Progress in Artificial Intelligence (2018) 7:385–398
https://doi.org/10.1007/s13748-018-0155-7

REGULAR PAPER

Reinforcement learning aided parameter control in multi-objective evolutionary algorithm based on decomposition

Weikang Ning[1] · Baolong Guo[1] · Xinxing Guo[1] · Cheng Li[1] · Yunyi Yan[1]

Received: 8 November 2017 / Accepted: 25 June 2018 / Published online: 4 July 2018

Abstract
Multi-objective evolutionary algorithm based on decomposition (MOEA/D) has been successfully applied in solving multi-objective optimization problems. However, the performance of MOEA/D could be severely influenced by its parameter settings. In this paper, we introduce reinforcement learning into MOEA/D as a generic parameter controller. The resulting algorithm, reinforcement learning enhanced MOEA/D (RL-MOEA/D), is used to adaptively control the neighborhood size T and the differential evolutionary operators used in MOEA/D. RL-MOEA/D is first compared with MOEA/D with a random parameter control mechanism and MOEA/Ds with some fixed parameter settings on ten widely used multi-objective test instances. Then, RL-MOEA/D is compared with FRRMAB to show the effectiveness of the proposed algorithm. The experimental results indicate that RL-MOEA/D is very competitive. Finally, the characteristics of RL-MOEA/D are studied.

Keywords Multi-objective optimization · Decomposition · Reinforcement learning · Parameter control

1 Introduction

Without loss of generality, a box-constrained multi-objective optimization problem (MOP) could be presented as:

$$\begin{aligned} \min \quad & \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))^{\mathrm{T}} \\ s.t. \quad & \mathbf{x} \in \Omega = [l_k, u_k]^n \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}}$ is an n-dimensional decision vector which lies in the decision space R^n. l_k, u_k are the lower and upper bounds of x_k $(k = 1, \ldots, n)$, respectively. $\mathbf{F} : \Omega \rightarrow R^m$ consists of m objective functions to be minimized, and R^m is called the objective space.

In solving a MOP, the improvement of one objective often leads to the deterioration of another one. Thus, multi-objective optimization algorithms aim to find the best trade-offs among the objectives. The best trade-offs of a MOP are called the Pareto set (PS), and its image in the objective space is called the Pareto front (PF).

✉ Weikang Ning
ningweikang@126.com

Baolong Guo
blguo@xidian.edu.cn

1 School of Aerospace Science and Technology, Xidian University, Xi'an 710071, China

Multi-objective evolutionary algorithm based on decomposition (MOEA/D) [19] decomposes a MOP into a series of single-objective subproblems with a set of uniformly distributed weighted vectors and solves them simultaneously in a single run. MOEA/D has been proved efficient in recent studies. However, the accuracy of the solutions and the robustness of MOEA/D, as many evolutionary algorithms, rely severely on its parameter settings such as the neighborhood size and the evolutionary operators.

Parameter control methods in evolutionary computing could adapt the parameters of the evolutionary algorithm on the fly based on the feedback of evolution and have been actively studied for a long time. [7] gives a comprehensive survey of parameter control methods in evolutionary algorithms.

Many MOEA/D variants [9,10,20,22] have been proposed to control the parameters of MOEA/D. In [22], Zhao proposed MOEA/D with an ensemble of neighborhood sizes (ENS-MOEA/D) to adapt the neighborhood size T. To adaptively select the operators used during evolution, many adaptive operator selection (AOS) methods [5,9,15] are proposed. Fitness-rate-rank-based multiarmed bandit (FRRMAB) [9] is proposed to select the differential evolutionary (DE) operators for MOEA/D. In [9], the operator selection problem is formulated as a multiarmed bandit problem, which is a simplified reinforcement learning problem. Inspired by

FRRMAB [9], Lin proposed MOEA/D-CDE [10]. In MOEA/D-CDE, both the operators and the parameters of the operators are adaptively controlled. The parameter control mechanism similar to the one of JADE [18] is used in MOEA/D-CDE. Venske proposed ADEMO/D [15] to select different strategies in MOEA/D, and two strategy selection methods including probability matching and adaptive pursuit are tested in ADEMO/D.

However, as a simplified reinforcement learning problem, only one state is considered in multiarmed bandit problem, and this may fail to fully describe the dynamics of the evolution process.

Reinforcement learning (RL) is goal-directed learning through the interaction between the agent and the environment. In RL, a learning agent chooses its action based on the state vector of the environment and the environment will emit a scalar reward and a new state vector describing the environment after the action is implemented on the environment [13]. Several attempts have been made to formulate the parameter control problem in evolutionary algorithms as a reinforcement learning problem [3,6,8,11,12,17]. Recently, the authors have proposed a generic parameter control mechanism with reinforcement learning to control the parameters of evolutionary algorithms in [6]. A regression tree is used to dynamically represent the state since the observable of the environment is continuous and TD(λ) [14] is adopted as the core learning method. The proposed parameter control mechanism based on reinforcement learning is combined with several evolutionary algorithms, and the algorithms are tested on some single-objective problems. The proposed parameter control mechanism is proved promising by comparing it with three other parameter control mechanisms including none (using the default parameters of an evolutionary algorithm), PRAM [16] and random parameter control mechanism.

However, the study of [6] is only limited to single-objective optimization, which is in essence different from multi-objective optimization. In this paper, the reinforcement learning mechanism proposed in [6] is combined with MOEA/D to control two of its parameters (the neighbor size T and the evolutionary operators used to generate new solutions). The immediate reward and state space used in RL-MOEA/D are redefined according to the nature of multi-objective optimization. Experimental results indicate that RL-MOEA/D could outperform MOEA/D with a random control mechanism and it is very promising for the joint control of different parameters. Besides, RL-MOEA/D could also outperform FRRMAB on most test instances.

The rest of this paper is organized as follows. Section 2 details the proposed algorithm. Section 3 presents the experimental results, and Sect. 4 concludes this paper.

2 Proposed algorithm

The proposed algorithm adopts the framework similar to MOEA/D-DRA [20]; thus, a set of evenly spread weighted vectors $W = \{\mathbf{w}^1, \ldots, \mathbf{w}^N\}$ is needed to decompose (1) into N subproblems. In this paper, the Tchebycheff approach that is most widely used in MOEA/D study is adopted (other decomposition methods such as boundary intersection approach [19] could also be adopted) as the decomposition method and the ith subproblem could be defined as:

$$\begin{aligned} \min \quad & g^{te}(\mathbf{x} \mid \mathbf{w}^i, \mathbf{z}^*) = \max_{1\le j\le m}\{w^i_j \cdot |f_j(\mathbf{x}) - z^*_j|\} \\ s.t. \quad & \mathbf{x} \in \Omega \end{aligned} \tag{2}$$

where $\mathbf{z}^* = (z^*_1, \ldots, z^*_m)$ is the ideal reference point, i.e. $z^*_j = \min\{f_j(\mathbf{x}) \mid \mathbf{x} \in \Psi \subset \Omega\}$ for each $j = 1, \ldots, m$, where Ψ denotes the set of solutions that have been encountered during the search of the evolutionary algorithm. $\mathbf{w}^i = \{ w^i_1,\ldots,w^i_m \}(i = 1, \ldots, N)$ is the ith weighted vector in W and $\sum_{j=1}^{m} w^i_j = 1$. All solutions of these N subproblems constitute a good approximation to the PF of (1).

Algorithm 1 presents the general framework of the proposed algorithm. Initially, N closest (based on the Euclidian distance between the corresponding weighted vectors of each subproblem) neighbors of the ith $(i = 1, \ldots, N)$ subproblem are indexed by $B(i)$, where $|B(i)| = N$ and the jth component of $B(i)$ indicates the jth nearest subproblem of the ith subproblem. Then, in line 3 of Algorithm 1, H uniformly distributed subproblems indexed by $I = \{I_1,\ldots,I_k,\ldots,I_H\}$ are selected, where I_k is selected from $\{1, \ldots, N\}$. Similar as $\mathbf{z}^*$, $\mathbf{z}^\dagger = (z^\dagger_1, \ldots, z^\dagger_m)$ and $z^\dagger_j = \max\{f_j(\mathbf{x}) \mid \mathbf{x} \in \Psi \subset \Omega\}$ for each $j = 1, \ldots, m$. $\mathbf{z}^*$ and $\mathbf{z}^\dagger$ are initialized and updated at each iteration. The reinforcement learning controller $RLcontroller$ is initialized by registering two parameters of MOEA/D (neighbor size $T \in \{T_{\min}, \ldots, N/2\}$ and the DE operators). The neighbor size T is uniformly discretized into L intervals, and four DE operators which are used in [9] are adopted. These DE operators are "DE/rand/1/bin," "DE/rand/2/bin," "DE/current-to-rand/2/bin" and "DE/current-to-rand/1/bin." Thus, there are $4 \cdot L$ actions in total and the initial action is chosen randomly from theses actions. For the neighbor size, once the level of this parameter is achieved from $RLcontroller$, the final value T_t is uniformly sampled from those neighborhood sizes belonging to that level. For example, if $T \in \{20, 21, 22, 23, 24, 25\}$ and T is uniformly discretized into $L = 2$ levels, then two intervals are [20, 22.5] and [22.5, 25]. If interval [20, 22.5] is selected by $RLcontroller$, then a random number rnd is uniformly sampled from this interval. For example, if $rnd = 20.3$, then $T_t = \lfloor rnd \rfloor = \lfloor 20.3 \rfloor = 20$ is used as the final value of neighbor size.

