Prediction Guided Meta-Learning for Multi-Objective Reinforcement Learning

Fei-Yu Liu
School of Computer Science and Technology
University of Science and Technology of China
Hefei, China
lfy0012@mail.ustc.edu.cn

Chao Qian
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
qianc@lamda.nju.edu.cn

2021 IEEE Congress on Evolutionary Computation (CEC) | 978-1-7281-8393-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/CEC45853.2021.9504972

***Abstract*—Many real-world control problems consist of several different, possibly conflicting, objectives, which require finding a high-quality set of policies that are optimal for different objective preferences. Extensive research mainly focused on how to obtain a high-quality approximated Pareto set of policies, while another important research direction studies how to adapt to new objective preferences quickly. In this paper, we propose a new multi-objective reinforcement learning (MORL) algorithm so-called PG-Meta-MORL for achieving both goals. PG-Meta-MORL frames MORL as a meta-learning problem and iteratively optimizes a meta-policy using multiple tasks with objective preferences selected based on a prediction model, which is trained to guide the optimization process towards best improving the quality of the current Pareto set of policies. The empirical results on several multi-objective continuous control problems show that PG-Meta-MORL can find a high-quality approximated Pareto set of policies, and meanwhile, the obtained meta-policy can be adapted well to new objective preferences using few-shot interactions with the environment.**

I. Introduction

Reinforcement learning (RL) is a framework for training agents to take actions in a given environment to maximize the expected cumulative rewards [1]. In most real-world scenarios, especially robotic control, one has to make tradeoffs for different performance metrics, because the concept of performance usually involves different conflicting objectives. For example, considering the problem that a MuJoCo [2] humanoid robot consumes energy to move forward, we need to consider two conflicting objectives: forward speed and energy efficiency. Traditional RL [3]–[5], addresses this issue by synthesizing multiple reward functions, corresponding to multiple objectives, as a single scalar reward function, and thus only one optimal policy exists. In contrast to traditional RL, MORL [6]–[8] measures the performance of a control policy by using multiple objectives explicitly and tries to solve the resulting multi-objective optimization problem. For MORL, there are multiple Pareto optimal policies, attaining different optimal trade-offs among multiple objectives. Considering the example of MuJoCo [2] humanoid robot as shown in Figure 1, when a task requires the robot to keep high speed, the corresponding Pareto optimal policy will have high speed at the cost of low energy efficiency; when a task requires the robot to reduce energy consumption, the corresponding Pareto optimal policy will have high energy efficiency at the cost of low speed. In general, the goal of MORL is to find the set of all Pareto optimal policies.

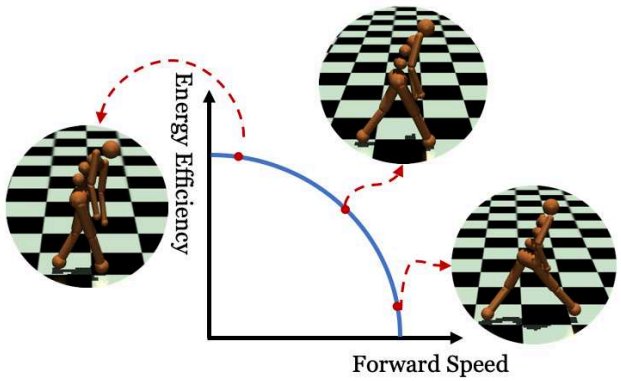


Fig. 1. Pareto optimal policies in the MuJoCo humanoid control problem.

However, learning Pareto optimal policies over multi-objective control problems has shown to be quite challenging [9], [10]. Two recent methods are proposed to solve this problem. Chen *et al.* [11] proposed the Meta-MORL algorithm that iteratively trains a meta-policy using multiple tasks over Markov decision processes (MDPs) with different objective preferences, sampled from a distribution. Such a meta-policy is not necessarily optimal, but can be quickly adapted to new objective preferences. Unfortunately, the control policies adapted from the meta-policy are often sub-optimal. This may be because, in many real-world problems, there is no sufficient prior knowledge to help us design a good distribution to sample tasks, and thus the tasks selected for training the meta-policy in each iteration are often sub-optimal, leading to a low-quality meta-policy. The PG-MORL algorithm proposed by Xu *et al.* [12] tries to find a high-quality and dense set of policies to approximate the optimal Pareto set, by selecting the tasks (i.e., optimization directions), which can bring the most improvement on the quality of the set of policies. However, PG-MORL does not generate a meta-policy, and thus, it is hard to adapt to new objective preferences quickly.

In this paper, by combining the merits of Meta-MORL [11] and PG-MORL [12], we propose a new algorithm so-called PG-Meta-MORL for solving multi-objective continuous control problems. PG-Meta-MORL iteratively trains a meta-policy using multiple tasks with objective preferences, which are selected to be those tasks improving the quality of the current Pareto set of policies the most based on a prediction model. Specifically, in each iteration, an analytical model is fitted for the meta-policy to predict the expected improvement along each optimization direction (i.e., objective preference). And then the meta-policy is trained with those selected optimization directions that are expected to best improve the quality of the current Pareto set of policies. As a result, PG-Meta-MORL can find a high-quality approximated Pareto set of policies when the objective preferences are not specified, and meanwhile, the obtained meta-policy can be quickly adapted to optimal control policies when new objective preferences are given.

The rest of this paper is organized as follows. We introduce the multi-objective control problem in Section II. In Section III, we outline the previous methods for solving multi-objective control problems. Section IV provides the overview and details of our algorithm. Section V demonstrates our empirical results on several multi-objective control problems. Finally, in Section VI, we conclude this paper.

II. Preliminaries

A. Multi-Objective Markov Decision Process

A multi-objective control problem can be characterized by a multi-objective MDP (MOMDP), which can be represented by the tuple $\langle \mathcal{S}, \mathcal{A}, P, \boldsymbol{R}, \boldsymbol{\gamma}, p(s_0) \rangle$, where $\mathcal{S}$ is the (continuous) state space, $\mathcal{A}$ is the (continuous) action space, $P(s'|s,a)$ is the state transition probability, i.e., the probability of jumping to state s' after performing action a at state s, $\boldsymbol{R} = [r_1, ..., r_m]^\top$ with $r_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a vector of the reward functions corresponding to m different objectives, $\boldsymbol{\gamma} = [\gamma_1, ..., \gamma_m]^\top$ with $\gamma_i \in [0, 1]$ is a vector of the discount factors corresponding to m different objectives, and $p(s_0)$ is the initial state distribution.

