Multi-agent Dynamic Algorithm Configuration

Anonymous Author(s)
Affiliation
Address
email

A Details of MA-DAC

A.1 Details of MOEA/D

First, we give a brief introduction to multi-objective optimization problems (MOPs), which can be defined as

$$\min \ \boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})) \quad \text{s.t.} \quad \boldsymbol{x} \in \Omega, \tag{1}$$

where $\boldsymbol{x} = (x_1, \ldots, x_D)$ is a solution, $\boldsymbol{F} : \Omega \to \mathbb{R}^m$ constitutes m objective functions, $\Omega = [x_i^L, x_i^U]^D \subseteq \mathbb{R}^D$ is the solution space, and $\mathbb{R}^m$ is the objective space.

Definition A.1. A solution $\boldsymbol{x}^*$ is Pareto-optimal with respect to Eq. (1), if $\nexists \boldsymbol{x} \in \Omega$ such that $\forall i : f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}^*)$ and $\exists i : f_i(\boldsymbol{x}) < f_i(\boldsymbol{x}^*)$. The set of all Pareto-optimal solutions is called Pareto-optimal set (PS). The set of the corresponding objective vectors of PS, i.e., $\{\boldsymbol{F}(\boldsymbol{x}) \mid \boldsymbol{x} \in \text{PS}\}$, is called Pareto front (PF).

Instead of focusing on a single optimal solution in single-objective optimization, the goal of MOP is to find at least one Pareto-optimal solution for each objective vector in the PF. However, as the size of PF can be prohibitively large or even infinite, it is often to find a set of solutions that can approximate the PS well, i.e., the set of their objective vectors can approximate the PF well.

Evolutionary algorithms have demonstrated their effectiveness in solving MOPs. Their population-based nature can approximate the Pareto optimal solutions within one execution, with each solution in the population representing a unique trade-off between the objectives. MOEA/D [17] is a representative multi-objective evolutionary algorithm. MOEA/D converts an MOP into a number of single-objective sub-problems through a number of weights, where neighboring solutions work cooperatively for the optimal solutions for the single-objective sub-problems. Note that an optimal solution for a single-objective sub-problem must be Pareto optimal for the MOP.

MOEA/D consists of two major processes, i.e., *decomposition* and *collaboration* [17, 14, 5]. In decomposition, MOEA/D transforms the task of approximating the PF into a number of sub-problems through a number of weights and an aggregation function. There have been several aggregation functions for MOEA/D. Here, we introduce the common Tchebycheff approach (TCH) that is also used in this paper. Given a weight vector $\boldsymbol{w} = (w_1, \ldots, w_m)$ where $w_i \geq 0, \forall i \in \{1, \ldots, m\}$ and $\sum_{i=1}^m w_i = 1$, the sub-problem by TCH is formulated as

$$\min_{\boldsymbol{x} \in \Omega} \ g(\boldsymbol{x} \mid \boldsymbol{w}, \boldsymbol{z}^*) = \max_{1 \leq i \leq m} \{w_i \cdot |f_i(\boldsymbol{x}) - z_i^*|\}, \tag{2}$$

where $\boldsymbol{z}^* = (z_1^*, \ldots, z_m^*)$ is the ideal point consisting of the best objective values obtained so far.

The basic idea of collaboration is that neighboring sub-problems are more likely to share similar properties, e.g., similar objective functions and/or optimal solutions [5]. In particular, the neighborhood of a sub-problem is determined by the Euclidean distance of its corresponding weight vector with respect to the others and the hyperparameter *neighborhood size*: if the distance between two sub-problems is smaller than the neighborhood size, they are the neighborhood of each other. In the mating selection of a sub-problem, the parent solutions are randomly selected from its neighborhood,

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Algorithm 1: MOEA/D

Parameters: Population size N, number T of iterations

1 Initialize a population $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$ of solutions, and a corresponding set $W = \{\boldsymbol{w}^{(i)}\}_{i=1}^{N}$ of weight vectors ;
2 $t = 0$;
3 **while** $t < T$ **do**
4 **for** $i = 1 : N$ **do**
5 Randomly select parent solutions from the neighborhood of $\boldsymbol{w}^{(i)}$, denoted as $\Theta^{\boldsymbol{w}^{(i)}}$;
6 Use crossover and mutation operators to generate an offspring solution $\boldsymbol{x}'^{(i)}$;
7 Evaluate the offspring solution to obtain $\boldsymbol{F}(\boldsymbol{x}'^{(i)})$;
8 Update the ideal point $\boldsymbol{z}^*$. That is, for any $j \in \{1, 2, \ldots, m\}$, if $f_j(\boldsymbol{x}'^{(i)}) < \boldsymbol{z}_j^*$, then $\boldsymbol{z}_j^* = f_j(\boldsymbol{x}'^{(i)})$;
9 Update the corresponding solution of each sub-problem within $\Theta^{\boldsymbol{w}^{(i)}}$ by $\boldsymbol{x}'^{(i)}$. That is, for each $\boldsymbol{w}^{(j)} \in \Theta^{\boldsymbol{w}^{(i)}}$, if $g(\boldsymbol{x}'^{(i)} \mid \boldsymbol{w}^{(j)}, \boldsymbol{z}^*) < g(\boldsymbol{x}^{(j)} \mid \boldsymbol{w}^{(j)}, \boldsymbol{z}^*)$, then $\boldsymbol{x}^{(j)} = \boldsymbol{x}'^{(i)}$
10 **end**
11 $t = t + 1$
12 **end**

and the newly generated offspring solution is used to update the solutions of sub-problems within the same neighborhood.

The procedure of MOEA/D used in this paper is described in Algorithm 1. Firstly, it generates a population $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$ of solutions with size N, associated with N weight vectors $\{\boldsymbol{w}^{(i)}\}_{i=1}^{N}$ in line 1. A weight vector $\boldsymbol{w}^{(i)}$ corresponds to a single-objective sub-problem, and $\boldsymbol{x}^{(i)}$ is the current best solution associated with this sub-problem. Then, in each iteration (i.e., lines 4–9) of MOEA/D, for each sub-problem, it selects parent solutions from the neighborhood, generates an offspring solution by reproduction operators, and updates the solutions of the sub-problem and its neighboring sub-problem(s). After obtaining parent solutions in line 5, it uses reproduction operators (e.g., simulated binary crossover (SBX) operator or differential evolution (DE) operator) and polynomial mutation (PM) operator [17], to generate an offspring solution $\boldsymbol{x}'^{(i)}$ in line 6. Then, it evaluates the offspring solution and obtain the objective vector $\boldsymbol{F}(\boldsymbol{x}'^{(i)})$ in line 7. Finally, it uses the offspring solution to update the ideal point $\boldsymbol{z}^*$ in line 8 and the solutions of the sub-problems within the neighborhood $\Theta^{\boldsymbol{w}^{(i)}}$ of the current sub-problem $\boldsymbol{w}^{(i)}$ in line 9. For each $j \in \{1, 2, \ldots, m\}$, z_j^* is the best found value of the j-th objective f_j, and thus if $f_j(\boldsymbol{x}'^{(i)})$ is better, i.e., $f_j(\boldsymbol{x}'^{(i)}) < z_j^*$, then z_j^* will be udpated accordingly. For each sub-problem $\boldsymbol{w}^{(j)}$ within $\Theta^{\boldsymbol{w}^{(i)}}$, if $\boldsymbol{x}'^{(i)}$ is better than its corresponding solution $\boldsymbol{x}^{(j)}$, i.e., $g(\boldsymbol{x}'^{(i)} \mid \boldsymbol{w}^{(j)}, \boldsymbol{z}^*) < g(\boldsymbol{x}^{(j)} \mid \boldsymbol{w}^{(j)}, \boldsymbol{z}^*)$, then $\boldsymbol{x}^{(j)}$ will be udpated accordingly.