Algorithm 1: General framework of the proposed algorithm

```
Input: a stopping criteria; $W$; $\Delta T$; $H$; $N$
1  $t = 0$;
2  Initialize $B(i)$ for each subproblem;
3  Uniformly choose $H$ weighted vectors from $W$ (indexed by $I = \{I_1,\ldots,I_H\}$);
4  Sample $N$ solutions randomly from the search space to form the initial population $P_t$;
5  Initialize $\mathbf{z}^*$ and $\mathbf{z}^\dagger$;
6  Initialize the reinforcement learning controller $RLcontroller$;
7  Get initial random $T_t$ and $DEvariant_t$ from $RLcontroller$;
8  while the stopping criteria is not satisfied do
9      Choose $|S|$ subproblems to be investigated using the MOEA/D-DRA resource allocation mechanism [20];
10     foreach $s \in S$ do
11         Select the parents for the $s$th subproblem from its neighborhood of size $T_t$ and generate a new solution $\mathbf{y}$ with operator $DEvariant_t$;
12         Update the solutions in $P_t$ that belong to the replacement neighborhood of the $s$th subproblem with $\mathbf{y}$;
13         Update $\mathbf{z}^*$ and $\mathbf{z}^\dagger$;
14     $t \leftarrow t + 1$;
15     Calculate the immediate scalar reward $R_t$ according to the relative fitness improvement of each subproblem;
16     Calculate the observable vector $obs_t$ of the population;
17     Update the value (including the state value and action value) of each state recorded in $RLcontroller$ with $R_t$ and $obs_t$;
18     Get new $T_t$ and $DEvariant_t$ from $RLcontroller$ with current observable $obs_t$;
19     if $t \bmod \Delta T = 0$ then
20         Update the utility of each subproblem according to [20];
```

At each iteration, $|S|$ subproblems to be investigated are first selected, where S is a set containing the indexes of the subproblems. For each subproblem indexed by $s \in S$, a new solution $\mathbf{y}$ is generated based on the current action setting in line 11 of Algorithm 1. The new solution $\mathbf{y}$ is also used to update the solutions in the nearest neighborhood (of size T_r) of the sth subproblem in line 12 of Algorithm 1. At the end of each iteration, the immediate scalar reward R_t is calculated as the sum of the relative fitness improvement of each subproblem.

$$R_t = \sum_{i=1}^{N} \frac{g^{te}(\mathbf{x}_{t-1}^i \mid \mathbf{w}^i, \mathbf{z}^*) - g^{te}(\mathbf{x}_t^i \mid \mathbf{w}^i, \mathbf{z}^*)}{g^{te}(\mathbf{x}_{t-1}^i \mid \mathbf{w}^i, \mathbf{z}^*)} \tag{3}$$

where $\mathbf{x}_t^i$ $(i = 1, \ldots, N)$ is the solution corresponding to the ith subproblem at iteration t.

The observable vector obs_t, which contains five observables, is used to describe the current state of the population. The first component of obs_t is the mean age of each subproblem. The age age_i $(i = 1, \ldots, N)$ of each subproblem will be set as zero if the ith subproblem is updated and be incremented by one otherwise. The second observable D^{obj} and the third observable V^{obj} are used to describe the properties of the objective space. For each subproblem indexed by I_k ($I_k \in I$ and $k = 1,\ldots,H$), a density estimator $D_{I_k}^{obj}$ is used to describe the density of the solutions belonging to it. $D_{I_k}^{obj}$ is calculated as the mean Euclidian distance in the objective space between the solution belonging to the I_kth subproblem and the solutions belonging to its nearest neighborhood of size NS. Note that $H = N/NS$ since NS subproblems are in the same group while estimating $D_{I_k}^{obj}$.

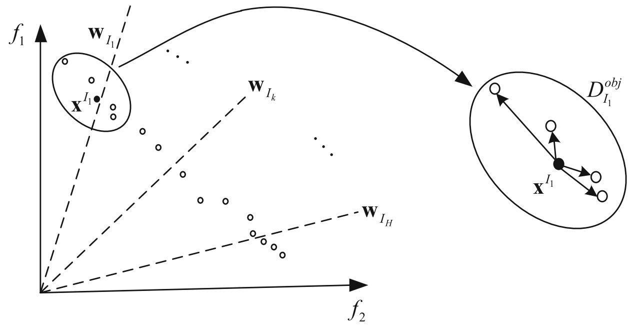


Fig. 1 Illustration of the calculation of $D_{I_1}^{obj}$ in the objective space of dimension two, where $\mathbf{w}_{I_1}$ is the weighted vector corresponding to the I_1th subproblem and $\mathbf{x}^{I_1}$ is the solution corresponding to it

$$D_{I_k}^{obj} = \frac{1}{NS} \sum_{j=1}^{NS} distance^{obj}(\mathbf{x}^{I_k}, \mathbf{x}^{B(I_k)_j}) \tag{4}$$

$$distance^{obj}(\mathbf{x}, \mathbf{y}) = \sum_{h=1}^{m} \left(\frac{x_h - y_h}{z_h^\dagger - z_h^*} \right)^2 \tag{5}$$

where $distance^{obj}(\mathbf{x}, \mathbf{y})$ is the normalized Euclidian distance between solution $\mathbf{x}$ and solution $\mathbf{y}$ in the objective space. m is the number of objectives. $\mathbf{x}^{I_k}$ indicates the solution corresponding to the I_kth subproblem, and $\mathbf{x}^{B(I_k)_j}$ indicates the solution corresponding to the jth nearest neighborhood subproblem of the I_kth subproblem. Figure 1 gives an illustration of the calculation of $D_{I_1}^{obj}$ in the objective space of dimension two.

D^{obj} is the mean of all $D_{I_k}^{obj}$, and V^{obj} is the standard variation of all $D_{I_k}^{obj}$. The fourth observable D^{var} and fifth observable V^{var} are used to describe the properties of the variable space. D^{var} and V^{var} are calculated in similar ways as D^{obj} and V^{obj}, respectively. The only difference lies in that the Euclidian distance in the variable space between two solutions is used.

The reward R_t and observable vector obs_t are then passed to the reinforcement controller *RLcontroller*. *RLcontroller* internally maintains a regression tree to dynamically represent the observable space. Each leaf in the tree is a called a state, and it may contain many observable vectors. The state value and action value for each state are then updated according to R_t and obs_t using SARSA on-policy rule. The leaf node of the tree will be transferred into an internal decision node if a certain criterion is satisfied. Refer [6] for more details of the updating procedure.

Finally, a new action setting (includes T_t and $DEvariant_t$) is selected by $RLcontroller$ according to current observable vector obs_t.

The main contributions of the proposed algorithm are as follows:

(1) The reinforcement learning algorithm proposed in [6] is combined with the MOEA/D framework;
(2) A reward and five observables are defined according to the nature of multi-objective optimization.

3 Experiments

In this section, the proposed algorithm RL-MOEA/D is first compared with MOEA/D with a random parameter control mechanism and MOEA/Ds with some fixed parameter settings. Then, RL-MOEA/D is compared with FRRMAB to show the effectiveness of the proposed parameter control mechanism. Finally, the characteristics of RL-MOEA/D are studied.

3.1 Test instances and performance metrics

Ten test instances (UF1-UF10) from CEC-2009 Special Session and Competition [21] are adopted in this paper. UF1-UF7 are problems with two objectives, and the other three are problems with three objectives.

Two indicators (hypervolume I_{hv} [23] and inverted generational distance I_{IGD} [1]) that quantify both the convergence and spread of the Pareto front approximation are adopted.