In MOMDPs, the performance of a policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ is measured by a vector of expected returns $\boldsymbol{F}(\pi) = \boldsymbol{G}^\pi = [G_1^\pi, ..., G_m^\pi]^\top$ with

$$G_i^\pi = \mathbb{E}\left[\sum_{t=0}^{T} \gamma_i^t r_i(s_t, a_t) | s_0 \sim p(s_0), a_t \sim \pi_\theta(s_t)\right],$$

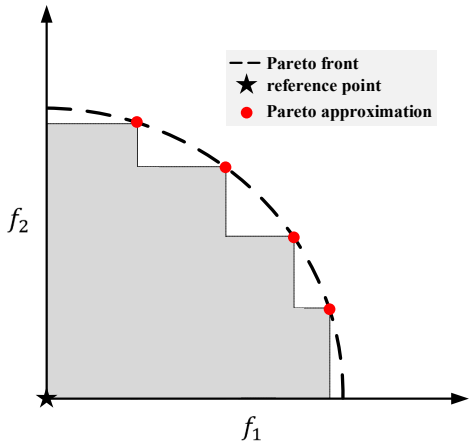
where T is the horizon, and π is used to denote π_θ for brevity.

B. Multi-Objective Control Problem

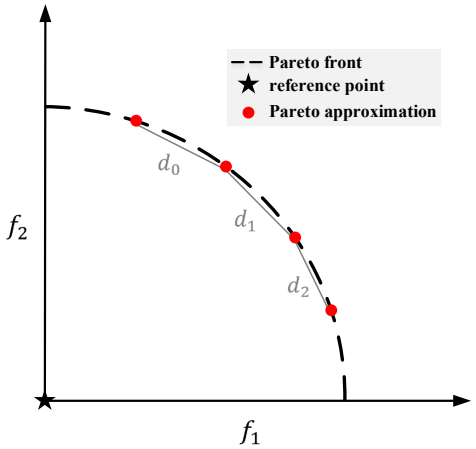
A multi-objective control problem can be formulated as

$$\max_\pi \boldsymbol{F}(\pi) = \max_\pi [G_1^\pi, ..., G_m^\pi]^\top.$$

A policy π dominates another policy π' if $\boldsymbol{F}(\pi) \geq \boldsymbol{F}(\pi')$ and $\boldsymbol{F}(\pi) \neq \boldsymbol{F}(\pi')$, i.e., $\forall i, G_i^\pi \geq G_i^{\pi'}, \exists j, G_j^\pi > G_j^{\pi'}$. A policy π is Pareto optimal if and only if it is not dominated by any other policy. A set of Pareto optimal policies is called the Pareto set. And the set of Pareto set outcomes in objective space, i.e., the set of objective vectors of the Pareto optimal policies in the Pareto set, is called the Pareto front. In multi-objective control problems, there is no single optimal policy that can maximize all the objectives. The goal of multi-objective control problems is to obtain the optimal Pareto set, i.e., the Pareto set containing all the Pareto optimal policies.



(a) Hypervolume metric



(b) Sparsity metric

Fig. 2. Metrics in bi-objective space. (a) Hypervolume metric measures the shaded area dominated by the Pareto optimal solutions and dominating the reference point. (b) Sparsity metric measures the average square distance between consecutive points in the Pareto front. In this case, $S = \frac{1}{3}(d_0^2 + d_1^2 + d_2^2)$.

In many complicated control problems, the optimal Pareto set is impossible to obtain, and thus most MORL methods try to find a set of policies that can best approximate the optimal Pareto set. To measure the quality of an approximated Pareto set, two metrics are often considered [13]:

Definition 1: (Hypervolume [14]) Let Q be the Pareto front of an approximated Pareto set and $\boldsymbol{r} \in \mathbb{R}^m$ be the reference point. The hypervolume metric $\mathcal{H}(Q)$ is

$$\mathcal{H}(Q) = \int \mathbf{1}_{H(Q)}(\boldsymbol{x}) d\boldsymbol{x},$$

where $H(Q) = \{\boldsymbol{x} | \exists 1 \leq i \leq |Q|, \boldsymbol{r} \preceq \boldsymbol{x} \preceq Q(i)\}$, $\preceq$ is the operator of objective dominance, $Q(i)$ is the i-th solution in Q, and $\mathbf{1}_{H(Q)}$ is a Dirac delta function that equals 1 if $\boldsymbol{x} \in H(Q)$ and 0 otherwise.

Definition 2: (Sparsity [12]) Let Q be the Pareto front of an approximated Pareto set. The sparsity metric $\mathcal{S}(Q)$ is

$$\mathcal{S}(Q) = \frac{1}{|Q|-1} \sum_{i=1}^{m} \sum_{j=1}^{|Q|-1} (\widehat{Q}_i(j) - \widehat{Q}_i(j+1))^2,$$

where $\widehat{Q}_i(j)$ is the j-th value in the sorted list of Q for the i-th objective.

The hypervolume metric (as illustrated in Figure 2(a)) measures the convergence of an approximated Pareto set towards the optimal Pareto set, and the sparsity metric (as illustrated in Figure 2(b)) measures the density of an approximated Pareto set. Intuitively, in a multi-objective control problem, we aim to obtain an approximated Pareto set that is expected to have a high hypervolume metric and a low sparsity metric.

III. Previous Methods

A. Multi-Objective Reinforcement Learning

As real-world control problems often involve multiple objectives, traditional RL methods adopt various scalarization techniques to convert the multi-objective problem into a single-objective problem [15], [16]. These methods aim to explore different forms of scalarization functions, including non-linear ones such as the minimum over all objectives [16]. However, these methods do not work when the objective preference is unknown or it changes over time.