A.2 State formulation

The state of our proposed benchmark, i.e., MaMo, can be divided into three parts.

1. The first part (i.e., indexes 0–1 in Table S1) contains the features of the problem instance, i.e., the numbers of objectives and variables.
2. The second part (i.e., indexes 2–3 in Table S1) contains the features of the optimization process, i.e., how much computational budget has been used and how many steps of the algorithm have not made any progress, i.e., stagnant count.
3. In the third part (i.e., indexes 4–21 in Table S1), we use several indicators, i.e., hypervolume [18], the ratio of non-dominated solutions in the population, and the average distance of the solutions [7], to reflect the state of the current population. For each indicator, we also use the gap between the current value and the value corresponding to the last population to reflect the immediate evolutionary progress. Besides, we use statistic metrics (i.e., the mean and variance) of these indicators in the last five steps and all steps from the beginning to characterize the short and long histories of the optimization, respectively.

The values of all features (except the number of objectives and the number of variables) in a state are in $[0, 1]$ to ensure the generalization of the learned MA-DAC policy. The detailed state features at step t are shown in Table S1. We use $\mathrm{List}(I, t, l)$ to denote a list of indicator values from step $t-l+1$ to t, i.e., $[I_{t-l+1}, \ldots, I_t]$, where I denotes a specific indicator and I_i denotes the indicator value at step i. Note that $\forall i < 0, I_i = 0$ as default. For the considered three indicators, i.e., hypervolume, the ratio of non-dominated solutions in the population and the average distance of the solutions, they are denoted as HV, $\mathrm{NDRatio}$ and Dist, respectively.

Table S1: State at step t in MaMo.

Index	Parts of state	Feature	Notes
0	1	m	Number of objectives
1	1	D	Number of variables
2	2	t/T	The computational budget has been used
3	2	N_{stag}/T	Stagnant count ratio
4	3	HV_t	Hypervolume value
5	3	$\mathrm{NDRatio}_t$	Ratio of non-dominated solutions
6	3	Dist_t	Average distance
7	3	$\mathrm{HV}_t - \mathrm{HV}_{t-1}$	Change of HV between steps t and $t-1$
8	3	$\mathrm{NDRatio}_t - \mathrm{NDRatio}_{t-1}$	Change of NDRatio between steps t and $t-1$
9	3	$\mathrm{Dist}_t - \mathrm{Dist}_{t-1}$	Change of Dist between steps t and $t-1$
10	3	$\mathrm{Mean}(\mathrm{List}(\mathrm{HV}, t, 5))$	Mean of HV in the last 5 steps
11	3	$\mathrm{Mean}(\mathrm{List}(\mathrm{NDRatio}, t, 5))$	Mean of NDRatio in the last 5 steps
12	3	$\mathrm{Mean}(\mathrm{List}(\mathrm{Dist}, t, 5))$	Mean of Dist in the last 5 steps
13	3	$\mathrm{Var}(\mathrm{List}(\mathrm{HV}, t, 5))$	Variance of HV in the last 5 steps
14	3	$\mathrm{Var}(\mathrm{List}(\mathrm{NDRatio}, t, 5))$	Variance of NDRatio in the last 5 steps
15	3	$\mathrm{Var}(\mathrm{List}(\mathrm{Dist}, t, 5))$	Variance of Dist in the last 5 steps
16	3	$\mathrm{Mean}(\mathrm{List}(\mathrm{HV}, t, t))$	Mean of HV in all the steps so far
17	3	$\mathrm{Mean}(\mathrm{List}(\mathrm{NDRatio}, t, t))$	Mean of NDRatio in all the steps so far
18	3	$\mathrm{Mean}(\mathrm{List}(\mathrm{Dist}, t, t))$	Mean of Dist in all the steps so far
19	3	$\mathrm{Var}(\mathrm{List}(\mathrm{HV}, t, t))$	Variance of HV in all the steps so far
20	3	$\mathrm{Var}(\mathrm{List}(\mathrm{NDRatio}, t, t))$	Variance of NDRatio in all the steps so far
21	3	$\mathrm{Var}(\mathrm{List}(\mathrm{Dist}, t, t))$	Variance of Dist in all the steps so far

A.3 Action formulation

We consider four heterogeneous types of configuration hyperparameters in MOEA/D as the actions of four different agents of MA-DAC.

Weights. In MOEA/D, weights are used to transform an MOP into multiple single-objective sub-problems, which should be as diverse as possible [17]. Inspired by MOEA/D-AWA [9], the action space for weights is discrete with two dimensions, i.e., adjusting (T) and not adjusting (N) the weights. Furthermore, we limit the frequency of adjustment because too frequent adjustment will lead to drastic changes in the sub-problems and is detrimental to the optimization process [9]. If the action is T, weights will be updated before selecting the parent solutions. The weights adaptation mechanism is as follows.

We first calculate the sparsity level of each solution $\boldsymbol{x}^{(i)}$ based on vicinity distance [4]:

$$SL\left(\boldsymbol{x}^{(i)}, \{\boldsymbol{x}^{(p)}\}_{p=1}^{N}\right) = \prod_{j=1}^{m} l(\boldsymbol{x}^{(i)}, j), \tag{3}$$

where $l(\boldsymbol{x}^{(i)}, j)$ is the Euclidean distance between $\boldsymbol{x}^{(i)}$ and its j-th nearest neighbor in the population $\{\boldsymbol{x}^{(p)}\}_{p=1}^{N}$. The m closest neighbors in the population are used for calculation, where m is the number of objectives. After calculating the sparsity level of each solution, the sub-problems corresponding to the solutions whose sparsity levels are ranked bottom 5%, i.e., the overcrowded solutions, will be removed.

