Multi-agent Dynamic Algorithm Configuration

Anonymous Author(s)
Affiliation
Address
email

Abstract

Automated algorithm configuration relieves users from tedious, trial-and-error tuning tasks. A popular algorithm configuration tuning paradigm is dynamic algorithm configuration (DAC), in which an agent learns dynamic configuration policies across instances by reinforcement learning (RL). However, in many complex algorithms, there may exist different types of configuration hyperparameters, and such heterogeneity may bring difficulties for classic DAC which uses a single-agent RL policy. In this paper, we aim to address this issue and propose a multi-agent DAC (MA-DAC) framework, with one agent working for one type of configuration hyperparameters. MA-DAC formulates the dynamic configuration of a complex algorithm with heterogeneous types of hyperparameters as a contextual multi-agent Markov decision process and solves it by a cooperative multi-agent RL (MARL) algorithm. To instantiate, we apply MA-DAC to a well-known optimization algorithm for multi-objective optimization problems. Experimental results show the effectiveness of MA-DAC in not only achieving superior performance compared with other configuration tuning approaches based on heuristic rules, multi-armed bandits, and single-agent RL, but also being capable of generalizing to different problems classes. Furthermore, we release the environments in this paper as a benchmark for testing MARL algorithms, with the hope of facilitating the application of MARL.

1 Introduction

Finding right configurations of hyperparameters is critical for many learning and optimization algorithms. Automated methods, such as algorithm configuration (AC) [15, 22], emerge to search for right configurations, with the aim of relieving users from tedious, trial-and-error tuning tasks. However, static configuration policies obtained by AC may not necessarily yield optimal performance (compared with dynamic policies) since algorithms may require different configurations at different stages of their execution [31].

Dynamic algorithm configuration (DAC) [2] is a well-known paradigm for obtaining dynamic configuration policies. Unlike AC, DAC can dynamically adjust the algorithm's configuration during the optimization process, through formulating it as a contextual Markov decision process (MDP) and then solving it by reinforcement learning (RL) [27]. DAC has been found to outperform static methods on many tasks, including the tuning learning rate of the deep neural network [6], the step-size adaptation of evolution strategy [29], and heuristic selection of planning [31].

The task of DAC typically focuses on a single type of configuration hyperparameter(s), such as tuning the step-size in CMA-ES [29]. However, due to the increasing complexity of real-world problem modeling, there are many algorithms whose performance rests on multiple types of hyperparameters. Tuning one type while fixing the rest may not produce promising results. Take a popular evolutionary algorithm for multi-objective optimization problems, MOEA/D [41], as an example. It has four types of configuration hyperparameters: weights, neighborhood size, reproduction opera-

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

tor type, and parameters associated with the reproduction operator used. All of them are critical and significantly affect the performance of the algorithm [36]. Finding a near-optimal configuration combination for even part of these hyperparameters requires massive manual effort [10].

In this paper, we attempt to extend DAC to dealing with tasks with multiple heterogeneous types of configuration hyperparameters. We model this as a cooperative multi-agent problem [24], with each agent handling one type of hyperparameters, for a shared goal (i.e., maximizing the team reward). Specifically, the proposed multi-agent DAC (MA-DAC) framework formulates the task as a contextual multi-agent MDP (MMDP) [4], and solves it by a common pay-off cooperative multi-agent RL (MARL). As an instantiation, we apply MA-DAC to the multi-objective evolutionary algorithm MOEA/D to learn the right configurations of its four types of hyperparameters by the classic MARL algorithm called value-decomposition networks (VDN) [33].

Experiments on well-established multi-objective optimization problems show that MA-DAC outperforms other configuration tuning methods based on heuristic rules [26], multi-armed bandits [17, 13] and single-agent RL [23]. Furthermore, we demonstrate the generalization of the learned MA-DAC policy to both inner classes (different instances with the same number of objectives) and outer classes (different instances with different numbers of objectives). Ablation studies demonstrate the importance of tuning every type of hyperparameter.

Our contributions are three-fold. 1) To the best of our knowledge, the MA-DAC framework is the first one to address dynamic configuration of algorithms with heterogeneous types of hyperparameters. 2) The contextual MMDP formulation of MA-DAC is analyzed, and experimental results show that the presented formulation works well and has good generalization ability. 3) The instantiation of configuring MOEA/D in this work can be used as a benchmark problem for MARL. The heterogeneity of MOEA/D's hyperparameters and the stochasticity of its search can promote the research of the MARL algorithms. Besides, the learned policies are useful for a specific type of optimization tasks - multi-objective optimization, which will facilitate the application of MARL.

2 Background

2.1 Dynamic algorithm configuration

Different from the static configuration of AC, DAC aims at dynamically adjusting the configuration hyperparameters of an algorithm during its optimization process. Biedenkapp et al. [2] formulated DAC as a contextual MDP and applied RL to solve it. Given a target algorithm A with its configuration hyperparameters space Θ, a DAC policy $\pi \in \Pi$ maps the state $s \in \mathcal{S}$ (e.g., history of changes in the objective value achieved by the target algorithm A) to the action $a \in \mathcal{A}$ (i.e., a hyperparameter $\theta \in \Theta$ of the target algorithm A). DAC aims at improving the performance of A on a set of instances (e.g., optimization functions). Given a probability distribution p over the the space of problem instance set $f \in \mathcal{F}$, the objective of DAC is to find an optimal policy π^*. That is,

$$\pi^* \in \arg\min_{\pi \in \Pi} \int_{\mathcal{F}} p(f) c(\pi, f) \mathrm{d}f, \tag{1}$$

where $f \in \mathcal{F}$ is a function to be optimized, and $c(\pi, f) \in \mathbb{R}$ is the cost function of the target algorithm with policy π on f. It has been shown that DAC outperforms static policies in learning rate adaptation in SGD [6], step-size adaptation in CMA-ES [29], and heuristic selection in planning [31]. These applications all only involve a single type of hyperparameter(s). Dynamic configurations of complex algorithms with heterogeneous types of hyperparameters have been found to be difficult for current DAC methods [9].