MORL methods try to solve the multi-objective problems directly by finding a set of policies that can best approximate the optimal Pareto set. The common approaches repeatedly solve the single-objective problems with strategically-designed scalarizations [6]–[8]. For example, the Radial algorithm (RA) [6] obtains an approximated Pareto set of policies by performing gradient ascent in the policy parameter space to solve different single-objective problems, which are generated by different convex combinations of the objectives. The Prediction Guided MORL (PG-MORL) [12] aims to find a high-quality and dense set of Pareto optimal policies using reinforcement learning strategies based on a novel prediction-guided evolutionary learning algorithm. Other approaches learn a set of optimal policies by using a multi-objective variant of the Q-learning update rule [17]–[20]. For example, the multi-objective fitted Q-iteration algorithm (MOFQI) [20] encapsulates objective preferences as input to a Q-function approximator and uses expanded historical trajectories to learn multiple optimal policies. However, these methods are either difficult to scale up to high-dimensional objective spaces or cannot be easily adapted to new objective preferences.

B. Multi-Objective Evolutionary Algorithms

Multi-objective evolutionary algorithms (MOEAs) are effective methods for solving multi-objective optimization problems, and thus can also be applied to solve MORL problems. MOEAs maintain a set of solutions (called a population), and iteratively improve the population by using genetic operators such as mutation and crossover operators [21]. For example, MOEA/D [22] and NSGA-II [23] are two popular MOEAs. MOEA/D [22] decomposes a multi-objective optimization problem into a number of single-objective optimization sub-problems (or simple multi-objective optimization sub-problems) and then uses a search heuristic to optimize these sub-problems simultaneously and cooperatively. NSGA-II [23] generates offsprings using a specific type of crossover and mutation and then selects the next population according to non-dominated sorting and crowding distance comparison. Though MOEAs are suitable for solving MORL, they may be inefficient in searching for optimal solutions when the parameter space is extremely large.

Algorithm 1 Meta-MORL Algorithm [11]

```
Randomly initialize the meta-policy π_θ.
while training π_θ do
    Sample trajectories D following π_θ.
    for task 1 to n do
        Sample a preference vector ω^i randomly.
        Update π_{θ_i} with D by
            θ_i = arg max_{θ'} L_{ω^i}(θ') | θ_init = θ.
        Sample trajectories D_i following π_{θ_i}.
    end for
    Update the meta-policy with D_i, ω^i by
        θ = arg max_{θ'} Σ_{i=1}^{n} L_{ω^i}(θ') | θ_init = θ.
end while
Construct the Pareto front by the fine-tuning phase.
```

C. Meta-Reinforcement Learning

MORL can also be solved in a meta-learning [24] framework, by training a meta-policy and adapting the meta-policy to the control policies w.r.t. different objective preferences. For example, Chen *et al.* [11] proposed the Meta-MORL algorithm to train deep policies for complex continuous control tasks based on meta-learning. Meta-MORL, as presented in Algorithm 1, iteratively trains a meta-policy using multiple tasks over MDPs with different objective preferences, sampled from a distribution. Specifically, Meta-MORL consists of three phases: (1) the adaptation phase, in which a number of tasks with different objective preferences are randomly selected and the corresponding policies are trained with the help of the current meta-policy, (2) the meta-policy training phase, in which the meta-policy is updated by aggregating data generated by the control policies trained in the adaptation phase, and (3) the fine-tuning phase, in which a set of Pareto optimal policies is obtained by initializing the policies with the meta-policy and training them for a number of iterations. Note that the trained meta-policy can be quickly adapted to control policies w.r.t new objective preferences. However, the goodness of adapted control policies depends on the quality of the meta-policy, which highly depends on the selected tasks. Meta-MORL selects the tasks randomly, which may lead to the low-quality of the resulting meta-policy. Thus, in this work, we will improve Meta-MORL by selecting tasks greedily based on a prediction model.

Algorithm 2 PG-Meta-MORL Algorithm

Input: Step size hyperparameters α, β.
Initialize: RL data buffer $\mathcal{R}$, Pareto archive $\mathcal{P}$, prediction model $\mathbf{\Delta}$.
Randomly initialize a meta-policy $\pi_{\boldsymbol{\theta}}$ and generate n evenly distributed preference weights $\{\boldsymbol{\omega}^1, ..., \boldsymbol{\omega}^n\}$.
while not done **do**
  Sample K trajectories $\mathcal{D} = \{(s_1, a_1, ..., s_H)\}$ using $\pi_{\boldsymbol{\theta}}$.
  for task 1 to n **do**
    Evaluate $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\omega}^i}(\pi_{\boldsymbol{\theta}})$ using $\mathcal{D}$ and $\mathcal{L}_{\boldsymbol{\omega}^i}$ in Eq. (1).
    Compute adapted policy $\pi_{\boldsymbol{\theta}_i}$ with gradient descent: $\boldsymbol{\theta}_i = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\omega}^i}(\pi_{\boldsymbol{\theta}})$.
    Sample trajectories $\mathcal{D}_i = \{(s_1, a_1, ..., s_H)\}$ using $\pi_{\boldsymbol{\theta}_i}$.
    Store $(\boldsymbol{\omega}^i, \boldsymbol{F}(\pi_{\boldsymbol{\theta}}), \boldsymbol{F}(\pi_{\boldsymbol{\theta}_i}))$ in $\mathcal{R}$.
    Data Augment $(\mathcal{D}_i, \pi_{\boldsymbol{\theta}_i}, \mathcal{R}, \mathcal{P})$.
  end for
  Update meta-policy $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{n} \mathcal{L}_{\boldsymbol{\omega}^i}(\pi_{\boldsymbol{\theta}_i})$ with $\mathcal{D}_i, \boldsymbol{\omega}^i$.
  Update $\mathcal{P}$ with $\{(\pi_{\boldsymbol{\theta}_i}, \boldsymbol{F}(\pi_{\boldsymbol{\theta}_i}))\}_{1...n}$.
  Fit the improvement prediction model $\mathbf{\Delta}$ with $\mathcal{R}$.
  $\{\boldsymbol{\omega}^1, ..., \boldsymbol{\omega}^n\} \leftarrow$ Task Selection $(n, \mathcal{P}, \mathbf{\Delta})$.
end while
Output: Pareto archive $\mathcal{P}$ and meta-policy $\pi_{\boldsymbol{\theta}}$.