Let $\mathbf{r}^*$ be a reference point which denotes an upper bound over all the objectives. It is defined as the biggest objective value of the real Pareto front. Let A be the Pareto front approximation, I_{hv} could be calculated as:

$$I_{hv}(A) = \Lambda\left(\bigcup_{\mathbf{a}\in A}\{\mathbf{x} \,|\, \mathbf{a} \prec \mathbf{x} \prec \mathbf{r}^*\}\right) \tag{6}$$

where Λ is the Lebesgue measure. Higher I_{hv} indicates better convergence and spread performance of the Pareto front approximation.

Let A be the Pareto front approximation and R be the real Pareto front. I_{IGD} is calculated as follows:

$$I_{IGD}(R, A) = \frac{\sum_{\mathbf{r}\in R} d(\mathbf{r}, A)}{|R|} \tag{7}$$

Table 1 Control parameters of the RL controller

TD		State tree		Traces	
ϵ	0.1	A_m	60	λ	0.8
γ	0.8	A_f	0.2	$e_{\min}$	0.001
α	0.9	$D_{\max}$	0.05		
α_0	0.2	p_s	0.1		

where $d(\mathbf{r}, A)$ is the minimum Euclidian distance between $\mathbf{r}$ and the points in A. Lower I_{IGD} indicates better convergence and spread performance of the Pareto front approximation.

3.2 Comparison with MOEA/Ds with different parameter settings

3.2.1 Experiment settings

Thirty independent runs are performed for each algorithm on each test instance. The maximum number of function evaluations is fixed as 300,000 for each algorithm. The population size is set as 600 and 1000 for the test instances with two objectives and three objectives, respectively. ΔT that is used to control the period of updating the utility of each subproblem is set as 50.

The parameters adopted by the reinforcement learning controller are the same as the ones adopted in [6]. Table 1 presents the parameters of the RL controller, where ϵ, γ, α and α_0 represent the probability of ϵ-greedy strategy, discount rate, step-size parameter of SARSA and a parameter of SARSA that prevents Q from dropping too quickly, respectively. λ and $e_{\min}$ represent the trace decay parameter and minimum eligibility, respectively. A_m, A_f, $D_{\max}$ and p_s represent the state's archive threshold, splitting fraction threshold, statistical test threshold and splitting probability of the state tree, respectively.

For RL-MOEA/D, the minimum neighbor size $T_{\min}$ equals 20 and T is discretized into $L = 5$ ranges. The replacement neighborhood size T_r is fixed as 20. Four DE operators, namely "DE/rand/1/bin," "DE/rand/2/bin," "DE/current-to-rand/2/bin" and "DE/current-to-rand/1/bin," are used. For these operators, the following four mutation strategies are used:

(1) DE/rand/1
$\mathbf{v}^i = \mathbf{x}^i + F \times (\mathbf{x}^{r_1} - \mathbf{x}^{r_2})$
(2) DE/rand/2
$\mathbf{v}^i = \mathbf{x}^i + F \times (\mathbf{x}^{r_1} - \mathbf{x}^{r_2}) + F \times (\mathbf{x}^{r_3} - \mathbf{x}^{r_4})$
(3) DE/current-to-rand/2
$\mathbf{v}^i = \mathbf{x}^i + K \times (\mathbf{x}^i - \mathbf{x}^{r_1}) + F \times (\mathbf{x}^{r_2} - \mathbf{x}^{r_3}) + F \times (\mathbf{x}^{r_4} - \mathbf{x}^{r_5})$
(4) DE/current-to-rand/1
$\mathbf{v}^i = \mathbf{x}^i + K \times (\mathbf{x}^i - \mathbf{x}^{r_1}) + F \times (\mathbf{x}^{r_2} - \mathbf{x}^{r_3})$

where $\mathbf{x}^i$ is called the target vector and $\mathbf{v}^i$ is called the donor vector. $\mathbf{x}^{r_1}$, $\mathbf{x}^{r_2}$, $\mathbf{x}^{r_3}$, $\mathbf{x}^{r_4}$ and $\mathbf{x}^{r_5}$ are mutually different solutions selected from the population. F and K are the scaling factor of the mutation operators.

After the mutation operation, the binomial crossover is then applied:

$$\mathbf{u}_j^i = \begin{cases} \mathbf{v}_j^i, & \textit{if rand} \leq CR \textit{ or } j = j_{rand} \\ \mathbf{x}_j^i, & \textit{otherwise} \end{cases} \tag{8}$$

where $\mathbf{u}^i$ is called the trial solution. $\mathbf{u}_j^i$, $\mathbf{v}_j^i$ and $\mathbf{x}_j^i$ represent the jth component of $\mathbf{u}^i$, $\mathbf{v}^i$ and $\mathbf{x}^i$, respectively. $CR \in [0, 1]$ is the crossover rate. *rand* is uniformly sampled from [0, 1] and $j \in \{1, \ldots, n\}$. j_{rand} is an integer randomly chosen from the set $\{1, \ldots, n\}$.

For these four operators, CR and F are fixed as 1.0 and 0.5, respectively. For the third and fourth operators, K is fixed as 0.5. While calculating the observables, the number of samples (NS) is set as 20.

3.2.2 Experiment results

Figures 2 and 3 present the box plots of I_{hv} for the comparison of MOEA/D with different parameter control mechanisms while solving UF1-UF10. *Rand* indicates the random control mechanism. The following eight combinations indicate eight different T-*DEvariant* combinations. For example, 20-0 indicates a parameter setting with $T = 20$ and the first DE variant DE/rand/1/bin. $d2$-3 indicates a parameter setting with $T = N/2$ and the fourth DE variant DE/current-to-rand/1/bin. RL indicates the reinforcement learning control mechanism. Besides, Wilcoxon rank sum test at a significance level of 0.05 is performed between RL and each of other parameter control mechanisms. Since the proposed algorithm is compared with multiple alternatives, Bonferroni correction is adopted. "− −" and "++" indicate that the results obtained by a specific control mechanism are significantly worse and better than the results obtained by RL, respectively. Similar results are obtained for I_{IGD}, and the box plots for I_{IGD} are presented in Figs. 4 and 5.

3.2.3 Discussion of the results

It could be observed from Figs. 2 and 3 that RL-MOEA/D significantly outperforms MOEA/D with random parameter control mechanism on five out of ten test instances. On the other five test instances, they have similar performance.

When RL-MOEA/D is compared with MOEA/D with fixed T-*DEvariant* combinations, the advantage of RL-MOEA/D may not be that obvious at first glance. As a matter of fact, MOEA/Ds with $T = 20$ seem to perform better, especially on UF4, UF8 and UF9. It could also be observed that with a larger $T = N/2$, the performance of MOEA/D generally gets worse. On the other test instances except UF4, UF8 and UF9, RL-MOEA/D performs similar as MOEA/D with $T = 20$. The slightly worse performance of RL-MOEA/D may be contributed to the exploration nature of reinforcement learning. In reinforcement learning, relatively worse parameter settings will also have a probability of being investigated even though a best parameter setting is found.

The performance of different DE variants could also be analyzed according to the experimental results. Different from T, no best DE variant is observed. When $T = 20$, the second, third and fourth DE variant perform the best on UF3/UF7, UF1/UF2/UF9 and UF6/UF10, respectively. When $T = N/2$, the fourth DE variant seems to perform generally better on most test instances, except UF1 and UF4. The joint effects of T and the DE variant seem harder to be obtained. For example, MOEA/D with $T = 20$ and the second DE variant performs the best on UF3. When $T = N/2$, however, the second DE variant performs the worst.

Indeed, there may be a best parameter setting for a specific parameter. However, prior knowledge of the best parameter setting of a specific parameter for a specific problem may be hard to be achieved. Besides, considering the joint effects of different parameters, the best parameter setting may be even more unavailable. In these situations, RL may be a competitive replacement.

