimental data show that the RL-HPSDE algorithm can solve the CEC2017 test functions well. It also shows that the reinforcement learning-based hybrid parameters and mutation strategies are effective for DE.

5.5. Evolutionary curve for all algorithms

The convergence curves of the median error values of the total 51 runs originating from JADE, SHADE, LSHADE, iLSHADE, jSO, and RL-HPSDE are plotted in Fig. 3. Due to page limit, only the functions F_5, F_7, F_8, F_{13}, F_{16}, F_{17}, F_{23}, F_{24}, and F_{26} on 30*D* experimental results are figured out to make a comparison. As shown in Fig. 3, for multimodal functions F_5, F_7, and F_8, RL-HPSDE converges and searches for the

Table 10
Friedman ranks for all algorithms.

Algorithm	Rank	F-rank				Mean Ranking
		D=10	D=30	D=50	D=100	
JADE	1	4.28	5.31	5.35	5.16	5.03
SHADE	2	4.01	4.36	4.28	4.93	4.40
LSHADE	3	3.45	3.76	4.21	3.93	3.34
iLSHADE	4	3.80	2.96	3.08	2.83	3.17
jSO	5	2.81	2.71	2.08	**1.93**	2.38
RL-HPSDE	6	**2.63**	**1.86**	**1.98**	2.20	**2.17**

optimal solution after about 15000 evaluations, which outperforms other algorithms. Among them, SHADE falls into the local optimum very early. For hybrid functions F_{13}, F_{16}, and F_{17}, RL-HPSDE can better balance exploration and exploitation. For composite functions F_{23}, F_{24}, and F_{26}, RL-HPSDE demonstrates better global convergence ability. Furthermore, the box plots of the mean error values of 51 independent runs on 30 dimensions derived from all the comparison algorithms are shown in Fig. 4. It can be easy to find that RL-HPSDE exhibits better stability, the performance of iLSHADE and jSO is not much different, the other three algorithms JADE, SHADE, and LSHADE show worse performance. To sum up all the analysis results, our proposed RL-HPSDE algorithm with RL strategy and hybrid parameters and mutation strategies works very well in the accuracy, convergence rate, and robustness than those comparative algorithms, and the results also demonstrate that the proposed RL-HPSDE algorithm is compatible with the resolution of various kinds of optimization features.

6. Conclusions

RL-HPSDE, a combination of reinforcement learning and hybrid parameters and mutation strategies differential evolution, is presented in this paper, which realizes the adaptive selection of parameters and mutation strategy through a reinforcement learning algorithm. The