2.2 Multi-agent reinforcement learning

A multi-agent system [40] under fully observable cooperative situation can be modeled as an MMDP [4], which can be formalized as $\mathcal{M} := \langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^n, \mathcal{T}, r \rangle$, where $\mathcal{N}$ is a set of n agents, $\mathcal{S}$ is the state space, and $\mathcal{A}_i$ is agent i's action space. At each time-step, agent $i \in \mathcal{N}$ acquires $s \in \mathcal{S}$ and then chooses the action $a_i \in \mathcal{A}_i$. The joint action $\boldsymbol{a} = \langle a^{(1)}, \ldots, a^{(n)} \rangle$ leads to next state $s' \sim \mathcal{T}(\cdot \mid s, \boldsymbol{a})$ and all agents get a shared global reward $r(s, \boldsymbol{a})$.

The goal of an MMDP is to find a joint policy that maps the states to probability distributions over joint actions, $\boldsymbol{\pi} : S \to \Delta(\mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n)$, where $\Delta(\mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n)$ stands for the

distribution over joint actions, with the goal of maximizing the global value function: $Q^{\boldsymbol{\pi}}(s, \boldsymbol{a}) = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \boldsymbol{a}_t) \mid s_0 = s, \boldsymbol{a_0} = \boldsymbol{a}\right]$.

3 Multi-agent DAC framework

This section is devoted to the MA-DAC framework, where Section 3.1 is concerned with formulating MA-DAC as a contextual MMDP and Section 3.2 is concerned with explaining detailed components of MA-DAC.

3.1 Problem formulation

We propose MA-DAC as a new paradigm for solving the dynamic configuration of algorithms with heterogeneous types of hyperparameters. We formulate MA-DAC as a contextual MMDP $\mathcal{M}_{\mathcal{I}} := \{\mathcal{M}_i\}_{i \sim \mathcal{I}}$, where $\mathcal{M}_i := \langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}_j\}_{j=1}^n, \mathcal{T}_i, r_i \rangle$ is a single MMDP as defined in Section 2.2. The notion of context $\mathcal{I}$ induces multiple MMDPs. Each $\mathcal{M}_i$ stands for a specific instance i sampled from a given distribution $\mathcal{I}$. Different $\mathcal{M}_i$s have shared state and action spaces, but with different transition and reward functions. Algorithms are often tasked with solving different problem instances from the same or similar domains. Searching for well-performing parameter settings on a specific instance might achieve strong performance on that individual instance but might perform poorly in new instances. Therefore, we explicitly take the instance distribution $\mathcal{I}$ as context into account to facilitate generalization.

Given a parameterized algorithm A with a configuration space $\Theta = \{\theta_j\}_{j \in \mathcal{N}}$, one agent in MA-DAC is to handle one type of configuration hyperparameter $\theta_j \in \Theta$ and all the agents work with the same goal, i.e., maximizing the team reward. The action space of each agent $j \in \mathcal{N}$ is the hyperparameter space of the corresponding θ_j. Note that MA-DAC can be seen as a common-payoff cooperative MARL problem and can be solved by cooperative MARL algorithms.

3.2 Components of MA-DAC

Next, we introduce several important components of MA-DAC, including the state, action, transition, reward, and instance set.

State. The state is used to describe the situation of the target algorithm, which is a key component in MA-DAC. We suggest that the state should have the following properties.

1. Accessibility. The state should be accessible in each step during the optimization process.
2. Representation. The state should reflect the information in the optimization process.
3. Generalization. The learned policy is expected to generalize to inner and even outer classes of instances. Thus, the state should consist of the common features across different instances.

Besides, it is better if the state can be easily obtained, reducing the computational cost.

Action. Each agent in MA-DAC focuses on one type of configuration hyperparameter. Its action is the value of the hyperparameter that should be adjusted, which can be discrete or continuous. MA-DAC agents are heterogeneous because they have completely different action spaces and affect different types of hyperparameters of the target algorithm.

Transition. The transition function describes the dynamics of the problem. For an iterative algorithm, each iteration can be defined as a step. Given state s_t, each agent acts $a_t^{(i)}$, and the probability of reaching state s_{t+1} can be expressed as $\mathcal{T}(s_{t+1} \mid s_t, \boldsymbol{a}_t)$. Different from the state and action spaces, the transition function depends on the given instance $f \in \mathcal{F}$. Thus, the policy should identify the characteristics of the current instance and take the optimal action under it.

Reward. The reward is used to guide the policy's learning process, whose quality has a significant impact on the policy's final performance. Many RL benchmark environments include a well-defined

reward function. However, in many application scenarios, we must define the reward function manually based on a specific metric that can indicate the final performance.

Given a metric $f(s_t)$ of the problem, assume the smaller the value, the better the performance. An intuitive way to define the reward function is $r_t = \max\{f(s_t) - f(s_{t+1}), 0\}$. However, this type of reward function can easily mislead agents, because higher rewards do not always imply better performance, e.g., repeated rewards obtained by switching back and forth between good and bad states. Furthermore, the policy's learning process will be influenced by different scales of instances. To address the issues above, we propose the following reward function at step t:

$$r_t = \begin{cases} (1/2) \cdot (p_t^2 - p_{t-1}^2) & \text{if } f(s_{t+1}) < f_t^* \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

where $p_t = \frac{f(s_0) - f(s_{t+1})}{f(s_0)}$ if $f(s_{t+1}) < f_t^*$, and $p_t = p_{t-1}$ otherwise; f_t^* is the minimum metric value found until step t.

As shown in Figure 1, the isosceles right-angled triangle indicates the maximum rewards we can get. The triangle's left side (colored by blue) indicates the cumulative rewards already achieved, the middle part (colored by green) indicates the immediate rewards obtained when the best-found metric value is improved, and the right side (colored by brown) indicates the rewards that can still be obtained in the future.