3.3 Comparison with FRRMAB

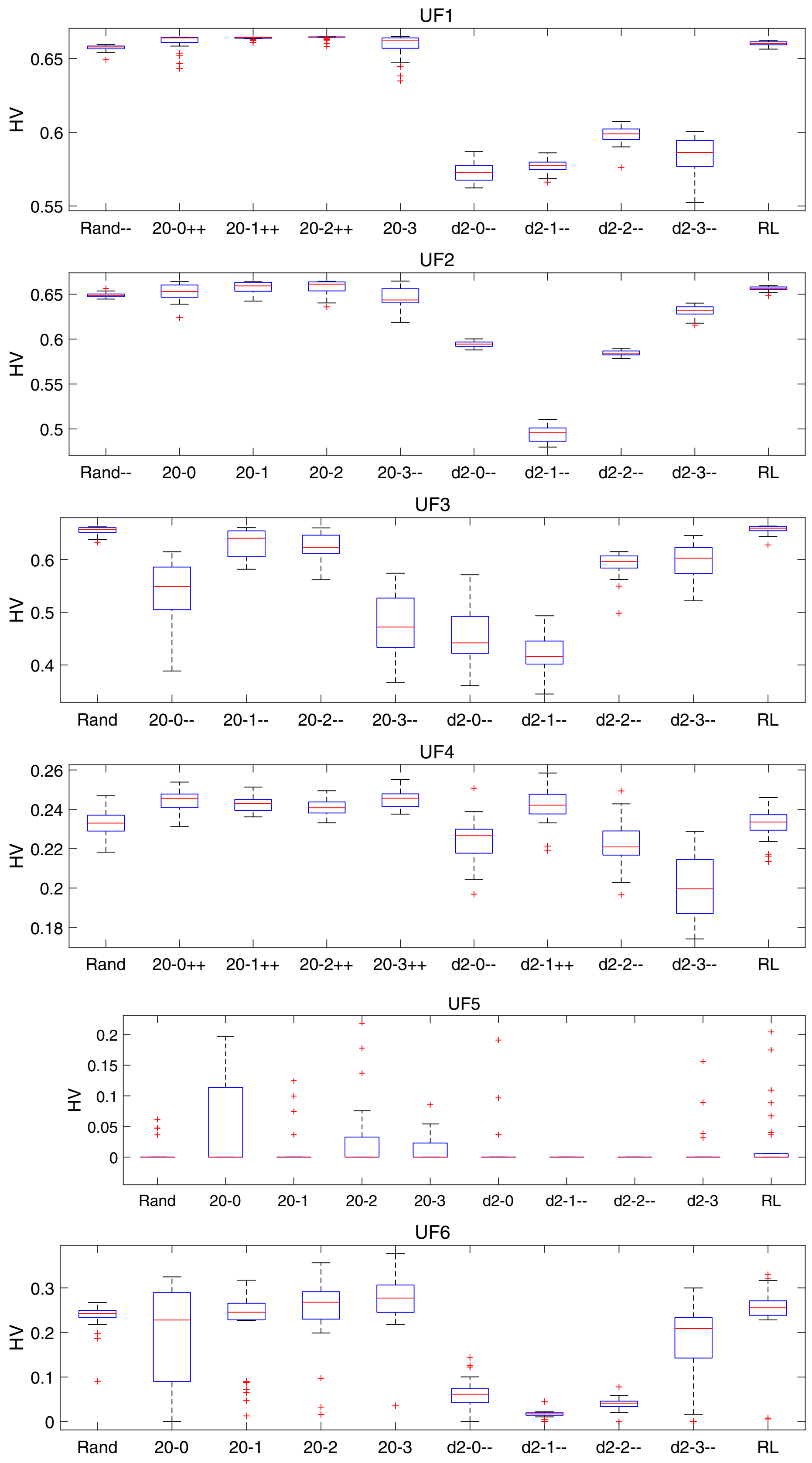
In this section, RL-MOEA/D is compared with FRRMAB [9] to show its effectiveness. Both FRRMAB and ADEMO/D [15] are used for operator selection but FRRMAB is most widely used in recent studies. Thus, FRRMAB is used as the benchmark algorithm in our study. MOEA/D-CDE [10] is based on FRRMAB, and it is used for both operator selection and parameter adaption of the operators. In RL-MOEA/D, the parameters of the operators are fixed; thus, we temporarily omit the comparison between RL-MOEA/D and MOEA/D-CDE.

3.3.1 Experiment settings

Thirty independent runs are performed for each algorithm on each test instance. The maximum number of function evaluations is fixed as 300,000 for each algorithm. The population size is set as 600 and 1000 for the test instances with two objectives and three objectives, respectively. ΔT that is used in MOEA/D-DRA is set as 50.

Other parameter values of RL-MOEA/D are set the same with those used in the last section. In the original FRRMAB, only the operators are adaptively selected and the neighbor size T is fixed as 20. In our study, FRRMAB is extended to adapt both operator and neighbor size selection for fair

Fig. 2 Box plots of I_{hv} for the comparison of MOEA/D with different parameter control mechanisms while solving UF1-UF6. *Rand* indicates the random control mechanism. The following eight combinations indicate eight different T-*DEvariant* combinations. For example, 20-0 indicates a parameter setting with $T = 20$ and the first DE variant. $d2$-3 indicates a parameter setting with $T = N/2$ and the fourth DE variant. *RL* indicates the reinforcement learning control mechanism



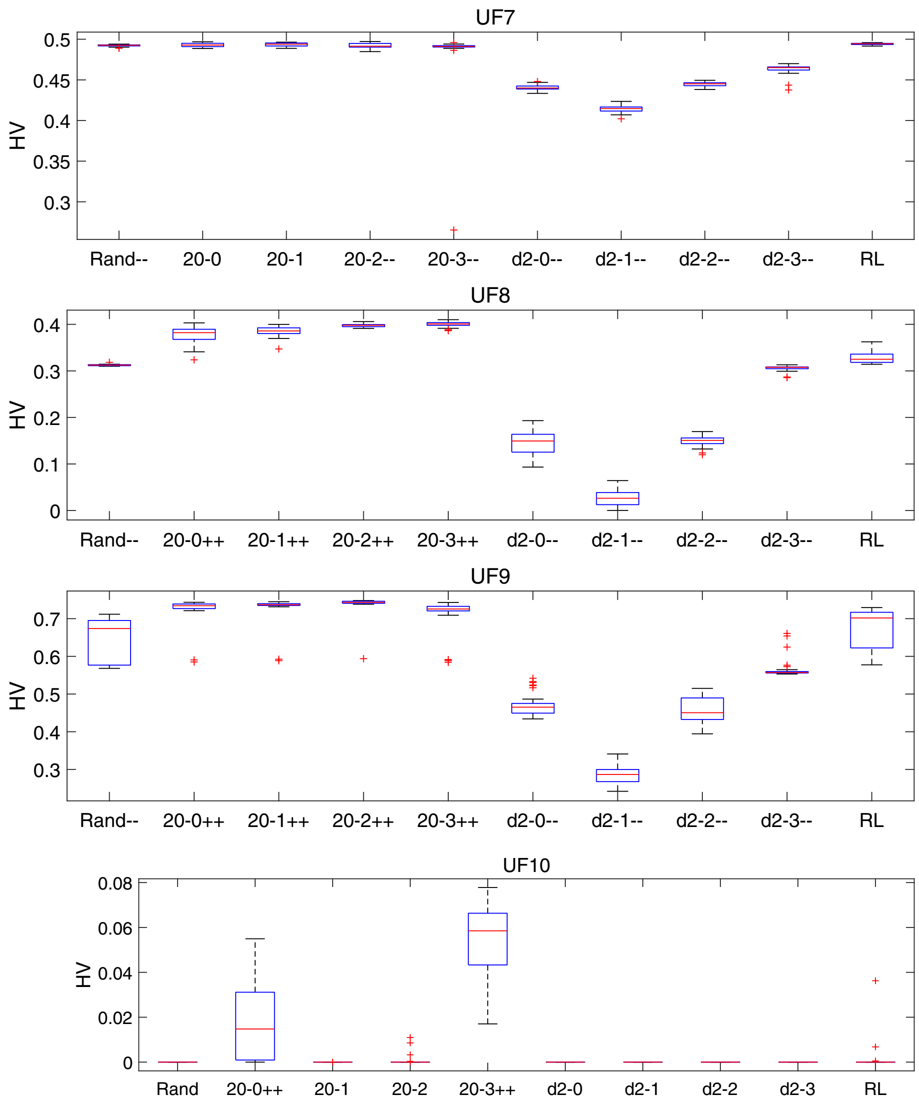


Fig. 3 Box plots of I_{hv} for the comparison of MOEA/D with different parameter control mechanisms while solving UF7-UF10. *Rand* indicates the random control mechanism. The following eight combinations indicate eight different T-*DEvariant* combinations. For example, 20-0 indicates a parameter setting with $T = 20$ and the first DE variant. $d2$-3 indicates a parameter setting with $T = N/2$ and the fourth DE variant. RL indicates the reinforcement learning control mechanism

comparison. The same action space is used by FRRMAB and RL-MOEA/D. Thus, for FRRMAB, the parameter control problem is formulated as a multiarmed bandit problem with 20 arms. As is suggested in [9], the scaling factor, the size of sliding window and the decay factor of FRRMAB are fixed as 5.0, $0.5 \times N$ and 1.0, respectively. The parameter values (CR, F and K) of four operators are set the same with those in the last section.