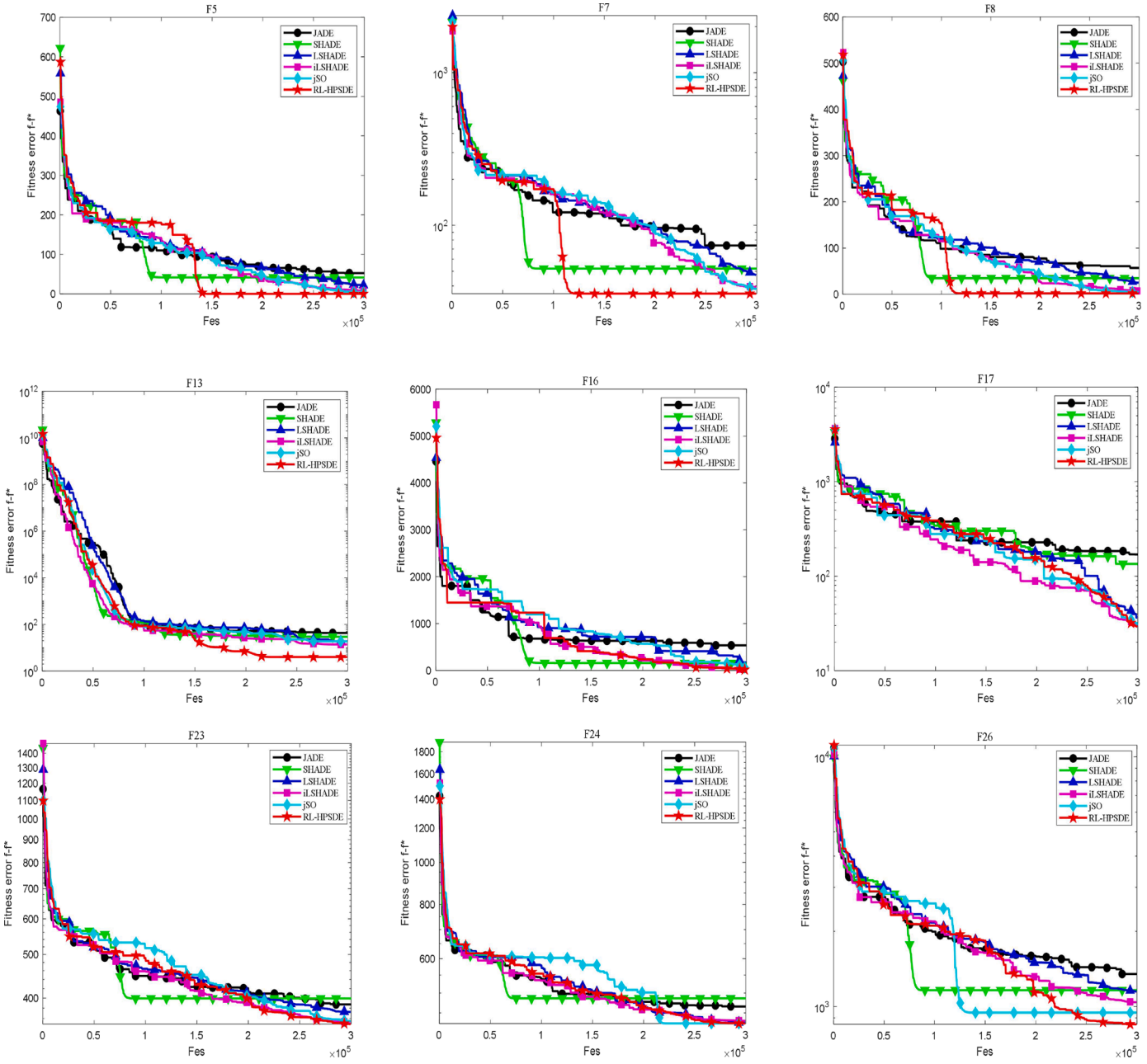

Fig. 3. Comparison of convergence performance for each algorithm.