The reward function defined in this way can prevent the agents from getting repeated rewards. The structure of the triangle also helps the agents focus on long-term rewards, because the reward increment becomes larger from left to right. Example explanations and comparative experiments will be provided in Appendix B. We believe that this design is applicable to a wide range of similar tasks.

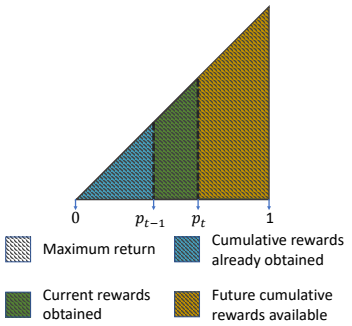

Figure 1: Illustration of the reward function.

Instance set. Instance set defines the optimization problem instances that have to be solved by the target algorithm. Note that the instances should have similar properties, otherwise the good generalization ability is hardly achievable [9].

4 Applying MA-DAC to dynamic configuration tuning of MOEA/D

As an instantiation, we apply MA-DAC to a well-established evolutionary algorithm for multi-objective optimization problems, multi-objective evolutionary algorithm based on decomposition (MOEA/D) [41]. As one of the most widely used multi-objective evolutionary algorithms, MOEA/D has several heterogeneous types of configuration hyperparameters (e.g., weights, neighborhood size, and reproduction operator type), which is a good fit for the proposed MA-DAC. We first briefly introduce multi-objective optimization and MOEA/D in Section 4.1, followed by detailed explanations of applying MARL to MOEA/D in Section 4.2, which results in an MARL benchmark, denoted by MaMo. Finally, we compare MaMo with other MARL benchmarks in Section 4.3.

4.1 Multi-objective optimization and MOEA/D

Multi-objective optimization refers to an optimization problem with more than one objective to be considered simultaneously. It is very common in the real world. For example, in deep neural networks, apart from the accuracy, one may also care about the model size, latency, and/or energy consumption [21, 12].

A prominent feature of multi-objective optimization problems (MOPs) is that unlike single-objective optimization problems, in MOPs there is no single optimal solution, but a set of Pareto optimal solutions. A solution is called Pareto optimal if there is no other solution in the search space that dominates it[1]. Unfortunately, the number of Pareto optimal solutions for a given MOP is typically

[1]For two solutions x and y, x is called to dominate y, if x is not worse than y on all objectives and better than y on at least one objective.

prohibitively large or even infinite. Therefore, the goal of a multi-objective optimization algorithm is to find a good approximation that well represents all the Pareto optimal solutions.

Evolutionary algorithms have demonstrated their effectiveness to solve MOPs. Their population-based nature can approximate the problem's optimal solutions within one execution, with each solution in the population representing a unique trade-off between the objectives. MOEA/D [41] is a representative multi-objective evolutionary algorithm. Unlike other mainstream multi-objective evolutionary algorithms like NSGA-II [7], which compare solutions based on their Pareto dominance relation and density in the population, 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. As such, MOEA/D entails different heterogeneous types of hyperparameters, e.g., weights and neighborhood size. Such hyperparameters significantly affect the performance of the algorithm. For example, different weight distributions are suitable for MOPs with different Pareto front shapes as the weights are used to control the distribution of the final population [19]. A large neighborhood has high search ability in the objective space while a small neighborhood is beneficial for diversity maintenance in the decision space [16]. In short, finding right configurations for these hyperparameters requires massive manual effort and different configurations work well on different MOPs.

Note that recently some efforts have been made to tune the hyperparameters of MOEA/D by heuristic rules [26], multi-armed bandit [17] and RL [35]. However, they all focus on a single type of hyperparameters, and also very few of them work on the dynamic algorithm configuration as well.

4.2 MA-DAC for MOEA/D

In this section, we show how to apply MA-DAC to learn the configuration policy of MOEA/D and also introduce the resulting environment MaMo. Each generation in the evolutionary process of MOEA/D is one step in the MaMo.

The state of MaMO can be divided into three parts. The first part is the features of the instance, including the number of variables and objectives. The second part includes 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. Besides, we use several indicators, i.e., hypervolume[2] [43], the ratio of non-dominated solutions in the population and the average distance of the solutions, to reflect the evolutionary progress of the current population. Lastly, we use statistic metrics (the mean and variance) of the indicators in several recent steps and all steps from the beginning to measure the short and long histories of the optimization, respectively. Note that all the states are normalized to $[0, 1]$ to ensure the generalization of the learned MA-DAC policy.

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

1. Weights. In MOEA/D, weights are used to maintain the diversity of the converted single-objective sub-problems [26]. The action space 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.
2. 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 [39]. We discretize the action space into four dimensions.
3. Types of the reproduction operators. We consider four types of differential evolution (DE) operators. Each type has different search ability [17].
4. Parameters of the reproduction operators. The parameters (e.g., scaling factor) of the reproduction operators in MOEA/D significantly affect the algorithm's performance [32], and the action space has four discrete dimensions.

The detailed definitions of the state and action are given in Appendix A due to the space limitation.

[2]Hypervolume is a quality indicator that can reflect both convergence and diversity of a solution set [18].

We use the MOPs with similar properties from the well-known MOP benchmarks DTLZ [8] and WFG [14] as the instance set of MaMo. These MOPs can have different number of objectives, and those with the same number of objectives can be seen as inner classes because the number of objectives considerably affects the properties and difficulty level of MOPs. We use the inverted generational distance (IGD) [3] as the metric in Eq. (2), resulting in the reward function.