3.3.2 Experiment results

Tables 2 and 3 present the median and interquartile range (IQR) of I_{hv} and I_{IGD}, respectively. Additionally, Wilcoxon rank sum test is implemented on the I_{hv}/I_{IGD} values obtained over 30 runs and the p values obtained are also presented. Better results in these two tables are set in boldface type.

3.3.3 Discussion of the results

It could be observed from Table 2 that RL-MOEA/D could significantly outperform FRRMAB on test instances except UF4, UF5, UF6 and UF10 with respect to I_{hv}. For UF4 and UF6, these two algorithms achieve similar results. For UF5 and UF10, the better algorithm could not be recognized according to the I_{hv} values.

It could be observed from Table 3 that RL-MOEA/D could significantly outperform FRRMAB on test instances except UF4, UF5 and UF6 with respect to I_{IGD}. For UF4, UF5 and UF6, these two algorithms achieve similar results. For UF4, UF5 and UF6, the medians achieved by RL-MOEA/D are also better than those achieved by FRRMAB.

On the whole, it could be observed that RL-MOEA/D is competitive compared with FRRMAB.

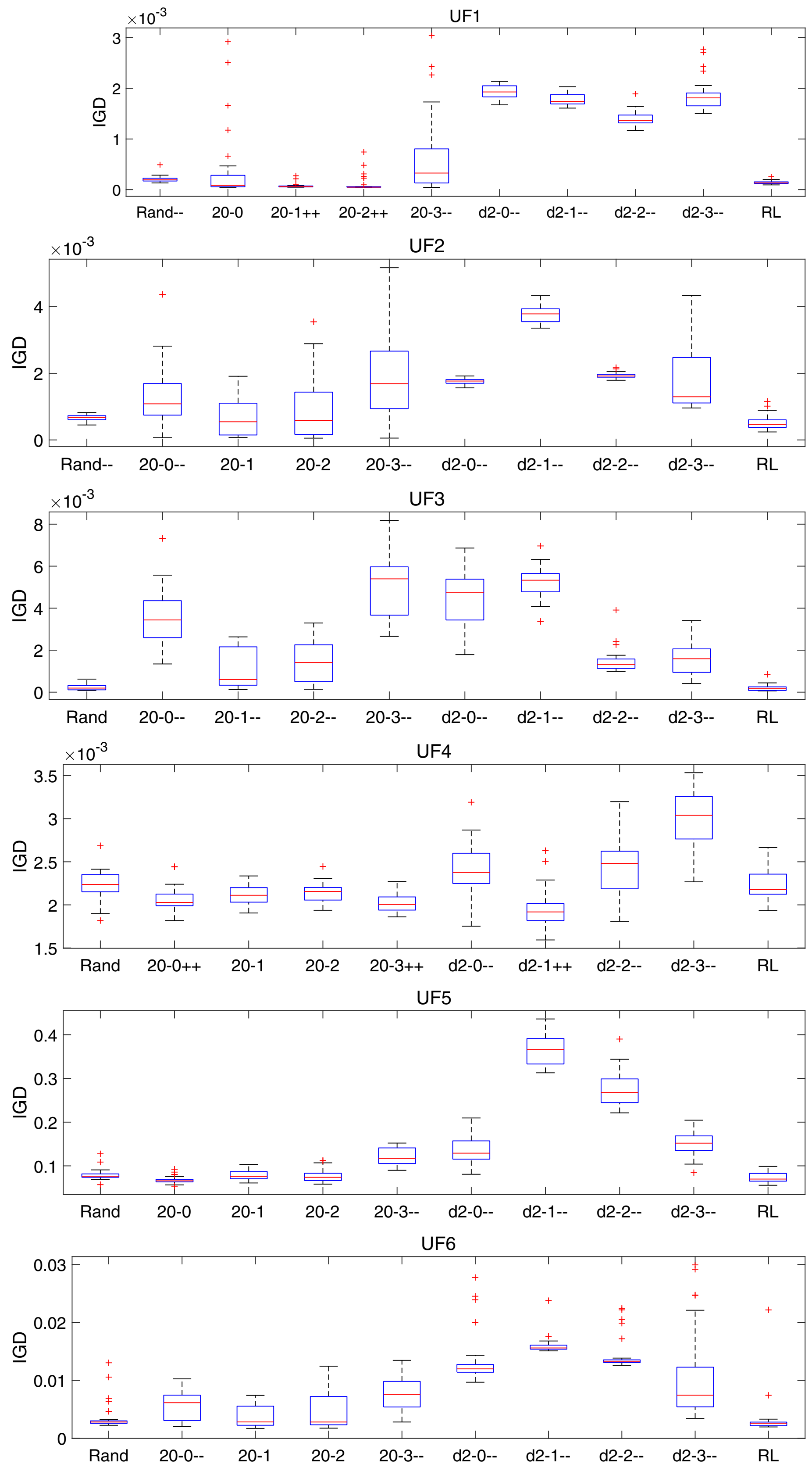


Fig. 4 Box plots of I_{IGD} for the comparison of MOEA/D with different parameter control mechanisms while solving UF1-UF6. *Rand* indicates the random control mechanism. The following eight combinations indicate eight different T-*DEvariant* combinations. For example, 20-0 indicates a parameter setting with $T = 20$ and the first DE variant. $d2$-3 indicates a parameter setting with $T = N/2$ and the fourth DE variant. RL indicates the reinforcement learning control mechanism

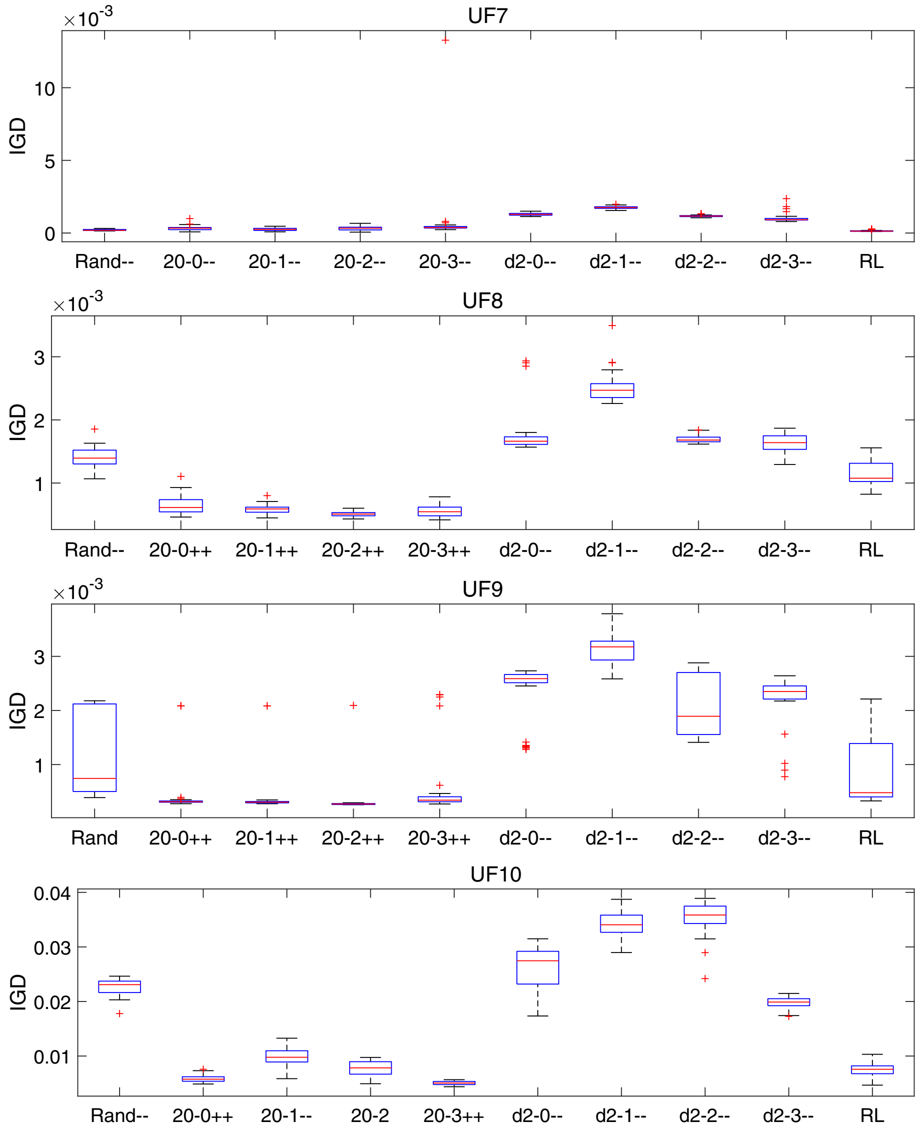


Fig. 5 Box plots of I_{IGD} for the comparison of MOEA/D with different parameter control mechanisms while solving UF7-UF10. *Rand* indicates the random control mechanism. The following eight combinations indicate eight different T-*DEvariant* combinations. For example, 20-0 indicates a parameter setting with $T = 20$ and the first DE variant. $d2$-3 indicates a parameter setting with $T = N/2$ and the fourth DE variant. RL indicates the reinforcement learning control mechanism

3.4 Analysis of RL-MOEA/D

In this section, the behavior of RL-MOEA/D is further analyzed. The obtained results may help us understand RL-MOEA/D better. First, the selection frequency of each action during the evolution of RL-MOEA/D is presented as a heatmap, which could help us identify the most frequently selected action easily. Second, the actions selected by different states are also presented, which may indicate why RL-MOEA/D could outperform FRRMAB.