To ensure that there are still N sub-problems in total, we should add $0.05N$ new sub-problems and their corresponding solutions. The newly added solutions are from an elite population, which stores

all historical non-dominated solutions with a capacity of $1.5N$. If the size of the elite population exceeds the capacity, the solutions with the lowest sparsity level will be removed. For each solution $\boldsymbol{x}'$ in the elite population, we calculate its sparsity level with respect to the current population, i.e., $SL(\boldsymbol{x}', \text{Pop})$, where Pop denotes the set of $0.95N$ solutions in the current population after removing the overcrowded solutions. Then, we select the solution from the elite population, which has the highest sparsity level with respect to the current population, and add it to the current population; this process is repeated for $0.05N$ times. For each newly added solution, the corresponding sub-problem (i.e., weight vector) is generated in a specific way, whose details can refer to Algorithm 3 in [9].

Neighborhood size. The neighborhood size is to control the distance between solutions in mating selection. A small size helps the search exploit the local area, while a large size helps the search explore a wide objective space [15]. We discretize the action space into four dimensions, i.e., 15, 20, 25, and 30, where 20 is the default value.

Types of the reproduction operators. We consider four types of DE operators with different search abilities introduced in [6]. Assuming that we are reproducing an offspring solution for the i-th sub-problem. Let $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}'^{(i)}$ denote its current solution and the generated offspring solution, respectively. The equations of four types of DE operators are shown as follows:

- OP1: $\boldsymbol{x}'^{(i)} = \boldsymbol{x}^{(i)} + F \times \left(\boldsymbol{x}^{(r_1)} - \boldsymbol{x}^{(r_2)}\right)$,
- OP2: $\boldsymbol{x}'^{(i)} = \boldsymbol{x}^{(i)} + F \times \left(\boldsymbol{x}^{(r_1)} - \boldsymbol{x}^{(r_2)}\right) + F \times \left(\boldsymbol{x}^{(r_3)} - \boldsymbol{x}^{(r_4)}\right)$,
- OP3: $\boldsymbol{x}'^{(i)} = \boldsymbol{x}^{(i)} + K \times \left(\boldsymbol{x} - \boldsymbol{x}^{(r_1)}\right) + F \times \left(\boldsymbol{x}^{(r_2)} - \boldsymbol{x}^{(r_3)}\right) + F \times \left(\boldsymbol{x}^{(r_4)} - \boldsymbol{x}^{(r_5)}\right)$,
- OP4: $\boldsymbol{x}'^{(i)} = \boldsymbol{x}^{(i)} + K \times \left(\boldsymbol{x} - \boldsymbol{x}^{(r_1)}\right) + F \times \left(\boldsymbol{x}^{(r_2)} - \boldsymbol{x}^{(r_3)}\right)$.

Here, $\boldsymbol{x}^{(r_1)}, \boldsymbol{x}^{(r_2)}, \boldsymbol{x}^{(r_3)}, \boldsymbol{x}^{(r_4)}$, and $\boldsymbol{x}^{(r_5)}$ are different parent solutions randomly selected from the neighborhood of $\boldsymbol{x}^{(i)}$. The scaling factor $F > 0$ controls the impact of the vector differences on the mutant vector, and $K \in [0, 1]$ plays a similar role to F.

Parameters of the reproduction operators. The parameters (e.g., scaling factor) of the reproduction operators in MOEA/D significantly affect the algorithm's performance [11]. We set the scaling factor K to a fixed value of 0.5 as recommended [6], and dynamically adjust the scaling factor F. The action space has four discrete dimensions, i.e., 0.4, 0.5, 0.6 and 0.7, where 0.5 is the default value.

B Additional results

B.1 Details of experimental settings

Common settings of MOEA/D We implement MOEA/D with `Platypus`.[1] All algorithms mentioned in this paper use the same common settings[17, 13], as shown in Table S2.

DQN We implement DQN with `tianshou`[2] [16] framework and adjust some of the hyperparameters to fit this new task. The network structure is:

$$\begin{aligned}\text{state} &\rightarrow \textbf{MLP}(128) \rightarrow \textbf{relu} \rightarrow \textbf{MLP}(128) \rightarrow \textbf{relu} \rightarrow \textbf{MLP}(128) \\ &\rightarrow \textbf{relu} \rightarrow \textbf{MLP}(\text{number of actions})\end{aligned}$$

where $\textbf{MLP}(n)$ means a fully-connected layer with output size of n, and **relu** means Rectified Linear Units. Here, the action apace is the concatenation of the four types of configuration hyperparameters, with a dimension of 128 (i.e., $4 \times 4 \times 4 \times 2$). Some key hyperparameters of DQN are as follows:

- The learning rate is $3e$-4.
- The discounting factor γ is 0.99.
- The buffer size is 50000 (unit is transition).
- The number of training steps is 400000.

[1] `https://github.com/Project-Platypus/Platypus`

[2] `https://github.com/thu-ml/tianshou`

Table S2: Common Settings of MOEA/D

General settings	
Population size N	210
Number T of iterations	$100 \times m$
Reproduction operators	
Crossover operator	Simulated binary crossover (SBX)
Distribution index of SBX	20
Mutation operator	Polynomial mutation (PM)
Probability of PM	$1/D$
Distribution index of PM	20
Aggregation function	
Aggregation function	Tchebycheff approach
Neighborhood size	20

MA-UCB MA-UCB uses four upper confidence bound (UCB) [1] agents to adjust the four types of hyperparameters [3]. Each agent follows the UCB action selection rule, i.e., the action taken by agent i at step t is

$$a_t^{(i)} \doteq \arg\max_{a^{(i)}} \left[Q_t(a^{(i)}) + c\sqrt{\frac{\ln t}{N_t(a^{(i)})}} \right], \tag{4}$$

where $Q_t(a^{(i)})$ denotes the estimated value of $a^{(i)}$ at step t, $N_t(a^{(i)})$ denotes the number of times that action $a^{(i)}$ has been selected at step t, and the number $c > 0$ (the value is 1.0 here) controls the degree of exploration.

MOEA/D-FRRMAB We modify the implementation FRRMAB from `PlatEMO`[3] [12] framework to make a fair comparison (i.e., the original adaptive operator selection mechanism and related hyperparameters are retained, except that it uses the same MOEA/D processes and settings as MA-DAC). MOEA/D-FRRMAB adjusts the four types of DE operators by MAB. In particular, we searched for some sensitive hyperparameters according to the suggestions in [6], and the best performing combination is shown as follows:

- Scaling factor is 2.0.
- Size of the sliding window is $0.5 \times N$.
- Decaying factor is 0.3.

MOEA/D-AWA We modify the implementation MOEA/D-AWA from `PlatEMO` framework to make a fair comparison (i.e., the original adaptive weight vector adjustment strategy and related hyperparameters are retained, except that it uses the same MOEA/D processes and settings as MA-DAC).