To learn the policy, we use a classical MARL algorithm named VDN [33], which is widely used in cooperative multi-agent systems. VDN learns to decompose the team value function into agent-wise value functions, alleviating the exponential growth of the action space. It follows the Individual-Global-Max (IGM) principle [30], i.e., the consistency between joint and local greedy action selections by the joint value function $Q_{\text{tot}}(s, \boldsymbol{a})$ and individual value functions $\left[Q_i(s, a^{(i)})\right]_{i=1}^{n}$. All parameters in the framework are updated using the standard TD loss from the global Q-values Q_{tot}, which follows additivity to factorize the global value function, i.e., $Q_{\text{tot}}(s, \boldsymbol{a}) = \sum_{i=1}^{n} Q_i\left(s, a^{(i)}\right)$. In the testing phase, each agent acts greedily with respect to its Q_i.

4.3 Comparison among MARL benchmarks

There are many benchmarks [25, 34] emerged in recent years. In order to clearly show the characteristics of MaMo, we compare it with some MARL benchmarks, as shown in Table 1. Heterogeneity refers to agents having different action spaces or skills [37]. For example, in MaMo, there are four entirely different action spaces, i.e., weights, neighborhood size, types of the reproduction operators and their paramters. Stochasticity means that performing the same action in the same state may lead to a different next state. The high stochasticity of MaMo comes from the randomness of MOEA/D.

Table 1: Overview of MARL benchmarks and their properties.

Benchmark	Heterogeneous	# of agents	Stochastic	Application scenarios
Matrix Games [5]	×	2	Low	Game
MPE [20]	×	2-3	Low	Game
MAgent [42]	×	2-1000	Low	Game
SMAC [28]	✓	2-30	Low	Game
Active Voltage Control [38]	×	3-38	Low	Control
MaMo (Ours)	✓	2-4	High	Optimization

The primary benchmarks are designed to investigate different aspects of multi-agent systems. For example, Matrix Games [5] and Multi-Agent Particle Environments (MPE) are popular and classical testbeds, to investigate cooperative or competitive behaviors in small scale settings (with 2–3 agents). MAgent [42] is used to test the scalability of MARL methods, where the number of agents is up to 1000. SMAC [28] has attracted wide attention in recent years to test the coordination ability in cooperative MARL. For the first time, Active Voltage Control [38] creates an exciting yet challenging real-world scenario for the application of MARL. To the best of our knowledge, none of the known benchmarks has focused on highly heterogeneous and stochastic scenarios in MARL. We hope our new MaMo benchmark can offer a good supplement that could benefit the MARL community.

5 Experiments

To examine the effectiveness of MA-DAC, we conduct experiments using MaMo. We investigate the following research questions (RQs). RQ1: How does MA-DAC perform compared with the baseline and other tuning algorithms? RQ2: How is the generalization ability of MA-DAC? RQ3: How do the different parts of MA-DAC affect the performance? We introduce the experimental settings in Section 5.1, and investigate the above RQs in Section 5.2.

5.1 Experimental settings

We consider several well-established multi-objective test functions (DTLZ2, DTLZ4, and WFG4-WFG9) with 3, 5, and 7 objectives following the practice in Bezerra et al. [1] to examine our method. Several arbitrary functions (here DTLZ2, WFG4 and WFG6) from the function set are used as the

Table 2: IGD values obtained by MOEA/D, DQN, MA-UCB and MA-DAC on different problems. Each result consists of the mean and standard deviation of 30 runs. The best mean value on 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	DQN	MA-UCB	MA-DAC
DTLZ2	3	4.605E-02 (3.54E-04) −	4.628E-02 (2.96E-04) −	4.671E-02 (3.70E-04) −	**3.807E-02** (5.05E-04)
	5	3.006E-01 (1.55E-03) −	3.016E-01 (1.34E-03) −	3.041E-01 (1.69E-03) −	**2.442E-01** (1.26E-02)
	7	4.455E-01 (1.41E-02) −	4.671E-01 (1.15E-02) −	4.826E-01 (9.59E-03) −	**3.944E-01** (1.17E-02)
DTLZ4	3	6.231E-02 (8.85E-02) ≈	**5.590E-02** (5.77E-03) −	6.011E-02 (5.08E-03) −	6.700E-02 (6.14E-02)
	5	3.133E-01 (4.45E-02) ≈	3.457E-01 (1.61E-02) −	3.492E-01 (1.69E-02) −	**2.995E-01** (2.10E-02)
	7	4.374E-01 (2.57E-02) −	4.552E-01 (1.47E-02) −	4.756E-01 (2.01E-02) −	**4.182E-01** (1.21E-02)
WFG4	3	5.761E-02 (5.41E-04) −	6.920E-02 (1.20E-03) −	7.165E-02 (1.83E-03) −	**5.200E-02** (1.19E-03)
	5	3.442E-01 (1.21E-02) −	2.810E-01 (6.86E-03) −	2.859E-01 (6.77E-03) −	**1.868E-01** (2.81E-03)
	7	4.529E-01 (1.79E-02) −	3.725E-01 (1.14E-02) −	3.868E-01 (1.54E-02) −	**3.033E-01** (3.66E-03)
WFG5	3	6.327E-02 (1.10E-03) −	6.212E-02 (5.54E-04) −	6.118E-02 (7.03E-04) −	**4.730E-02** (7.89E-04)
	5	3.350E-01 (9.77E-03) −	3.077E-01 (6.36E-03) −	3.036E-01 (8.83E-03) −	**1.811E-01** (3.02E-03)
	7	4.101E-01 (2.08E-02) −	4.996E-01 (1.32E-02) −	5.024E-01 (1.38E-02) −	**3.206E-01** (8.04E-03)
WFG6	3	6.938E-02 (5.50E-03) −	6.834E-02 (1.78E-02) −	6.601E-02 (1.00E-02) −	**4.831E-02** (8.95E-03)
	5	3.518E-01 (2.82E-03) −	3.160E-01 (2.40E-02) −	3.359E-01 (1.47E-02) −	**1.942E-01** (6.90E-03)
	7	4.869E-01 (3.03E-02) −	4.322E-01 (2.95E-02) −	4.389E-01 (3.41E-02) −	**3.112E-01** (4.93E-03)
WFG7	3	5.811E-02 (6.31E-04) −	5.930E-02 (7.32E-04) −	6.014E-02 (7.11E-04) −	**4.066E-02** (5.31E-04)
	5	3.572E-01 (5.47E-03) −	2.993E-01 (1.43E-02) −	3.207E-01 (1.71E-02) −	**1.858E-01** (2.12E-03)
	7	5.236E-01 (2.19E-02) −	4.576E-01 (2.38E-02) −	4.879E-01 (2.75E-02) −	**3.258E-01** (1.25E-02)
WFG8	3	8.646E-02 (3.44E-03) −	9.280E-02 (1.06E-03) −	9.612E-02 (1.48E-03) −	**7.901E-02** (1.19E-03)
	5	4.258E-01 (8.42E-03) −	3.969E-01 (1.26E-02) −	3.956E-01 (1.32E-02) −	**2.479E-01** (7.20E-03)
	7	5.816E-01 (1.30E-02) −	5.575E-01 (1.39E-02) −	5.642E-01 (1.38E-02) −	**4.127E-01** (5.93E-03)
WFG9	3	5.817E-02 (1.24E-03) −	5.628E-02 (7.29E-04) −	7.953E-02 (2.45E-02) −	**4.159E-02** (6.10E-04)
	5	3.633E-01 (1.20E-02) −	3.258E-01 (1.61E-02) −	3.396E-01 (1.55E-02) −	**1.832E-01** (7.10E-03)
	7	5.538E-01 (2.63E-02) −	5.115E-01 (2.15E-02) −	5.227E-01 (1.79E-02) −	**3.278E-01** (7.21E-03)
+/−/≈		0/22/2	0/24/0	0/24/0	