3.4.1 Selection frequency of different actions

The selection frequency of each action at different stages of RL-MOEA/D is presented in Fig. 6. Due to the space limitation, only the results for UF2, UF4, UF6, UF8 and UF10 are given.

The actions with action ID 1 to action ID 5 correspond to the parameter settings using the first DE variant combined with five different neighbor size ranges (L = 5). The actions with action ID 16 to action ID 20 correspond to the parameter settings using the fourth DE variant combined with five different neighbor size ranges. The relationship between an action ID and a specific parameter setting thus could be inferred.

The selection frequency for an action is recorded every 5000 function evaluations.

It could be observed from Fig. 6 that for UF2, UF6 and UF8, the action with action ID 16 (corresponds to the parameter setting with the first neighbor size range and the fourth operator) is more preferable at early stages of evolution. At later stages of evolution, no action could dominate all other actions. For UF4, no action is in the dominant position over the whole evolution process. For UF10, the action with action

Table 2 $Median_{IQR}$ of I_{hv} and the p value of the Wilcoxon rank sum test implemented on I_{hv}

	FRRMAB	RL	p value
UF1	$6.58e{-}01_{1.9e-03}$	$\mathbf{6.60e{-}01}_{2.0e-03}$	6.01e−8
UF2	$6.50e{-}01_{4.9e-03}$	$\mathbf{6.57e{-}01}_{3.1e-03}$	3.50e−9
UF3	$6.56e{-}01_{1.2e-02}$	$\mathbf{6.59e{-}01}_{7.7e-03}$	0.01
UF4	$\mathbf{2.35e{-}01}_{6.6e-03}$	$2.34e{-}01_{7.9e-03}$	0.42
UF5	$0.00e{+}00_{0.0e+00}$	$0.00e{+}00_{5.5e-03}$	0.20
UF6	$2.47e{-}01_{2.5e-02}$	$\mathbf{2.55e{-}01}_{3.3e-02}$	0.25
UF7	$4.93e{-}01_{9.7e-04}$	$\mathbf{4.94e{-}01}_{1.4e-03}$	2.32e−6
UF8	$3.13e{-}01_{2.4e-03}$	$\mathbf{3.25e{-}01}_{1.8e-02}$	1.29e−9
UF9	$6.83e{-}01_{1.2e-01}$	$\mathbf{7.02e{-}01}_{9.4e-02}$	0.003
UF10	$0.00e{+}00_{0.0e+00}$	$0.00e{+}00_{0.0e+00}$	0.08

Table 3 $Median_{IQR}$ of I_{IGD} and the p value of the Wilcoxon rank sum test implemented on I_{IGD}

	FRRMAB	RL	p value
UF1	$1.82e{-}04_{4.4e-05}$	$\mathbf{1.32e{-}04}_{3.2e-05}$	9.53e−7
UF2	$6.71e{-}04_{1.8e-04}$	$\mathbf{4.67e{-}04}_{2.3e-04}$	4.46e−4
UF3	$2.14e{-}04_{2.8e-04}$	$\mathbf{1.62e{-}04}_{1.7e-04}$	0.04
UF4	$2.23e{-}03_{2.2e-04}$	$\mathbf{2.18e{-}03}_{2.3e-04}$	0.66
UF5	$7.01e{-}02_{1.2e-02}$	$\mathbf{6.98e{-}02}_{1.8e-02}$	0.95
UF6	$2.70e{-}03_{6.0e-04}$	$\mathbf{2.60e{-}03}_{6.2e-04}$	0.35
UF7	$1.91e{-}04_{3.6e-05}$	$\mathbf{1.31e{-}04}_{4.2e-05}$	7.74e−6
UF8	$1.33e{-}03_{3.1e-04}$	$\mathbf{1.08e{-}03}_{2.9e-04}$	0.001
UF9	$6.72e{-}04_{1.6e-03}$	$\mathbf{4.80e{-}04}_{9.9e-04}$	0.06
UF10	$2.26e{-}02_{1.9e-03}$	$\mathbf{7.57e{-}03}_{1.4e-03}$	3.02e−11

ID 16 is in the dominant position over the whole evolution process. Also, it could be observed from Figs. 3 and 5 that for UF10, the algorithm with fixed parameter setting 20-3 (the neighbor size is 20 and the fourth operator is used) performs the best among all parameter settings. This indicates that the best action is indeed found by RL-MOEA/D for UF10.

3.4.2 Actions selected by different states

The normalized Q value of each action for each state in the regression tree at the end of the iteration of RL-MOEA/D is presented in Fig. 7. Each state is a leaf node in the regression tree used by RL-MOEA/D. For each state, 20 actions are available. For a state, the normalized Q value for the jth ($j = 1, \ldots, 20$) action is noted as $\overline{Q}_j$ and it is calculated as follows:

$$\overline{Q}_j = \frac{Q_j}{\sum_{i=1}^{20} Q_i} \tag{9}$$

For space limitation, only the results for UF2, UF4, UF6, UF8 and UF10 are given.

It could be observed from Fig. 7 that for each test instance, there are some states with more preferable actions. For different states, the selected action is also not always the same. This may indicate why RL-MOEA/D could outperform FRRMAB on some test instances.

3.4.3 Access frequency of different states

The access frequency of different states in RL-MOEA/D is analyzed in this section. It should be noted, however, that a dynamic regression tree is used in RL-MOEA/D and the states (leaves of the tree) are always changing with evolution. To get the access frequency of each state, the dynamic tree is freezed (stop the split operation even though many observables hit the same state) during a fixed period in the middle of the evolution of RL-MOEA/D. The access frequency of each state in ascending order and the Q value of each action for the states with nonzero access frequencies are presented in Fig. 8.

It could be observed from Fig. 8 that for UF2, UF4, UF6 and UF8, the number of states with nonzero access frequencies is rather small. For these most frequently accessed states, several actions preferred are observed and this may indicate that these states need further division to be distinguished.

For UF10, more states with nonzero access frequencies are observed and different actions are strongly preferred for different states. This may indicate why RL-MOEA/D is more advantageous on UF10 compared with FRRMAB.

4 Conclusions

In this paper, a generic parameter control mechanism based on reinforcement learning is introduced into MOEA/D to control its neighborhood size T and DE operators. A reward and five observables are defined according to the nature of multi-objective optimization. The proposed algorithm, RL-MOEA/D, provides a generic framework for the control of different parameters in MOEA/D.

RL-MOEA/D is tested and compared with some other algorithms on ten test instances (UF1-UF10) that are most widely used in recent studies of MOEA/D. By comparing RL-MOEA/D with the algorithms using fixed parameter settings, it could be observed that for UF1-UF10, the algorithms with neighbor size $T = 20$ seem to perform better. The performance of RL-MOEA/D is slightly worse than this best parameter setting on some test instances. Though 20 is the empirical value of T, this value may not suit other test instances and RL-MOEA/D is more applicable under this circumstance. By observing the performance of four operators, it could be found that no operator could dominate all