MA-DAC We use default VDN policy network without parameter sharing in `EPyMARL`[4] [8] framework. MA-DAC and all its other variants use the same hyperparameters. Some key hyperparameters are as follows:

- The learning rate is $1e$-4.
- The discounting factor γ is 0.99.
- The buffer size is 5000 (unit is episode).
- The number of training steps is 400000.

All hyperparameters of the above algorithms can be found in the code.

[3] `https://github.com/BIMK/PlatEMO`

[4] `https://github.com/uoe-agents/epymarl`

Table S3: IGD values obtained by MA-DAC-R1, MA-DAC-R2, MA-DAC-R3, and MA-DAC on different problems. Each result consists of the mean and standard deviation of 30 runs. The best mean value of each problem is highlighted in **bold**. The symbols '$+$', '$-$' and '$\approx$' indicate that the result is significantly superior to, inferior to, and almost equivalent to the MA-DAC, respectively, according to the Wilcoxon rank-sum test with confidence level 0.05.

Problem	M	MA-DAC-R1	MA-DAC-R2	MA-DAC-R3	MA-DAC
DTLZ2	3	4.223E-02 (2.50E-03) $-$	3.853E-02 (5.58E-04) $-$	3.809E-02 (4.64E-04) $\approx$	**3.807E-02** (5.05E-04)
	5	2.401E-01 (8.27E-03) $-$	2.726E-01 (1.51E-02) $\approx$	**2.364E-01** (1.04E-02) $+$	2.442E-01 (1.26E-02)
	7	4.142E-01 (1.12E-02) $-$	4.248E-01 (1.30E-02) $-$	4.215E-01 (9.03E-03) $-$	**3.944E-01** (1.17E-02)
DTLZ4	3	**5.567E-02** (7.33E-03) $-$	7.236E-02 (6.19E-02) $-$	6.144E-02 (5.10E-02) $\approx$	6.700E-02 (6.14E-02)
	5	3.119E-01 (1.91E-02) $-$	3.221E-01 (2.12E-02) $-$	3.119E-01 (1.58E-02) $-$	**2.995E-01** (2.10E-02)
	7	4.354E-01 (1.29E-02) $-$	4.385E-01 (1.23E-02) $-$	4.275E-01 (1.60E-02) $-$	**4.182E-01** (1.21E-02)
WFG4	3	5.989E-02 (5.60E-03) $\approx$	5.255E-02 (1.14E-03) $-$	5.309E-02 (8.02E-04) $-$	**5.200E-02** (1.19E-03)
	5	1.848E-01 (2.61E-03) $+$	1.851E-01 (2.43E-03) $+$	**1.846E-01** (2.20E-03) $+$	1.868E-01 (2.81E-03)
	7	3.028E-01 (3.19E-03) $+$	**3.008E-01** (3.51E-03) $\approx$	3.029E-01 (3.36E-03) $\approx$	3.033E-01 (3.66E-03)
WFG5	3	4.841E-02 (7.78E-04) $-$	4.763E-02 (7.73E-04) $-$	4.773E-02 (6.58E-04) $-$	**4.730E-02** (7.89E-04)
	5	1.823E-01 (2.49E-03) $\approx$	1.818E-01 (2.90E-03) $-$	1.812E-01 (3.06E-03) $\approx$	**1.811E-01** (3.02E-03)
	7	3.212E-01 (6.60E-03) $\approx$	**3.174E-01** (6.43E-03) $\approx$	3.196E-01 (5.99E-03) $\approx$	3.206E-01 (8.04E-03)
WFG6	3	7.920E-02 (1.81E-02) $+$	4.909E-02 (1.50E-02) $-$	**4.814E-02** (1.22E-02) $\approx$	4.831E-02 (8.95E-03)
	5	1.977E-01 (6.17E-03) $-$	2.037E-01 (4.49E-03) $-$	1.975E-01 (5.78E-03) $-$	**1.942E-01** (6.90E-03)
	7	**3.110E-01** (4.86E-03) $-$	3.151E-01 (5.01E-03) $\approx$	3.148E-01 (4.05E-03) $-$	3.112E-01 (4.93E-03)
WFG7	3	4.555E-02 (1.26E-03) $\approx$	4.076E-02 (5.41E-04) $-$	4.168E-02 (6.40E-04) $-$	**4.066E-02** (5.31E-04)
	5	1.842E-01 (3.28E-03) $\approx$	1.865E-01 (2.93E-03) $+$	**1.841E-01** (3.95E-03) $+$	1.858E-01 (2.12E-03)
	7	3.335E-01 (1.09E-02) $+$	**3.199E-01** (9.86E-03) $-$	3.271E-01 (9.65E-03) $\approx$	3.258E-01 (1.25E-02)
WFG8	3	8.914E-02 (2.96E-03) $\approx$	7.911E-02 (1.06E-03) $-$	8.199E-02 (1.96E-03) $-$	**7.901E-02** (1.19E-03)
	5	2.551E-01 (1.02E-02) $-$	2.628E-01 (1.22E-02) $-$	2.541E-01 (9.08E-03) $-$	**2.479E-01** (7.20E-03)
	7	4.163E-01 (9.54E-03) $\approx$	**4.115E-01** (9.80E-03) $\approx$	4.197E-01 (7.52E-03) $-$	4.127E-01 (5.93E-03)
WFG9	3	5.003E-02 (9.00E-03) $-$	4.208E-02 (6.56E-04) $-$	4.428E-02 (9.97E-03) $-$	**4.159E-02** (6.10E-04)
	5	1.929E-01 (8.84E-03) $\approx$	**1.819E-01** (5.73E-03) $-$	1.951E-01 (9.83E-03) $-$	1.832E-01 (7.10E-03)
	7	3.342E-01 (8.56E-03) $-$	3.322E-01 (8.89E-03) $-$	3.327E-01 (8.02E-03) $-$	**3.278E-01** (7.21E-03)
$+/-/\approx$		4/12/8	2/17/5	3/14/7	
average rank		3.04	2.71	2.46	1.79

Computing Resources The experiments are conducted on six PCs with an AMD Ryzen 9 3950X 16-Core Processor and an NVIDIA GeForce RTX 3090 GPU.

B.2 Analysis of the reward function

In this subsection, we compare our proposed reward function with the three types of reward functions proposed by [10], as shown in the following:

$$r_t^1 = \max\{f(s_t) - f(s_{t+1}), 0\}, \tag{5}$$

$$r_t^2 = \begin{cases} 10 & \text{if } f(s_{t+1}) < f_t^* \\ 1 & \text{else if } f(s_{t+1}) < f(s_t) \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

$$r_t^3 = \max\left\{\frac{f(s_t) - f(s_{t+1})}{f(s_{t+1}) - f_{\text{opt}}}, 0\right\}, \tag{7}$$

where $f(s_t)$ is the metric value at step t, f_t^* is the minimum metric value achieved until step t, and f_{opt} is the optimal metric value, i.e., the global minimum value. Here, we use IGD [2] as the metric $f(\cdot)$, and thus $f_{\text{opt}} = 0$. We train MA-DAC policy with these three reward functions r_t^1, r_t^2 and r_t^3, which are denoted as MA-DAC-R1, MA-DAC-R2 and MA-DAC-R3, respectively.