training set, and all the functions are considered in the testing. IGD [3] is considered as the metric to measure the performance of the algorithms. The mean and standard deviation of the IGD values obtained by each algorithm on each MOP for 30 independent runs are reported. We apply the Wilcoxon rank-sum test with confidence level 0.05. For a fair comparison, other parameters (such as population size and computational budget) are set to be the same across all the compared algorithms. More details about the settings can be found in Appendix B due to space limitations. The code used in this paper is provided in the supplementary material.

5.2 Experimental results

RQ1: How does MA-DAC perform compared with the baseline and other tuning algorithms? We compare the MA-DAC policy with the original MOEA/D [41], DQN [23], and MA-UCB [13]. DQN is a single-agent RL algorithm that shares the same state, reward, and transition as our MA-DAC but has a different action space that is the concatenation of the four types of hyperparameters. DQN can be seen as an instantiation of DAC in MaMo. MA-UCB is a simplified algorithm that uses four UCB agents to adjust the four types of hyperparameters. For MA-DAC and DQN, we perform the training and testing on the same number of objectives.

The results are shown in Table 2. As can be seen in the table, MA-DAC is significantly superior to the other algorithms on almost all the 24 problems. Furthermore, MA-DAC (which is trained only on DTLZ2, WFG4, and WFG6) shows excellent performance on previously unseen problems during the training process, demonstrating its good generalization ability. DQN policy is significantly worse than MA-DAC on all the problems, which is consistent with the previous observation that DQN performs poorly in MARL tasks [11]. The superiority of MA-DAC over MA-UCB also demonstrates the necessity of learning configuration policy by an MARL algorithm.

We also compare our algorithm with two adaptive MOEA/D algorithms, i.e., MOEA/D-AWA [26] and MOEA/D-FRRMAB [17] which adjust the weights and reproduction operator type based on heuristic rules and MAB, respectively. The results are in Appendix B due to space limitation.

RQ2: How is the generalization ability of MA-DAC? We compare MA-DAC policies trained from different training sets to test the generalization ability of MA-DAC policies. MA-DAC (M) denotes that the policy is trained on the problems with all 3, 5, and 7 objectives. Meanwhile, MA-DAC (3), (5), and (7) denote that the policies are trained on the problems with 3, 5, and 7 objectives, respectively.

As shown in Table 3, it is unsurprising that the results of average rank show that the MA-DAC (3), (5), and (7) policies have best performance on the problems with 3, 5, and 7 objectives, respectively. For example, MA-DAC (3) has the best average rank (i.e., 1.625) on the problems with 3 objectives but has the worst average rank, i.e., 3.0 and 3.25, on the problems with 5 and 7 objectives, respectively. To obtain a more robust policy, we mix the problems with multiple numbers of objectives as training sets, resulting in the MA-DAC (M). As can be seen in Table 3, MA-DAC (M) demonstrates its robustness. Among all the policies, MA-DAC always takes the first or second place in terms of its performance in all three types of problems. Lastly, the last row of the table shows the out-performance of these MA-DAC policies compared with other policies – the IGD values of all the four MA-DAC policies perform significantly better than the best result obtained by the three peer algorithms MOEA/D, DQN and MA-UCB in Table 2.

Table 3: IGD values obtained by MA-DAC (M), MA-DAC (3), MA-DAC (5) and MA-DAC (7) 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 the best value except for MA-DAC in Table 2, respectively, according to the Wilcoxon rank-sum test with confidence level 0.05.