IV. The PG-Meta-MORL Algorithm

A. Algorithm Overview

A general flow of our algorithm is shown in Algorithm 2. It first initializes an RL data buffer $\mathcal{R}$ which is used to collect data for training the prediction model $\mathbf{\Delta}$, and a Pareto archive $\mathcal{P}$ which contains all non-dominated policies generated-so-far and their corresponding expected returns. Then it starts with initializing a meta-policy $\pi_{\boldsymbol{\theta}}$ and generating n evenly distributed preference weights $\{\boldsymbol{\omega}^1, ..., \boldsymbol{\omega}^n\}$. Each iteration consists of two phases: (1) meta RL phase and (2) task selection phase, which are inspired by Meta-MORL [11] and PG-MORL [12], respectively. In the meta RL phase, a number of policies starting from the meta-policy are updated (i.e., adapted from the meta-policy) with different preference weights for a number of iterations, and then a data augment step (i.e., Algorithm 3) will be conducted for collecting RL data, afterward the meta-policy is updated by aggregating data generated by these adapted policies. In the task selection phase, an analytical model is fitted with the RL data buffer $\mathcal{R}$ for the meta-policy to predict the expected improvement along each optimization direction, and then the algorithm will select n new preference weights that are expected to best improve the quality of the Pareto archive $\mathcal{P}$ for the next iteration (Algorithm 4).

B. Meta RL Phase

Given a policy $\pi_{\boldsymbol{\theta}}$ and a preference weight $\boldsymbol{\omega}$ where $\sum_i \omega_i = 1$, the policy $\pi_{\boldsymbol{\theta}}$ is optimized by maximizing the weighted-sum returns $\mathcal{J}(\boldsymbol{\theta}, \boldsymbol{\omega})$:

$$\mathcal{J}(\boldsymbol{\theta}, \boldsymbol{\omega}) = \boldsymbol{\omega}^\top \boldsymbol{G}^\pi = \sum_{i=1}^{m} \omega_i G_i^\pi.$$

Because scalar value functions have proven to be inefficient in multi-objective control problems [18], we use a vectorized value function $\boldsymbol{V}^\pi(s)$ for a better representation, which maps a state s to the vector of expected returns under policy π. The vectorized value function $\boldsymbol{V}^\pi(s)$ is updated with the mean squared error loss $\mathbb{E}_{s\sim\mathcal{S}} \|\boldsymbol{V}^\pi(s) - \widehat{\boldsymbol{V}}(s)\|^2$, where $\widehat{\boldsymbol{V}}(s)$ is the target value. The policy $\pi_{\boldsymbol{\theta}}$ is updated with the clipped version of PPO loss [25]:

$$\mathcal{L}_{\boldsymbol{\omega}}(\boldsymbol{\theta}) = \min\left(\delta \boldsymbol{A}_{\boldsymbol{\omega}}(s_t, a_t), \text{clip}\left(\delta, 1-\epsilon, 1+\epsilon\right) \boldsymbol{A}_{\boldsymbol{\omega}}(s_t, a_t)\right),$$

where ϵ is a threshold hyperparameter, δ is the important sampling ratio $\pi_{\boldsymbol{\theta}}(a_t|s_t)/\pi_{\boldsymbol{\theta}_{old}}(a_t|s_t)$, clip is a function to limit the value range of δ, and $\boldsymbol{A}_{\boldsymbol{\omega}}(s_t, a_t)$ is the weighted-sum scalarization of the vectorized advantage function $\boldsymbol{A}(s_t, a_t)$ for individual objectives.

The meta RL phase contains two steps: the adaptation step and the meta-policy training step. In the adaptation step, each worker optimizes one policy based on the meta-policy $\pi_{\boldsymbol{\theta}}$ and one preference weight. That is, the policies are updated for a few iterations using trajectories generated by running the meta-policy with their corresponding preference weights. In short, each policy $\pi_{\boldsymbol{\theta}_i}$ is trained as,

$$\boldsymbol{\theta}_i = \arg\max_{\boldsymbol{\theta}'} \mathcal{L}_{\boldsymbol{\omega}^i}(\boldsymbol{\theta}') | \boldsymbol{\theta}_{\text{init}} = \boldsymbol{\theta}. \tag{1}$$

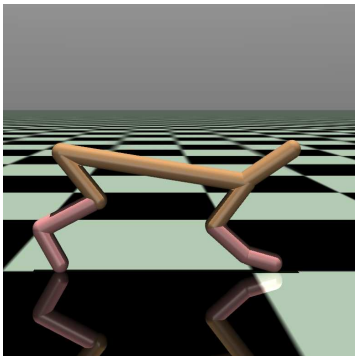
In the meta-policy training step, the meta-policy $\pi_{\boldsymbol{\theta}}$ is updated by aggregating data generated by those adapted policies $(\pi_{\boldsymbol{\theta}_i}, i = 1, ..., n)$ in the adaptation step, i.e.,

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}'} \sum_{i=1}^{n} \mathcal{L}_{\boldsymbol{\omega}^i}(\boldsymbol{\theta}') | \boldsymbol{\theta}_{\text{init}} = \boldsymbol{\theta}. \tag{2}$$

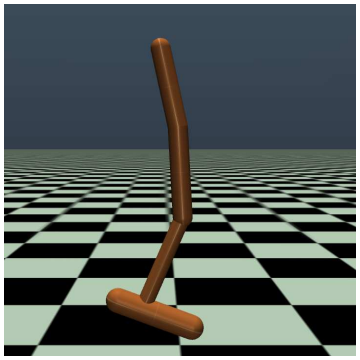
Before the meta-policy training step, there is a data augment step for collecting enough RL data to fit a prediction model later. The data augment algorithm as presented in Algorithm 3 randomly samples K preference weights and updates each adapted control policy for a few iterations using trajectories generated by itself. Note that, the data augment step does not need additional interactions with the environment.