The experimental results are shown in Table S3. We can see that MA-DAC has the best average rank, indicating the effectiveness of our proposed reward function. For the other three methods, MA-DAC-R2 and MA-DAC-R3 are better than MA-DAC-R1, which is consistent with the observation in [10].

B.3 Analysis of the reproduction operators

In this subsection, we give a detailed analysis of the reproduction operators, including the four types of DE operators introduced in Appendix A.3, and also further compare MA-DAC with MOEA/D-FRRMAB, which applies the MAB-based adaptive tuning method FRRMAB [6] to dynamically adjust the types of DE operators used in MOEA/D.

First, we examine the performance of MOEA/D equipped with each type of DE operator, where the DE operator is used as the crossover operator with a default scaling factor $F = 0.5$. The results are shown in Table S4. Compared with the original MOEA/D using the SBX operator, these methods using the DE operator achieve a similar performance, as the numbers of '+' and '-' are close. Among the methods using the DE operator, MOEA/D-OP2 has the best average rank, which has thus also been used as the default DE operator in MA-DAC (M) w/o 3. Note that MA-DAC (M) w/o 3 denotes MA-DAC (M) without tuning the types of reproduction operators, which is used to validate the effectiveness of adjusting all configuration hyper-parameters simultaneously in RQ3 of the main paper.

Table S4: IGD values obtained by MOEA/D-OP1, MOEA/D-OP2, MOEA/D-OP3, and MOEA/D-OP4 on different problems. Each result consists of the mean and standard deviation of 30 runs. The best mean value of each problem is highlighted in **bold**. The symbols '$+$', '$-$' and '$\approx$' indicate that the result is significantly superior to, inferior to, and almost equivalent to the original MOEA/D (i.e., the column MOEA/D in Table 2 of the main paper or Table S5), respectively, according to the Wilcoxon rank-sum test with confidence level 0.05.

Problem	M	MOEA/D-OP1	MOEA/D-OP2	MOEA/D-OP3	MOEA/D-OP4
DTLZ2	3	**4.681E-02** (2.95E-04) $-$	4.691E-02 (3.97E-04) $-$	6.050E-02 (2.64E-03) $-$	5.033E-02 (1.07E-03) $-$
	5	3.037E-01 (9.85E-04) $-$	**3.012E-01** (1.51E-03) $\approx$	3.391E-01 (1.06E-02) $-$	3.083E-01 (2.69E-03) $-$
	7	4.735E-01 (9.68E-03) $-$	**4.551E-01** (4.43E-03) $-$	4.988E-01 (1.01E-02) $-$	4.887E-01 (9.51E-03) $-$
DTLZ4	3	7.897E-02 (6.36E-02) $-$	**6.226E-02** (4.05E-03) $-$	1.296E-01 (1.23E-02) $-$	7.890E-02 (9.62E-03) $-$
	5	3.504E-01 (2.77E-02) $-$	**3.413E-01** (1.48E-02) $-$	3.631E-01 (7.24E-03) $-$	3.521E-01 (1.23E-02) $-$
	7	4.923E-01 (1.89E-02) $-$	**4.519E-01** (1.15E-02) $-$	4.766E-01 (1.35E-02) $-$	4.975E-01 (2.23E-02) $-$
WFG4	3	**6.934E-02** (1.54E-03) $-$	7.293E-02 (1.43E-03) $-$	9.046E-02 (4.01E-03) $-$	7.998E-02 (2.25E-03) $-$
	5	2.930E-01 (1.03E-02) $+$	2.761E-01 (6.39E-03) $+$	2.762E-01 (5.86E-03) $+$	**2.761E-01** (7.63E-03) $+$
	7	4.057E-01 (1.45E-02) $+$	3.711E-01 (9.79E-03) $+$	**3.617E-01** (6.46E-03) $+$	3.696E-01 (1.06E-02) $+$
WFG5	3	6.181E-02 (5.85E-04) $+$	6.177E-02 (8.01E-04) $+$	6.128E-02 (5.59E-04) $+$	**6.113E-02** (5.30E-04) $+$
	5	3.138E-01 (6.20E-03) $+$	3.052E-01 (7.19E-03) $+$	**3.031E-01** (7.37E-03) $+$	3.116E-01 (8.26E-03) $+$
	7	**4.945E-01** (1.24E-02) $-$	4.988E-01 (1.04E-02) $-$	5.197E-01 (1.01E-02) $-$	5.189E-01 (1.22E-02) $-$
WFG6	3	7.470E-02 (2.22E-02) $+$	**6.714E-02** (1.59E-02) $+$	7.665E-02 (9.41E-03) $-$	9.557E-02 (1.71E-02) $-$
	5	3.513E-01 (1.46E-02) $+$	3.285E-01 (2.33E-02) $+$	**3.254E-01** (1.39E-02) $+$	3.421E-01 (1.30E-02) $+$
	7	4.918E-01 (3.31E-02) $\approx$	4.797E-01 (3.04E-02) $\approx$	**4.328E-01** (2.81E-02) $+$	4.478E-01 (3.08E-02) $+$
WFG7	3	**5.929E-02** (6.35E-04) $-$	6.033E-02 (8.84E-04) $-$	8.382E-02 (4.86E-03) $-$	6.699E-02 (1.75E-03) $-$
	5	3.286E-01 (1.55E-02) $+$	2.941E-01 (9.66E-03) $+$	**2.924E-01** (1.12E-02) $+$	3.148E-01 (1.58E-02) $+$
	7	5.062E-01 (2.46E-02) $+$	4.739E-01 (2.51E-02) $+$	**4.479E-01** (2.28E-02) $+$	4.859E-01 (2.72E-02) $+$
WFG8	3	**9.314E-02** (9.12E-04) $-$	9.598E-02 (1.22E-03) $-$	1.213E-01 (3.36E-03) $-$	1.070E-01 (2.16E-03) $-$
	5	4.112E-01 (1.14E-02) $+$	3.884E-01 (1.19E-02) $+$	**3.808E-01** (7.26E-03) $+$	3.925E-01 (1.33E-02) $+$
	7	5.743E-01 (1.09E-02) $+$	5.587E-01 (1.56E-02) $+$	**5.564E-01** (1.13E-02) $+$	5.570E-01 (1.22E-02) $+$
WFG9	3	**5.993E-02** (1.32E-02) $+$	8.122E-02 (2.54E-02) $-$	8.912E-02 (1.83E-02) $-$	8.652E-02 (2.15E-02) $-$
	5	**3.246E-01** (1.54E-02) $+$	3.300E-01 (1.47E-02) $+$	3.325E-01 (1.63E-02) $+$	3.389E-01 (1.18E-02) $+$
	7	5.179E-01 (2.68E-02) $+$	**5.001E-01** (2.59E-02) $+$	5.252E-01 (2.03E-02) $+$	5.472E-01 (2.12E-02) $\approx$
$+/-/\approx$		13/10/1	12/10/2	12/12/0	11/12/1
average rank		2.62	1.88	2.67	2.83

Then, we examine the performance of MOEA/D, MOEA/D-OP2, MOEA/D-FRRMAB, and MA-DAC on different problems. The operator pool of FRRMAB is just the four types of DE operators. The results in Table S5 show that MOEA/D-FRRMAB is better than MOEA/D and MOEA/D-OP2, disclosing the effectiveness of adjusting the type of reproduction operators. We can also observe that the proposed MA-DAC clearly performs the best.