Problem	M	MA-DAC (M)	MA-DAC (3)	MA-DAC (5)	MA-DAC (7)
DTLZ2	3	3.839E-02 (5.35E-04) +	**3.807E-02 (5.05E-04)** +	3.830E-02 (7.24E-04) +	3.837E-02 (5.69E-04) +
	5	2.468E-01 (7.55E-03) +	2.472E-01 (1.56E-02) +	**2.442E-01 (1.26E-02)** +	2.569E-01 (1.39E-02) +
	7	3.921E-01 (8.84E-03) +	**3.880E-01 (1.02E-02)** +	4.081E-01 (1.52E-02) +	3.944E-01 (1.17E-02) +
DTLZ4	3	**6.171E-02 (3.67E-02)** +	6.700E-02 (6.14E-02) +	6.618E-02 (4.62E-02) +	8.088E-02 (7.12E-02) +
	5	3.044E-01 (1.66E-02) ≈	**2.974E-01 (1.94E-02)** +	2.995E-01 (2.10E-02) ≈	3.036E-01 (1.69E-02) ≈
	7	4.271E-01 (1.45E-02) +	4.313E-01 (1.39E-02) ≈	4.327E-01 (2.15E-02) ≈	**4.182E-01 (1.21E-02)** +
WFG4	3	5.220E-02 (9.83E-04) +	**5.200E-02 (1.19E-03)** +	5.236E-02 (1.10E-03) +	5.302E-02 (9.78E-04) +
	5	**1.850E-01 (3.14E-03)** +	1.867E-01 (3.01E-03) +	1.868E-01 (2.81E-03) +	1.853E-01 (2.67E-03) +
	7	3.091E-01 (5.80E-03) +	3.104E-01 (7.14E-03) +	3.100E-01 (5.89E-03) +	**3.033E-01 (3.66E-03)** +
DTLZ2	3	3.839E-02 (5.35E-04) +	**3.807E-02** (5.05E-04) +	3.830E-02 (7.24E-04) +	3.837E-02 (5.69E-04) +
	5	2.468E-01 (7.55E-03) +	2.472E-01 (1.56E-02) +	**2.442E-01** (1.26E-02) +	2.569E-01 (1.39E-02) +
	7	3.921E-01 (8.84E-03) +	**3.880E-01** (1.02E-02) +	4.081E-01 (1.52E-02) +	3.944E-01 (1.17E-02) +
DTLZ4	3	**6.171E-02** (3.67E-02) +	6.700E-02 (6.14E-02) +	6.618E-02 (4.62E-02) +	8.088E-02 (7.12E-02) +
	5	3.044E-01 (1.66E-02) ≈	**2.974E-01** (1.94E-02) +	2.995E-01 (2.10E-02) ≈	3.036E-01 (1.69E-02) ≈
	7	4.271E-01 (1.45E-02) +	4.313E-01 (1.39E-02) ≈	4.327E-01 (2.15E-02) ≈	**4.182E-01** (1.21E-02) +
WFG4	3	5.220E-02 (9.83E-04) +	**5.200E-02** (1.19E-03) +	5.236E-02 (1.10E-03) +	5.302E-02 (9.78E-04) +
	5	**1.850E-01** (3.14E-03) +	1.867E-01 (3.01E-03) +	1.868E-01 (2.81E-03) +	1.853E-01 (2.67E-03) +
	7	3.091E-01 (5.80E-03) +	3.104E-01 (7.14E-03) +	3.100E-01 (5.89E-03) +	**3.033E-01** (3.66E-03) +
WFG5	3	**4.721E-02** (7.15E-04) +	4.730E-02 (7.89E-04) +	4.733E-02 (8.10E-04) +	4.746E-02 (5.90E-04) +
	5	1.811E-01 (2.59E-03) +	1.817E-01 (2.96E-03) +	1.811E-01 (3.02E-03) +	**1.808E-01** (2.83E-03) +
	7	3.256E-01 (5.49E-03) +	3.266E-01 (8.98E-03) +	3.263E-01 (9.73E-03) +	**3.206E-01** (8.04E-03) +
WFG6	3	5.078E-02 (1.20E-02) +	4.831E-02 (8.95E-03) +	**4.599E-02** (9.48E-03) +	5.206E-02 (1.64E-02) +
	5	1.971E-01 (6.40E-03) +	2.003E-01 (6.26E-03) +	**1.942E-01** (6.90E-03) +	1.957E-01 (6.67E-03) +
	7	3.114E-01 (5.08E-03) +	3.242E-01 (9.24E-03) +	3.129E-01 (5.71E-03) +	**3.112E-01** (4.93E-03) +
WFG7	3	4.076E-02 (5.33E-04) +	**4.066E-02** (5.31E-04) +	4.077E-02 (5.12E-04) +	4.124E-02 (4.98E-04) +
	5	1.839E-01 (2.38E-03) +	1.881E-01 (3.70E-03) +	1.858E-01 (2.12E-03) +	**1.836E-01** (2.21E-03) +
	7	3.368E-01 (1.54E-02) +	3.461E-01 (1.97E-02) +	3.390E-01 (1.38E-02) +	**3.258E-01** (1.25E-02) +
WFG8	3	**7.828E-02** (1.46E-03) +	7.901E-02 (1.19E-03) +	7.921E-02 (1.36E-03) +	7.944E-02 (1.30E-03) +
	5	2.506E-01 (1.11E-02) +	2.653E-01 (1.51E-02) +	**2.479E-01** (7.20E-03) +	2.532E-01 (9.28E-03) +
	7	4.303E-01 (1.49E-02) +	4.364E-01 (1.38E-02) +	4.242E-01 (9.08E-03) +	**4.127E-01** (5.93E-03) +
WFG9	3	4.324E-02 (7.07E-04) +	**4.159E-02** (6.10E-04) +	4.359E-02 (1.00E-02) +	6.415E-02 (2.64E-02) +
	5	1.858E-01 (7.63E-03) +	**1.814E-01** (4.59E-03) +	1.832E-01 (7.10E-03) +	1.918E-01 (1.13E-02) +
	7	3.328E-01 (1.02E-02) +	3.298E-01 (1.03E-02) +	3.307E-01 (1.37E-02) +	**3.278E-01** (7.21E-03) +
average rank	3	2.0	1.625	2.5	3.875
	5	2.375	3.0	2.125	2.5
	7	2.375	3.25	3.125	1.25
+/−/≈	3	8/0/0	8/0/0	8/0/0	8/0/0
	5	7/0/1	8/0/0	7/0/1	7/0/1
	7	8/0/0	7/0/1	7/0/1	8/0/0

RQ3: How do the different parts of MA-DAC affect the performance? We conduct ablation studies to show the importance of different types of hyperparameters, and the results are given in Table 4. We use MA-DAC (M) w/o i to denote the MA-DAC policy that does not include the i-th agent, and we use a reasonable setting as the default for each ablated agent; the detailed settings are provided in Appendix B. That is, MA-DAC (M) w/o 1, 2, 3, and 4 represent MA-DAC (M) without tuning weights, neighborhood size, types of reproduction operators, and parameters of the reproduction operators, respectively.