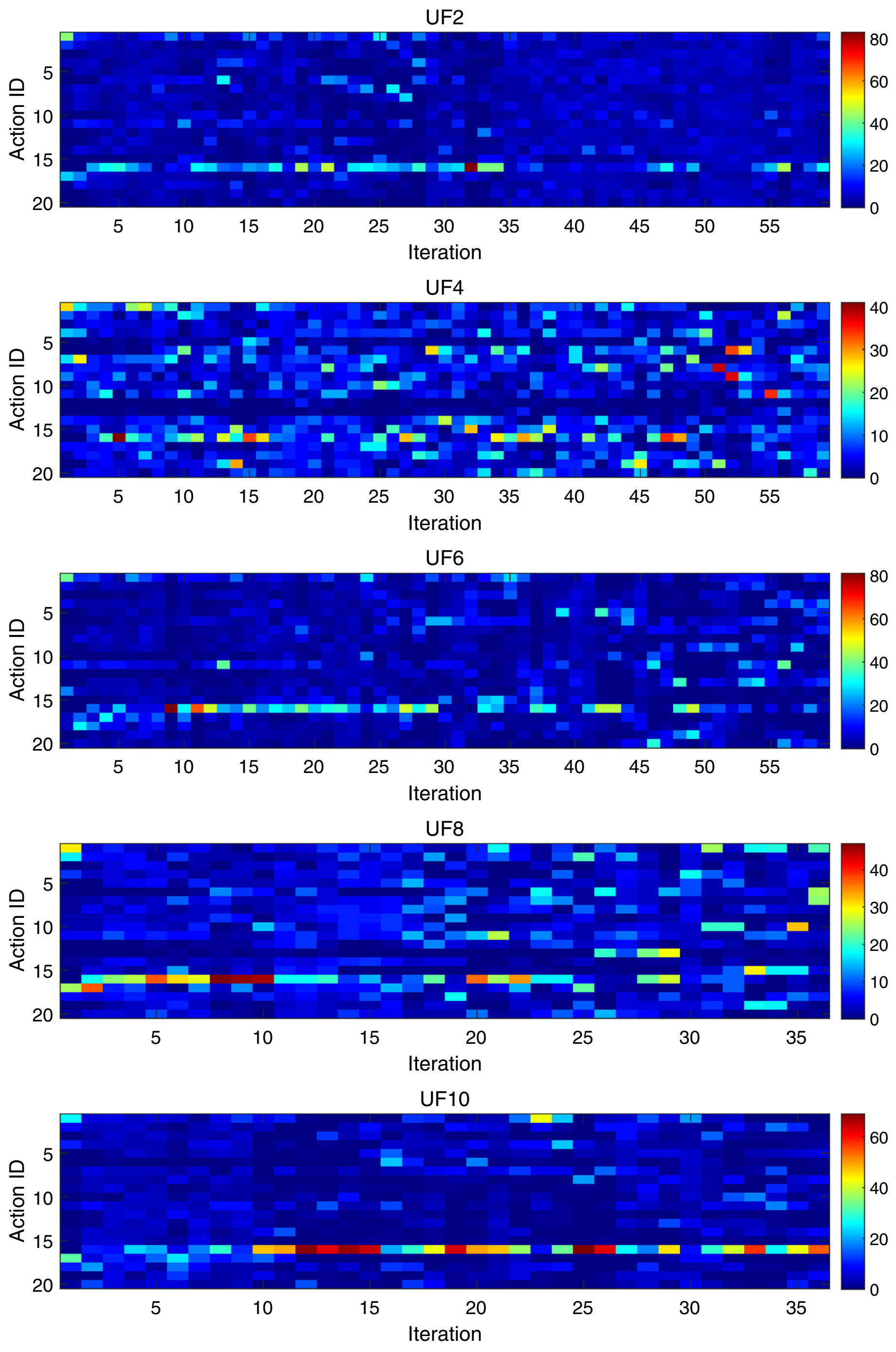


Fig. 6 Evolution of the selection frequency of each action in RL-MOEA/D

other operators on all test instances. For UF5, the PF of which is discontinuous and DE/rand/1/bin seems to perform better. For UF10, which has three objectives and is a multimodal extension of UF8, DE/current-to-rand/1/bin seems to perform better. By comparing RL-MOEA/D with FRRMAB, it could be observed that RL-MOEA/D could outperform FRRMAB on most test instances. Finally, by analyzing the characteristics of RL-MOEA/D, it could be observed that different actions are selected by different states in RL-MOEA/D and this may explain why RL-MOEA/D performs better than FRRMAB.

However, the proposed algorithm RL-MOEA/D is just our preliminary work and a lot could be studied in the future. First, the combination of some prior knowledge (such as the

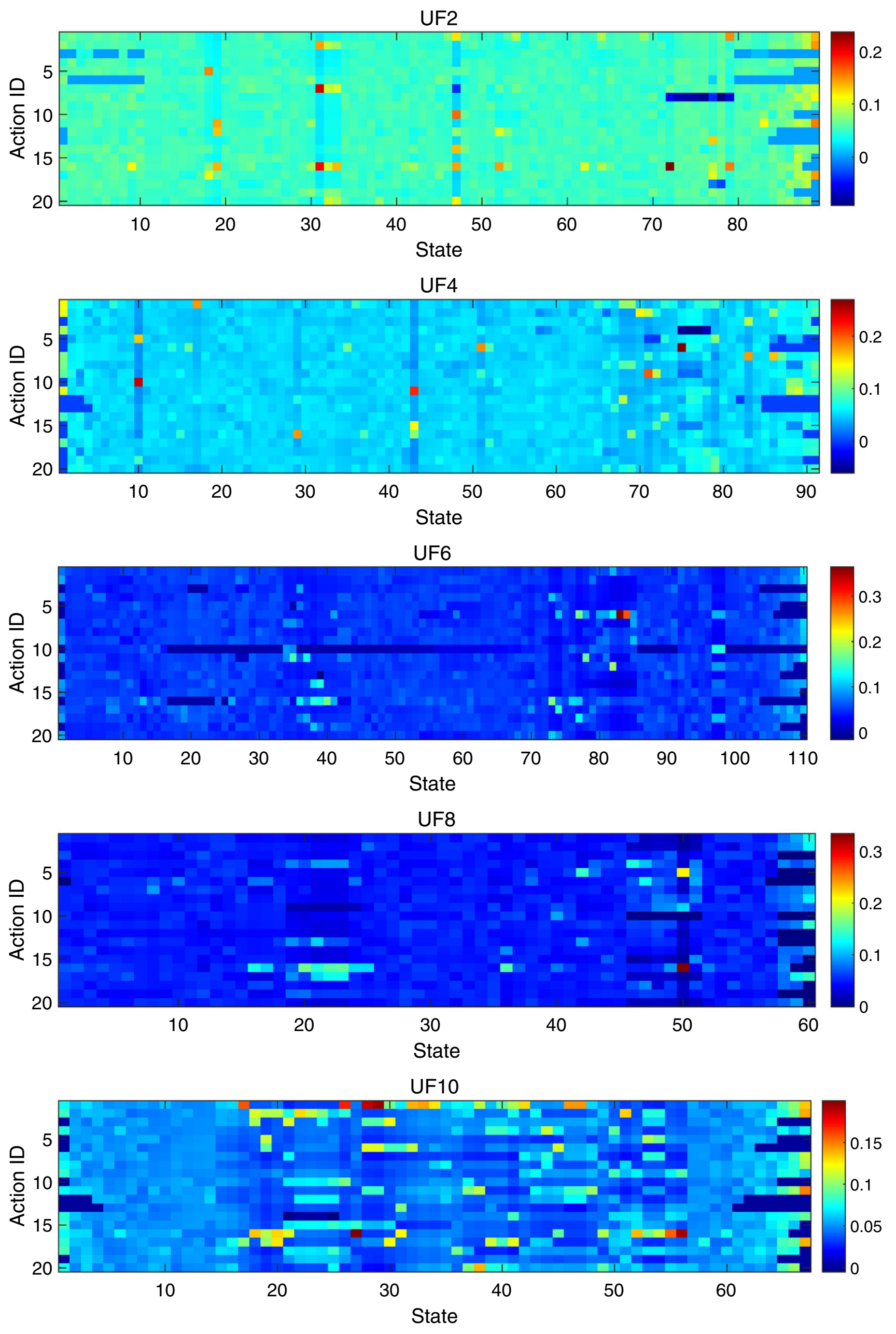


Fig. 7 Normalized Q value of each action for each state in the regression tree at the end of the iteration of RL-MOEA/D

best empirical setting for a specific parameter) with RL is expected to improve the learning efficiency of RL. Thus, it is worthwhile to try different ways to utilize this prior knowledge. Second, with more and more parameters to be controlled, discretization of each parameter may result in a too large action space, which may influence the learning efficiency. In [12], the authors proposed a mechanism to dynamically discretize the parameter range. Some other reinforcement learning techniques that are used for handling large action space (including continuous action space) are worth a try. Third, the observables used may have a significant influence on the performance of a RL controller. Thus, as is recommended by [6], it is worthwhile to try other observables against the background of multi-objective optimization. Some landscape analysis techniques [2,4] may provide some inspirations. Finally, future studies of the rela-