Algorithm 3 Data Augment Algorithm

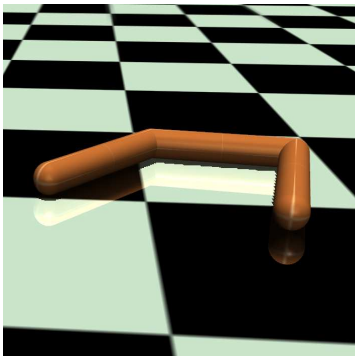
Input: Trajectories $\mathcal{D}$, policy $\pi_{\boldsymbol{\theta}}$, RL data buffer $\mathcal{R}$, Pareto archive $\mathcal{P}$.
Sample M preference weights $\{\boldsymbol{\omega}^i\}_{i=1}^M$ randomly.
for i from 1 to M **do**
  Evaluate $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\omega}^i}(\pi_{\boldsymbol{\theta}})$ using $\mathcal{D}$ and $\mathcal{L}_{\boldsymbol{\omega}^i}$ in Eq. (1).
  Compute new policy $\pi_{\boldsymbol{\theta}_i}$ with gradient descent: $\boldsymbol{\theta}_i = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\omega}^i}(\pi_{\boldsymbol{\theta}})$.
  Store $(\boldsymbol{\omega}^i, \boldsymbol{F}(\pi_{\boldsymbol{\theta}}), \boldsymbol{F}(\pi_{\boldsymbol{\theta}_i}))$ in $\mathcal{R}$.
end for
Update $\mathcal{P}$ with $\{(\pi_{\boldsymbol{\theta}_i}, \boldsymbol{F}(\pi_{\boldsymbol{\theta}_i}))\}_{1...M}$.



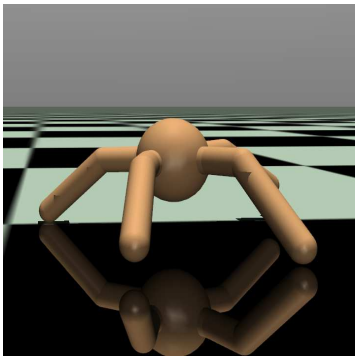
(a) HalfCheetah



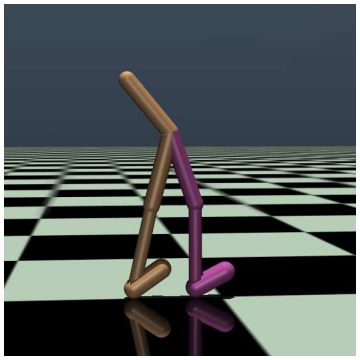
(b) Hopper



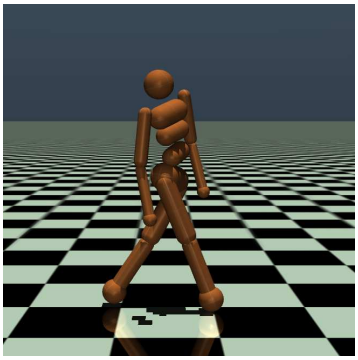
(c) Swimmer



(d) Ant



(e) Walker2d



(f) Humanoid

Fig. 3. Illustration of the MuJoCo control environments.

Algorithm 4 Task Selection Algorithm

1: **Input**: Number n of preference weights, Pareto archive $\mathcal{P}$, prediction model $\mathbf{\Delta}$.
2: **Initialize**: Virtual Pareto archive $\widehat{\mathcal{P}} = \mathcal{P}$.
3: **for** i from 1 to n **do**
4: Initialize new preference weight $\boldsymbol{\omega}^i = \mathbf{0}$.
5: **for each** candidate weights $\widehat{\boldsymbol{\omega}}$ **do**
6: **if** $\mathcal{O}(\widehat{\mathcal{P}}, \widehat{\boldsymbol{\omega}}) > \mathcal{O}(\widehat{\mathcal{P}}, \boldsymbol{\omega}^i)$ **then**
7: $\boldsymbol{\omega}^i = \widehat{\boldsymbol{\omega}}$.
8: **end if**
9: **end for**
10: Update $\widehat{\mathcal{P}}$ by inserting $\boldsymbol{\omega}^i$ and its predicted expected return $F(\pi_{\boldsymbol{\theta}}) + \Delta(\boldsymbol{\omega}^i)$.
11: **end for**
12: **Output**: New preference weights $\{\boldsymbol{\omega}^1, ..., \boldsymbol{\omega}^n\}$.

C. Task Selection Phase

In each iteration of PG-Meta-MORL, instead of selecting tasks randomly, it aims to select the most important tasks (each task corresponding to a preference weight) that can best improve the quality of the current Pareto archive $\mathcal{P}$, measured by the hypervolume $\mathcal{H}$ and sparsity $\mathcal{S}$ metrics in Definitions 1 and 2. Specifically, in line 16 of Algorithm 2, it needs to select n new tasks to be processed in the next iteration. Let $F(\mathcal{P})$ denote the set of objective vectors in $\mathcal{P}$, and $\widetilde{F}(\mathcal{P})$ denote the set of non-dominated objective vectors in $\mathcal{P}$. The selected tasks seek to maximize a hybrid metric $\mathcal{H}(\widetilde{F}(\widehat{\mathcal{P}})) - \eta\mathcal{S}(\widetilde{F}(\widehat{\mathcal{P}}))$, where $\eta \in [0, 1]$ and $\widehat{\mathcal{P}}$ is a virtual Pareto archive by combining the Pareto archive $\mathcal{P}$ with the predicted policies from those tasks. This optimization problem can be formulated as:

$$\arg \max_{\{\boldsymbol{\omega}^i\}_{i=1}^n} \mathcal{O}(\mathcal{P}, \{\boldsymbol{\omega}^i\}_{i=1}^n) = \mathcal{H}(\mathcal{Q}) - \eta\mathcal{S}(\mathcal{Q}), \tag{3}$$

$$\text{where } \mathcal{Q} = \widetilde{F}(\widehat{\mathcal{P}}) = \Phi(\boldsymbol{F}(\mathcal{P}) \cup \{\boldsymbol{F}(\pi_{\boldsymbol{\theta}}) + \mathbf{\Delta}(\boldsymbol{\omega}^i)\}_{1...n}). \tag{4}$$

Φ is the function computing the set of non-dominated objective vectors, and $\mathbf{\Delta}$ is the prediction model trained for the current meta-policy and predicts the expected objective vector for each new task as $\boldsymbol{F}(\pi_{\boldsymbol{\theta}}) + \mathbf{\Delta}(\boldsymbol{\omega})$.