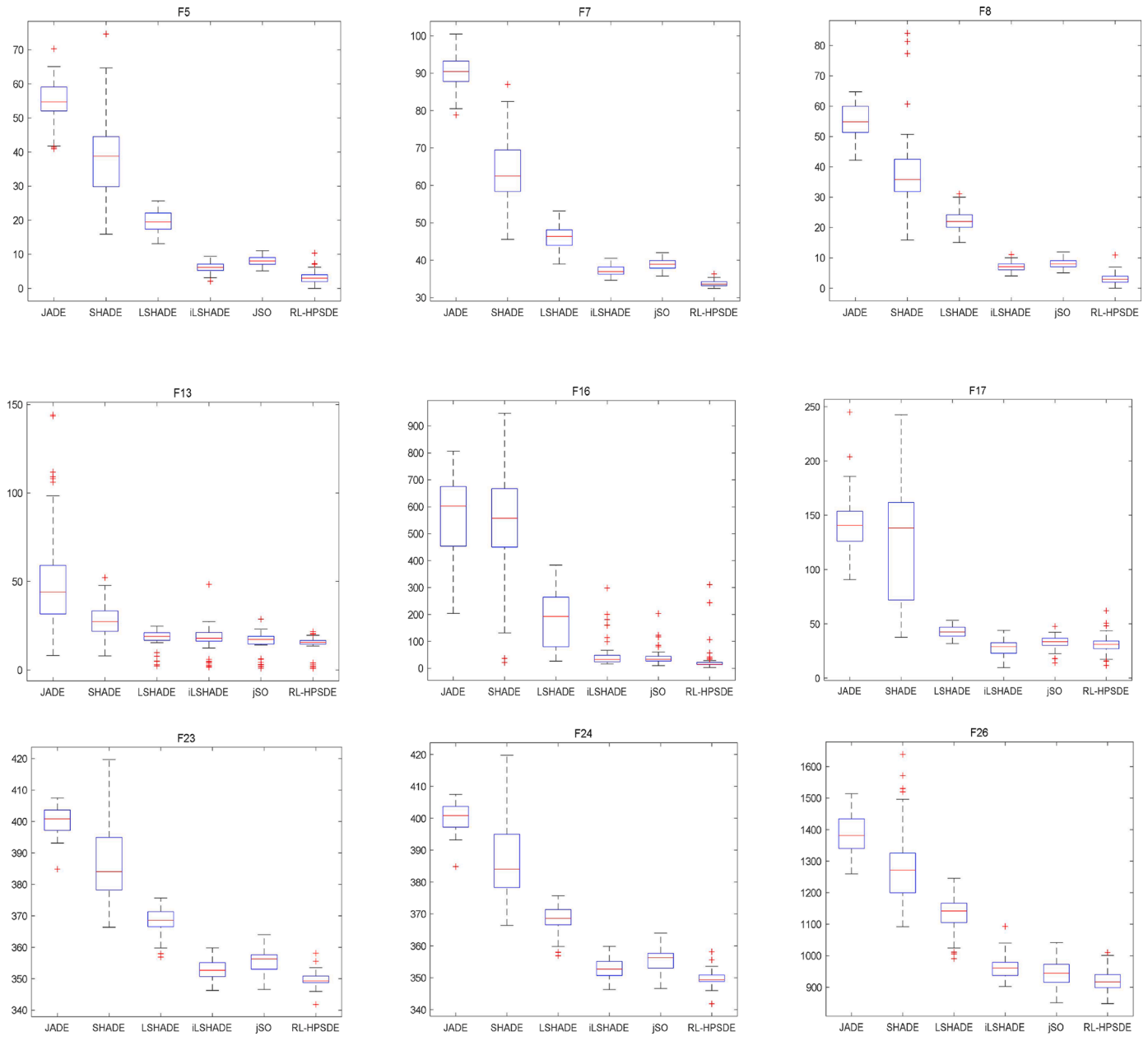

Fig. 4. Comparison of box plots for each algorithm.

environmental states have consisted of the optimization problem's dynamic fitness landscape analysis results, which are used to determine the evolution direction for DE. Since reinforcement learning with hybrid parameters and mutation strategies helps guide the population to converge gradually and improve diversity, RL-HPSDE shows obvious superiority compared to JADE, SHADE, LSHAD, iLSHADE, and jSO in terms of accuracy, the velocity of convergence, and robustness. It can be concluded from the experimental results that DE combined with the Q-learning algorithm can significantly improve the efficiency of the population search in the solution space, and dynamic fitness landscape analysis can effectively analyze the evolution process. The reinforcement learning algorithm enables the DE algorithm to select the mutation strategy and control parameters effectively. In future research, fitness landscape analysis will be applied to multi-objective problems.

CRediT authorship contribution statement

Zhiping Tan: Conceptualization, Methodology, Software, Writing – original draft. **Yu Tang:** Writing – review & editing. **Kangshun Li:** Supervision, Resources. **Huasheng Huang:** Formal analysis, Data curation. **Shaoming Luo:** Investigation, Visualization.

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.

Acknowledgments

The authors acknowledge support from the Planned Science and Technology Project of Guangdong Province, China (Grant nos. 2019B020216001, 2019A050510045, and 2021A0505030075), the Fundamental and Applied Basic Research Fund of Guangdong Province (Grant no. 2021A1515110637), the National Natural Science Foundation of China (Grant no. 32071895), the Natural Science Foundation of

Guangdong Province, China (Grant nos. 2020B1515120070, and 2021A1515010824), the Key Project of Universities in Guangdong Province, China (Grant no. 2020ZDZX1061), the Innovation Team Project of Universities in Guangdong Province, China (Grant no. 2021KCXTD010), the Planned Science and Technology Project of Guangzhou, China (Grant nos. 202002020063, 202007040007, and 202103000028), the Rural Revitalization Strategy Project of Guangdong Province, China (Grant no. 2019KJ138), and the Academy of Heyuan Project of Guangdong Polytechnic Normal University, China (Grant no. 20210101), Guangdong Provincial Department of Education characteristic innovation project Grant no. 2020KTSCX068) .