B.4 Analysis of the adaptive weights

In this subsection, we compare MA-DAC with MOEA/D-AWA [9], which dynamically adjusts the weights of MOEA/D based on predefined heuristic intervals. The concrete way of adjusting the weights of MOEA/D-AWA and MA-DAC are the same, as described in Appendix A.3. Table S6

Table S5: IGD values obtained by MOEA/D, MOEA/D-OP2, MOEA/D-FRRMAB, and MA-DAC on different problems. Each result consists of the mean and standard deviation of 30 runs. The best mean value of each problem is highlighted in **bold**. The symbols '$+$', '$-$' and '$\approx$' indicate that the result is significantly superior to, inferior to, and almost equivalent to MA-DAC, respectively, according to the Wilcoxon rank-sum test with confidence level 0.05.

Problem	M	MOEA/D	MOEA/D-OP2	MOEA/D-FRRMAB	MA-DAC
DTLZ2	3	4.605E-02 (3.54E-04) $-$	4.691E-02 (3.97E-04) $-$	4.668E-02 (2.50E-04) $-$	**3.807E-02** (5.05E-04)
	5	3.006E-01 (1.55E-03) $-$	3.012E-01 (1.51E-03) $-$	3.031E-01 (1.29E-03) $-$	**2.442E-01** (1.26E-02)
	7	4.455E-01 (1.41E-02) $-$	4.551E-01 (4.43E-03) $-$	4.724E-01 (7.80E-03) $-$	**3.944E-01** (1.17E-02)
DTLZ4	3	6.231E-02 (8.85E-02) $\approx$	6.226E-02 (4.05E-03) $-$	**5.782E-02** (3.48E-03) $-$	6.700E-02 (6.14E-02)
	5	3.133E-01 (4.45E-02) $\approx$	3.413E-01 (1.48E-02) $-$	3.373E-01 (1.70E-02) $-$	**2.995E-01** (2.10E-02)
	7	4.374E-01 (2.57E-02) $-$	4.519E-01 (1.15E-02) $-$	4.681E-01 (1.87E-02) $-$	**4.182E-01** (1.21E-02)
WFG4	3	5.761E-02 (5.41E-04) $-$	7.293E-02 (1.43E-03) $-$	7.097E-02 (1.63E-03) $-$	**5.200E-02** (1.19E-03)
	5	3.442E-01 (1.21E-02) $-$	2.761E-01 (6.39E-03) $-$	2.799E-01 (9.44E-03) $-$	**1.868E-01** (2.81E-03)
	7	4.529E-01 (1.79E-02) $-$	3.711E-01 (9.79E-03) $-$	3.778E-01 (1.01E-02) $-$	**3.033E-01** (3.66E-03)
WFG5	3	6.327E-02 (1.10E-03) $-$	6.177E-02 (8.01E-04) $-$	6.120E-02 (7.38E-04) $-$	**4.730E-02** (7.89E-04)
	5	3.350E-01 (9.77E-03) $-$	3.052E-01 (7.19E-03) $-$	3.033E-01 (8.69E-03) $-$	**1.811E-01** (3.02E-03)
	7	4.101E-01 (2.08E-02) $-$	4.988E-01 (1.04E-02) $-$	5.045E-01 (9.70E-03) $-$	**3.206E-01** (8.04E-03)
WFG6	3	6.938E-02 (5.50E-03) $-$	6.714E-02 (1.59E-02) $-$	6.266E-02 (8.47E-03) $-$	**4.831E-02** (8.95E-03)
	5	3.518E-01 (2.82E-03) $-$	3.285E-01 (2.33E-02) $-$	3.272E-01 (1.61E-02) $-$	**1.942E-01** (6.90E-03)
	7	4.869E-01 (3.03E-02) $-$	4.797E-01 (3.04E-02) $-$	4.417E-01 (3.29E-02) $-$	**3.112E-01** (4.93E-03)
WFG7	3	5.811E-02 (6.31E-04) $-$	6.033E-02 (8.84E-04) $-$	5.976E-02 (7.44E-04) $-$	**4.066E-02** (5.31E-04)
	5	3.572E-01 (5.47E-03) $-$	2.941E-01 (9.66E-03) $-$	3.042E-01 (1.52E-02) $-$	**1.858E-01** (2.12E-03)
	7	5.236E-01 (2.19E-02) $-$	4.739E-01 (2.51E-02) $-$	4.762E-01 (2.74E-02) $-$	**3.258E-01** (1.25E-02)
WFG8	3	8.646E-02 (3.44E-03) $-$	9.598E-02 (1.22E-03) $-$	9.536E-02 (1.14E-03) $-$	**7.901E-02** (1.19E-03)
	5	4.258E-01 (8.42E-03) $-$	3.884E-01 (1.19E-02) $-$	3.917E-01 (9.00E-03) $-$	**2.479E-01** (7.20E-03)
	7	5.816E-01 (1.30E-02) $-$	5.587E-01 (1.56E-02) $-$	5.570E-01 (1.60E-02) $-$	**4.127E-01** (5.93E-03)
WFG9	3	5.817E-02 (1.24E-03) $-$	8.122E-02 (2.54E-02) $-$	6.445E-02 (1.72E-02) $-$	**4.159E-02** (6.10E-04)
	5	3.633E-01 (1.20E-02) $-$	3.300E-01 (1.47E-02) $-$	3.312E-01 (1.70E-02) $-$	**1.832E-01** (7.10E-03)
	7	5.538E-01 (2.63E-02) $-$	5.001E-01 (2.59E-02) $-$	5.145E-01 (2.82E-02) $-$	**3.278E-01** (7.21E-03)
$+/-/\approx$		0/22/2	0/24/0	0/24/0	
average rank		3.12	2.92	2.83	1.12