As can be seen in Table 4. MA-DAC (M) outperforms all the ablations, demonstrating the importance of tuning every type of hyperparameter. On the other hand, we notice that the importance of hyperparameters varies. For example, adaptive weights are in general more important as the performance of MA-DAC (M) w/o 1 drops significantly. In addition, we have conducted the ablation studies for the reward; this can be found in Appendix B due to space limitations.

Table 4: IGD values obtained by MA-DAC (M) w/o 1, MA-DAC (M) w/o 2, MA-DAC (M) w/o 3 and MA-DAC (M) w/o 4 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 (M), respectively, according to the Wilcoxon rank-sum test with confidence level 0.05.

Problem	M	MA-DAC (M) w/o 1	MA-DAC (M) w/o 2	MA-DAC (M) w/o 3	MA-DAC (M) w/o 4
DTLZ2	3	4.656E-02 (3.80E-04) $-$	**3.914E-02** (8.43E-04) $-$	3.935E-02 (6.72E-04) $-$	3.919E-02 (5.91E-04) $-$
	5	3.086E-01 (7.24E-03) $-$	2.619E-01 (8.99E-03) $-$	2.503E-01 (1.30E-02) $-$	**2.433E-01** (1.59E-02) $\approx$
	7	4.970E-01 (1.26E-02) $-$	4.067E-01 (1.20E-02) $-$	**4.003E-01** (1.19E-02) $-$	4.228E-01 (1.25E-02) $-$
DTLZ4	3	6.463E-02 (3.85E-02) $-$	6.242E-02 (4.07E-02) $-$	**4.496E-02** (2.45E-03) $+$	4.496E-02 (2.45E-03) $+$
	5	3.497E-01 (1.41E-02) $-$	3.061E-01 (2.12E-02) $\approx$	**3.054E-01** (1.55E-02) $\approx$	3.130E-01 (1.81E-02) $-$
	7	4.853E-01 (1.82E-02) $-$	4.275E-01 (1.90E-02) $-$	**4.208E-01** (1.52E-02) $\approx$	4.289E-01 (2.07E-02) $-$
WFG4	3	7.222E-02 (1.93E-03) $-$	5.484E-02 (1.01E-03) $-$	**5.410E-02** (8.85E-04) $-$	5.410E-02 (8.85E-04) $-$
	5	2.868E-01 (1.01E-02) $-$	1.879E-01 (3.76E-03) $\approx$	**1.845E-01** (2.17E-03) $+$	1.846E-01 (2.39E-03) $+$
	7	3.758E-01 (1.33E-02) $-$	3.102E-01 (6.34E-03) $-$	**3.020E-01** (3.99E-03) $\approx$	3.032E-01 (4.32E-03) $\approx$
WFG5	3	6.189E-02 (7.40E-04) $-$	4.827E-02 (6.31E-04) $-$	**4.766E-02** (5.72E-04) $\approx$	4.766E-02 (5.72E-04) $\approx$
	5	3.202E-01 (9.77E-03) $-$	**1.821E-01** (2.73E-03) $\approx$	1.835E-01 (2.90E-03) $-$	1.822E-01 (2.31E-03) $-$
	7	4.948E-01 (1.47E-02) $-$	3.290E-01 (8.87E-03) $-$	3.310E-01 (7.70E-03) $-$	**3.261E-01** (9.26E-03) $-$
WFG6	3	6.864E-02 (8.14E-03) $-$	**5.338E-02** (1.37E-02) $-$	6.543E-02 (1.69E-02) $-$	6.067E-02 (2.11E-02) $\approx$
	5	3.480E-01 (1.34E-02) $-$	2.005E-01 (5.21E-03) $-$	1.996E-01 (6.51E-03) $-$	**1.979E-01** (6.76E-03) $-$
	7	4.784E-01 (3.37E-02) $-$	**3.147E-01** (5.85E-03) $-$	3.162E-01 (6.15E-03) $-$	3.147E-01 (5.73E-03) $-$
WFG7	3	6.004E-02 (9.45E-04) $-$	4.250E-02 (5.82E-04) $-$	**4.150E-02** (6.60E-04) $-$	4.150E-02 (6.60E-04) $-$
	5	3.402E-01 (2.49E-02) $-$	1.873E-01 (3.67E-03) $\approx$	**1.826E-01** (2.60E-03) $+$	1.847E-01 (2.80E-03) $\approx$
	7	4.877E-01 (5.70E-02) $-$	3.393E-01 (1.23E-02) $-$	**3.373E-01** (1.16E-02) $-$	3.377E-01 (1.59E-02) $-$
WFG8	3	9.661E-02 (1.87E-03) $-$	8.374E-02 (1.70E-03) $-$	**8.029E-02** (1.29E-03) $-$	8.029E-02 (1.29E-03) $-$
	5	4.119E-01 (1.30E-02) $-$	2.695E-01 (1.49E-02) $-$	**2.571E-01** (1.09E-02) $-$	2.632E-01 (9.96E-03) $-$
	7	5.830E-01 (1.59E-02) $-$	4.345E-01 (9.27E-03) $-$	**4.260E-01** (8.71E-03) $-$	4.322E-01 (1.12E-02) $-$
WFG9	3	5.894E-02 (9.24E-04) $-$	**4.321E-02** (7.50E-04) $-$	4.650E-02 (1.40E-02) $-$	4.650E-02 (1.40E-02) $-$
	5	3.148E-01 (2.06E-02) $-$	1.875E-01 (5.14E-03) $-$	1.941E-01 (6.45E-03) $-$	**1.865E-01** (9.02E-03) $-$
	7	5.069E-01 (2.53E-02) $-$	3.433E-01 (1.35E-02) $-$	3.402E-01 (1.00E-02) $-$	**3.368E-01** (1.05E-02) $-$
$+/-/\approx$		0/24/0	0/20/4	3/17/4	2/17/5