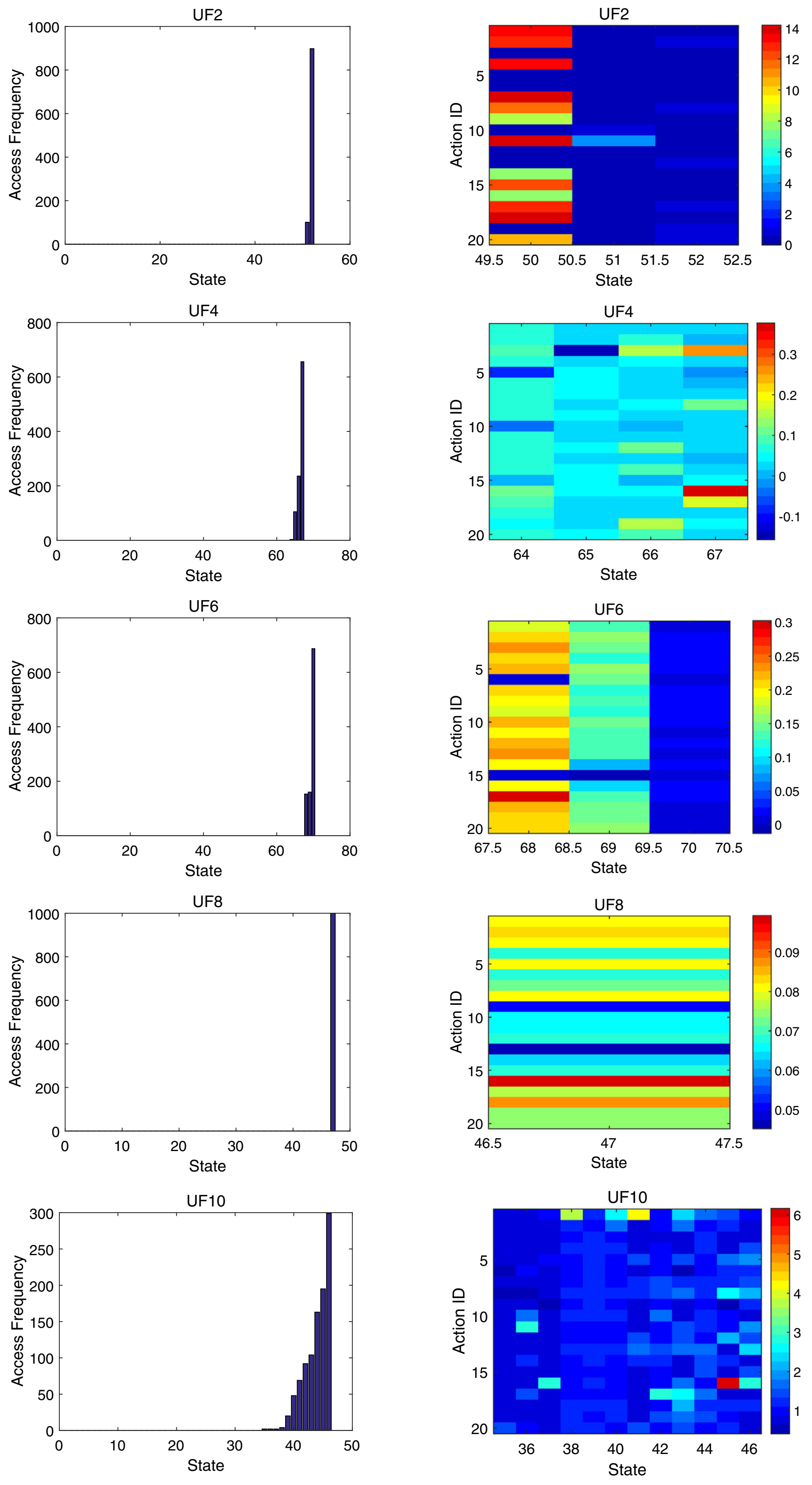


Fig. 8 Access frequency of each state in ascending order (left column) and the Q value of each action for the states with nonzero access frequencies (right column)

tionship between the states and their actions in RL-MOEA/D are desired since no obvious pattern is discovered in our study.

Acknowledgements This work is supported by the National Natural Science Foundation of China under grant number 61571346. The authors would like to thank the reviewers and the editor for their comments for improving this paper.

References

1. Bosman, P.A.N., Thierens, D.: The balance between proximity and diversity in multiobjective evolutionary algorithms. IEEE Trans. Evol. Comput. **7**(2), 174–188 (2003). https://doi.org/10.1109/TEVC.2003.810761
2. Consoli, P.A., Mei, Y., Minku, L.L., Yao, X.: Dynamic selection of evolutionary operators based on online learning and fitness landscape analysis. Soft. Comput. **20**(10), 3889–3914 (2016)
3. Eiben, A.E., Horvath, M., Kowalczyk, W., Schut, M.C.: Reinforcement learning for online control of evolutionary algorithms. In: International Workshop on Engineering Self-Organising Applications, pp. 151–160 (2006)
4. Ginley, B.M., Maher, J., O'Riordan, C., Morgan, F.: Maintaining healthy population diversity using adaptive crossover, mutation, and selection. IEEE Trans. Evol. Comput. **15**(5), 692–714 (2011). https://doi.org/10.1109/tevc.2010.2046173
5. Goncalves, R.A., Almeida, C.P., Pozo, A.: Upper confidence bound (UCB) algorithms for adaptive operator selection in MOEA/D. In: Evolutionary Multi-criterion Optimization, Lecture Notes in Computer Science, pp. 411–425. Springer, Cham (2015)
6. Karafotias, G., Eiben, A.E., Hoogendoorn, M.: Generic parameter control with reinforcement learning. In: Conference on Genetic and Evolutionary Computation, pp. 1319–1326 (2014)
7. Karafotias, G., Hoogendoorn, M., Eiben, A.: Parameter control in evolutionary algorithms: trends and challenges. IEEE Trans. Evol. Comput. **19**(2), 167–187 (2015). https://doi.org/10.1109/TEVC.2014.2308294
8. Karafotias, G., Hoogendoorn, M., Eiben, A.E.: Evaluating reward definitions for parameter control. In: European Conference on the Applications of Evolutionary Computation, pp. 667–680 (2015)
9. Li, K., Fialho, A., Kwong, S., Zhang, Q.: Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **18**(1), 114–130 (2014). https://doi.org/10.1109/TEVC.2013.2239648
10. Lin, Q., Liu, Z., Yan, Q., Du, Z., Coello, C.A.C., Liang, Z., Wang, W., Chen, J.: Adaptive composite operator selection and parameter control for multiobjective evolutionary algorithm. Inf. Sci. **339**, 332–352 (2016). https://doi.org/10.1016/j.ins.2015.12.022
11. Muller, S.D., Schraudolph, N.N., Koumoutsakos, P.D.: Step size adaptation in evolution strategies using reinforcement learning. In: Proceedings of the World on Congress on Computational Intelligence, vol. 1, pp. 151–156. IEEE Computer Society, Los Alamitos, CA, USA (2002). https://doi.org/10.1109/CEC.2002.1006225
12. Rost, A., Petrova, I., Buzdalova, A.: Adaptive parameter selection in evolutionary algorithms by reinforcement learning with dynamic discretization of parameter range. In: Genetic and Evolutionary Computation Conference Companion, pp. 141–142 (2016)
13. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
14. Sutton, R.S.: Learning to predict by the methods of temporal differences. Mach. Learn. **3**(1), 9–44 (1988). https://doi.org/10.1023/A:1022633531479
15. Venske, S.M., Gonçalves, R.A., Delgado, M.R.: ADEMO/D: multiobjective optimization by an adaptive differential evolution algorithm. Neurocomputing **127**(127), 65–77 (2014)
16. Wong, Y.Y., Lee, K.H., Leung, K.S., Ho, C.W.: A novel approach in parameter adaptation and diversity maintenance for genetic algorithms. Soft. Comput. **7**(8), 506–515 (2003)
17. Zhang, H., Lu, J.: Adaptive evolutionary programming based on reinforcement learning. Inf. Sci. **178**(4), 971–984 (2008). https://doi.org/10.1016/j.ins.2007.09.026
18. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009)
19. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**(6), 712–731 (2007). https://doi.org/10.1109/TEVC.2007.892759
20. Zhang, Q., Liu, W., Li, H.: The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In: IEEE Congress on Evolutionary Computation, 2009. CEC '09, pp. 203–208 (2009). https://doi.org/10.1109/CEC.2009.4982949
21. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P.N., Liu, W., Tiwari, S.: Multiobjective optimization test instances for the CEC 2009 special session and competition. University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report (2008)
22. Zhao, S.Z., Suganthan, P., Zhang, Q.: Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes. IEEE Trans. Evol. Comput. **16**(3), 442–446 (2012). https://doi.org/10.1109/TEVC.2011.2166159
23. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999). https://doi.org/10.1109/4235.797969