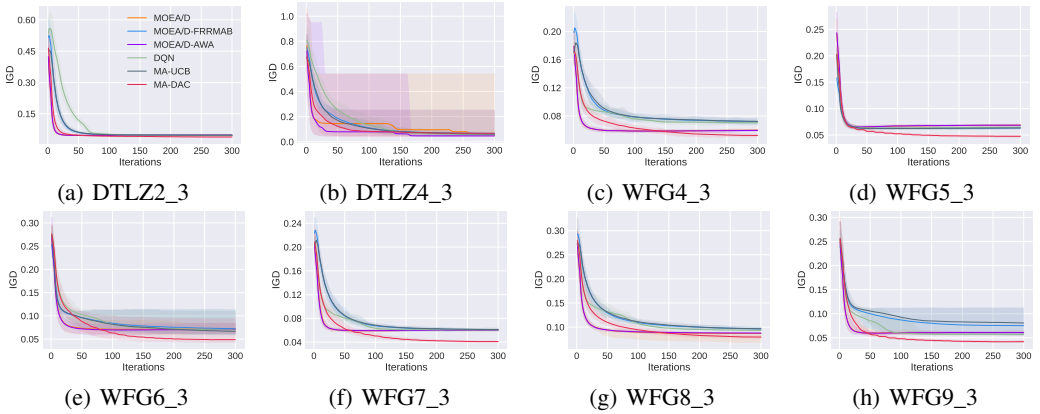
shows the results, where MOEA/D-OP2-AWA refers to MOEA/D-AWA using the DE operator OP2 (which has been shown to be the best among the four investigated DE operators in the last subsection) instead of the SBX operator. We can observe that MA-DAC performs the best in all problems except DTLZ4, where MOEA/D-AWA is better. Note that DTLZ4 is not used for training MA-DAC, and the worse performance than MOEA/D-AWA on this problem also implies that MA-DAC can be further improved in the future.

B.5 IGD values during the optimization process

Finally, we plot the curves of IGD value of all the compared methods (i.e., MOEA/D, MOEA/D-FRRMAB, MOEA/D-AWA, DQN, MA-UCB, and MA-DAC), on the problems with 3, 5 and 7 objectives, as shown in Figures S1, S2 and S3, respectively. We can observe that MA-DAC performs the best in general, and the superiority is more clear on the problems with 5 and 7 objectives. As the number of objectives increases, the problems become more difficult, thus requiring a powerful policy of adjusting the configuration hyperparameters. This also implies the applicability of MA-DAC in solving difficult problems.

Table S6: IGD values obtained by MOEA/D, MOEA/D-AWA, MOEA/D-OP2-AWA, and MA-DAC on different problems. Each result consists of the mean and standard deviation of 30 runs. The best mean value of each problem is highlighted in **bold**. The symbols '+', '−' and '≈' indicate that the result is significantly superior to, inferior to, and almost equivalent to MA-DAC, respectively, according to the Wilcoxon rank-sum test with confidence level 0.05.

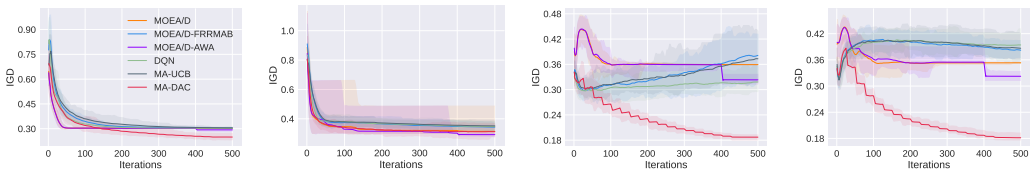
Problem	M	MOEA/D	MOEA/D-AWA	MOEA/D-OP2-AWA	MA-DAC
DTLZ2	3	4.605E-02 (3.54E-04) −	4.596E-02 (3.54E-04) −	4.670E-02 (3.30E-04) −	**3.807E-02** (5.05E-04)
	5	3.006E-01 (1.55E-03) −	2.900E-01 (2.73E-03) −	2.764E-01 (3.40E-03) −	**2.442E-01** (1.26E-02)
	7	4.455E-01 (1.41E-02) −	4.167E-01 (2.37E-02) −	4.436E-01 (8.67E-03) −	**3.944E-01** (1.17E-02)
DTLZ4	3	6.231E-02 (8.85E-02) ≈	**4.597E-02** (3.66E-04) ≈	6.219E-02 (3.90E-03) −	6.700E-02 (6.14E-02)
	5	3.133E-01 (4.45E-02) ≈	**2.816E-01** (3.24E-03) +	3.283E-01 (1.08E-02) −	2.995E-01 (2.10E-02)
	7	4.374E-01 (2.57E-02) −	**3.696E-01** (1.32E-02) +	4.437E-01 (9.46E-03) −	4.182E-01 (1.21E-02)
WFG4	3	5.761E-02 (5.41E-04) −	5.748E-02 (7.11E-04) −	7.280E-02 (1.33E-03) −	**5.200E-02** (1.19E-03)
	5	3.442E-01 (1.21E-02) −	3.168E-01 (5.37E-03) −	2.648E-01 (8.15E-03) −	**1.868E-01** (2.81E-03)
	7	4.529E-01 (1.79E-02) −	4.285E-01 (1.55E-02) −	3.676E-01 (1.06E-02) −	**3.033E-01** (3.66E-03)
WFG5	3	6.327E-02 (1.10E-03) −	6.376E-02 (9.85E-04) −	6.168E-02 (4.61E-04) −	**4.730E-02** (7.89E-04)
	5	3.350E-01 (9.77E-03) −	3.173E-01 (5.33E-03) −	3.024E-01 (6.02E-03) −	**1.811E-01** (3.02E-03)
	7	4.101E-01 (2.08E-02) −	4.095E-01 (1.94E-02) −	4.865E-01 (1.28E-02) −	**3.206E-01** (8.04E-03)
WFG6	3	6.938E-02 (5.50E-03) −	6.846E-02 (4.70E-03) −	6.078E-02 (1.16E-03) −	**4.831E-02** (8.95E-03)
	5	3.518E-01 (2.82E-03) −	3.190E-01 (3.93E-03) −	3.143E-01 (2.52E-02) −	**1.942E-01** (6.90E-03)
	7	4.869E-01 (3.03E-02) −	4.727E-01 (3.05E-02) −	4.770E-01 (3.24E-02) −	**3.112E-01** (4.93E-03)
WFG7	3	5.811E-02 (6.31E-04) −	5.837E-02 (6.25E-04) −	6.017E-02 (6.74E-04) −	**4.066E-02** (5.31E-04)
	5	3.572E-01 (5.47E-03) −	3.227E-01 (4.19E-03) −	2.885E-01 (1.25E-02) −	**1.858E-01** (2.12E-03)
	7	5.236E-01 (2.19E-02) −	5.004E-01 (3.80E-02) −	4.560E-01 (2.56E-02) −	**3.258E-01** (1.25E-02)
WFG8	3	8.646E-02 (3.44E-03) −	8.742E-02 (6.36E-04) −	9.572E-02 (8.39E-04) −	**7.901E-02** (1.19E-03)
	5	4.258E-01 (8.42E-03) −	4.216E-01 (1.18E-02) −	3.824E-01 (9.74E-03) −	**2.479E-01** (7.20E-03)
	7	5.816E-01 (1.30E-02) −	5.790E-01 (1.06E-02) −	5.632E-01 (1.27E-02) −	**4.127E-01** (5.93E-03)
WFG9	3	5.817E-02 (1.24E-03) −	5.809E-02 (1.45E-03) −	6.470E-02 (1.75E-02) −	**4.159E-02** (6.10E-04)
	5	3.633E-01 (1.20E-02) −	3.517E-01 (2.19E-02) −	3.024E-01 (1.36E-02) −	**1.832E-01** (7.10E-03)
	7	5.538E-01 (2.63E-02) −	5.108E-01 (2.65E-02) −	4.861E-01 (2.78E-02) −	**3.278E-01** (7.21E-03)
+/−/≈		0/22/2	2/21/1	0/24/0	
average rank		3.50	2.54	2.75	1.21