Intuitively, the more weight one objective is put on, the better the objective can be optimized. Thus, we choose a monotonic hyperbolic function as the prediction model:

$$\Delta_i(\omega_i) = d \cdot \frac{e^{a(\omega_i - b)} - 1}{e^{a(\omega_i - b)} + 1} + c,$$

where i corresponds to the i-th objective, and $\{a, b, c, d\}$ are the four parameters that need to be learned in each iteration. In order to fit this model, the algorithm stores the tuple $(\boldsymbol{\omega}, \boldsymbol{F}(\pi), \boldsymbol{F}(\pi'))$ in the meta RL phase, where $\boldsymbol{F}(\pi)$ and $\boldsymbol{F}(\pi')$ are the objective vectors of the meta-policy and the adapted policy, respectively.

The problem in Eq. (3) is actually a subset selection problem [26], [27], which is to select n preference weights to maximize the hybrid metric $\mathcal{O}$. We adopt a greedy procedure

TABLE I

RESULTS OF THE COMPARED ALGORITHMS ON THE SEVEN ENVIRONMENTS. WE RUN ALL ALGORITHMS ON EACH ENVIRONMENT WITH FIVE RANDOM SEEDS AND REPORT THE MEAN VALUE OF THE HYPERVOLUME $\mathcal{H}$ AND SPARSITY $\mathcal{S}$ METRICS. IN EACH ROW, THE BEST VALUES ARE BOLDED. THE NUMBER IN () DENOTES THE RANK (THE SMALLER THE BETTER) OF EACH ALGORITHM ON EACH ENVIRONMENT W.R.T. ONE METRIC. THE LAST ROW SHOWS THE AVERAGE RANK OF EACH ALGORITHM ON ONE METRIC.

Environment	Metric	RA	MOEA/D	Meta-MORL	OURS
HalfCheetah-v2	$\mathcal{H}(\times 10^6)$	5.66 (3)	5.69 (2)	5.21 (4)	**5.72** (1)
	$\mathcal{S}(\times 10^3)$	21.73 (4)	17.81 (3)	**4.25** (1)	5.96 (2)
Hopper-v2	$\mathcal{H}(\times 10^7)$	**2.13** (1)	2.09 (2)	1.18 (4)	1.94 (3)
	$\mathcal{S}(\times 10^4)$	4.00 (4)	3.24 (2)	3.92 (3)	**0.50** (1)
Swimmer-v2	$\mathcal{H}(\times 10^4)$	**2.67** (1)	1.58 (3)	1.16 (4)	2.34 (2)
	$\mathcal{S}(\times 10^1)$	8.40 (4)	**0.30** (1)	2.17 (3)	1.10 (2)
Walker2d-v2	$\mathcal{H}(\times 10^6)$	3.49 (3)	**3.63** (1)	2.17 (4)	3.57 (2)
	$\mathcal{S}(\times 10^4)$	0.47 (2)	2.09 (3)	3.68 (4)	**0.02** (1)
Ant-v2	$\mathcal{H}(\times 10^6)$	6.07 (3)	6.37 (2)	2.53 (4)	**6.60** (1)
	$\mathcal{S}(\times 10^4)$	4.61 (4)	2.40 (3)	0.85 (2)	**0.57** (1)
Humanoid-v2	$\mathcal{H}(\times 10^7)$	3.29 (3)	4.25 (2)	2.11 (4)	**4.51** (1)
	$\mathcal{S}(\times 10^4)$	7.93 (4)	1.14 (3)	0.97 (2)	**0.25** (1)
Hopper-v3	$\mathcal{H}(\times 10^{10})$	3.29 (3)	3.58 (2)	2.52 (4)	**3.62** (1)
	$\mathcal{S}(\times 10^3)$	0.78 (2)	**0.72** (1)	9.87 (4)	4.56 (3)
Average Rank	$\mathcal{H}$	2.43	2.14	4	**1.57**
	$\mathcal{S}$	3.43	2.29	2.71	**1.57**

in Algorithm 4 to select the n preference weights. The greedy algorithm maintains a virtual Pareto archive $\widehat{\mathcal{P}}$, which is initialized as $\mathcal{P}$. It iteratively selects the task that can best improve the hybrid metric of $\widehat{\mathcal{P}}$, and updates $\widehat{\mathcal{P}}$ by inserting the predicted policy of this selected task. As the number of candidate preference weights can be infinite, we use $2M$ candidate preference weights for approximation, where M preference weights are evenly sampled by discretizing the continuous weights and the other M ones are randomly sampled. That is, the best one from these $2M$ candidate preference weights is selected in each iteration.

V. EXPERIMENT

In this section, we aim to answer two questions: (1) how is the quality of the approximated Pareto set obtained from PG-Meta-MORL? (2) given some new objective preferences, can the use of meta-policy as the initial policy speed up the training process and find better policies? To answer these questions, we conduct experiments to examine the performance of PG-Meta-MORL on several multi-objective control environments designed by Xu *et al.* [12], which are based on the MuJoCo control environments [2], as illustrated in Figure 3.

A. Experimental Setup

1) Benchmarks: We use seven multi-objective control environments to examine the performance of the proposed algorithm PG-Meta-MORL, including six bi-objective environments and one three-objective environment. All of these environments are modified from the original MuJoCo environments with different objectives.

HalfCheetah-v2: The dimensions of observation space and action space are 17 and 6, respectively. Policies are optimized for maximizing forward speed and energy efficiency.

Hopper-v2: The dimensions of observation space and action space are 11 and 3, respectively. Policies are optimized for maximizing forward speed and jump height.