6 Conclusion

This paper considers the dynamic configuration of algorithms with heterogeneous types of hyperparameters. We propose the framework MA-DAC to solve it, where one agent works to handle one type of configuration hyperparameters. Experimental results show that MA-DAC works well and has good generalization ability. The instantiation of configuring MOEA/D forms the benchmark MaMo for MARL, with the hope of facilitating the application of MARL. In the future, we would like to propose a better contextual MMDP formulation and better cooperative MARL algorithm based on the heterogeneity and stochasticity of MA-DAC. It is also interesting to include more real-world optimization problems into MaMo.

References

[1] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. A large-scale experimental evaluation of high-performing multi- and many-objective evolutionary algorithms. *Evolutionary Computation*, 26(4):621–656, 2018.

[2] André Biedenkapp, H. Furkan Bozkurt, Theresa Eimer, Frank Hutter, and Marius Lindauer. Dynamic algorithm configuration: Foundation of a new meta-algorithmic framework. In *24th European Conference on Artificial Intelligence*, pages 427–434, 2020.

[3] 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.

[4] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*, pages 195–210, 1996.

[5] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference*, pages 746–752, 1998.

[6] Christian Daniel, Jonathan Taylor, and Sebastian Nowozin. Learning step size controllers for robust neural network training. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1519–1525, 2016.

[7] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2): 182–197, 2002.

[8] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 825–830, 2002.

[9] Theresa Eimer, André Biedenkapp, Maximilian Reimer, Steven Adriaensen, Frank Hutter, and Marius Lindauer. DACBench: A benchmark library for dynamic algorithm configuration. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1668–1674, 2021.

[10] Alessio Greco, Salvatore Danilo Riccio, Jon Timmis, and Giuseppe Nicosia. Assessing algorithm parameter importance using global sensitivity analysis. In *Analysis of Experimental Algorithms - Special Event, SEA*^{2}, pages 392–407, 2019.

[11] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, 2019.

[12] Wenjing Hong, Peng Yang, Yiwen Wang, and Ke Tang. Multi-objective magnitude-based pruning for latency-aware deep neural network compression. In *Parallel Problem Solving from Nature - PPSN XVI - 16th International Conference*, pages 470–483, 2020.

[13] 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.

[14] Simon Huband, Philip Hingston, Luigi Barone, and Lyndon While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

[15] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. Paramils: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36: 267–306, 2009.

[16] Hisao Ishibuchi, Naoya Akedo, and Yusuke Nojima. Relation between neighborhood size and MOEA/D performance on many-objective problems. In *Evolutionary Multi-Criterion Optimization - 7th International Conference*, pages 459–474, 2013.

[17] 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.

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

[19] Miqing Li and Xin Yao. What weights work for you? adapting weights for any pareto front shape in decomposition-based evolutionary multiobjective optimisation. *Evolutionary Computation*, 28(2):227–253, 2020.

[20] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30*, pages 6379–6390, 2017.

[21] Zhichao Lu, Kalyanmoy Deb, Erik D. Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. In *Computer Vision - ECCV 2020 - 16th European Conference*, pages 35–51, 2020.

[22] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[24] A. OroojlooyJadid and D. Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1908.03963*, 2019.

[25] 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 1*, 2021.

[26] 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.

[27] Sutton Richard S. and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[28] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2186–2188, 2019.

[29] Gresa Shala, André Biedenkapp, Noor H. Awad, Steven Adriaensen, Marius Lindauer, and Frank Hutter. Learning step-size adaptation in CMA-ES. In *Proceedings of the 16th Parallel Problem Solving from Nature*, pages 691–706, 2020.

[30] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5887–5896, 2019.

[31] David Speck, André Biedenkapp, Frank Hutter, Robert Mattmüller, and Marius Lindauer. Learning heuristic selection with dynamic algorithm configuration. In *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling*, pages 597–605, 2021.

[32] 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.

[33] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2085–2087, 2018.

[34] Justin K. Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S. Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, Niall L. Williams, Yashas Lokesh, and Praveen Ravi. Pettingzoo: Gym for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems 34*, pages 15032–15043, 2021.

[35] 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.

[36] 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.

[37] Ceyer Wakilpoor, Patrick J Martin, Carrie Rebhuhn, and Amanda Vu. Heterogeneous multi-agent reinforcement learning for unknown environment mapping. *arXiv preprint arXiv:2010.02663*, 2020.

[38] Jianhong Wang, Wangkun Xu, Yunjie Gu, Wenbin Song, and Tim C. Green. Multi-agent reinforcement learning for active voltage control on power distribution networks. In *Advances in Neural Information Processing Systems 34*, pages 3271–3284, 2021.

[39] 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.

[40] Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*, 2020.

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

[42] Lianmin Zheng, Jiacheng Yang, Han Cai, Ming Zhou, Weinan Zhang, Jun Wang, and Yong Yu. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[43] 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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Abstract and Section1.
   (b) Did you describe the limitations of your work? [Yes] We describe the limitations in Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] The authors do not see obvious negative societal impacts of the proposed method
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Our paper conforms to them.

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Yes, see Section 5, Appendix and Supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Yes, see Section 5 and Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Yes, see Section 5 and Appendix.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Yes, see Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See References.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]