References

[1] R Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[2] H Li, F He, Y Chen, et al., MLFS-CCDE: multiobjective large-scale feature selection by cooperative coevolutionary differential evolution, Meme. Comput. 13 (1) (2021) 1–18.
[3] L Liu, D Zhao, F Yu, et al., Performance optimization of differential evolution with slime mould algorithm for multilevel breast cancer image segmentation, Comput. Biol. Med. 138 (2021), 104910.
[4] M Kaur, D Singh, V. Kumar, Drug synergy prediction using dynamic mutation based differential evolution, Curr. Pharm. Des. 27 (8) (2021) 1103–1111.
[5] G Luo, L Zou, Z Wang, et al., A novel kinematic parameters calibration method for industrial robot based on Levenberg-Marquardt and Differential Evolution hybrid algorithm, Rob. Comput. Integr. Manuf. 71 (2021), 102165.
[6] M Pant, H Zaheer, L Garcia-Hernandez, et al., Differential Evolution: A review of more than two decades of research, Eng. Appl. Artif. Intell. 90 (2020), 103479.
[7] R D Al-Dabbagh, F Neri, N Idris, et al., Algorithmic design issues in adaptive differential evolution schemes: review and taxonomy, Swarm Evol. Comput. 43 (2018) 284–311.
[8] K R Opara, J. Arabas, Differential evolution: a survey of theoretical analyses, Swarm Evol. Comput. 44 (2019) 546–558.
[9] J Zhang, AC Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945–958.
[10] R Mallipeddi, P N Suganthan, QK Pan, et al., Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2) (2011) 1679–1696.
[11] G Wu, R Mallipeddi, P N Suganthan, et al., Differential evolution with multi-population based ensemble of mutation strategies, Inform. Sci. 329 (2016) 329–345.
[12] R Tanabe, A. Fukunaga, Evaluating the performance of SHADE on CEC 2013 benchmark problems, in: 2013 IEEE Congress on evolutionary computation, IEEE, 2013, pp. 1952–1959.
[13] V Stanovov, S Akhmedova, E. Semenkin, LSHADE Algorithm with rank-based selective pressure strategy for solving CEC 2017 benchmark problems, in: 2018 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2018, pp. 1–8.
[14] G Sun, G Xu, N. Jiang, A simple differential evolution with time-varying strategy for continuous optimization, Soft Comput. 24 (4) (2020) 2727–2747.
[15] G Sun, B Yang, Z Yang, et al., An adaptive differential evolution with combined strategy for global numerical optimization, Soft Comput. 24 (9) (2020) 6277–6296.
[16] Y Song, D Wu, W Deng, et al., MPPCEDE: multi-population parallel co-evolutionary differential evolution for parameter optimization, Energy Convers. Manage. 228 (2021), 113661.
[17] TN Huynh, DTT Do, J Lee, Q-Learning-based parameter control in differential evolution for structural optimization, Appl. Soft Comput. 107 (2021), 107464.
[18] M Sharma, A Komninos, M López-Ibáñez, et al., Deep reinforcement learning based parameter control in differential evolution, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2019, pp. 709–717.
[19] W Deng, S Shang, X Cai, et al., Quantum differential evolution with cooperative coevolution framework and hybrid mutation strategy for large scale optimization, Knowl.-Base. Syst. 224 (2021), 107080.
[20] J Liu, J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft Comput. 9 (6) (2005) 448–462.
[21] J Brest, S Greiner, B Boskovic, et al., Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (6) (2006) 646–657.
[22] Z H Zhan, Z J Wang, H Jin, et al., Adaptive distributed differential evolution, IEEE Trans. Cybern. 50 (11) (2019) 4633–4647.
[23] M.Z. Ali, N.H. Awad, P.N. Suganthan, R.G. Reynolds, An adaptive multipopulation differential evolution with dynamic population reduction, IEEE Trans. Cybern. 47 (2017) 2768–2779.
[24] W Deng, J Xu, Y Song, et al., Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem, Appl. Soft Comput. 100 (2021), 106724.
[25] Y Li, S Wang, B Yang, An improved differential evolution algorithm with dual mutation strategies collaboration, Exp. Syst. Appl. 153 (2020), 113451.
[26] Z Meng, J S Pan, L. Kong, Parameters with adaptive learning mechanism (PALM) for the enhancement of differential evolution, Knowl.-Base. Syst. 141 (2018) 92–112.
[27] 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.
[28] 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) (2008) 398–417.
[29] Z Tan, K Li, Y Wang, Differential evolution with adaptive mutation strategy based on fitness landscape analysis, Inform. Sci. 549 (2021) 142–163.
[30] C.J. Watkins, P. Dayan, Q-learning, Int. J. Mach. Learn. Cybern. 8 (3–4) (1992) 279–292.
[31] H. Richter, Fitness landscapes: from evolutionary biology to evolutionary computation. Recent Advances in the Theory and Application of Fitness Landscapes, Springer, Berlin, Heidelberg, 2014, pp. 3–31.
[32] KM Malan, AP Engelbrecht, A progressive random walk algorithm for sampling continuous fitness landscapes, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 2507–2514.
[33] S. Yang, Evolutionary computation for dynamic optimization problems, in: Proceedings of the 15th annual conference companion on Genetic and evolutionary computation, 2013, pp. 667–682.
[34] K M Malan, A P Engelbrecht, A survey of techniques for characterising fitness landscapes and some possible ways forward, Inform. Sci. 241 (2013) 148–163.
[35] Y Huang, W Li, F Tian, et al., A fitness landscape ruggedness multiobjective differential evolution algorithm with a reinforcement learning strategy, Appl. Soft Comput. 96 (2020), 106693.
[36] H. Richter, Dynamic fitness landscape analysis. Evolutionary Computation for Dynamic Optimization Problems, Springer, Berlin, Heidelberg, 2013, pp. 269–297.
[37] A W Mohamed, A A Hadi, A M Fattouh, et al., LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems, in: 2017 IEEE Congress on evolutionary computation (CEC), IEEE, 2017, pp. 145–152.
[38] J Brest, M S Maučec, B. Bošković, iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization, in: 2016 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2016, pp. 1188–1195.
[39] J Brest, M S Maučec, B. Bošković, Single objective real-parameter optimization: Algorithm jSO, in: 2017 IEEE congress on evolutionary computation (CEC), IEEE, 2017, pp. 1311–1318.