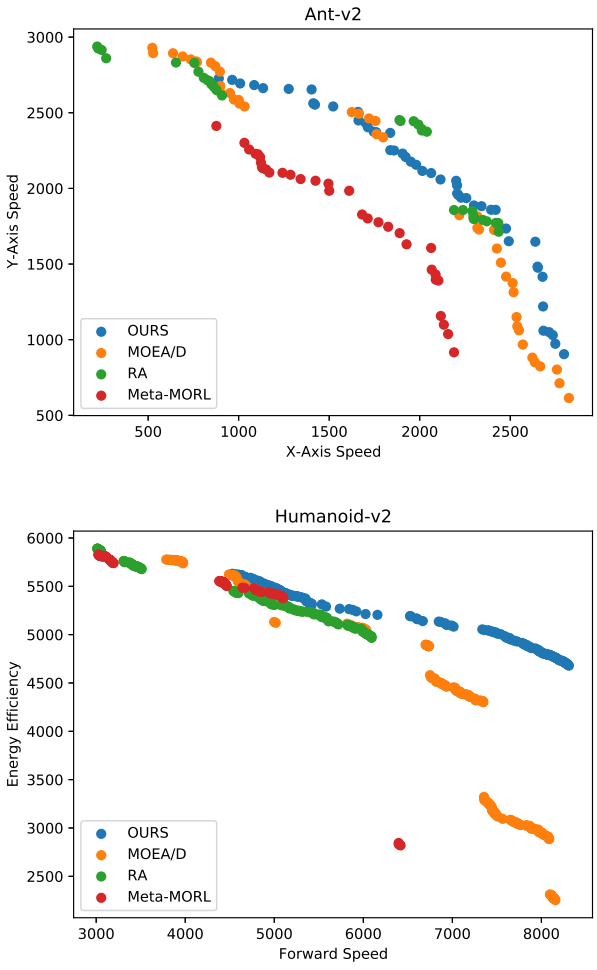


Fig. 4. The set of non-dominated objective vectors obtained from the compared four algorithms on the Ant-v2 and Humanoid-v2 environments.

Swimmer-v2: The dimensions of observation space and action space are 8 and 2, respectively. Policies are optimized for maximizing forward speed and energy efficiency.
Ant-v2: The dimensions of observation space and action space are 27 and 8, respectively. Policies are optimized for maximizing x-axis speed and y-axis speed.
Walker2d-v2: The dimensions of observation space and action space are 17 and 6, respectively. Policies are optimized for maximizing forward speed and energy efficiency.
Humanoid-v2: The dimensions of observation space and action space are 376 and 17, respectively. Policies are optimized for maximizing forward speed and energy efficiency.
Hopper-v3: The dimensions of observation space and action space are 11 and 3, respectively. Policies are optimized for maximizing forward speed, jump height, and energy efficiency.

2) Baselines: The proposed algorithm PG-Meta-MORL is compared with RA [6], MOEA/D [22], and Meta-MORL [11]. RA assigns a set of objective preferences and runs an RL algorithm to optimize the policies for each objective preference separately. MOEA/D decomposes a multi-objective optimization problem into a number of single-objective optimization sub-problems and then uses a search heuristic to optimize these sub-problems simultaneously and cooperatively. Meta-MORL iteratively trains a meta-policy with multiple randomly sampled tasks and then adapts the meta-policy to the control policies w.r.t. new objective preferences.

To compare all algorithms as fairly as possible, we set all the shared hyperparameters to be the same and run all algorithms with the same total interaction steps in each environment. For the Meta-MORL, we obtain the approximated Pareto set by maintaining a Pareto archive that contains all non-dominated policies and their corresponding expected returns during the optimization process, as same as our algorithm.

B. Experimental Results

For answering the first question, we test all the algorithms on the seven multi-objective control environments and compare the hypervolume and sparsity of their found set of policies. For answering the second question, we randomly select 30 new objective preferences and compare the performance of a random initialization policy with the meta-policy trained by our algorithm on the Ant-v2 and Humanoid-v2 environments.

1) Pareto Quality Comparison: We use the hypervolume metric and the sparsity metric to compare the quality of the approximated Pareto set obtained from our algorithm PG-Meta-MORL and three other algorithms on the seven multi-objective control environments. We run each algorithm on each environment for five random seeds and report the mean value of both metrics in Table I. The results in Table I demonstrate that PG-Meta-MORL obtains the smallest average rank in both metrics, implying that PG-Meta-MORL performs the best. We also plot the set of objective vectors of found non-dominated policies on the Ant-v2 and Humanoid-v2 environments in Figure 4, which clearly shows that PG-Meta-MORL is the best. The approximated Pareto set obtained from Meta-MORL is worse than others, which may be because the selected tasks for training the meta-policy do not contribute to improving the quality of the current Pareto archive, resulting in a sub-optimal meta-policy. MOEA/D is the most competitive algorithm as it periodically shares better solutions across sub-problems. However, it does not provide a way to optimize with new objective preferences.

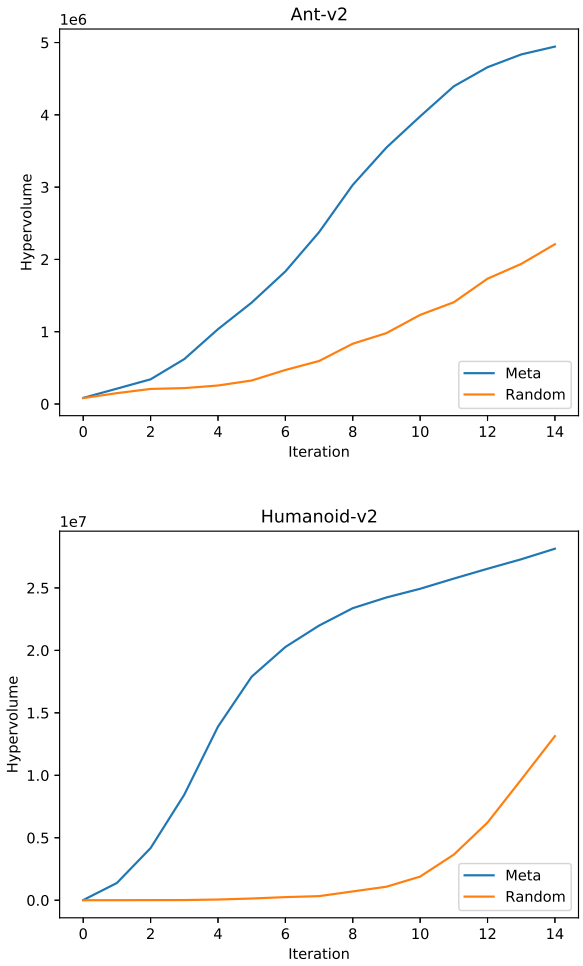


Fig. 5. Comparison of a random initialization policy with the meta-policy trained by the proposed algorithm PG-Meta-MORL as the initial policy on the Ant-v2 and Humanoid-v2 environments.