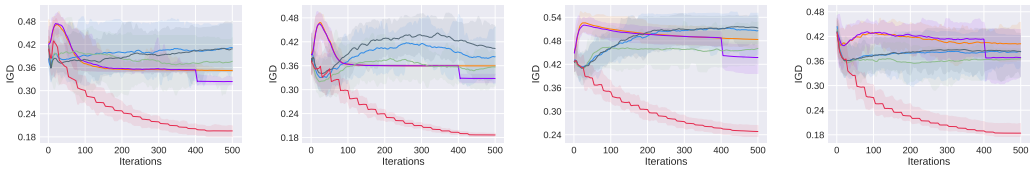
(a) DTLZ2_3 (b) DTLZ4_3 (c) WFG4_3 (d) WFG5_3

(e) WFG6_3 (f) WFG7_3 (g) WFG8_3 (h) WFG9_3

Figure S1: Curves of IGD value obtained by the compared methods on the 3-objective problems.

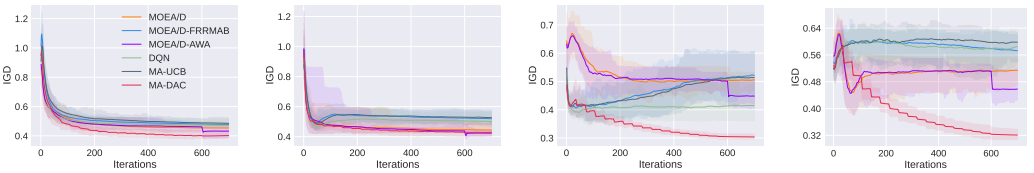


(a) DTLZ2_5 (b) DTLZ4_5 (c) WFG4_5 (d) WFG5_5

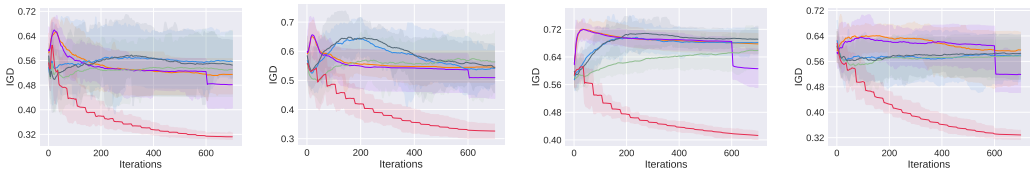


(e) WFG6_5 (f) WFG7_5 (g) WFG8_5 (h) WFG9_5

Figure S2: Curves of IGD value obtained by the compared methods on the 5-objective problems.



(a) DTLZ2_7 (b) DTLZ4_7 (c) WFG4_7 (d) WFG5_7



(e) WFG6_7 (f) WFG7_7 (g) WFG8_7 (h) WFG9_7

Figure S3: Curves of IGD value obtained by the compared methods on the 7-objective problems.

References

[1] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

[2] Peter A. N. Bosman and Dirk Thierens. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(2): 174–188, 2003.

[3] Safwan Hossain, Evi Micha, and Nisarg Shah. Fair algorithms for multi-agent multi-armed bandits. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34*, pages 24005–24017, 2021.

[4] Saku Kukkonen and Kalyanmoy Deb. A fast and effective method for pruning of non-dominated solutions in many-objective problems. In *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference*, pages 553–562, 2006.

[5] Ke Li. Decomposition multi-objective evolutionary optimization: From state-of-the-art to future opportunities. *arXiv preprint arXiv:2108.09588*, 2021.

[6] Ke Li, Alvaro Fialho, Sam Kwong, and Qingfu Zhang. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130, 2013.

[7] Miqing Li and Xin Yao. Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Computing Surveys*, 52(2):26:1–26:38, 2019.

[8] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021.

[9] Yutao Qi, Xiaoliang Ma, Fang Liu, Licheng Jiao, Jianyong Sun, and Jianshe Wu. MOEA/D with adaptive weight adjustment. *Evolutionary computation*, 22(2):231–264, 2014.

[10] Mudita Sharma, Alexandros Komninos, Manuel López-Ibáñez, and Dimitar Kazakov. Deep reinforcement learning based parameter control in differential evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 709–717. ACM, 2019.

[11] Jianyong Sun, Xin Liu, Thomas Bäck, and Zongben Xu. Learning adaptive differential evolution algorithm from optimization experiences by policy gradient. *IEEE Transactions on Evolutionary Computation*, 25(4):666–680, 2021.

[12] Ye Tian, Ran Cheng, Xingyi Zhang, and Yaochu Jin. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. *IEEE Computational Intelligence Magazine*, 12(4): 73–87, 2017.

[13] Ye Tian, Xiaopeng Li, Haiping Ma, Xingyi Zhang, Kay Chen Tan, and Yaochu Jin. Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization. *IEEE Transactions on Emerging Topics in Computational Intelligence*, pages 1–14, 2022.

[14] Anupam Trivedi, Dipti Srinivasan, Krishnendu Sanyal, and Abhiroop Ghosh. A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation*, 21(3):440–462, 2017.

[15] Zhenkun Wang, Qingfu Zhang, Aimin Zhou, Maoguo Gong, and Licheng Jiao. Adaptive replacement strategies for MOEA/D. *IEEE Transactions on Cybernetics*, 46(2):474–486, 2016.

[16] Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Hang Su, and Jun Zhu. Tianshou: A highly modularized deep reinforcement learning library. *arXiv preprint arXiv:2107.14171*, 2021.

[17] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[18] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms - A comparative case study. In *Parallel Problem Solving from Nature - PPSN V, 5th International Conference*, pages 292–304, 1998.