2) Adaptation Comparison: We randomly select 30 new objective preferences and use the hypervolume metric to compare the performance of a random initialization policy with the meta-policy trained by our algorithm as the initial policy on the Ant-v2 and Humanoid-v2 environments. We only train 15 iterations for each new objective preference. The results in Figure 5 show that compared to training a random initialization policy from scratch, using the meta-policy as the initial policy can speed up the training process and result in a better approximated Pareto set. This observation verifies that the meta-policy obtained from our algorithm can adapt faster and better to new objective preferences.

VI. Conclusion

In this work, we introduce a new algorithm PG-Meta-MORL to solve multi-objective control problems based on meta-learning, with the goal of adapting faster and better to new objective preferences. PG-Meta-MORL iteratively trains a meta-policy using multiple tasks with objective preferences selected towards improving the quality of the current Pareto set of policies based on a prediction model. We conduct experiments on several multi-objective control problems to verify the effectiveness of PG-Meta-MORL. The experimental results show that PG-Meta-MORL can not only find a high-quality approximated Pareto set but adapt faster and better to new objective preferences. In the current work, we only selected the MOEA/D as a representative MOEA for comparison. It would be interesting to examine the performance of more MOEAs, e.g., NSGA-II, for solving multi-objective control problems.

Acknowledgment

This work was supported by the NSFC (62022039) and the Jiangsu NSF (BK20201247). Chao Qian is the corresponding author.

References

[1] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction". MIT Press, 1998.

[2] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control". In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'12), pp. 5026-5033, Vilamoura, Portugal, 2012.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning". Nature, 518(7540):529-533, 2015.

[4] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization". In Proceedings of the 32nd International Conference on Machine Learning (ICML'15), pp. 1889-1897, Lille, France, 2015.

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning". In Proceedings of the 4th International Conference on Learning Representations (ICLR'16), San Juan, PR, 2016.

[6] S. Parisi, M. Pirotta, N. Smacchia, L. Bascetta, and M. Restelli, "Policy gradient approaches for multi-objective sequential decision making". In Proceedings of the 2014 International Joint Conference on Neural Networks (IJCNN'14), pp. 2323-2330, Beijing, China, 2014.

[7] S. Natarajan and P. Tadepalli, "Dynamic preferences in multi-criteria reinforcement learning". In Proceedings of the 22nd International Conference on Machine learning (ICML'05), pp. 601-608, Bonn, Germany, 2005.

[8] M. Pirotta, S. Parisi, and M. Restelli, "Multi-objective reinforcement learning with continuous Pareto frontier approximation". In Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15), pp. 2928-2934, Austin, TX, 2015.

[9] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision making". Journal of Artificial Intelligence Research, 48(1):67-113, 2013.

[10] C. Liu, X. Xu, and D. Hu, "Multi-objective reinforcement learning: A comprehensive overview". IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45(3):385-398, 2015.

[11] X. Chen, A. Ghadirzadeh, M. Bjorkman, and P. Jensfelt, "Meta-learning for multi-objective reinforcement learning". In Proceedings of 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'19), pp. 977-983, Macau, China, 2019.

[12] J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik, "Prediction-guided multi-objective reinforcement learning for continuous robot control". In Proceedings of the 37th International Conference on Machine Learning (ICML'20), pp. 10607-10616, Vienna, Austria, 2020.

[13] N. Riquelme, C. Von Lucken, and B. Baran, "Performance metrics in multi-objective optimization". In Proceedings of the 2015 Latin American Computing Conference (CLEI'15), pp. 1-11, Arequipa, Peru, 2015.

[14] E. Zitzler and L. Thiele, "Multi-objective evolutionary algorithms: A comparative case study and the strength Pareto approach". IEEE Transactions on Evolutionary Computation, 3(4):257-271, 1999.

[15] Z. Gabor, Z. Kalmar, and C. Szepesvari, "Multi-criteria reinforcement learning". In Proceedings of the 15th International Conference on Machine Learning (ICML'98), pp. 197-205, Madison, WI, 1998.

[16] S. Mannor and N. Shimkin, "The steering approach for multi-criteria reinforcement learning". In Advances in Neural Information Processing Systems 14 (NIPS'01), pp. 1563-1570, British Columbia, Canada, 2001.

[17] L. Barrett and S. Narayanan, "Learning all optimal policies with multiple criteria". In Proceedings of the 25th International Conference on Machine learning (ICML'08), pp. 41-47, Helsinki, Finland, 2008.

[18] R. Yang, X. Sun, and K. Narasimhan, "A generalized algorithm for multi-objective reinforcement learning and policy adaptation". In Advances in Neural Information Processing Systems 32 (NeurIPS'19), pp. 14610-14621, Vancouver, Canada, 2019.

[19] M. Reymond and A. Nowe, "Pareto-DQN: Approximating the Pareto front in complex multi-objective decision problems". In Proceedings of the Adaptive and Learning Agents Workshop at the 18th International Conference on Autonomous Agents and MultiAgent Systems, 2019.

[20] A. Castelletti, F. Pianosi, and M. Restelli, "Multi-objective fitted q-iteration: Pareto frontier approximation in one single run". In Proceedings of the IEEE International Conference on Networking, Sensing and Control (ICNSC'11), pp. 260-265, Delft, The Netherlands, 2011.

[21] K. Deb, "Multi-objective optimisation using evolutionary algorithms: An introduction". In Multi-objective Evolutionary Optimisation for Product Design and Manufacturing, pp. 3-34, 2011.

[22] Q. Zhang and H. Li, "MOEA/D: A multi-objective evolutionary algorithm based on decomposition". IEEE Transactions on Evolutionary Computation, 11(6):712-731, 2007.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II". IEEE Transactions on Evolutionary Computation, 6(2): 182-197, 2002.

[24] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks". In Proceedings of the 34th International Conference on Machine Learning (ICML'17), pp. 1126-1135, Sydney, Australia, 2017.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms". arXiv preprint arXiv:1707.06347, 2017.

[26] C. Qian, Y. Yu, and Z.-H. Zhou, "Subset selection by Pareto optimization". In Advances in Neural Information Processing Systems 28 (NIPS'15), pp. 1765-1773, Montreal, Canada, 2015.

[27] C. Qian, Y. Yu, and K. Tang. "Approximation guarantees of stochastic greedy algorithms for subset selection". In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18), pp.1478-1484, Stockholm, Sweden, 2018.